ISSN: 2153-1234 Vol.1, No.1, July 2010

Scientific Research

Journal of Information Security

ISSN: 2153-1234

www.scirp.org/journal/jis

Journal Editorial Board

ISSN: 2153-1234 (Print) ISSN: 2153-1242 (Online)

http://www.scirp.org/journal/jis

Journal of Information Security, 2010, 1, 1-44
Published Online July 2010 in SciRes (http://www.SciRP.org/journal/jis/)

TABLE OF CONTENTS

Volume 1 Number 1 July 2010

Journal of Information Security (JIS)
Journal Information

SUBSCRIPTIONS

The *Journal of Information Security* (Online at Scientific Research Publishing, www.SciRP.org) is published quarterly by Scientific Research Publishing, Inc., USA.

Subscription rates:

Print: $50 per issue.
Electronic: free, available on www.SciPR.org.
To subscribe, please contact Journals Subscriptions Department, E-mail: service@scirp.org

SERVICES

Advertisements
Advertisement Sales Department, E-mail: service@scirp.org

Reprints (minimum quantity 100 copies)
Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.
E-mail: sub@scirp.org

COPYRIGHT

Statements and opinions expressed in the articles and communications are those of the individual contributors and not the statements and opinion of Scientific Research Publishing, Inc. We assumes no responsibility or liability for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained herein. We expressly disclaim any implied warranties of merchantability or fitness for a particular purpose. If expert assistance is required, the services of a competent professional person should be sought.

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:
E-mail: jis@scirp.org

Journal of Information Security, **2010, 1, 1-10**
doi:10.4236/jis.2010.11001 Published Online July 2010 (http://www.SciRP.org/journal/jis)

Micro-Architecture Support for Integrity Measurement on Dynamic Instruction Trace

Hui Lin[1], Gyungho Lee[2]
[1]*ECE Department, University of Illinois at Chicago, Chicago, USA*
[2]*College of Information and Communications, Korea University, Seoul, Korea*
E-mail: *hlin33@uic.edu*, *ghlee@korea.ac.kr*

Received July 12, 2010; *revised July* 16, 2010; *accepted July* 20, 2010

Abstract

Trusted computing allows attesting remote system's trustworthiness based on the software stack whose integrity has been measured. However, attacker can corrupt system as well as measurement operation. As a result, nearly all integrity measurement mechanism suffers from the fact that what is measured may not be same as what is executed. To solve this problem, a novel integrity measurement called dynamic instruction trace measurement (DiT) is proposed. For DiT, processor's instruction cache is modified to stores back instructions to memory. Consequently, it is designed as a assistance to existing integrity measurement by including dynamic instructions trace. We have simulated DiT in a full-fledged system emulator with level-1 cache modified. It can successfully update records at the moment the attestation is required. Overhead in terms of circuit area, power consumption, and access time, is less than 3% for most criterions. And system only introduces less than 2% performance overhead in average.

Keywords: Integrity Measurement, Remote Attestation, Software Vulnerability, Trusted Computing

1. Introduction

Nowadays, computer under different platforms interacts with each other through internet environment. Although this provides convenience and increased functionality, it is necessary to securely indentify software stack running in remote systems. Effective remote attestation mechanism has drawn lots of research interests. Trusted Computing Group (TCG) first standardized the procedure to launch a remote attestation [1]. As defined, the protocol consists of three stages: integrity measurement, integrity logging, and integrity reporting [2]. The function of integrity measurement is to derive a proper measure that is an effective representation of a given platform status. In order to narrow down the range of such measures, Trusted Computer Base (TCB) is defined as hardware components and/or software modules whose integrity decides the status of a whole platform. Consequently, integrity measurement can simply based on measures from the TCB, which reduce performance overhead in measurement and attestation. Integrity logging is the process of storing aforementioned integrity measure in protected storing space. This process is not mandatory, but highly recommended to reduce the overhead due to repeated calculation for integrity measurement. The last step, which is called integrity reporting, is to attest system based on the stored or calculated integrity measures.

Computer systems emphasize different security goals per contexts. While system integrity is more important in one situation, the other may concern more about data privacy. Integrity measurement is strongly related to security policy applied to specific computer system and consequently results in different attestation mechanism. TCG's specification describes an integrity measurement during system's booting process. This mechanism is called "trusted boot". At the very beginning, a hardware signature, which is stored in some security-related hardware components, is used as the root of the trust. Current hardware vendors design Trusted Platform Module (TPM) to provide such functionality. As each entity is loaded into memory, the integrity measures on the binaries are calculated one by one and form a trust chain at last. Unlike secure booting, system takes measurements and leaves them to the remote party to determine system's trustworthiness. TCG's attestation based on such a trusted booting is also called binary attestation [2].

Other integrity measurements still follow TCG's "measure-before-load" principle. Property attestation and semantic attestation both try to extract the high level property or semantic information from binary measurement. So it will be more efficient and effective to validate whether a security policy is hold or violated on such a measured property a priori. IBM's Integrity Measurement Architecture (IMA) based on the TCG's trusted booting extends the approach into application software stack. IMA is now a security module provided by Linux kernel since version 2.30 [3].

A good integrity measurement should be able to derive a reliable measure that represents the status of computer system. From the resulting measure, a challenger (the remote entity which is interested in attesting the system) should be able to tell the system's updated security-related capability such as whether the memory has been ever corrupted by attacker, or whether programs can be properly executed in isolation, or whether cryptography keys are securely stored, and so on. On the other hand, measurement procedure should be transparent to the local user and introduces little performance overhead.

Current integrity measurements face problems of gathering sufficient history information on what has been done to the computing device. When each entity is loaded into memory, measurement of its binary codes is recorded. However, there will be a "measurement gap" at the moment when measurement results are requested. System status may be different from the recording in measure. Furthermore, measurements are made directly on program's executable code residing in main memory. There exists another "behavior gap" between instructions executed in the processor and executable codes in the memory. The integrity measure of executable code in the memory can be a good measure to represent the system state. However, as different attacks occur from internet, this is becoming less sufficient for a remote challenger. For those programs running for a long time, such as server programs, a static measurement prior to execution may have little relation to the system status at the current moment. As a result, more accurate measurement, which can include program behavior, needed to tell challenger all history of bad behavior. This results in a better decision on trustworthiness of the system.

However, with more information included, overhead to measure programs' state increases. As a result, some measurements are targeted to specific data, such as processor control data, function pointer in memory, network traffic, intrusion detection, and so on. Measurement is often restricted in order to utilize only limited amount of information. Consequently validation of system against a certain security policy introduces little performance overhead. This policy-driven attestation or validation schemes are largely based on limited information specific per intended attack scenarios. The problem is that although it is efficient in their proposed situation, portability of such measurement is very low. In different situation, attestation may require a big modification which also exerts a large performance penalty.

In order to provide updated integrity measurement as system evolves, we propose an original dynamic instruction trace measurement (DiT) to include in the metric dynamic instructions-level behaviour in the processor with the help of simple micro-architecture modification. However, instruction-level trace can vary from time to time, with some part of the program being executed more frequent than the other. Directly recording the processor behavior causes lots of performance overhead and without increasing any accuracy. In stead of applying measurement in processor, we still perform the operations on the memory. As a result, most function interfaces provided before, such as the ones proposed in TCG or IBM's IMA, can be maintained.

Cache is an evolutionary design building a bridge between the memory and processor to reduce access delay. However, in this paper, we modify the structure of the instruction cache to the one similar to the data cache. The consequence is that instructions can also be written back to the memory. As program continues its execution, code region in its address space no longer stores codes loaded before execution but records instructions which are executed. We improve the integrity measurement for trusted computing in the following aspects:

1) Extending the measurement scope. When the security-sensitive program is loaded and starts execution, DiT writes back instructions into memory. Consequently, binary code located in its address space records instructions which are actually executed.

2) Facilitating attestation for different security policy. DiT only replaces static measurement with dynamic one. As a result, it changes little on the high level interface and provides a better general solution to diverse scenarios.

3) Writing back instructions does not require the involvement of operating system. Thus, DiT builds a connection between what has seen inside processor and what resides in memory. This procedure does not require trusting operating system, which in some cases can be corrupted by attackers.

The paper is structured as follows. Section 2 presents the background on trusted computing and integrity measurement. In Section 3, we present DiT's design in details. To avoid potential hazards from attacks, we propose several hardware-wise recommendations in Section 4. The experimental results and analysis are given in Section 5. Finally, the related work and conclusion are made in Section 6 and Section 7.

2. Background

2.1. Trusted Computing

Trusted computing deals with computer system in a hazard environment. Though there is lack of ubiquitous definition of trust, this paper refers the one from Trusted Computing Group (TCG) specification. Trust is mentioned as the expectation that a device will behave in a particular manner for a specific purpose [2].

Trusted Computing Base (TCB) is specified as any hardware and/or software components within the interested platform, whose safety can affect the status of the whole system. The assumption is made that if TCB is safe, system can be trusted. However, TCB's components vary from systems. In some situations, it may work with integrity validation mechanism; as a result, run-time critical data values are included in TCB. However, on other situations, execution of security-sensitive programs, such as encryption/decryption operation, is important to system's proper function; some architecture components, which guarantee privacy of such application program, are chosen in TCB. TCG has summarized diverse application scenarios and concludes that it should include the following two characteristics:

1) Isolated Execution, or protected execution. The computing platform should be able to equip security-related application program with an isolated environment. As a result, no other legacy programs can access or corrupt information it relies on. To achieve this property, many researchers adopt the virtualization approach or hardware extension to legacy computer architecture [4].

2) Remote Attestation. Each computing platform should be able to provide mechanisms to: (1) securely measure TCB's safety state; (2) protect measure log stored locally; (3) transmit measure to remote challenger.

2.2. TCG's Binary Attestation

TCG defines a binary attestation to provide a trusted booting. Whenever an entity is loaded into memory from the moment machine is physically turned on, TPM applies cryptographic hash function, say *Hash*, on its executable code to make a measurement result, say *M*. The binary measurement for each entity is logged separately. Additionally, each measurement is also stored in one of Platform Configuration Registers (PCRs) in TPM by making the cryptographically *extend* operations with PCR's current value, PCR_t, *i.e.*, new PCR values $\text{PCR}_{t+1} = Hash(\text{PCR}_t|M)$, where|denotes concatenation. When verifier requires attestation, TPM sends measurement logs (in local hard disk) and the corresponding PCR value to the verifier. He will recalculate hash result based on measurement logs. The comparison between newly-computed hash result and PCR value can tell whether untrusted behaviour within the environment has ever modified PCR value, measurement log, or executable code itself.

Using binary attestation facilitate verification in mainly two aspects. 1) measurement with such format hides many different high-level implementations and reduces the complexity to calculate measure log and PCR value; 2) It successfully separates measuring and verification. Attestation does not try to prevent a system from illegal behaviour that might compromise system. It only records the history of loaded code, securely sends them to the verifier and leaves the verifier to make trustworthiness decision.

2.3. Integrity Measurement on the Application Program

Starting from the root of trust provided by TCG, Integrity measurement architecture (IMA) from IBM takes the first step to extend measurements from booting process to application level programs. IMA is provided as a software module to Linux kernel from the version 2.30. It provides measurements regarding to current system's software stack. The whole project provides integrity measurement but does not propose any detailed attestation mechanism. Measurements provide evidences showing whether system is corrupted by certain rootkit attacks or not.

IMA measures each individual component before it is loaded. With the help of *extend* operation, trusted booting forced execution to follow only one legal order. However, in application level, programs can execute different threads in parallel; program order does not related to trusted condition any more. So IMA groups measure together instead of applying extend operation one by one.

But IMA's is following TCG's "measure before loading" principle, therefore it inevitable maintains shortcomings of the binary attestation, such as its ineffectiveness to reveal hardware attacks or the software attacks after the program is loaded and executing.

3. Architecture Extension to Measure Instruction Level Behaviour

3.1. Design of Integrity Measurement in Application Level

DiT is based on IBM's IMA which provides comprehensive measurement over software stack. In IMA, all executable codes and chosen structured data are included in the measurement log. Any data which are loaded by operating system, dynamic loaders, or applications with identifiable integrity semantics are hashed. Measurement can be made automatically at the moment when codes or data are loaded into main memory. As programs continue their execution, kernel is able to measure its own changes. Similarly, every user level process can measure its own security sensitive inputs, such as its configuration files or scripts. The consequent 160 bit value from hash calculation becomes an unambiguous identity for such software module. Challenger can distinguish different file types, versions, and extensions by this unique fingerprint.

As system evolves, IMA collects hash results into a measurement list which is stored locally. The integrity of this list is of a great importance. Therefore, IMA uses TPM to prevent any modifications made on measurement list. Platform Configuration Register whose value can only be changed by physically system rebooting or TPM extend operation provides protected storage. Extend operation is applied on each value stored in the measurement list. Since it is impossible to restrict application-level softwares into a small number of orders, order of each value in the list is not used to validate the trustworthiness of the system.

3.2. Writing Back the Instructions

Although IMA provides measurement of all loaded software, it still follows TCG's "measure-before-loading" mechanism. As a result, "metric gap" and "behaviour gap" can largely degrade efficacy of measure log.

The "metric gap" occurs when measurement does not represent the updated state of the system. Application program can run for a long time, such as server program. So it may be a long period since the measurement is made. During this time interval, memory is possible to be corrupted. Attacks, who can take root privilege, can modify loaded executable codes. However, it is possible to detect such modification when the codes are being executed again. This is the basic assumption made in former tamper resistance design [5]. As executable code is hashed again, resulting measure will be different. However, attestation is made asynchronously to system's operation. It is possible that attestation is made before executable codes are hashed again. As a result, measurements may give challenger a misinformation about what is running at the moment.

Figure 1 makes a comparison between three measurement mechanisms: DiT proposed in this paper, IMA and Aegis which is a typical secure processor design to achieve tamper evidence and resistance environment [5]. When IMA measures executable code, it makes comparison to values which are calculated before. In Aegis, if software's execution relies on a program, the measurement of this program is calculated again and comparison is made to former calculated value. In these two situations, the challenger may still get measurements from which the system can regarded as trusted but actually the memory is already corrupted.

"Metric gap" can be resolved by applying a measure to executable code at the moment of attestation is made (which is also reflected in **Figure 1**). However, "behaviour gap" can further introduce more severe problem. This describes the fact that static codes in memory are different from instructions executed in processor. But it is instructions executed in processor finally corrupt the system. On the other words, executing instructions are truly represent the trustworthiness of the system. What makes things worse is that many attacks do not rely on the modification on program's executable code to launch malicious behaviour any longer. For example, buffer overflow attack has diverse implementations. One of them is to insert codes directly in stacks which make detection only possible for a very short period of time. Challenger should also be able to know such deleterious execution since this system is vulnerable to attacks in the future.

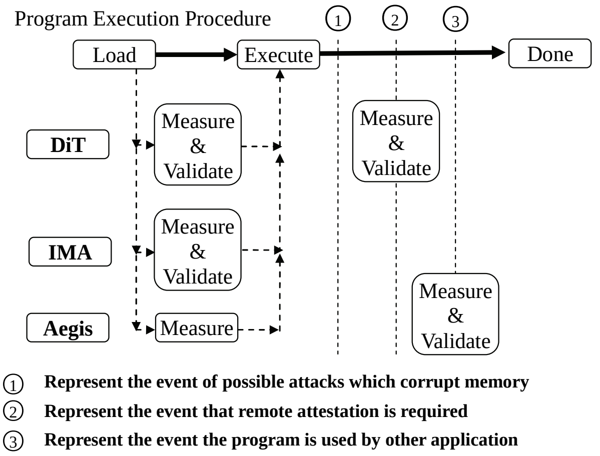

Program Execution Procedure
①
②
③
Load
Execute
Done
DiT
Measure & Validate
Measure & Validate
IMA
Measure & Validate
Aegis
Measure
Measure & Validate
① Represent the event of possible attacks which corrupt memory
② Represent the event that remote attestation is required
③ Represent the event the program is used by other application

Figure 1. "Metric Gap" occurs in the design of IMA and Aegis.

No matter how attacks exploit software vulnerability, it finally needs to execute its code in the processor. As a result, researchers also propose to records behaviour in the processor. To reduce performance overhead, they only analyse behaviour of critical instructions, such as indirect branch or critical data. Measuring those data may work for certain security policy but lacks of portability and extendibility to future unknown attacks.

Measuring all instructions is a challenge. Instructions are fetched from memory, but dynamic execution flow varies from situation to situation. It is impossible to provide limited number of unique state to represent safety of such execution. On the other hand, collecting all possible states are computationally impossible to make.

DiT does not directly measure all executed instruction in processor. It maintains large part of original measurement interfaces which measure codes in memory. What DiT successfully makes is to extend architecture's pipeline to build connection between processor and memory (**Figure 2**). It proposes to store back instructions into its original locations after they are fetched into pipeline. The

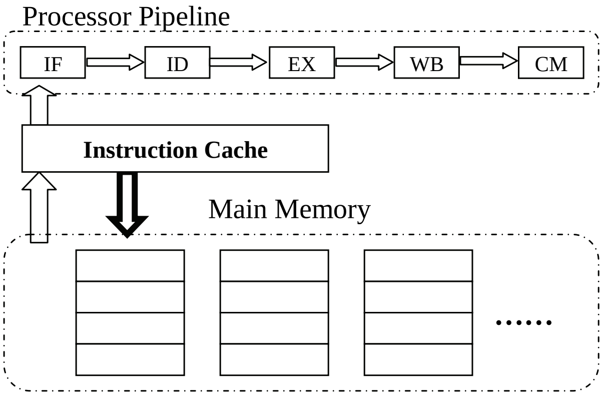


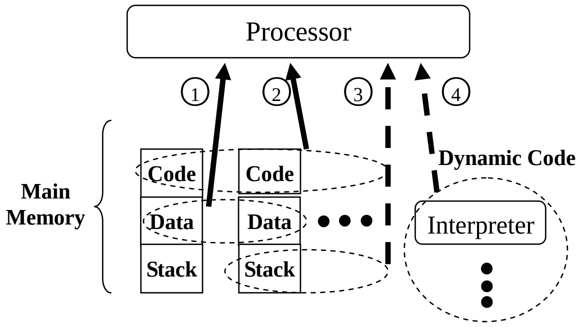
Figure 2. Strcture to measure dynamic instruction trace.

purpose is to resolve “behaviour gap” between processor and memory. This is not an intention to record all possible run-time execution paths but to store instructions which are truly executed into measure log.

With such modification, what to measure and when to measure have to be carefully designed. Program’s address space consists of data region, code region, and stack to record program execution context. In IMA, all executable code and part of related data, which are dynamically loaded by operating system, are measured (**Figure 3**). DiT will cover all code regions, data regions and stack as long as there are some instructions being written back to them.

Due to attacks, instructions can come from other locations rather the code region. This not only makes DiT to expand measurement range to include memory region such as stack, but also require it to add several temporal points to make such measurement. We can still use the aforementioned buffer overflow as the example. Stack contents vary as program enters into different contexts. Malicious code hidden there may soon be overlapped by unrelated information, such as parameter passed by following function call. As a result, malicious code should be measured on time before it is eliminated by legal ones.

To insert proper temporal points is a trade-off between detection ability and performance overhead. The performance overhead in original integrity measurement mechanisms is amortized, which is due to the fact that hash calculation is made at the frequency of program loading. From many former anomaly-detection approaches, successful corruption usually results in some changes in instructions level behaviour, such as cache miss, prediction miss and so on [6]. Furthermore, hash operation, which calculates memory code, is easily performed in parallel with program’s normal operation. In the current work, one inevitable measurement is added. DiT launches the measurement at the moment of attestation requirement is made, which at least resolve the metric gap between measure and system state.



①: **Data with integrity semantic is loaded by operating**

②: **Executable codes are fetched from memory**

③: **Malicious codes are fetched in statck or other illegal location**

④: **Codes are executed dynamically**

Figure 3. Behavior gap occurs due to attacks or dynamic generated code.

3.3. Introduce Randomization through the Use of Cache

Most personal computers usually have two level caches. Instruction and data are divided in the level-1 cache while level-2 cache is usually a unified cache which stores them together. DiT includes cache into the procedure of writing back instructions to the memory which “reverse” the procedure when instructions are fetching from it. In order to make write back work, instruction cache should be appended with few state bit just as data cache does.

By replacing structure of individual cache to the one of data cache, processor actually does not need to have the actual action of “writing back”. It only needs to set a corresponding status bit and leave the work to cache and memory management unit. Whenever cache miss occurs, instruction cache first stores values in cache entry back to the memory and then read other instructions instead of overwriting it directly.

Usually, it is hard to predict cache miss. This randomizes the time to write back instructions. As a result, another level of protection which prevents attacks from learning this measurement and hide its malicious codes can be made. Besides, this operation does not need the involvement of operating system. Even when OS is not trusted, such as the kernel is corrupted, writing back operation can always be executed properly.

Current micro-architecture design can further help our design to write back instructions. Since level-2 cache is unified, only level-1 instruction cache requires modification. And the modification is restricted to small number of status bits added to each cache entry. As a result, overhead on chip area, power consumption and access time to cache entry (which is also called cache hit latency) is reasonable. Furthermore, instructions usually holds much better locality references than data cache which results in much less cache miss. Consequently, performance effect from writing back instructions is also possible to be restricted to a small amount.

4. Further Micro-Architecture Recommendations

With the proposed design, DiT is able to measure large amount of program's execution. However, it may still miss some situation due to current operating system design as well as diverse attacking mechanism. In this section, we propose several extra hardware recommendations to further resolve those issues.

4.1. Adding Measurement Point

With the aid of DiT, measurement will be recalculated with program's execution. There is still a possible hazard that attacker replaces correct codes to the malicious ones (that he injects before) in memory to avoid proper measuring (similar to the way he/she can insert malicious code) after malicious codes are stored back. As a result, adding more measurement points is necessary to provide another protection level on DiT itself.

The cache miss or branch prediction miss indicate a behaviour change in instruction level, which can be used as a point to recalculate measurement. To further reduce performance, we propose to make the measurement at the moment when the potential attacks are going to happen. However, from current study in software vulnerability, to detect the proper attacking potential is proved to be another difficult issue. As program is running, its address space records its execution state through the use of stack and/or heap and so on. However, its code space remains stable. Operating system design provides a good protection when it launches different code space to execute, such as the design of context switch. However, attackers successfully inject or exploit new or existing code space to avoid reliable operation provided from operating system.

As a result, we can make measurement when instructions are written back to the memory location which is outside of the code region (not address space) for the current running programs. As each program is loading its code, we can records its physical address in memory into a table and store it in a memory management unit. A comparison between written back instruction and each physical address of a code region can indicate which program this instruction is belong to. If it does not belong to any legal program, we can raise an exception. On seeing this exception, measurement is not also necessary since action of avoiding measurement is made.

By such architecture recommendation, DiT can achieve the validation such that every instruction executed in processor should be from executable code space which is properly loaded into memory before. Consequently, DiT can prevent injected code attacks while making measurement.

4.2. Measuring the Run-Time Generated Code

Different from compiler which generates executable codes, interpreter executes machine instructions on the fly. In our proposed design, integrity measurement is only capable of measuring binary codes of interpreter itself, dynamic codes generate by interpreter to processor are not recorded (**Figure 3**). On the other hand, more popular attacks begin to adopt this mechanism. Such attacks, including sql injection, cross script attacks, dominate current web applications. This presents a big challenge to provide accurate measurements to remote challenge, as malicious behaviours are extracted from user input and getting execution one instruction by another. Measuring executable codes from memory becomes impossible.

When instructions are generated from interpreter, DiT finds that there is no source memory location to which such dynamic instructions can transmit. Our proposed method is to "deceive" the interpreter that the dynamic executed codes is actually dynamic loaded. As a result, it can follow the predefined procedure to make such measurement.

This is achieved by creating a new memory region which can be linked to the memory space of interpreter's process. Current operating system, such as Linux kernel, provides safe interface to dynamically add or remove memory region from process' address space. It will be easy to include such secondary code region to interpreter's address space.

This is equivalent to adding a container to store dynamically executed code; however, the measurement will not be possible at the "load" time, since the container is empty at this moment. Only at the end of execution when all executed codes are written back, proper measure is going to be made on the full container.

5. Experiment and Result Analysis

In order to analyse applicability of DiT, two sets of experiments are conducted respectively. The first one simulated measurement mechanism, especially the situation to hash program's code upon asynchronous attestation. Then another set of experiments are made to detect hardware and performance overhead caused by modification on level-1 instruction cache.

5.1. Implementing Measurement

Different from IBM's IMA which implements all integrity measurements within Linux kernel, we implement it in the hardware level. DiT is integrated into Bochs which is a full-fledged open source × 86 PC emulator. It is used to emulate entire system from × 86 architecture to virtually instrumented monitor.

Through our experiment, we find that write back instructions to memory causes some instability for emulated system. As a result, DiT focuses on certain target program and only stores its on-fly instructions into memory. As mentioned before, TCB provides an isolation execution environment for the security-related programs. By implementing writing back instructions for only interested program, we believe that DiT can more practically simulate TCB's execution model.

We install Gentoo Linux with the kernel of version 2.6.29 in the emulation. To track process information, kernel is modified so that hardware emulator becomes aware of software context switch. Since version 2.6.x, kernel introduces the late binding for the context switch, so both *exec* () and *sched* () functions are modified. Consequently, process identity, such as Process ID and Process name is updated into a global variable as soon as process is created and loaded into memory.

Besides the operating system modification, we also implement several virtual debugging monitor. One of the most critical interfaces which DiT inserts is the one that halts the execution of current program in emulated operating system and hashes the code region in the address space of current active process. This efficiently emulates the situation that measurement is made upon the attestation request is sent from remote challenger.

5.2. Performance Overhead

In order to make instructions cache to write back, several extra status bits are required to each cache entry which is similar to the structure in data cache. Since in most micro-architecture design, level-2 cache is designed as a unified cache, only level-1 instructions cache needs modifications. To make a comprehensive analysis of such change, area, power consumption and access time is emulated under CACTI 5.0 [7]. The parameter of unmodified cache is the same as the one used in **Table 1**, which is also used in SimpleScalar for performance experiment. Five extra bits are added to each entry of the instruction cache to implement the write back mechanism. With the simulation results given from **Table 2**, largest power overhead is less than 10%. Overhead of other criterion is actually ignorable. Especially, modification has little effect on access time of level-1 cache.

Table 1. Architecture parameters.

Parameter	Value
Fetch/dispatch/issue width	4
Instruction window	128 entries
register update unit size	128 entries
Load/Store Queue	64 entries
I-cache	128K 1 way set-asso., 1-cycle hit time
D-cache	128K 1 way set-asso., 1-cycle hit time
L2 cache	Unified, 1M, 4 way set-asso, 6 –cycle hit time
Memory	100 cycles access time, 2 memory ports
Function unit	4 Int ALUs, 1 Int MUL/DIV, 4 FP Adder, 1 FP MUL/DIV

Table 2. Area, power and access time overhead for modified L1 cache.

Technology node	Overhead criterion	Normal L1 Cache	Modified L1 Cache	Overhead
90 nm	Area (mm^2)	2.59811765	2.66909173	2.73%
	Power (W)	5.23044172	5.23787143	0.142%
	Access time (ns)	1.40756434	1.40756434	0.00%
32 nm	Area (mm^2)	0.36714162	0.36929974	0.588%
	Power (W)	3.54005779	3.87976541	9.59%
	Access time (ns)	0.43442463	0.43875809	0.998%

We tested SPEC2000 benchmarks running in Simplescalar which models an out-of-order superscalar processor [8]. Reference inputs are adopted and we skip instructions of the number which is specified by SimPoint [9].

Writing back instructions are not supported in Simplescalar, as a result, we modify source codes of sim-outorder (the out of order simulators) such that right after each time a read access is performed to the level-1 cache, a write access to the same entry in the cache is launched. The parameter to run Simplescalar is given in **Table 1**.

We collect all number of level 2 cache access and cache misses for each program in SPEC 2000. The number of level-2 cache access varies to different programs. In *eon*, *perlbmk* and *vortex*, the modified level-1 cache increases more than 50% of level 2 cache accesses. But for other benchmarks, the change is not that obvious. We only select the increase of level-2 cache access with more than 0.01% among all 26 programs (**Figure 4**).

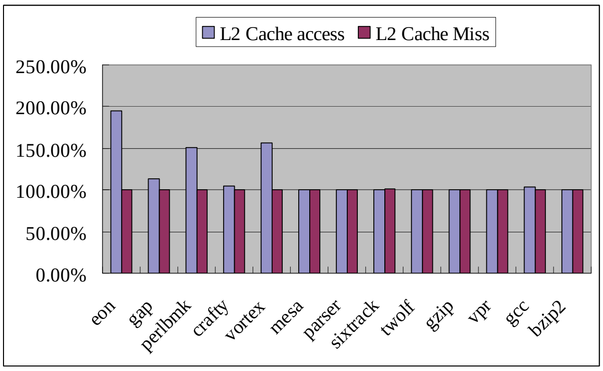


Figure 4. Normalized Level-2 cache access and cache misss.

Although there are big increases in level-2 caches access, this does not simply increase the corresponding cache miss. All cache miss due to the modification of level-1 cache is increased with less than 1%. This is probably due to the fact that level-2 cache holds a good locality references for instructions. As a result, performance overhead for all benchmark programs is ignorable as shown in **Figure 5**. The largest performance overhead measured in IPC is less than 5%.

6. Related Work

6.1. Tamper Resistance Design

Execute Only Memory (XOM) has included whole memory space in the trusted computing base as most adversaries launch the attacks to corrupt memory [10]. In order to guarantee both integrity and privacy of the data in memory, encryption components are included in the legacy architecture design. Data transmitted from processor to memory is encrypted and reversely, they are decrypted for execution in processor

Aegis [5] follows the same assumption that memory can not be trusted. It hashes executable code when a program is loaded into the memory for execution. At this moment, any other code and data that the program relies on is checked to guarantee that the program is started in a trusted environment. In the situation that operating system can not be trusted, Aegis introduce security related module and hardware component into the legacy processor. Tamper resistance design does not make assumption on how memory is corrupted thus it is able to detect simple hardware attacks.

Tamper resistance design is similar to our approach in the way of measuring untrusted code. However, they are holding the assumption that detection of static code can be found on moment the software is used again. As mentioned before, attestation can be made before next-use of software modules, so directly adopting tamper resistance approach introduces "metric gap". On the other hand, they are unable to measure program's runtime behaviour as well.

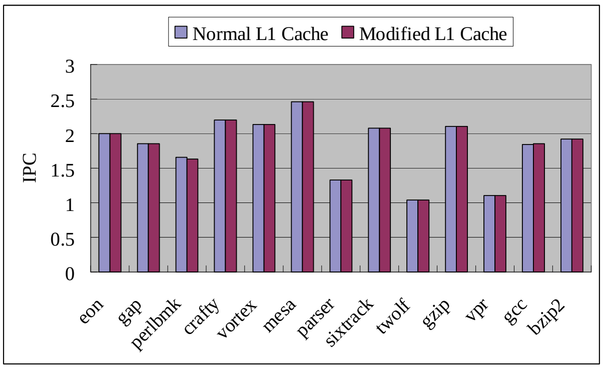


Figure 5. Comparison of IPC number with normal Level-1 cache and modified L1 cache in processor.

6.2. Integrity Measurement

TCG first standardize the procedure to make a remote attestation, besides, it also recommends an integrity measurement methods which is efficient during system booting. This binary attestation can only record what the programs are running on the platform and use the identity and the loading order of programs to system state after booting.

IBM's IMA, Integrity Measurement Architecture, inserts measurement interface into Linux kernel. As each program is loaded into memory, its executable code is hashed. When a program is further loading other codes or security-critical data structure, measurement is made as program transfer its control flow. However, software vulnerability which is exploited by attackers during each individual program's execution can also spoil measurement.

Based on the observation that modifications made in kernel space is usually permanent, Loscocco *et al.* propose to measure dynamic data structure which is critical to kernel control flow [11]. Such dynamic data structure is called contextual information, which is used to represent the state of the whole computing system. But this method is not efficient to be used in the user space operations.

6.3. Property Driven Remote Attestation

Binary measurement has the advantage of easy calculation and application-independence. Since hash calculation is irreversible, directly exploiting such metrics provides a big challenge and performance overhead. As a result, different attestation, which adopts different metrics, is proposed.

With specific security policy being set for the attesting system, property attestation and semantic attestation [12-14] propose to derive system high level information instead of the pure software stack. The extracted metrics can be directly used against security policy. Measurement methods may be implemented differently, but measurement is decided by security policy. As security policy changes, it is less flexible to change measurement implementation accordingly. As they indirectly include validation part into attesting platform, attesting platform's performance overhead is increased and validation procedure is also put under the hazardous environment. We propose DiT which designs an application-independent measurement which separates validation and measurement just as binary attestation does.

Some other researches also consider that program's run-time behaviour as a validation metrics, however, with many limitations. Alam *et al*. propose a behaviour attestation method [15]. However, the behaviour is defined as the quality of service the system can provide, connection latency, and so on. Consequently, this attestation implementation designed for web services only which lack the portability to be applied to other applications programs.

7. Conclusions

Ever since TCG standardized the procedure to launch a remote attestation, how to exchange the trust measure efficiently between computer systems under diverse platoforms has been a popular open research issue. Locally, attesting mechanism derives integrity measure based on software stacks on which trust decision is made. TCG introduces a binary attestation during system booting and many integrity measurement implementations are proposed following the "measure-before-loading" principle. Those measurements do not take into the account the actions after each program begins its execution. As a result software vulnerability which can corrupt both system status as well as measurement operation can introduce the "behavior gap" and the "metric gap" between program runtime behavior and consequent measurement. DiT, the dynamic instruction trace integrity measurement, is proposed as assistance to the current integrity measurement methods. By changing the structure of instruction cache, instructions are stored back into memory when cache miss occurs. As a result, code region in programs address space actually contains dynamic instructions trace executed in processor. By applying integrity measurement based on this change, DiT successfully include most updated system state to the moment when attestation is required.

We have experimented this attestation mechanism in *bochs*, a full-fledged emulator, with a current updated version of Linux kernel installed. We have successfully simulated the procedure of measuring program's code (or trace) at the time when attestation is made. To further analyze the change made in level-1 instruction cache, Cacti is exploited to check area, power consumption and access time overhead. SPEC2000 benchmarks are run on the modified Simplescalar to analyze the performance overhead. As we only limit our small modification in level 1 instruction cache, the overhead in terms of circuit area, power consumption, and access time are all reasonable, and also the performance overhead is marginal.

8. Acknowledgement

This work was supported by the IT R&D Program of MKE/KEIT (2010-KI002090, Development of Technology Base for Trustworthy Computing).

9. References

[1] "Trusted Computing Group." http://www.trustedcomputinggroup.org

[2] TCG Specification Architecture Overview Specification Revision 1.4, Trusted Computing Group (TCG), 2007.

[3] IBM Integrity Measurement Architecture (IMA). http: //domino.research.ibm.com/comm/research_people.nsf/pages/sailer.ima.html

[4] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter and A. Seshadri, "How Low can you Go Recommendations for Hardware-Supported Minimal TCB Code Execution," *Proceedings of ASPLOS*'08, Seattle, Vol. 43, No. 3, 2008, pp. 14-25.

[5] G. Edward Suh, D. Clarke, B. Gassend, M. Dijk and S. Devadas, "AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing," *Proceedings of ICS*'03, San Francisco, 2003, pp. 160-171.

[6] Y. X. Shi and G. H. Lee, "Augmenting Branch Predictor to Secure Program Execution," *Proceedings of DSN* 07.

[7] http://www.hpl.hp.com/research/cacti/

[8] T. Austin and D. Burger, "The SimpleScalar Tool Set," University of Wisconsin CS Department, Technical Report No. 1342, June 1997.

[9] T. Sherwood, E. Perelman, G. Hamerly and B. Calder, "Automatically Characterizing Large Scale Program Behavior," *Proceedings of the* 10*th ASPLOS*, California, Vol. 37, No. 10, 2002, pp. 45-57.

[10] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, *et al.*, "Archi-

tectural Support for Copy and Tamper Resistant Software," SIGPLAN Notice, Vol. 35, No. 11, 2000, pp. 178-179.

[11] P. Loscocco, P. Wilson, A. Pendergrass and C. McDonell, "Linux Kernel Integrity Measurement Using Contextual Inspection," *STC*'07: *Proceedings of the* 2007 *ACM Workshop on Scalable Trusted Computing*, Virginia, 2007.

[12] L. Chen, R. Landfermann, H. Lohr and C. Stuble, "A Protocol for Property-Based Attestation," *Proceedings of STC*'06, the ACM Press, Virginia, 2006, pp. 7-16.

[13] A. Sadeghi and C. Stuble, "Property-Based Attestation for Computing Platforms: Caring about Properties, not Mechanisms," *Proceedings of NSPW*'04, New York, 2004, pp. 67-77.

[14] V. Haldar, D. Chandra and M. Franz, "Semantic Remote Attestation: A Virtual Machine Directed Approach to Trusted Computing," *Proceedings of VM*'04, San Jose, 2004, p. 3.

[15] M. Alam, X. W. Zhang, M. Nauman and T. Ali, "Behavioral Attestation for Web Services (BA4WS)," *Proceedings of the* 2008 *ACM Workshop on Secure Web Services*, 2008.

Journal of Information Security, **2010, 1, 11-22**
doi:10.4236/jis.2010.11002 Published Online July 2010 (http://www.SciRP.org/journal/jis)

Fast Forgery Detection with the Intrinsic Resampling Properties

Cheng-Chang Lien, Cheng-Lun Shih, Chih-Hsun Chou
Department of Computer Science and Information Engineering,
Chung Hua University, Hsinchu, Taiwan, China
E-mail: *cclien@chu.edu.tw*

Received June 21, 2010; *revised July* 20, 2010; *accepted July* 25, 2010

Abstract

With the rapid progress of the image processing software, the image forgery can leave no visual clues on the tampered regions and make us unable to authenticate the image. In general, the image forgery technologies often utilizes the scaling, rotation or skewing operations to tamper some regions in the image, in which the resampling and interpolation processes are often demanded. By observing the detectable periodic distribution properties generated from the resampling and interpolation processes, we propose a novel method based on the intrinsic properties of resampling scheme to detect the tampered regions. The proposed method applies the pre-calculated resampling weighting table to detect the periodic properties of prediction error distribution. The experimental results show that the proposed method outperforms the conventional methods in terms of efficiency and accuracy.

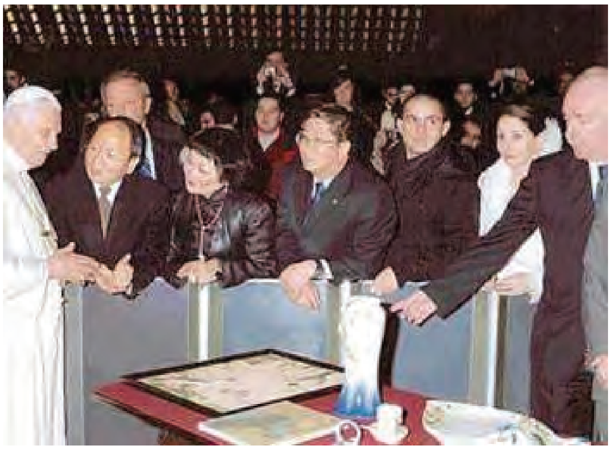
Keywords: Image Forgery, Resampling, Forgery Detection, Intrinsic Properties of Resampling

1. Introduction

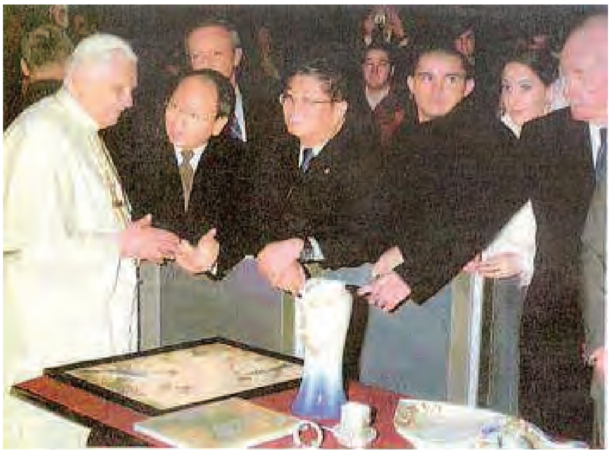
In recent years, with the rapid progress of image processing software, it becomes a great challenge to verify whether the digital image is tampered or not because the image processing software can create a sophisticated digital forgery and leave no visual clues on the tampered regions. For example, the *Liberty Times* newspaper in January 2008 (newspaper in Taiwan) published a photograph shown in **Figure 1(b)** in which the picture “Miss Wang” had been removed intentionally.

In general, the digital forgery detection methods can be roughly categorized into the active [1-4] and passive methods [5-16]. In the active methods [1-4], the digital watermarking or signatures are hid in the image for the purpose of authentication [1-4]. In addition, the embedded watermarks need to be robust enough to resist the various kinds of image attacks. On the contrary, the passive approaches [5-17] do not need any prior information for the forgery detection and can be further categorized into the methods of detecting copy-pasted regions, defocus blur edges, resampling, sensor noise pattern, different lighting conditions and block artifact inconsistency.

In [5], the author provided a method to identify the digital forgery regions that are copied and pasted from the same image by applying the method of block matching. However, the matching process can fail if the tampered region is cropped from different images. Zhou *et al.* [6] proposed a method to identify the digital forgeries by using the edge preserving smoothing filter in which the manual blur edge is discriminated from the defocus blur edge and the erosion operation is applied for detecting the manual blur edge. Another typical method developed by Popescu [7] detected the digital forgeries by tracing the characteristic of the resampled signals. The major concept of this method is to apply the EM (expectation/maximization) algorithm to acquire the resampling coefficients and then calculate the resampling probability map. Based on the spectral analysis of the probability map, the magnitude peak can be used to identify the forgery patterns. Moreover, Popescu [8] utilized the specific interpolation coefficients of color filter array for each brand of digital camera to identify the digital forgery. Kirchner [9] proposed a more efficient method by directly applying the converged resampling coefficients to detect the tempered regions. As same as tracing the periodic characteristic of the resampled signals, Prasad [10] and Mahdian [11,12] proposed their method to extract the periodical property of the resampled signals based on analyzing the periodic characteristic of the covariance of the second order derivatives. In [13,14], Lukáš *et al.*



(a)



(b)

Figure 1. (a) The original image; (b) The tampered image.

proposed a method that utilize the imaging sensor noise as a unique stochastic characteristic to detect the forgeries. Johnson *et al.* [15] discovered that the light condition of the tampered area will be inconsistent to the original image. For the compressed image, Ye *et al.* [16] proposed a method based on the different block artifacts caused by different quantization tables.

Generally, each kind of digital forgery detection method can solve only one kind of forgery pattern. In this study, we only address on the detection of resampling forgery. Two related researches addressed on the detection of resampling forgery are the methods proposed by Popescu [7] and Mahdian [11]. However, there exist two major drawbacks in the above-mentioned algorithms. For the Popescu's method [7], high computation cost in the iterative computing procedure is required. It takes almost 5 minutes to generate the probability map for the image with resolution 512 × 512 pixels. For the method proposed by Mahdian [11], we found that the derivative kernel used in [11] will destroy the periodicity of the correlation function at the high texture regions. Hence, in this study, we try to investigate and analyze the intrinsic properties of resampling scheme and develop a new more efficient algorithm based on the intrinsic properties of resampling.

Based on the periodical property that the original values can be selected from the resampling process, some of the reconstructed values would exactly overlap the original values in resampled signal and then the error between the predicted value and the resampled value would be very small. By analyzing the prediction error distribution generated by the weighting tables from different resampling rates, we can detect the digital forgeries. To enhance the periodical property, the projection operation is used for creating one-dimensional prominent periodical patterns. In addition, both of the vertical and horizontal predicting error variations are considered simultaneously.

The rest of this paper is organized as follows. In Section 2, two typical forgery detection methods are described. In Section 3, a new forgery detection method based on the intrinsic properties of resampling is proposed, which can detect the tampered regions more efficiently. In Section 4, we present the efficiency and accuracy analyses among the proposed method and other approaches. Finally, we summarize the contributions and future works in Section 5.

2. Related Works

In this section, two typical forgery detection methods for the resampling forgery techniques are introduced. These methods detect the forgery by tracing the interpolation clues of resampled signal

2.1. The Popescu's Method

A well known forgery detection method proposed by Popescu [7] assume that the interpolated samples are the linear combination of their neighboring pixels and try to train a set of resampling coefficients to estimate the probability map. In this method, a digital sample can be categorized into two models: M_1 and M_2. M_1 denotes the model that the sample is correlated to their neighbors; while M_2 denotes that the sample isn't correlated to its neighbors. The resampling coefficients can be acquired by the EM algorithm. In the E-step, the probability for M_1 model for every sample is calculated. In the M-step, the specific correlation coefficients are estimated and updated continuously. The detailed description of the forgery detection algorithm is described in the sequel.

2.1.1. E-Step

The conditional probability for sample $y\,[i]$ belonging to M_1 model is calculated by the following formula.

$$\Pr\left\{y[i]\middle|y[i]\in M_1\right\} = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{-\left(y[i]-\sum_{k=-N}^{N}\alpha_k y[i+k]\right)^2}{2\sigma^2}\right] \quad (1)$$

2.1.2. M-Step

Minimize the quadratic error function defined in Equation (2) by updating the correlation coefficients α iteratively.

$$E(\vec{\alpha}) = \sum_i \omega(i)\left(y[i]-\sum_{k=-N}^{N}\alpha_k y[i+k]\right)^2 \quad (2)$$

where $\omega(i) \equiv \Pr\left\{y[i]\in M_1\middle|y[i]\right\}$.

After applying the Popescu's method to the image, we can obtain a probability map. The peak ratio of frequency response of the probability map can be used to identify the digital forgery. **Figure 2** illustrates that the peaks of frequency response exist in the tampered image. On the contrary, no peaks exist in the original image shown in **Figure 2(a)**.

2.2. The Mahdian's Method

Another method proposed by Mahdian and Saic [11] demonstrates that the interpolation operation can exhibit periodicity in their derivative distributions. To emphasize the periodical property, they employ the radon transformation to project the derivatives along a certain orientation. The radon transformation is defined as:

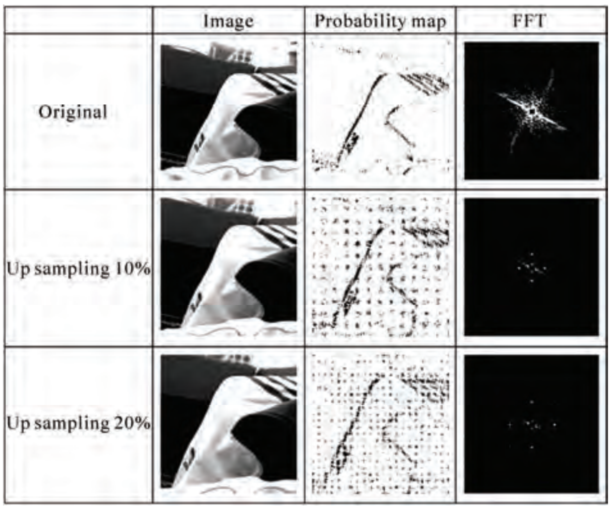


Figure 2. The frequency response of the probability maps generated from Popescu's method for the original image, resampled images with up-sampling rate 10% and 20% respectively.

$$\rho D^2\{b\}(x,y) = \int_L \left|D^2\{b(x,y)\}\right| dl \quad (3)$$

where, $b\ (x, y)$ denotes the pixel in the block with size of $R \times R$ and $D^2\{*\}$ denotes the derivative kernel of order 2. The radon transform along angle θ (0 ~ 179°) is defined in Equation (4).

$$\rho_\theta(x') = \int_{-\infty}^{\infty} \left|D^2\{b(x,y)\}\right| \cdot \left(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta\right) dy' \quad (4)$$

After projecting all the derivatives to one direction and forming 1-*D* projection vectors, the autocovariance function can be used to emphasize the periodicity and defined as:

$$R_{\rho_\theta}(k) = \sum_i \left(\rho_\theta(i+k) - \overline{\rho_\theta}\right)\left(\rho_\theta(i) - \overline{\rho_\theta}\right) \quad (5)$$

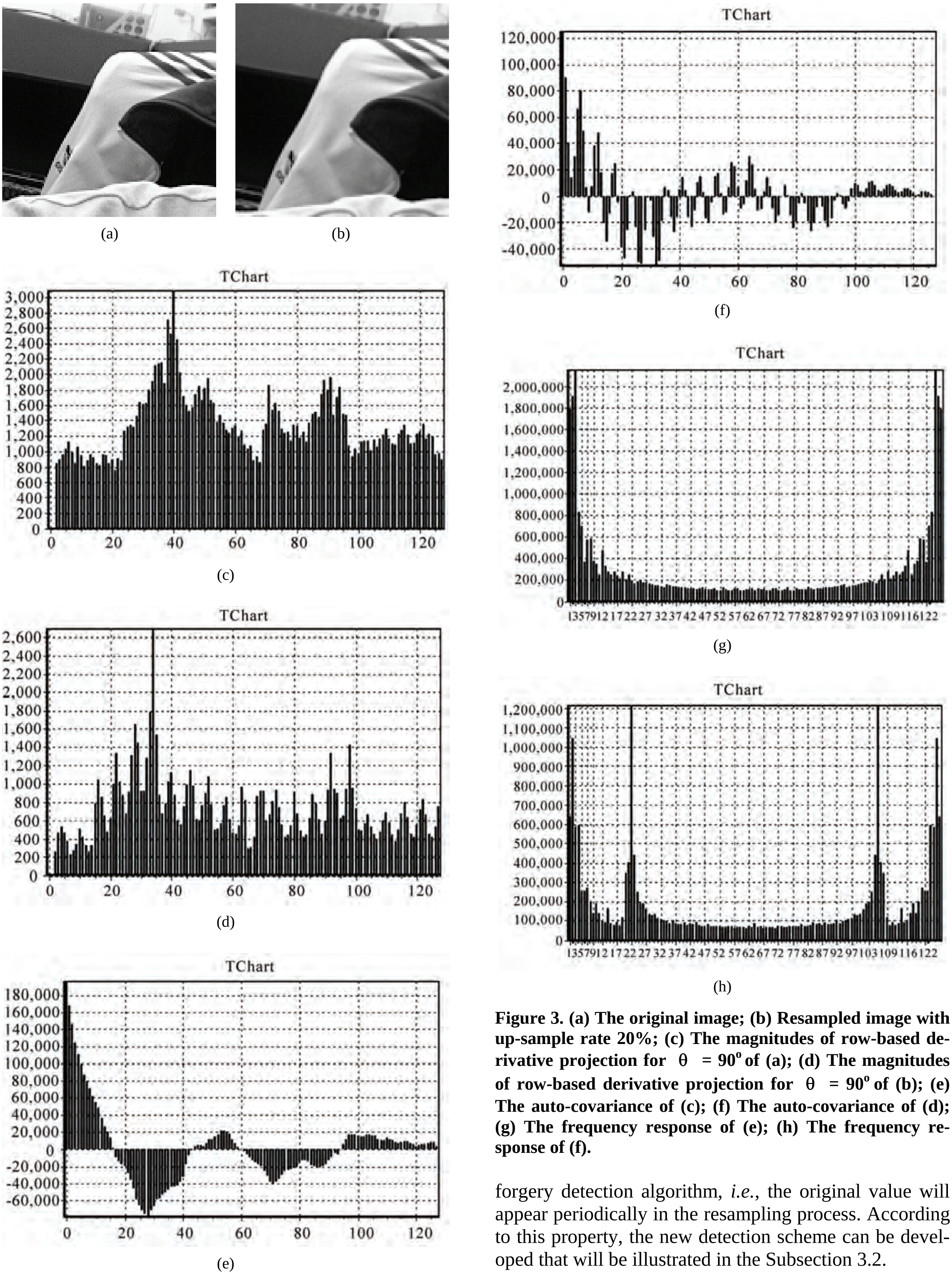
Then, the Fourier transformation of R_{ρ_θ} are also computed to identify the periodic peaks which can indicate the existing of digital forgery. The simulation results are shown in **Figure 3**. It shows that the resampled image can have strong peaks in the frequency response of the derivative covariance.

3. Forgery Detection Using the Resampling Intrinsic Properties

There exist two major drawbacks in the above-mentioned algorithms. For the Popescu's method [7], high computation cost in the iterative computing procedure is required. It takes almost 5 minutes to generate the probability map for an image with resolution 512 × 512 pixels. For the method proposed by Mahdian [11], we found that the derivative kernel used in [11] can reduce the periodicity of the correlation function at the high texture region. Hence, in this study we try to investigate and analyze the intrinsic properties of resampling process and develop a new more efficient algorithm. The system flowchart is shown in **Figure 4** and the detailed function for each block will be described in the following subsections.

3.1. Intrinsic Properties of Resampled Signal

In this section, we firstly introduce the procedures of general resampling process. The up-sampling process is illustrated in **Figure 5(a)** and the original values are denoted as red bars. **Figure 5(b)** shows that interpolation operation fills the empty points with the linear combination of the adjacent signals' values which are denoted as yellow bars. Finally, the samples selected for decimation process which are denoted as blue bars are shown in **Figure 5(c)**. Through the observation of the resampling process, it gives us an important clue to design a new



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3. (a) The original image; (b) Resampled image with up-sample rate 20%; (c) The magnitudes of row-based derivative projection for θ = 90^o of (a); (d) The magnitudes of row-based derivative projection for θ = 90^o of (b); (e) The auto-covariance of (c); (f) The auto-covariance of (d); (g) The frequency response of (e); (h) The frequency response of (f).

forgery detection algorithm, *i.e.*, the original value will appear periodically in the resampling process. According to this property, the new detection scheme can be developed that will be illustrated in the Subsection 3.2.

3.2. Periodicity of the Prediction Error

Every resampled value denoted as blue bar in **Figure 5** can be approximated by the linear combinations of the adjacent original values denoted as red bar with different weights according to their positions, *i.e.*, the weighting in the linear interpolation algorithm is propositional to the distance to their neighbors. Here, we pre-calculate the weighing table (shown in **Table 1**) for each resampling rates. If the resampling rate is known, then the original values can be approximated by the linear combination of the interpolated values. Based on the periodical property of the original values selected from resampling, some of the approximated values would exactly overlap the original values in resampled signal (see the green bar in **Figure 6**). Ideally, the error between the predicted value and the resampled value would be very small at the position where the original value is resampled (the green bar in **Figure 6**). Moreover, the variation of the prediction error will distribute periodically. The weighting table *WT* [*i*], *i* = 1, 2,…, *N*, should be calculated in advance for each resampling rate. The prediction process is described in **Figure 6**.

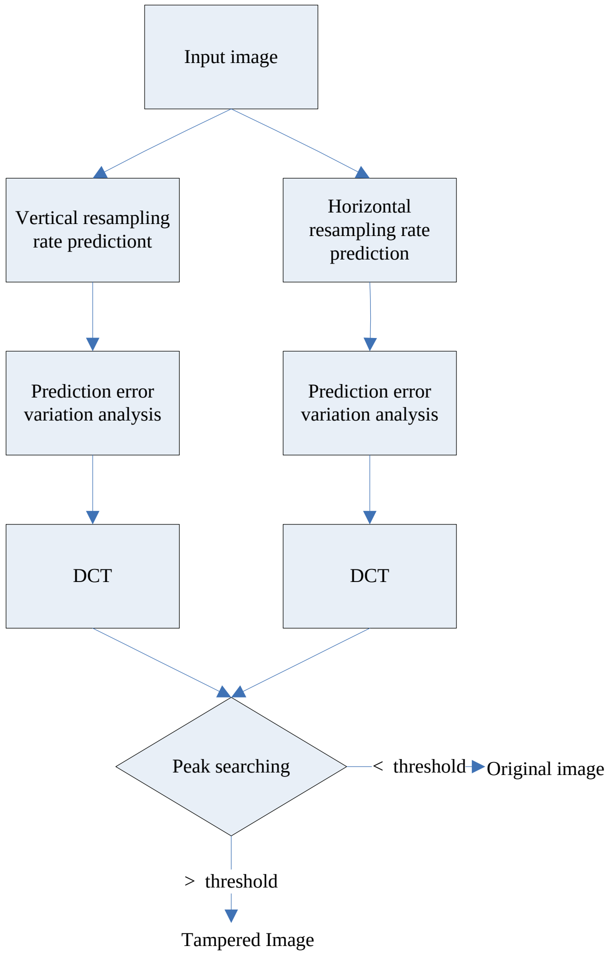

Input image
Vertical resampling rate predictiont
Horizontal resampling rate prediction
Prediction error variation analysis
Prediction error variation analysis
DCT
DCT
Peak searching
< threshold
Original image
> threshold
Tampered Image

Figure 4. Flowchart of the proposed forgery detection system.

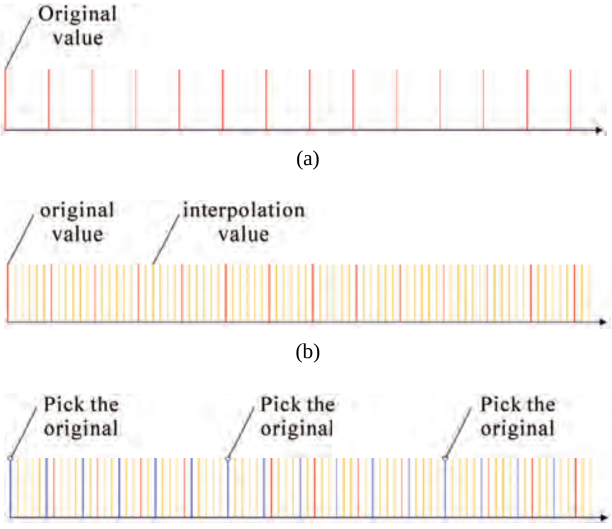


Figure 5. An example for illustrating the intrinsic property of resampled signal. The scaling factor used here is 6/5. (a) The up-sampling for the original values (red bars); (b) Linear interpolation denoted as yellow bars; (c) Down sampling of signal in (b). The resampled signal is denoted as blue bars. The blue bars labeled the white node denote that the original values are chosen.

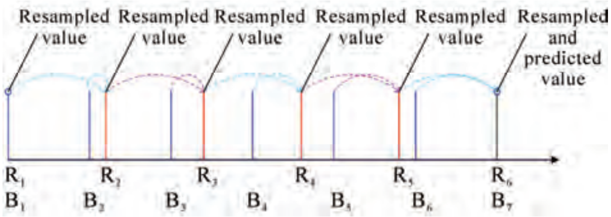


Figure 6. The values (red bar) could be predicted by the resampled values (blue bar). After a certain periodical time interval, the predicted value will overlap the original value denoted as green bar.

Table 1. Weighting table for resampling rate 6/5.

	$WT_L[i]$	$WT_R[i]$
1	1/6	5/6
2	2/6	4/6
3	3/6	3/6
4	4/6	2/6
5	5/6	1/6

In **Figure 6**, the interpolated values can be computed as:

$$B_i = R_{i-1} \times WT_L[i-1] + R_i \times WT_R[i-1] \quad (6)$$

Then, the predicted resampling values can be computed as:

$$
\begin{aligned}
pre_1 &= R_2 = \frac{B_2 - R_1^{*} WT_L[i]}{WT_R[i]} \\
pre_2 &= R_3 = \frac{B_3 - pre_1^{*} WT_L[i]}{WT_R[i]} \\
&\;\;. \\
&\;\;. \\
pre_m &= R_N = \frac{B_N - pre_{m-1}^{*} WT_L[i]}{WT_R[i]} = B_{N+1}
\end{aligned}
\tag{7}
$$

Finally, the prediction error within the certain sliding window can be computed as:

$$\text{Prediction error} = \left| B_{N+1} - pre_m \right| \tag{8}$$

For the case of resampling rate 120%, the difference between pre_5 and B_7 will be very small. When the sliding window for calculating the sample prediction is moving (shown in **Figure 7**), the prediction error will increase and then decrease to the minimum value until the sliding window moves to the next periodical position (B_{14}, B_{21}…). Such a periodical property makes the sequence of prediction error distribute periodically shown in **Figure 8**. In order to enhance this property, the projection operation is also performed for every row and column (two directions are considered separately) before we utilize the frequency analysis to detect the forgery patterns (peaks in frequency response). If the test samples are not resampled or the wrong weighting table is selected, the distribution of prediction error would be irregular.

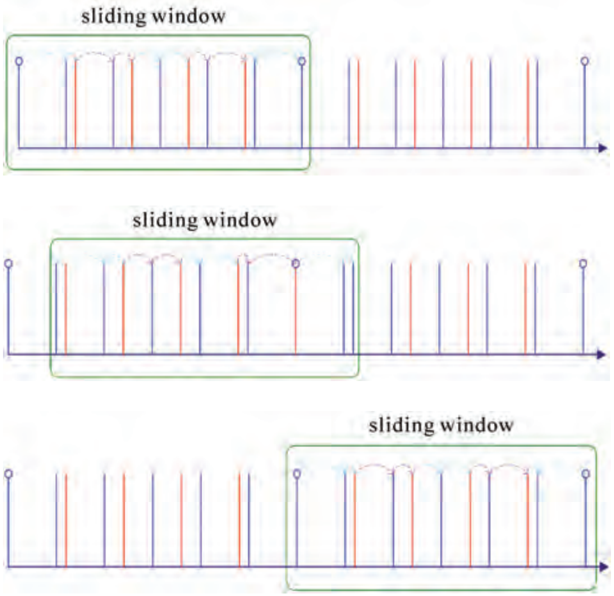
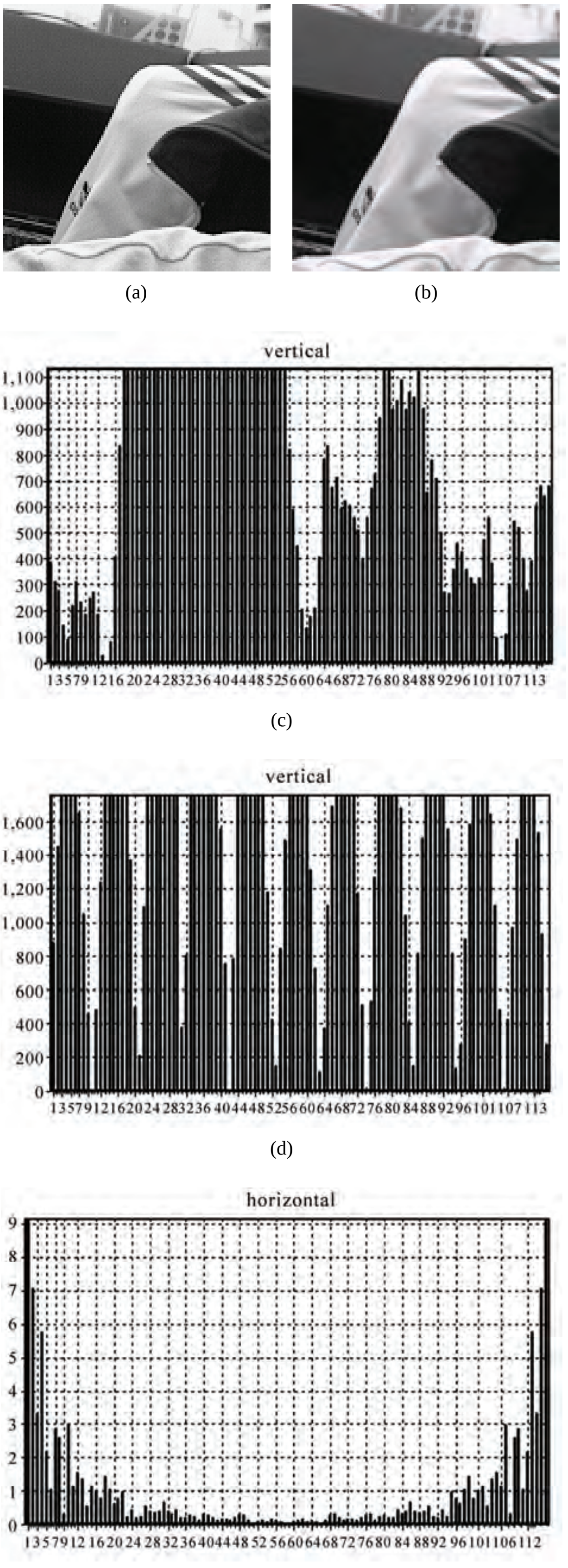


Figure 7. The sliding window for calculating the sample prediction using the pre-calculated weighting table.



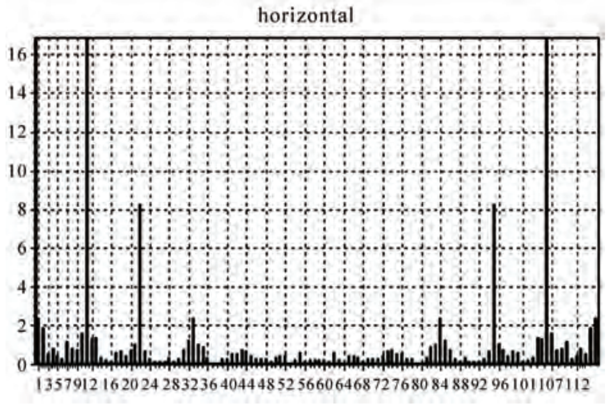
(a)

(b)

(c)

(d)

(e)



(f)

Figure 8. (a) The original image; (b) Resampled image with up-sampled rate 10%; (c) The magnitudes of row-based prediction error variation projection of (a); (d) The magnitudes of row-based prediction error variation projection of (b); (e) The frequency response of (c); (f) The frequency response (d).

To develop an automatic forgery detection method, there are two main criteria should be considered. The first one is the position where the peak occurs and the second one is the peak ratio. According to the different weighting tables (different resampling rate) for the forgery detection and the specific periodical property for each resampling rate, the expected position where the peak occurs could be forecasted. Then, we can match the peak position to the forecasted position where the specific resampling rate generates for identifying the existence of digital forgery. If the ratio is larger than a specified threshold, we can identify that existence of digital forgery. Finally, the flowchart of the proposed system is shown in **Figure 9**. To detect the tampered region, the image is scanned from left-top to right-bottom with different block sizes. In each block, the proposed method is applied to detect the tampered regions.

4. Experimental Results

In this section, the efficiency and accuracy for Popescu'd method [7], Mahdian's method [11], and the proposed method are analyzed. The experimental database is constructed with 160 gray level images with resolution 512 × 512 and each image is partial tampered in BMP format. The image tampering is based on the resampling process with the different bi-linear sampling rates: 105%, 110%, 120% and 125%. The forgery detections are performed by scanning the image with the block size of 128 × 128 pixels.

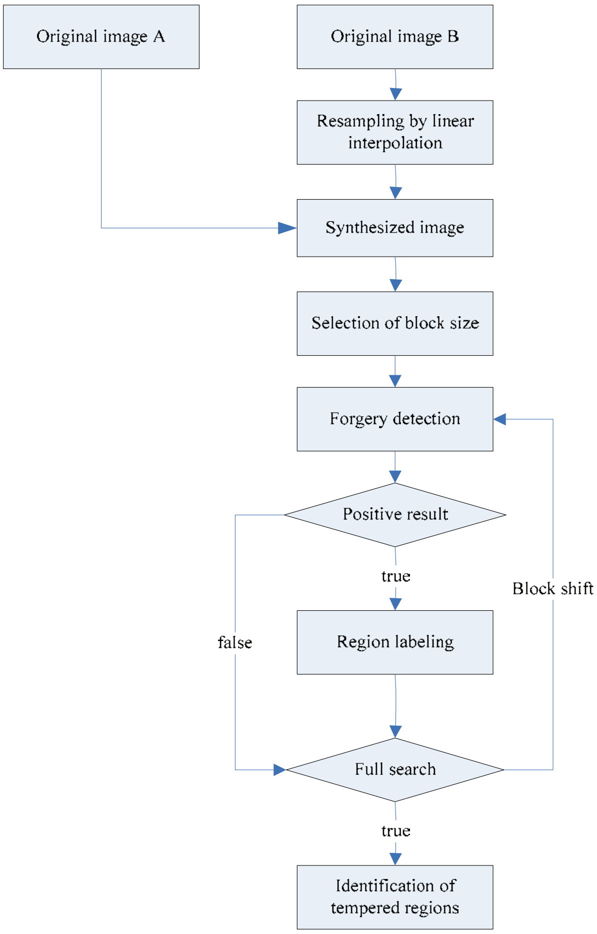


Figure 9. The flowchart of the proposed method.

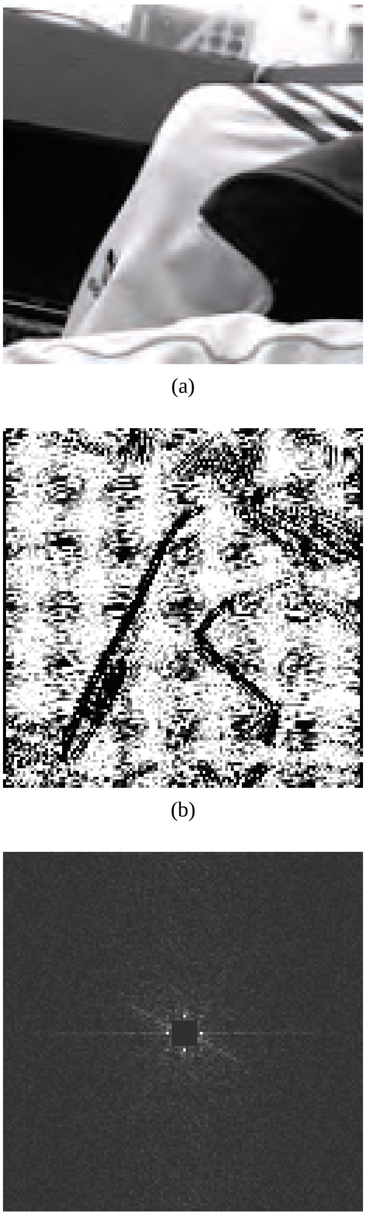
Before analyzing the accuracy of forgery detection, we firstly describe the detection rules for the Popescu's [7], Mahdian's [11], and our methods. Here, the forgery detection of Popescu's and Mahdian's methods is determined by evaluating whether the ratio of peak-to-average frequency response is larger than a predefined threshold value or not. The ratio of peak-to-average frequency response is defined as:

$$R_{Popsecu} = R_{Mahdian} = \frac{magnitude_{\text{maximum}}}{magnitude_{\text{average}}}$$

For our method, the forgery detection is determined by evaluating whether the ratio of forecasted peak-to-average frequency response is larger than a predefined threshold value or not. The ratio of forecasted peak-to-average frequency response is defined as:

$$R_{our} = \frac{magnitude_{\text{forecasted position}}}{magnitude_{\text{average}}}$$

The resampled image with rate 120% shown in **Figure 10(a)** is used as the tampered image for analyzing the detection accuracy for the three methods. **Figure 10(b)** shows the probability map produced by the Popescu's method and **Figure 10(c)** shows the frequency response of the probability map. **Figure 11(a)** shows the radon transformation of the derivative along horizontal direction generated by Mahdian's method and **Figure 11(b)** shows its auto-covariance. **Figure 11(c)** shows the frequency response of the auto-covariance values. Based on the proposed method, the prediction error generated by the novel algorithm is shown in **Figure 12(a)**. **Figure 12(b)** presents the frequency response of the prediction error. An obvious drawback of the Mahdian's method is that the weak periodical patterns occur at the high texture regions shown in **Figure 11(c)**. The accuracy analyses of forgery detections for different resampling rates are analyzed in **Table 2**.

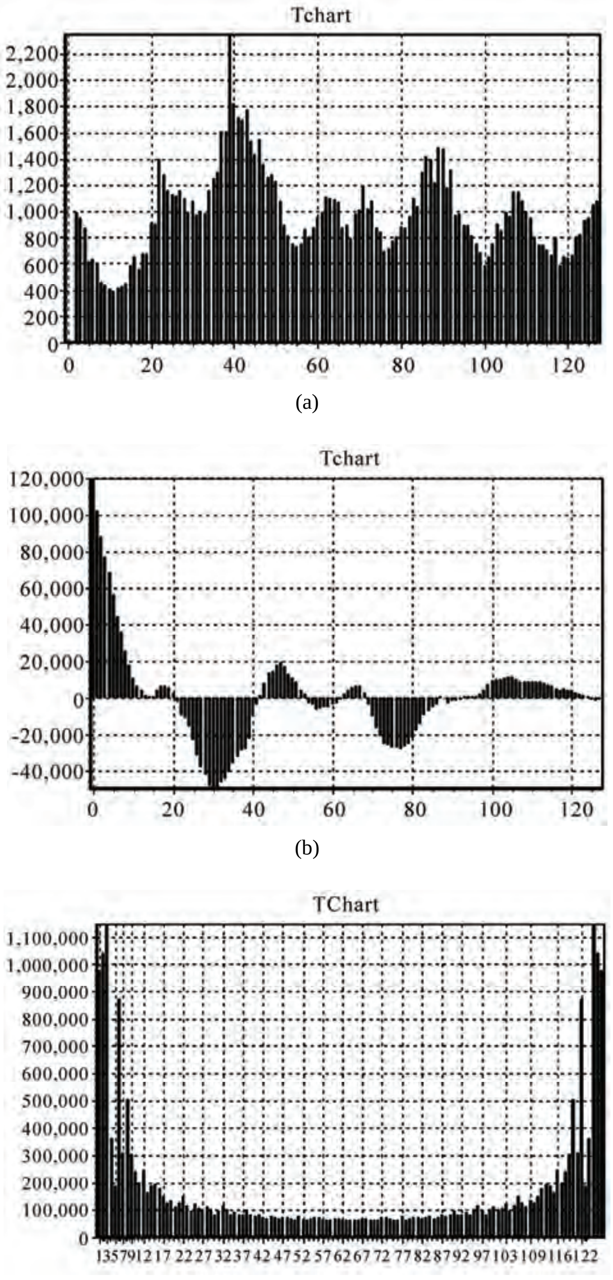


(a)

(b)

(c)

Figure 10. (a) The tampered image; (b) The probability map generated by the Popescu's method; (c) The frequency response of (b).



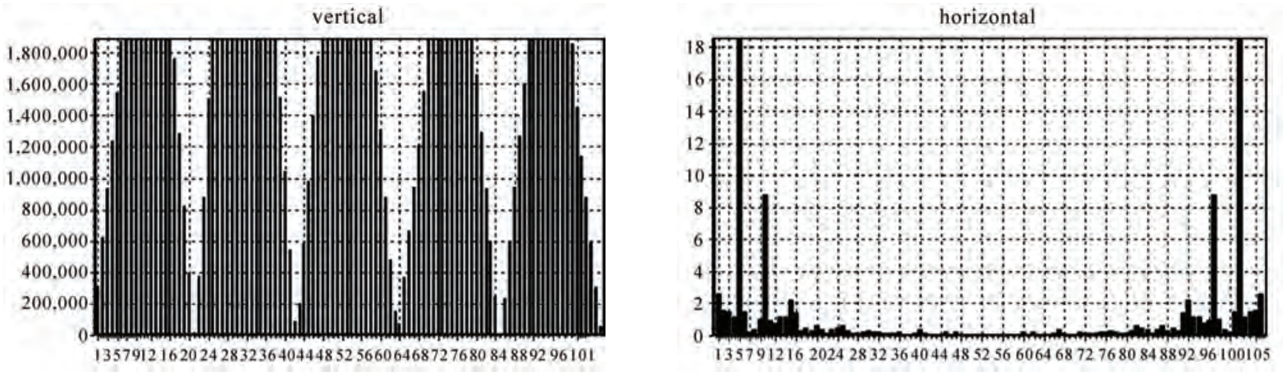
(a)

(b)

(c)

Figure 11. (a) The radon transformation output of Figure 13 by the Mahdian's method; (b) The autocovariance of (a); (c) The frequency response of (b).

The ROC curves with different up-sampling rates for Popescu's, Mahdian's and our methods are shown in **Figure 13**. In this Figure, the detection accuracy of Popescu's method is the highest one and the detection accuracy of our method is close to the Popescu's curve. However, our method is the fastest one that will be mentioned later. The detection accuracy of Mahdian's method is the lowest because the detection accuracy is affected by the high texture regions.

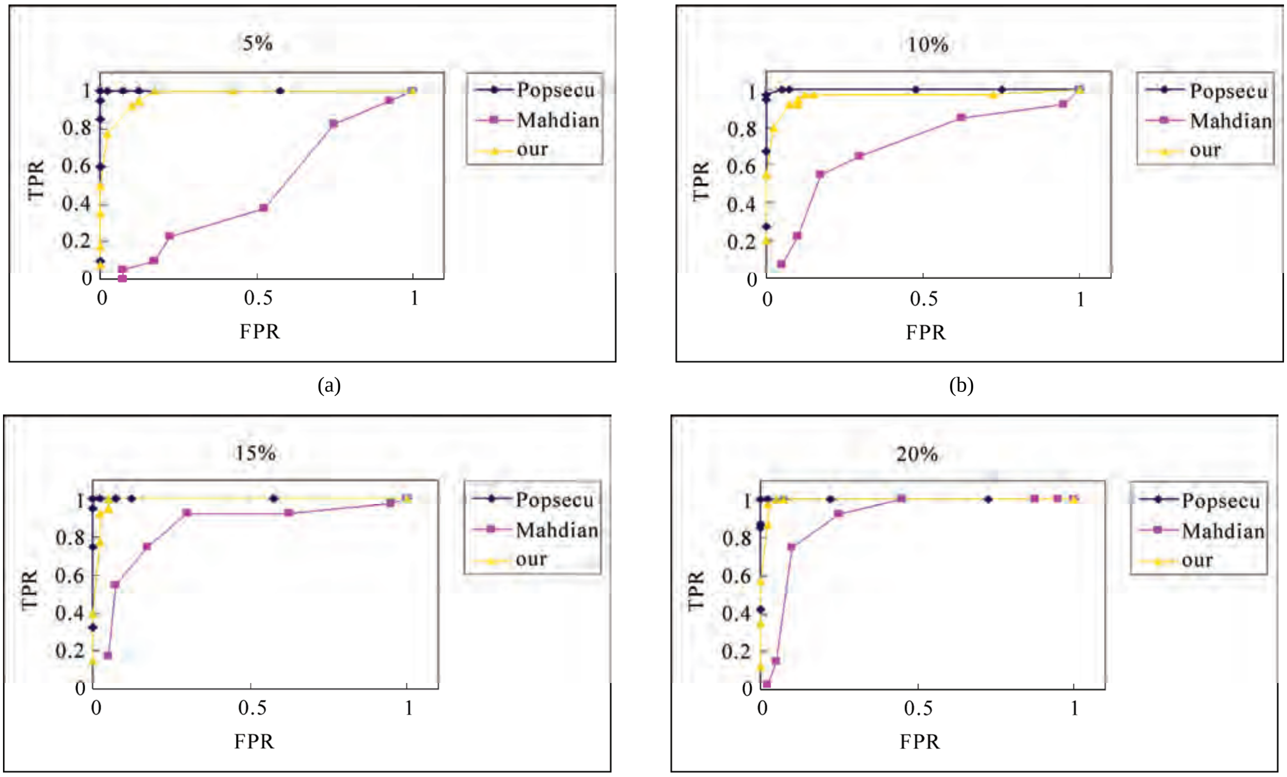


(a) (b)

Figure 12. (a) The prediction error of the tampered image shown in Figure 10, which is generated by the proposed method; (b) The frequency response of (a).

Table 2. The accuracy analysis for the methods of our, Popescu's and Mahdian's with 40 resampled images for different rates.

	Popescu's method				Our method				Mahdian's method			
Up-sampling	5%	10%	20%	25%	5%	10%	20%	25%	5%	10%	20%	25%
Positive	40	40	40	40	40	40	40	40	40	40	40	40
Negative	40	40	40	40	40	40	40	40	40	40	40	40
True positive	40	39	40	40	38	39	40	40	21	22	37	37
True negative	40	40	40	40	35	37	38	38	25	33	28	30
Accuracy	100%	98.7%	100%	100%	91.2%	95%	97.5%	97.5%	57.5%	68.7%	81.2%	83.7%



(a) (b)

(c) (d)

Figure 13. The ROC curves of (a) Up-sampling 5%; (b) Up-sampling 10%; (c) Up-sampling 20%; (d) Up-sampling 25%.

In addition, we compare the efficiency among Popescu's [7], Mahdian's [11] and our methods with the PC of 1.8 GHz. The efficiency analysis is shown in **Figure 14**. Here, we perform the efficiency analysis from block size 64 × 64 to 512 × 512 and assume there are 21 weighting tables for 21 resampling rates used in [7]. Because the iterative EM algorithm is very time-consuming, the efficiency of Popescu's method is the lowest. On the contrary, the highest efficiency is presented in Mahdian's method because the operations in his method are very simple. It's worthy to conclude that detection accuracy and efficiency of our method can approach both of the benefits of Popescu's and Mahdian's methods.

Figures 15-16 show the detection results of the pro-

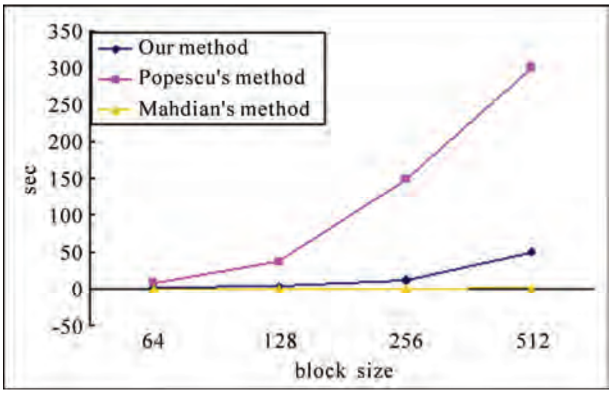
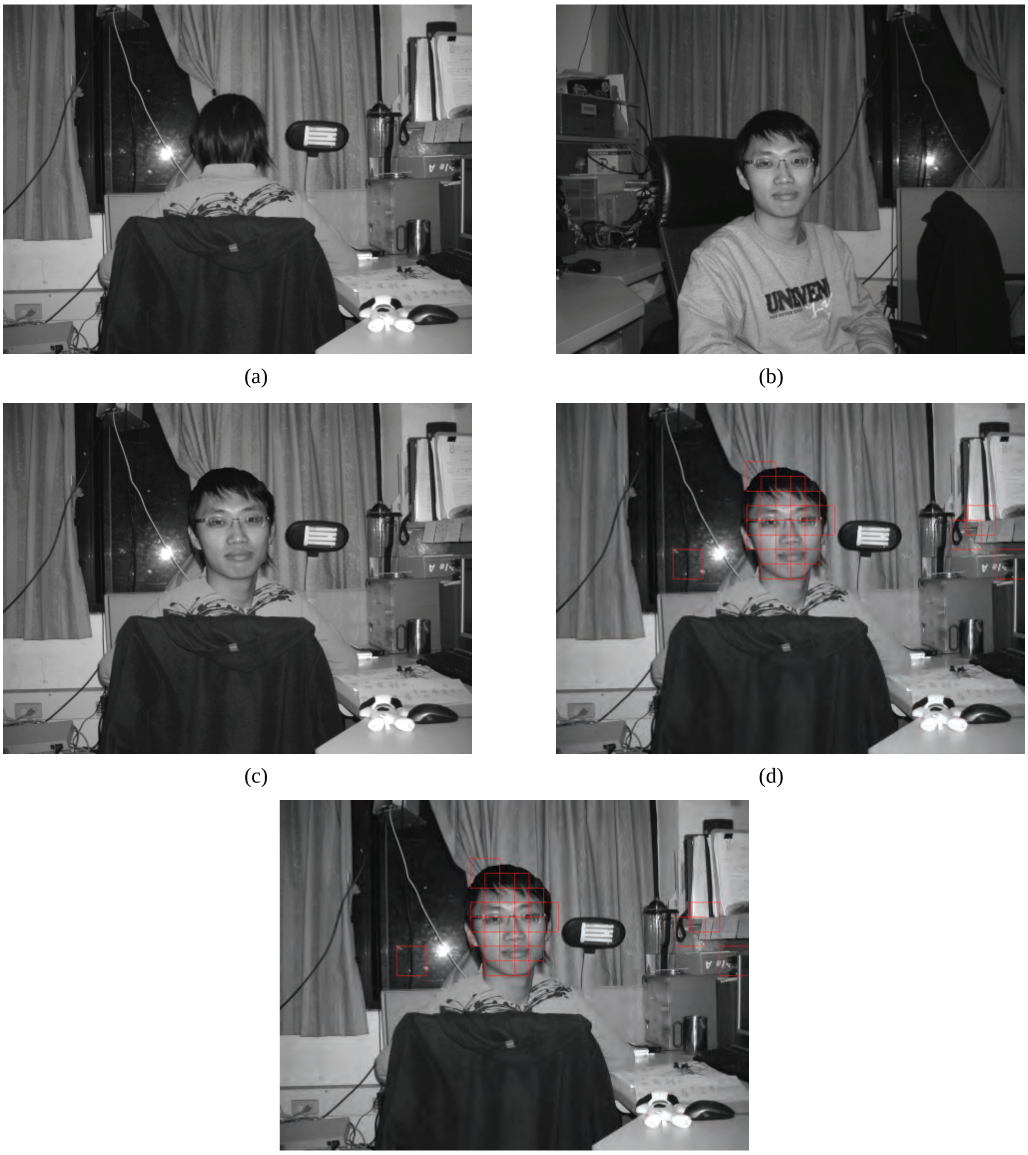


Figure 14. Efficiency analysis.



(a)

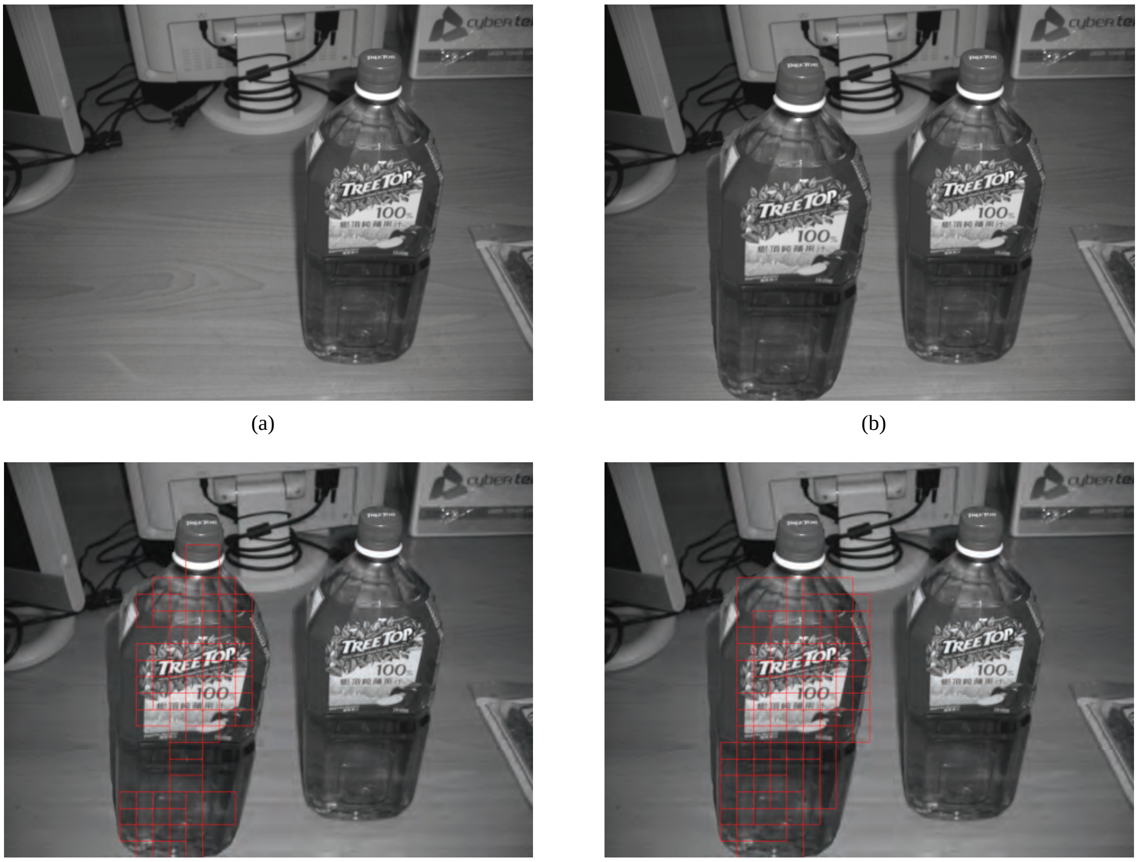
(b)

(c)

(d)

(e)

Figure 15. (a) Original image; (b) Image with up-sample rate 5%; (c) Forgery image composed from (a), (b); (d) Detection result with 64 × 64 block size; (e) Detection result with 128 × 128 block size.


(a)

(b)

(c)

(d)

Figure 16. (a) Original image; (b) Forgery image composed from up-sample (a) 10% and put the bottle near beside; (c) Detection result with 64 × 64 block size; (d) Detection result with 128 × 128 block size.

-posed method for different resampling rates with two block sizes. In **Figure 15**, the man's head in **Figure 15(b)** is cropped and replaced the head region in **Figure 15(a)** to synthesize the forgery image shown in **Figure 15(c)**. **Figure 15(d)** and **Figure 15(e)** show the detection result with 64 × 64 and 128 × 128 block sizes. **Figure 16(a)** shows an original bottle image and **Figure 16(b)** shows that a resized bottle is put on the left side of the tampered image. **Figures 16(c)** and **16(d)** show the detection results with different block sizes. Here, we observe that the detection accuracy for the smaller block size is lower than the accuracy with larger block size because more periodical patterns can be collected in larger blocks.

5. Conclusions

In this paper, we propose a novel method based on the intrinsic properties of resampling scheme to detect the forgery regions with the pre-calculated resampling weighting tables and the detecting of periodic patterns for the vertical and horizontal prediction error. In Popescu's method, high accuracy can be obtained with high computation cost. On the contrary, in Mahdian's method, the detecting accuracy can be affected on the high texture regions. The detection accuracy and efficiency of our method can approach both of the benefits of Popescu's and Mahdian's methods. The detection accuracy of our method is about 95% and the time for detecting a 512 × 512 image needs only 50 seconds.

6. References

[1] R. B. Wolfgang and E. J. Delp, "A Watermark for Digital Image," *Proceedings of the International Conference on Image Processing*, Vol. 3, 1996, pp. 219-222.

[2] R. B. Wolfgang, C. I. Podilchuk and E. J. Delp, "Perceptual Watermarks for Digital Images and Video," *Proceedings of the IEEE, Special Issue on Identification and Protection of Multimedia Information*, Vol. 87, No. 7, 1999, pp. 1108-1126.

[3] M. Wu and B. Liu, "Watermarking for Image Authentication," *IEEE International Conference on Image Processing*, Vol. 2, 1998, pp. 437-441.

[4] M. Yeung and F. Mintzer, "An Invisible Watermarking Technique for Image Verification," *Proceedings of the International Conference on Image Processing*, Vol. 1,

1997, pp. 680-683.

[5] J. Fridrich, D. Soukal and J. Lukáš, “Detection of Copy-Move Forgery in Digital Images,” *Proceedings of the Digital Forensic Research Workshop*, Cleveland, 2003.

[6] L. Zhou, D. Wang, Y. Guo and J. Zhang, “Blue Detection of Digital Forgery Using Mathematical Morphology,” *Technical Report*, *KES AMSTA*, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 990-998.

[7] A. C. Popescu and H. Farid, “Exposing Digital Forgeries by Detecting Traces of Resampling,” *IEEE Transactions on Signal Processing*, Vol. 53, No. 2, 2005, pp. 758-767.

[8] A. C. Popescu and H. Farid, “Exposing Digital Forgeries in Color Filter Array Interpolated Images,” *IEEE Transactions on Signal Processing*, Vol. 53, No. 10, 2005, pp. 3948-3959.

[9] M. Kirchner, “Fast and Reliable Resampling Detection by Spectral Analysis of Fixed Linear Predictor Residue,” *MM* & *Sec*’08, *Proceedings of the Multimedia and Security Workshop*, 2008, pp. 11-20.

[10] S. Prasad and K. Ramakrishnan, “On Resampling Detection and its Application to Detect Image Tampering,” *Proceedings of the* 2006 *IEEE International Conference on Multimedia and EXPO*, 2006, pp. 1325-1328.

[11] B. Mahdian and S. Saic, “Blind Authentication Using Periodic Properties of Interpolation,” *IEEE Transactions on Information Forensics and Security*, Vol. 3, No. 3, 2008, pp. 529-538.

[12] B. Mahdian and S. Saic, “Detection of Resampling Supplemented with Noise Inconsistencies Analysis for Image Forensics,” *International Conference on Computational Sciences and its Applications*, Vol. 81, No. 4, 2008, pp. 546-556.

[13] J. Lukáš, J. Fridrich and M. Goljan, “Detecting Digital Image Forgeries Using Sensor Pattern Noise,” *Proceedings of the SPIE Conference on Security*, *Steganography and Watermarking of Multimedia Contents*, Vol. 6072, 2006, pp. 362-372.

[14] J. Lukáš, J. Fridrich and M. Goljan, “Digital Camera Identification from Sensor Pattern Noise,” *IEEE Transactions on Information Security and Forensics*, Vol. 1, No. 2, 2006, pp. 205-214.

[15] M. K. Johnson and H. Farid, “Exposing Digital Forgeries in Complex Lighting Environments,” *IEEE Transactions on Information Forensics and Security*, Vol. 2, No. 4, 2007, pp. 450-461.

[16] S. Ye, Q. Sun and E. Chang, “Detecting Digital Image Forgeries by Measuring Inconsistencies of Blocking Artifact,” *IEEE International Conference on Multimedia and EXPO*, 2007, pp. 12-15.

[17] M. C. Stamm and K. J. R. Liu, “Forensic Detection of Image Tampering Using Intrinsic Statistical Fingerprints in Histograms,” *Proceedings of the APSIPA Annual Summit and Conference*, Sapporo, 2009.

Journal of Information Security, **2010, 1, 23-28**
doi:10.4236/jis.2010.11003 Published Online July 2010 (http://www.SciRP.org/journal/jis)

Eliminating Forgers Based on Intra Trial Variability in Online Signature Verification Using Handglove and Photometric Signals

Andrews Samraj[1], Shohel Sayeed[1], Loo Chu Kiong[1], Nikos E. Mastorokis[2]
[1]*Faculty of Information Science and Technology, Multimedia University, Malacca, Malaysia*
[2]*Faculty of Engineering, Industrial Engineering Department, Technical University of Sofia, Sofia, Bulgaria*
E-mail: *andrews.samraj@mmu.edu.my*

Received March 5, 2010; *revised June* 9, 2010; *accepted July* 15, 2010

Abstract

The novel reinforcement to the data glove based dynamic signature verification system, using the Photometric measurement values collected simultaneously from photo plethysmography (PPG) during the signing process is the emerging technology. Skilled forgers try to attempt the genuine signatures in many numbers of trials. The wide gap in the Euclidian distances between forgers and the genuine template features prohibits them from successful forging. This has been proved by our repeated experiments on various subjects using the above combinational features. In addition the intra trial features captured during the forge attempts also differs widely in the case of forgers and are not consistent that of a genuine signature. This is caused by the pulse characteristics and degree of bilateral hand dimensional similarity, and the degrees of pulse delay. Since this economical and simple optical-based technology is offering an improved biometric security, it is essential to look for other reinforcements such the variability factor considerations which we proved of worth considering.

Keywords: Photo Plethysmography, Data Glove, Signature Verification, Multimodal Biometrics

1. Introduction

Enhancements to the signature verification systems has been suggested by many researchers [1-3] and bio signal based security features are also considered as a unique alternatives for applications that require some document evidence like signing cheques and security documents. The signal based biometrics in the only applicable means for people with physical disability [4].

Making use of multimodal biometric technology which provides unique and robust identification features for every individual is in great demand in security environments that require high quality authentication methods. Using PPG wave forms to distinguish individuals using their biometric component was suggested by researchers in 2007 [5] and is employed in protected applications like e-transactions and access control mechanisms.

As a fortification to the current signal based dynamic signature verification system, we have used a new method by using the combination of the plethysmographic component along with the data glove signals to make the authentication process more robust and distinctive.

The possibility of skilled forging is reduced by the PPG feature that brings in the hand and heart dimensions of an individual into the signature feature vector. In order to further reinforce the effectiveness of the system here in this research work we have considered the intra trial variation approach to further validate the signature process. This method assures the elimination of skilled forging by a multi level authentication.

2. The Equipment

The plethysmographic system, a simple equipment that functions on the intensity of light reflected from the skin's surface. The red cells count below the skin is considered to determine the volume of blood in the particular area. The recorded signal posses the measurement of changes in venous blood volume and the arterial blood pulsation in the arterioles, hence representing the heart rate. There are two values supplied by the system and are the measurements of transmission and reflectance. A sample signal produced by the PPG is shown in **Figure 1**.

Similarly, the data glove is used for dynamic signature verification and that is easy to use, free from image and material of signature medium as well as no scanning processes is required. It involves only a direct acquisition of signals from the subjects while they write down their signatures, preprocess it, extract the feature, match it to classify and decision making. The data glove offers the users comfort, ease of application, and it comes with a small form factor with multiple application drivers, high data quality, low cross-correlation and high frequency data lodging. It measures finger flexure (2 sensors per finger) as well as the abduction between fingers. The system interfaces with the computer via a cable to the USB port (Platform Independent). It features an auto calibration function, 8-bit flexure and abduction resolution, extreme comfort, low drift and an open architecture. It can also be operated wirelessly to interface with the computer via Bluetooth technology up to 20 m distance.

Similarly, the data glove is used for dynamic signature verification and that is easy to use, free from image and material of signature medium as well as no scanning processes is required. It involves only a direct acquisition of signals from the subjects while they write down their signatures, preprocess it, extract the feature, match it to classify and decision making. The data glove offers the users comfort, ease of application, and it comes with a small form factor with multiple application drivers, high data quality, low cross-correlation and high frequency data lodging. It measures finger flexure (2 sensors per finger) as well as the abduction between fingers. The system interfaces with the computer via a cable to the USB port (Platform Independent). It features an auto calibration function, 8-bit flexure and abduction resolution, extreme comfort, low drift and an open architecture. It can also be operated wirelessly to interface with the computer via Bluetooth technology up to 20 m distance.

One glove fits many hands since it is made-up of stretchable material "A".

The data Glove and the signature verification process using the glove is shown in **Figure 2**. The output of the probe is fed into the serial port of a pulse oximetry module (from Dolphin Medical, Inc.) Measurements were taken for 50 signatures from 14 sensors of the data glove, and four led's of plethysmogram fixed on the subject as seen in **Figure 3**.

3. The Fusion of Photo Plethysmography System with Data Glove Signals

3.1. Subjects and Signal Acquisition Methods

In this study, the data glove signals and photo plethysmographic signals were recorded from 6 volunteered

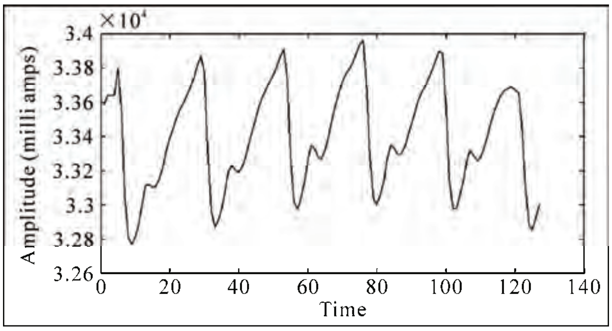


Figure 1. The PPG signal pattern during one signature.

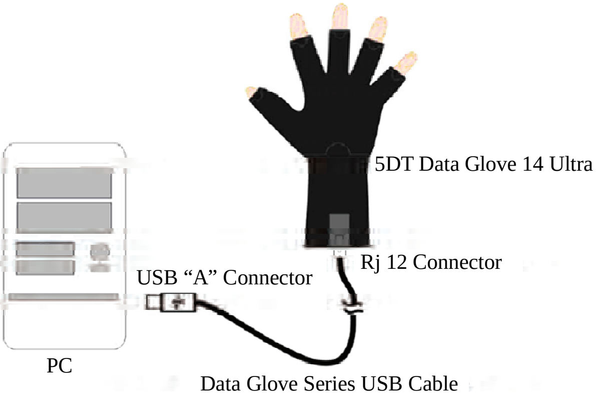


Figure 2. The data glove.

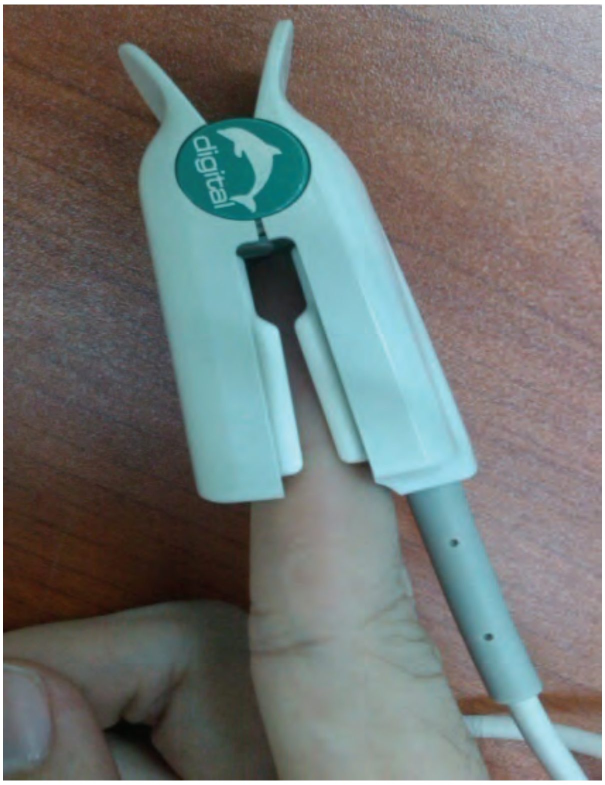


Figure 3. The plethysmogram.

subjects. Two subjects were considered as original signers and other four were the skilled forgers. The data glove signals and the peripheral volume pulses (PPG) were sampled at 61 Hz. Both the signals were recorded from the subjects simultaneously while they were signing.

The subjects were selected among our co-researchers and the average age of the subjects is 34.

The dynamic features of the data glove signal comprises of

1) Distinctive patterns to an individual's signature,
2) The hand dimensions,
3) Time taken to complete a signature process,
4) Hand trajectory dependent rolling.

These factors contributes to the feature of the signal captured from the data glove and make it more suitable to trust for use in signature verification since it provides data on the dynamics of pen movement and the individual's hand dimensions. Along with these four components that represent a person's identity, the heart rate reflected by the plethysmographic signals is also measured as the fifth component to reinforce the system's distinctiveness. The photometric signal consists on the volume of blood that flows through the blood vessels per pulse during every beat of the heart.

3.2. Experimental Setup

The recordings of signals are arranged in such a way that one of the subject writing original signatures was allowed to sign 50 original (own) signatures and the subjects who are assigned to forge are allowed to observe it to do the skilled forgery [6].

The forgers are given generous exercise to forge against the two genuine signatures by giving special sessions to practice the original signatures. The subjects are seated in a comfortable chair located in a sound protected room. The data glove was fixed on their right hand and the photoelectric probe was fixed to the index finger of their left hand.

All the subjects appointed for forging were allowed to sign 50 forge signatures one by one each with the help of a tracing paper placed on the original signatures after successful training. The subjects were asked to write the signatures, in two sessions, with an interval of 24 hours. Fifty signatures were collected per subject in one session. The skilled forging of the original signatures from forging subjects were also collected in the same intervals. Forging with 50 signatures per original subject takes a total of 100 signatures per session for two original signatures.

The PPG signal during the signing in process was also recorded for every subject from all the 14 electrodes embedded in the data glove.

Hence there were 200 original signatures and 800 forgings were recorded and considered for analysis. Similarly 1000 simultaneous PPG recordings were also included in the analysis.

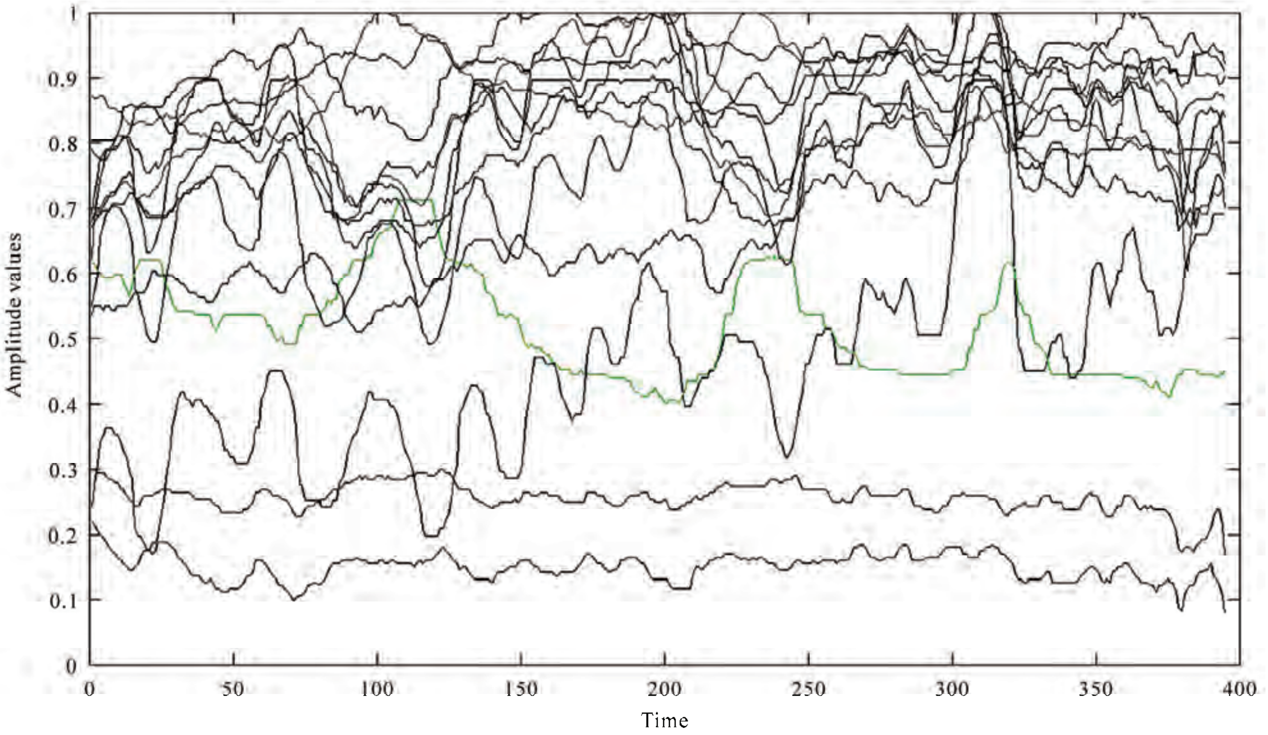


Figure 4. The data glove signal pattern during one signature.

3.3. Preprocessing and Feature Vector Construction

The dimension of each recordings of hand glove signal, ***A*** is of order *n* by *m*, where *n* is the number of electrodes and *m* is the number of sequential samples per second. *n* was fixed as 14 throughout the experiment, and *m* differs in milliseconds as the intra and inter subject vary in signature timings. A sample of the plotted handglove signals are shown in **Figure 4**.

The dimension of every PPG, matrix ***B*** is *j* by *k*, where *j* is the number of LEDs and *k* is the number of sequential samples in one second. Throughout the experiment, *j* was fixed as 4 and *k* was taken up to the exact time length of *m*.

To condense the dimension and to reduce the effects of overlapping spectral information between noise and signature features, singular value decomposition (SVD) approach was applied to both matrix ***A*** and ***B***.

Since there is a real factorization for any real *nX m* matrices, The SVD of matrix ***A*** & ***B*** are is given by

$$A = U.S.V^T \tag{1}$$

$$B = R.F.Q^T \tag{2}$$

where ***U*** (m by m), ***R*** (*j* by *j*) ***Q*** (*k* by *k*) and ***V*** (*n* by *n*) are orthogonal matrices and ***S*** (*m x n*) and ***F*** (*j x k*) are the diagonal matrices. The columns, u_i and v_i of *U* and *V* are the left and right singular vectors respectively, and the diagonal elements of σ_i of ***S*** are called the singular values. The columns, r_i and q_i of ***R*** and ***Q*** are the left and right singular vectors respectively, and the diagonal elements of σ_i of ***S*** are called the singular values.

Next, the singular values for each signal are arranged on the main diagonal in such an order:

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \cdots \geq \sigma_{r+1} = \cdots = \sigma_p = 0 \tag{3}$$

The singular values calculated from the Matrix A are considered as the total Energy of matrix *A*. [7], and are measured in the direction of ith left singular vector of the matrix *A*.

$$E[A] = \|A\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2 \tag{4}$$

Similarly through SVD, the diagonal entries σ_i are the singular values of any matrix *A*, *A* can be written as the sum of rank one matrices as *r* = rank (*A*).

$$A = \sum_{i=1}^{r} ui.\sigma i.v_i^T \tag{5}$$

where (u_i, σ_i, v_i) is the ith singular triplet of matrix *A*.The oriented energy of matrix *A*, E_q is measured in direction *q* is delineated as

$$E_q[A] = \sum_{K=1}^{n} \left(q^T.a_k\right)^2 \tag{6}$$

In general the energy E_Q measured in subspace $Q \in R^m$ is given as

$$E_Q[A] = \sum_{K=1}^{n} \left\|P_Q(a_k)\right\|^2 \tag{7}$$

The SVD can be related to the minima or maxima of the oriented energy distribution as follows.

$$max_{q\in UB} E_q[A] = E_{u1}[A] = \sigma_1^2 \tag{8}$$

$$min_{q\in UB} E_q[A] = E_{un}[A] = \sigma_n^2 \tag{9}$$

From this it is proved that the oriented energy measured in the direction of the *i*th left singular vector of the matrix *A* is equal to the square of *i*th singular value. Hence it is determined that the singular value decomposition protects the characteristics of the source signal matrix given by the *m* signal samplings from *n* electrodes.

Matrix *B*, used to incorporate PPG representation was also subjected to exactly similar SVD process to estimate the singular values for use in feature vector.

The average size of the glove signature matrix is (14,234) as well as the average size of the PPG signal matrix is (4,234). After the application of SVD the features are reduced to (14,1) and (4,1) respectively.

We were used the *l*-largest singular values of ***A*** as well as *q*-largest singular values of ***B*** as feature contents representing every single data glove signal and PPG respectively. Therefore, the entire signal ***A*** is now represented by a highly discriminate feature vector of length ***A*** (*l*) and the entire PPG is represented by ***B*** (*q*). These *l* and *q* largest singular value features of ***A*** and ***B*** contain the feature component of the subjects' unique signature ID that discriminates the original from forge signatures. To minimize computational complexity, we set the *l* value to be five and *q* to 2 throughout these experiments, subsequent to its superior performance during our preliminary simulations.

$$Fs = \left[A_i, \ldots A_j\right] \tag{10}$$

$$Fp = \left[B_i, \ldots B_q\right] \tag{11}$$

where *i* = 1, *J* = 5 and *q* = 2;

The fused feature

$$F = \left[F_s, F_p\right] \tag{12}$$

reflects the pattern of integrated signature components with the heart rate variability for further matching and classification.

3.4. Matching and Classification

The reference signature along with the reference PPG F_G (Genuine Factor) was computed from a set of reference enrollment samples. The pair having minimal overall angle to the rest of signature, PPG pairs was selected as the reference signature to which all the comparisons where carried out. The genuineness of any factor pair F_i

is decided by the similarity factor (SF) to both the components of F_G & F_i are calculated as the angle between their principle subspaces.

4. Results and Discussions

Figure 5 gives Euclidean distance between the genuine reference signature PPG fusion factors with other genuine signature and forge signatures with the corresponding PPG of the subjects. The Euclidian distance is calculated using

$$d=\sqrt{\sum_{j=1}^{L}\left(x_j-t_j\right)^2} \qquad (13)$$

Ten random sample distances across the two sessions were shown taken for considerations and other signatures were also giving similar results. These results were reported in our previous works [4].

The Equal Error Rate (EER) can be calculated if and only if a set of False Acceptance rate (FAR) and False Rejection Rate (FRR) are available. In this experiment, both are found to be zero and hence could not able to draw a curve of FAR and FRR to find the intersection point which is EER.

As an enhancement to this system we intended to find the consistency of the signatures written by the forgers with that of the consistency of the genuine signatories. This is to identify the best forger and later this factor may be used to enhance the authenticity of the entire system using the distinct quality of inter trial coherence.

Table 1 shows the results in terms of Euclidian distance between the signatures cum PPG fusion template to every subject's with their own signature cum PPG features found in different trials. We found that the consistency of the genuine signatory is consistence and all the other four skilled forgers were not able to retain their consistency across the trails. This can be seen from the zigzag lines from **Figure 6**.

The performance of the data glove declines with the reduction of sensors. The performance degradation with the reduction of electrode channels from the hand glove were reported in our previous works [8] to minimize the hardware and volume overheads in data processing. But the proposed technique of this paper helps to eliminate the said problem by providing strong reinforcements provided in two levels. The first one being the PPG factors and the second one is the intra trial variability. Table 1 shows the values of intra trial variability with the corresponding reference temples of every subject.

In all the ten trials the Euclidean distance is calculated against the templates of the individual subjects' training to write the same signature. The proposed signature and PPG combination is not altered through out the experiment.

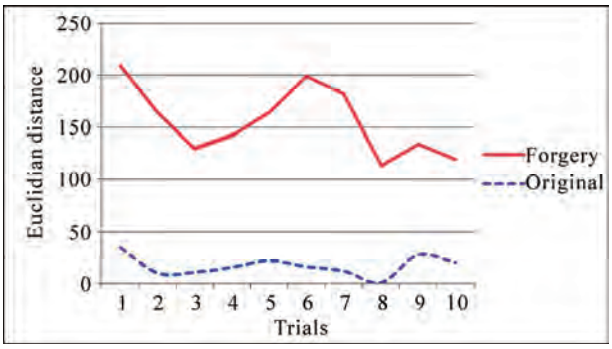


Figure 5. Euclidian distance between genuine signature + ppg factors and forgery.

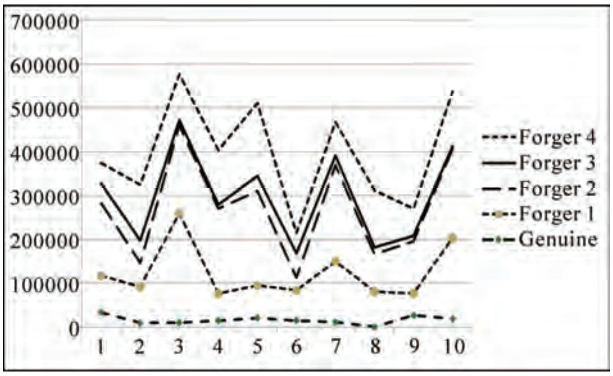


Figure 6. Intra trial variability shown in euclidean distance across ten trials among individual forgers compared with the intra trial variability of genuine features.

Table 1. Intra trial variations in euclidian distance between individual templates to corresponding individual signatures.

	Trial1	Trial2	Trial3	Trial4	Trial5	Trial6	Trial7	Trial8	Trial9	Trial10	Average Rnd
Genuine	34370	9934	10759	15392	21582	15866	11819	875.38	27423	19917	16793
Forger1	165410	53161	198290	191950	215300	28236	218790	83738	118500	201180	147455
Forger2	44223	51702	12488	8906.1	33876	54862	21046	15533	12128	6096.4	26086
Forger3	46886	127180	103760	123760	165250	46779	75789	129740	62206	124990	100634
Forger4	82633	82805	249120	62179	73780	68638	138800	81469	49637	184130	107319

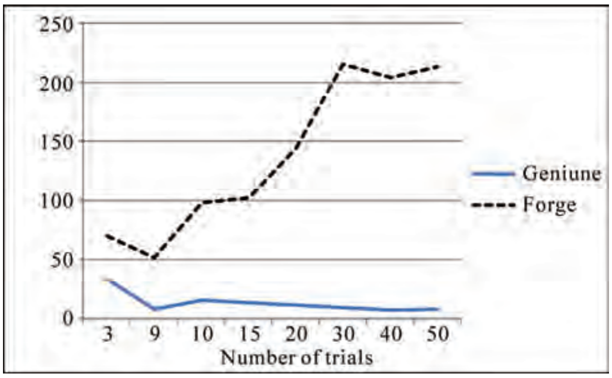


Figure 7. Variability rates against the number of trials.

This wide gap between the intra trial variability shown in **Figure 7** reveals that this factor can be considered as an improvement factor in reinforcing the robustness of data glove + PPG based signature verification system.

5. Conclusions

The proposed intra trial variability measurement of multi-modal signature + PPG based signature verification system, is found to be reliable in strengthening the identification of genuine subjects of Data glove based signature verification system. The novelty lies in two levels of using PPG factors and its augment to the robustness of the data glove features as well as the counting of Intra trial variability factors against the reference signatures. In feature the data glove may be fabricated as additionally accommodating the PPG sensor to make it easy for everyone. This technique is verified with the present easy modeling of data glove signals using SVD. Two sets of singular vectors produced by SVD are fused as the feature, where the primary set is from data glove with the maximum energy of the signature during the process of signature writing, and the secondary set is derived from the photometric signals extracted by PPG simultaneously during the process has been presented. These selected set of vectors are known as the principle subspace of data glove output matrix ***A*** and PPG output matrix ***B*** respectively. These principle subspace set are used to model a reinforced signature feature robust against any forge attack.

This research work is a venture to demonstrate the intra trial variability factor enhances the signature verification system that uses the PPG as its combination. This novel system is much potential to offer a sensitive high level security for applications like banking, electronic commerce and legal proceedings, than the existing similar systems. On the other hand this founding strongly supports the reduction in hardware by manufacture data glove integrated with PPG with minimum sensors so that the size of the equipment can be made simple and efficient for handling by a single hand. Since the possibility of reducing the feature size by means of reducing sensor in the data glove as well as reducing timing in the PPG, we can achieve a low cost signature verification system suitable to a common user in common place.

6. References

[1] R. Plamondon and S. N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, 2000, pp. 63-84.

[2] S. Rhee, B.-H. Yang and H. H. Asada, "Modelling of Finger Photoplethysmography for Wearable Sensors," *Proceedings of* 21*st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Atlanta, 1999.

[3] Y. Y. Gu, Y. Zhang and Y. T. Zhang, "A Novel Biometric Approach in Human Verification by Photophlytesmograpic Signals," *Proceedings of the* 4*th IEEE conference on Information Technology Applications in Biomedicine*, Birmingham, 2003, pp. 13-14.

[4] A. Samraj, N. G. Noma and S. Sayed, "Quantification of Emotional Features on Phtoplethysomogrpic Wave Forms Using Box Counting Method of Fractal Dimention," *Proceedings of the* 8*th WSEAS International Conference on Circuits*, *Systems*, *Electronics*, *Control & Signal, Processing* (*CSECS*'09), Puerto De La Cruz, 2009, pp. 24-29.

[5] J. C. Yao, X. D. Sun and Y. B. wan, "A Pilot Study on Using Derivatives of Photop Phlythesomogrpic Signals as Biometric Identifier," *Proceedings of* 24*th Annual International Conference of the IEEE EMBS*, 2007, pp. 4576-4579.

[6] B. Majhi, Y. Santhosh Reddy and D. Prassanna Babu, "Novel Features for off-Line Signature Verification," *International Journal of Computers Communication & Control*, Vol. 1, No. 1, 2006, pp. 17-24.

[7] "SVD and Signal Processing: Algorithms, Applications and Architectures," F. Deprettere, Ed., North Holland Publishing Co., Amsterdam, 1989.

[8] N. S. Kamel, S. sayeed and G. A. Ellis, "Glove Based Approach to Online Signature Verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 5, 2008, pp. 1-5.

Journal of Information Security, **2010, 1, 29-40**
doi:10.4236/jis.2010.11004 Published Online July 2010 (http://www.SciRP.org/journal/jis)

Design and Implementation of Multilevel Access Control in Synchronized Audio to Audio Steganography Using Symmetric Polynomial Scheme

Jeddy Nafeesa Begum[1], Krishnan Kumar[1], Vembu Sumathy[2]
[1]*Department of Computer Science and Engineering, Government College of Engineering, Bargur, India*
[2]*Department of Electronics and Communications Engineering, Government College of Technology, Coimbatore, India*
E-mail: {*nafeesa_jeddy, pkk_kumar*}*@yahoo.com, sumi_gct*2001*@yahoo.co.in*
Received March 5, 2010; *revised July* 15, 2010; *accepted July* 19, 2010

Abstract

Steganography techniques are used in Multimedia data transfer to prevent adversaries from eaves dropping. Synchronized audio to audio steganography deals with recording the secret audio, hiding it in another audio file and subsequently sending to multiple receivers. This paper proposes a Multilevel Access control in Synchronized audio steganography, so that Audio files which are meant for the users of low level class can be listened by higher level users, whereas the vice-versa is not allowed. To provide multilevel access control, symmetric polynomial based scheme is used. The steganography scheme makes it possible to hide the audio in different bit locations of host media without inviting suspicion. The Secret file is embedded in a cover media with a key. At the receiving end the key can be derived by all the classes which are higher in the hierarchy using symmetric polynomial and the audio file is played. The system is implemented and found to be secure, fast and scalable. Simulation results show that the system is dynamic in nature and allows any type of hierarchy. The proposed approach is better even during frequent member joins and leaves. The computation cost is reduced as the same algorithm is used for key computation and descendant key derivation. Steganography technique used in this paper does not use the conventional LSB's and uses two bit positions and the hidden data occurs only from a frame which is dictated by the key that is used. Hence the quality of stego data is improved.

Keywords: Steganography, Multilevel Access Control, Synchronized Audio, Symmetric Polynomial, Dynamic, Scalable

1. Introduction

Transmission of audio files is very important for many applications and it is found that this transmission takes place through an insecure medium. For a live session where on the go audio transmission takes place, efficient techniques should be used. One way of preventing the dissemination of secret audio is through digital audio stenography. This protects valuable information from unauthorized persons. For real time audio transmissions the secret data is recorded and send subsequently to the receivers. For hiding the message there is a need for a secret key that is available with all receivers. This key has to be changed to preserve forward and backward secrecy. There is an additional very important requirement called the multilevel access control. There are many scenarios in which situation arises, that only some users should be able to hear the data or all higher level users should also be able to hear the message that are relayed to the descendant users. To implement such a multilevel access control in steganography symmetric polynomial approach is used. In most existing schemes, key derivation is different from key computation. Key derivation needs iterative computation of keys for nodes along the path from a node to its descendant, which is inefficient if the path is long. In this scheme, both operations are same by substituting (different) parameters in the same polynomial function assigned to node *v*. Thus, the key derivation efficiency can be improved. Our scheme also supports full dynamics at both node and user levels and permits any random access hierarchies. More importantly, removing nodes and/or users is an operation as simple as adding nodes and/or users in the hierarchy. A trusted Central Authority (CA) can assign secrets (*i.e.*, polynomials) to corresponding nodes so that nodes can compute their keys. Also, nodes can derive their descendants' keys without involvement of the CA once polynomial functions were distributed to them. In addition, the storage requirement and computation complexity at the CA are almost same as that at individual nodes, thus, the CA would not be a performance bottleneck and can deal with dynamic operations efficiently.

The rest of the paper is as follows, Section 2 deals with related work Section 3 gives an insight into multilevel access control problem. Section 4 gives the system Overview, Section 5 describes the audio steganography method Section 6 deals about the symmetric polynomial approach Section 7 shows the simulation results and Section 8 gives the performance analysis and Section 8 concludes the paper.

2. Related Work

2.1. Related Work in Steganography

Information hiding using steganography [1] relates to protection of text, image, audio and digital content on a cover medium [2-5]. The cover media in many cases has been an image [1]. Aoki presented a method in which information that is useful for widening the base band is hidden into the speech data [6]. Sub band Phase shifting was also proposed for acoustic data hiding [3]. All these schemes focus on data that is stored in a hard disk or any other hardware whereas there are many applications like military warfare where the audio data is to be given in real time as in live broadcast system. Techniques for hiding the audio in real time came into existence [7] and systems for synchronized audio steganography has been developed and evaluated [8]. In our scheme secret speech data is recorded and at the same time it is sent to the receiver and a trusted receiver extracts the speech from the stego data using the key which is shared between the server and the receiver.

2.2. Related Work in Multilevel Access Control

The first multi level access solution was proposed by Akl *et al.* [9,10] in 1983 and followed by many others [11-21]. These schemes basically rely on a one-way function so that a node *v* can easily compute *v*'s descendants' keys whereas *v*'s key is computationally difficult to compute by *v*'s descendant nodes. In this paper, we propose a new scheme based on symmetric polynomials for synchronized audio data. Unlike many existing schemes based on one-way functions, our scheme is based on a secret sharing method which makes the scheme unconditionally secure [21,22]. Also, this multilevel access control requires two types of key operations: (1) key computation. A node *v* computes its own key and (2) key derivation. A node *v* computes its descendants' keys.

3. Multi Level Access Control Problem

In practice, many group applications contain multiple related data streams and have the members with various access privileges. These applications prevail in various scenarios.

1) Multimedia applications distributing data in multilayer coding format. For example, in a video broadcast, users with a normal TV receiver can receive the normal format, while others with HDTV receivers can receive both the normal format and the extra information needed to achieve HDTV resolution.

2) Communications in hierarchically managed organizations, such as military group communications where participants have different access authorizations.

3) Multilevel access control can be effectively used in Audio library and patient monitoring system.

4) E-newspaper subscription service may have multiple data streams. The service provider classifies the users into membership groups and provides data streams according to the subscription.

5) Video multicasting service in which users can subscribe to services with different video quality.

Defense messaging systems where the server sends messages and one or more can see the message according to the access rights.

In these applications, group members subscribe to different data streams, or possibly multiple of them. Thus, it is necessary to develop group access control mechanism that supports the multi-level access privilege, which is referred to as the Multilevel Access Control.

4. System Overview

Multilevel Access Control applied to real time audio to audio steganography is useful for organizations which have a hierarchical structure. e.g., in the Indian Military system the following hierarchy exists in "**Figure 1**".

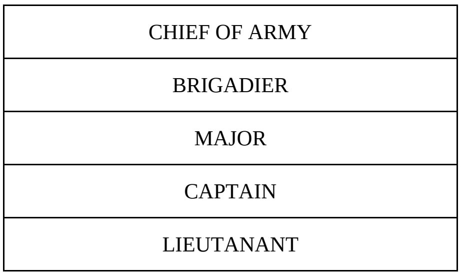


Figure 1. Military hierarchy.

In such a type of system, audio messages sent to a lower class should be heard by the active members of lower class and also by all active members of the higher class. It is not only essential to maintain the access control but the data should be hidden as well. The sequence of events is as follows.

At the server:

1) Generate a general polynomial.

2) Give a symmetric polynomial to each of the classes.

3) Record the real time audio on a microphone.

4) Use Steganography technique to hide the audio into another audio.

5) A text can also be hidden in an audio file.

6) The file is encrypted by the class key for whom the Message is to be relayed.

7) The symmetric polynomial generates a key in this case. The server takes care to include class dynamics so the hierarchy can be changed at any time.

8) Users can join or leave a class at all instances. Keys are recalculated so that Forward and Backward secrecy is maintained.

9) If the users within the group need to transfer message among themselves. The private key of the users is used.

The above steps are given pictorially in "**Figure 2**".

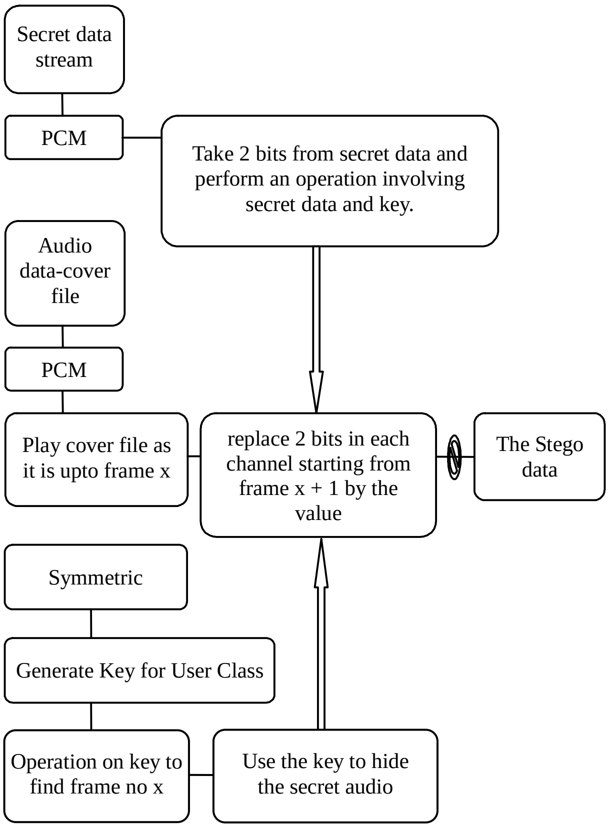

Secret data stream
PCM
Take 2 bits from secret data and perform an operation involving secret data and key.
Audio data-cover file
PCM
Play cover file as it is upto frame x
replace 2 bits in each channel starting from frame x + 1 by the value
The Stego data
Symmetric
Generate Key for User Class
Operation on key to find frame no x
Use the key to hide the secret audio

Figure 2. At the server.

At the receiver:

1) All the active receivers will receive the audio file.

2) If the recipient belongs to the actual intended class he can use the polynomial to get the hidden audio file instantaneously.

3) If the recipient belongs to a class lower than the Actual intended class in the hierarchy, he will not be able to derive the key .The polynomial derivation method will give a null value.

4) If the recipient belongs to a higher class he can derive the key and hear to the audio file and in case a text message was sent it can be seen.

5) The users at the same class can transfer messages among them.

6) When a user leaves or joins. The new polynomials are given by the server and the private keys also get updated according to the new polynomial. Other classes are not affected by this.

7) Service messages can be sent from higher class users to lower class users.

The above steps are explained pictorially in "**Figure 3**".

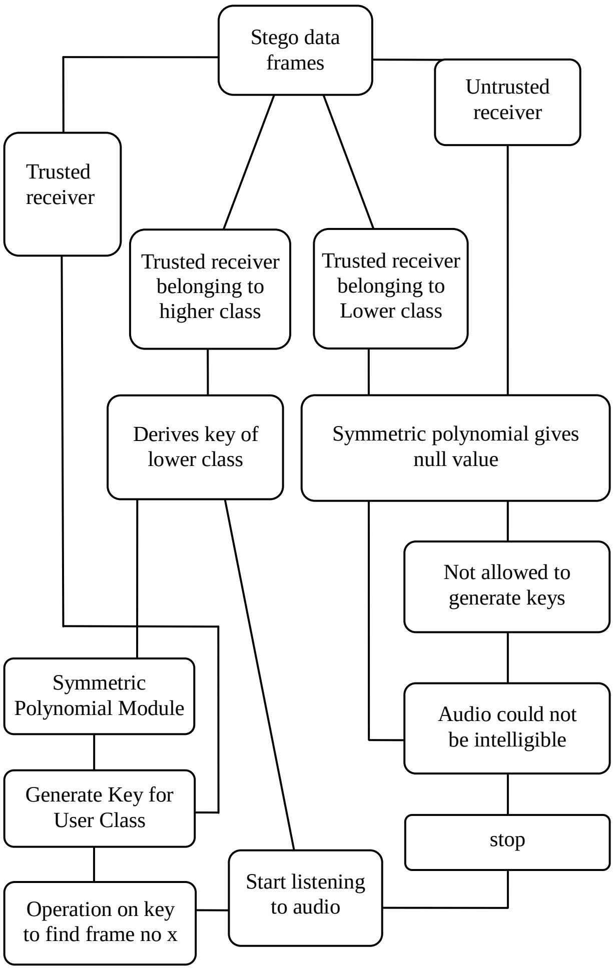

Stego data frames
Untrusted receiver
Trusted receiver
Trusted receiver belonging to higher class
Trusted receiver belonging to Lower class
Derives key of lower class
Symmetric polynomial gives null value
Not allowed to generate keys
Symmetric Polynomial Module
Audio could not be intelligible
Generate Key for User Class
stop
Start listening to audio
Operation on key to find frame no x

Figure 3. At the receiver.

4.1. Features of our Model

Solutions of a synchronized steganography have been given in past [8]. In this once the stego data reaches the destination the audio can be listened by the trusted receiver. Our contributions are

1) The key is used during the embedding process also.

2) The key is not a simple key it identifies a class of users.

3) If the key used belongs to a low level group in the hierarchy, the higher level class of user can derive the key using the symmetric polynomial approach and listen to it.

4) There can be normal message transfer among the Group elements and also service messages from higher classes.

5) Forward and backward secrecy is maintained.

6) It is a dynamic one where new hierarchies can be introduced, User level and class level dynamics are taken care.

5. Synchronized Audio to Audio Steganography

The data to be sent is not available. It is recorded in real time before starting the steganography scheme. When the covering media is being played at the same time the audio file is recorded and put into the cover file. The stego bit stream is then transmitted to the receivers. Multilevel Access control using symmetric polynomial is used at this stage to generate the key to make secure transmission of the audio file. According to the hierarchy the trusted users are able to retrieve the hidden audio file.In this system, both of cover data and secret data are divided into fix-sized frames according to pulse code modulation setting. To cover low size and high phonetic quality the sampling rate of the hidden audio is set to 8 kHz. Three main processes are involved in the synchronized audio to audio steganography.

1) Using data sampling, acoustic signals are embedded into another audio.

2) Bit Embedding: The key used helps in hiding the audio file in bit positions and once the bit positions are found data is hidden after performing an operation on secret data and the key.

3) Synchronized Process: Malicious and intentional attacks can be avoided as the secret data is real time.

5.1. Algorithm

Step 1: Record the Secret Speech Data: The audio files are divided into fix-sized frames and set to be specific PCM format. PCM quantification is decided by sampling rate, sampling size, and sampling channel. The PCM property of cover audio wav is set to be 32kHz-16bit-2ch, while the secret wav data is 16kHz-8bit-1ch.
Formula used for steganography:
Steganography process:

cover_medium + hidden_data + stego_key
= stego_medium

Part of The Wave File Format opened using Notepad:
52 49 46 46 24 40 01 00 57 41 56 45 66 6D 74 RIFF $ @.
Wave Fmt10 00 00 00 01 00 02 00 11 2B 00 00 44 AC 00 00 ……. + …D…04 00 10 00 64 61 74 61 00 40 01 00 00 00 00 00 …. data. @..........
Wave_Format_PCM: 01 11 channel count: 02 00
Samples: 11 2B 00 00 bytes: 44 AC 00 00
Block align: 04 00 bits per sample: 10 00
Step 2: Use Symmetric Polynomial to calculate key of the class
Step 3: Perform calculation and decide the frame from which the data is to be embedded.
Step 4: Decide two bit locations in each frame and clear the bit in the locations.

*cmask*1 = (2^{loc1}−1) xor (keybit)

cmask2 = (2^{loc2}−1) xor (Keybit)

cmask = cmask1 $\wedge$ cmask2

hide the secret data bits into these bit locations by again performing an operation on the secret data along with the key. The cover media has two channels so data is written on both the channels. Other bits are not changed.
Step 5: The next set of data will go to the next frame.
Step 6: Do the repetitive process till the recoding is over.
Step 8: Transmit using sockets.
Step 9: At the receiving end, use the key and play the audio.
Step 10: If the receiving user belongs to higher class, he can derive the key and listen to the audio.

6. Symmetric Polynomial Approach

6.1. Symmetric Polynomial

A CA selects a large positive integer *P* as the system modulus, *p* need not be a prime and a threshold number *t* so that less than *t* + 1 users cannot collaborate together to disclose their ancestors' keys. Then, the CA can randomly generate a symmetric polynomial in *m* variables with co-efficient from Z_p in which the degree of any variables is at most *t* as:

$$P\left(x_1, x_2, \ldots, x_m\right) = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_m=0}^{t} a_{i_1, i_2, \ldots, i_m} x_1^{i_1} x_2^{i_2} \cdots x_m^{i_m} \left(\bmod P\right)$$

where $a_{i_1, i_2, \ldots, i_m}$ are randomly generated coefficients by the CA. The polynomial function $P\left(x_1, x_2, \ldots, x_m\right)$ is kept as a secret to the CA. Every class in the hierarchy has a polynomial function which is derived from $P\left(x_1, x_2, \dots, x_m\right)$ and the polynomial function is transmitted to each class securely by the CA.

Example for Symmetric Polynomial

The following polynomial function is a suitable example for symmetric polynomials.

$$f\left(x_1, \dots, x_n\right) = \sum_{i_1=0}^{w} \cdots \sum_{i_n=0}^{w} a_{i_1, i_2, \dots, i_n} x_1^{i_1} \dots x_n^{i_n}$$

where $a_{i_1, i_2, \dots, i_n} = a_{\pi(i_1, i_2, \dots, i_n)}$

For all permutations π of {1,…*n*}

For example,

suppose $n = 3$ and $w = 2$,

Let i_1, i_2, i_3 be as follows

$a_{0,0,0} = 13$

$a_{0,0,1} = a_{0,1,0} = a_{1,0,0} = 3$

$a_{0,0,2} = a_{0,2,0} = a_{2,0,0} = 7$

$a_{0,1,1} = a_{1,0,1} = a_{1,1,0} = 4$

$a_{0,1,2} = a_{0,2,1} = a_{1,0,2} = a_{2,0,1} = a_{2,1,0} = 8$

$a_{0,2,2} = a_{2,0,2} = a_{2,2,0} = 9$

$a_{1,2,2} = a_{2,1,2} = a_{2,2,1} = 11$

$a_{2,2,2} = 5$

6.2. Polynomial Function

To derive proper keys in the hierarchy, the CA generates some publicly known numbers

1) *n* random numbers s_{i} associated with C_i for $i = 1, 2, \dots n$ and 2) and $(m - 1)$ additional random numbers r_j for $j = 1, 2, \dots, m - 1$

(Note: s_i and r_j belong to Z_p).

For each class C_i with an ancestor set $S_i = \{C_{i_1}, C_{i_1}, \dots, C_{i_n}\}$ where i_j is an ordinal number such that $1 \le i_j \ne i \le n$, class C_i is given a polynomial function, g_{i} derived by the CA as,

$$g_i(x_{m_i+2}, x_{m_i+3}, \dots, x_m) = P\big(s_i, s_{i_1}, s_{i_2}, \dots, s_{i_m}, \\ x_{m_i+2}, x_{m_i+3}, \dots, x_m\big)$$

A symmetric polynomial based scheme:

A is an set of n classes – $\{C_1, C_2, C_3, \dots, C_n\}$

B is a set of ancestral classes of set *A*.

$$B = \{S_1, S_2, S_3, \dots, S_n\}$$

mi is calculated as the number of the ancestral classes $m_i = |S_i|$. We choose m such that $m \ge \max\{m_1, m_2, \dots, m_n\} + 1$. Here *m* is the number of parameter in the polynomial function *P*, where *P* is to construct our multi level access control scheme.

We illustrate with a sample hierarchy "**Figure 4**"

Here we have nine classes

$\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9\}$

Ancestral classes' sets are

$S_1 = \{\varphi\}$, $S_2 = \{\varphi\}$, $S_3 = \{C_1, C_2\}$, $S_4 = \{C_2\}$

$S_5 = \{C_2\}$, $S_6 = \{C_1, C_2, C_3\}$,

$S_7 = \{C_1, C_2, C_3, C_4\}$

$S_8 = \{C_2, C_3, C_5\}$, $S_9 = \{C_2, C_5\}$

From the previous step, we need to choose m such that $m \ge \max\{m_1, m_2, m_3, \dots, m_9\}$. Let us choose $m = 7$, it will allow to expand the hierarchy without changing the value of *m*.

Symmetric polynomial, we are using here is as follows

$$P(x_1, x_2, \dots x_m) = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_m=0}^{t} a_{i_1, i_2, \dots i_m}, \\ x_1^{i_1}, x_2^{i_2}, \dots x_m^{i_m} \bmod P$$

Here *t* is threshold number. We can classify our work into two Key Calculation and Key Derivation.

Key Calculation:

We can calculate key K_i of class C_i as follows

$$K\mathrm{i} = P\left(s_i, s_{i_1}, s_{i_2}, \dots, s_{i_m}, s_1{}', s_2{}', \dots, s_{m\text{-}m_i\text{-}1}{}'\right) \quad (1)$$

Key Derivation:

In key derivation, we are using a term *Sj*/*I* which is

$$S_{j/I} = S_j \big/ \left(S_i U \{C_i\}\right) = \left\{C_{(j/i)1}, C_{(j/i)2}, \dots, C_{(j/i)rj}\right\}$$

Consider a class C_i which is ancestor to class C_j and key K_j can be calculated by C_i as,

$$K_j = g_i\left(s_j, s_{(j/i)1}, s_{(j/i)2}, \dots, s_{(j/i)r_j}, s'_1, s'_2, \dots, s'_{m\text{-}m_i\text{-}2\text{-}r_j}\right) \\ = P\Big(s_i, s_j, s_i, s_{i_1}, s_{i_2}, \dots, s_{i_{m_i}}, s_{(j/i)1}, s_{(j/i)2}, \dots, s_{(j/i)r_j}, \\ s'_1, s'_2, \dots, s'_{m\text{-}m_i\text{-}2\text{-}r_j}\Big) \quad (2)$$

Example Key Derivation

Consider that C_3 is an ancestor class to class C_7.

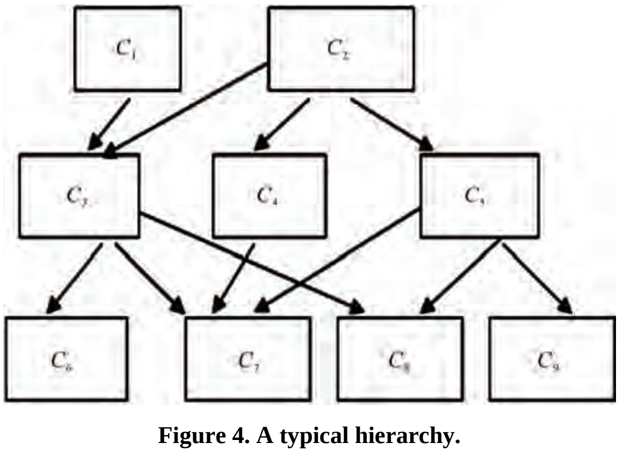


Figure 4. A typical hierarchy.

Then K_7 can be derived by C_3 in the following steps.

$$S_{7/3} = \{C4\}$$
$$K7 = P(s_3, s_7, s_1, s_2, s_4, s_1{}', s_2{}')$$

Key Calculation for the Classes using Equation (1)

$$K1 = P(s_1, r_1, r_2, r_3, r_4. r_5, r_6)$$
$$K2 = P(s_2, r_1, r_2, r_3, r_4. r_5, r_6)$$
$$K3 = P(s_1, s_2, s_3, r_1, r_2, r_3, r_4)$$
$$K4 = P(s_4, s_1, s_2, r_1, r_2, r_3, r_4)$$
$$K5 = P(s_5, s_1, s_2, r_1, r_2, r_3, r_4)$$
$$K6 = P(s_6, s_1, s_2, s_3, r_1, r_2, r_3)$$
$$K7 = P(s_7, s_1, s_2, s_3, s_4, r_1, r_2)$$
$$K8 = P(s_8, s_1, s_2, s_3, s_4, s_5, r_1)$$
$$K9 = P(s_9, s_1, s_2, s_3, s_4, s_5, r_1)$$

Key Derivation of class 7 by class 3 using Equation (2)

$$S3 = \{C1, C2\}$$
$$S7 = \{C1, C2, C3, C4\}$$
$$S3U\{C3\} = \{C1, C2, C3\}$$
$$S3/7 = \{C4\}$$
$$K7 = P(s_3, s_7, s_1, s_2, s_4, r_1, r_2)$$

Which is equal to the key calculated by class7 itself.

Key Derivation of class 3 by class 7 using Equation (2)

$$S3 = \{C1, C2\}$$
$$S7 = \{C1, C2, C3, C4\}$$
$$S7U = \{C7\} = \{C1, C2, C3, C4, C7\} \qquad (3)$$
$$S7/3 = \{\varphi\}$$
$$K7 = P(s_7, s_3, s_1, s_2, s_3, s_4, s_1{}')$$

It can be seen that when the class derives its own key and when a ancestor of this class derives the key same parameters are passed in the polynomial but the combination differs when a wrong ancestor derives the key, the parameters are not the same.

The default values, we have taken are
$m = 7$, $P = 2147483646$, $s_1 = 5$, $s_2 = 10$, $s_3 = 13$, $s_4 = 9$, $s_5 = 6$, $s_6 = 22$, $s_7 = 18$, $s_8 = 30$, $s_9 = 39$, $r_1 = 11$, $r_2 = 12$, $r_3 = 13$, $r_4 = 14$, $r_5 = 15$, $r_6 = 16$, $r_7 = 17$, $r_8 = 18$, $r_9 = 19$ (instead of *s*’ we have used *r*)

For a small Hierarchy “**Figure 5**”, with more than two classes, we can easily illustrate our key calculations, where each class consists of several users.

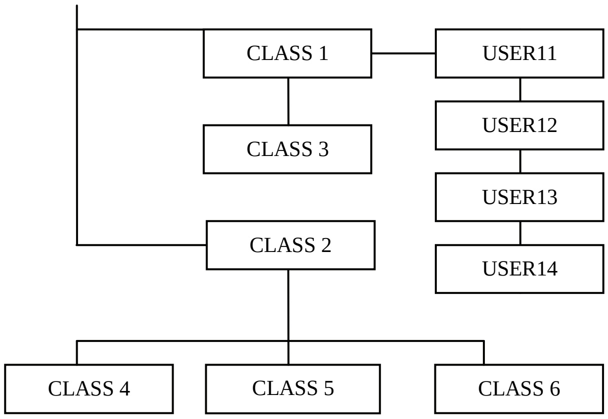


Figure 5. Sample hierarchy to illustrate calculations.

Key Calculations:
The parameters to be passed for calculating group key class 2 are $P(S_2, r_1, r_2, r_3, r_4, r_5, r_6)$
Group key class C_2 = 699615258
1) Private key for user 21 = 699615258 + (699615258/21) = 732930270
2) Private key for user 22 = 699615258 + (699615258/22) = 731415951
3) Private key for user 23 =699615258 + (699615258/23) = 730033312
4) Private key for user 24 = 699615258 + (699615258/24) = 728765893
Key Derivation:
Deriving the group key of class C_4 using its ancestral class C_2

$$S_2 = \{\ \} \quad S_4 = \{s_2\}$$

Sj|i can be calculated as
$S_{4|2} = \{s_4\} \Big| \left(S_2 U \{C_2\}\right)$ Set Difference $S_{4|2} = \{\Phi\}$
The parameters to be passed for deriving the key of class C_4 using C_2

$$\text{Key} = p(s_2, s_4, r_1, r_2, r_3, r_4, r_5) = p(10, 9, 11, 12, 13, 14, 15)$$
$$= 1947982264$$

Private Keys are used for local communication.

6.3. Class Level Dynamics

6.3.1. Adding a Class

When a new class C_r is added, we need to verify whether m value satisfies the new node constraints

1) If $m < \max\{m_1, m_2, ..., m_n, m_r\} + 1$, a new m value will be generated so that $m \geq \max\{m_1, m_2, ..., m_n, m_r\} + 1$. Also, the CA will regenerate a new polynomial functions $P(x_1, x_2, \ldots, x_m)$ accordingly. In addition, all polynomial functions of classes are recomputed and retransmitted securely.

2) If $m \geq \max\{m_1, m_2, ..., m_n, m_r\} + 1$, the CA selects a random number s_r for the new class C_r so that a new polynomial function g_r can be computed and transmitted to class C_r securely. However, if class C_r is added as a parent class of any existing classes, we need to modify keys of C_r's descendant classes to prevent class C_r from obtaining old keys of its descendant.

6.3.2. Deleting a Class

When a class C_r is removed from the hierarchy, we need to determine whether the class C_r is a leaf node or a parent node. Here, a leaf node is defined as a node without any descendant:

1) class C_r is a leaf node: The CA can simply discard the public parameter s_r without changing any other keys.

2) class C_r is a parent node: Once class C_r is deleted from the hierarchy, we cannot allow it to compute keys of C_r's descendant classes using polynomial function g_r. We need to prevent class C_r from accessing its descendants' resources.

6.3.3. Moving a Class

A class C_r can be moved from one node to another node in the hierarchy. There are four cases:

1) leaf node to another leaf node: the CA simply recomputes new polynomial function g_r according the new hierarchy and securely transmits g_r to C_r.

2) leaf node to parent node: the CA recomputes polynomial functions of class C_r and C_r's new descendant classes according to the new hierarchy. The CA securely transmits polynomial functions to the affected classes;

3) parent node to leaf node: the CA recomputes polynomial functions of previous descendant classes of C_r and class C_r according to the new hierarchy and then, securely transmits these polynomial functions to the affected classes

4) parent node to parent node: the CA recomputes polynomial functions of previous and present descendant classes of C_r and class C_r according to the new hierarchy and then, securely transmits these polynomial functions to the affected classes.

6.3.4. Merging a Class

Two or more classes can merge together and become one class C_r. Similarly, the CA needs to find previous and present descendant classes of the merging classes. The CA randomly chooses a new number sr and then, generates polynomial functions for all corresponding classes.

6.3.5. Splitting a Class

A class C_r splits into two classes C_{r1} and C_{r2}. Depending on whether C_r is a parent node or leaf node, the CA has to determine what previous and present descendant classes are associated with these classes (C_r, C_{r1} and C_{r2}). The CA then selects two new numbers s_{j1} and s_{j2} and generates polynomial functions for these affected classes.

6.3.6. Adding a Link

If two classes C_r and C_k are linked together, we establish a new direct parent-child relationship between two classes, say class C_r is the parent of class C_k. There are two different cases: 1) class C_r was an ancestor of class C_k through other classes. The CA does not need to perform anything; and 2) class C_r is the only parent for class C_k in the new hierarchy. The CA selects a new number S_k, and generates new polynomial functions for class C_k and its descendants classes. The CA securely transmits new polynomial functions to these affected classes.

6.3.7. Deleting a Link

If two linked classes C_r and C_k are disconnected, we destroy a direct parent-child relationship between two classes, say class C_r will not be the parent of class C_k in the new hierarchy. Again, there are two different cases: 1) class C_r is still an ancestor of class C_k through other classes in the new hierarchy. The CA does not need to perform anything; and 2) class C_r is not an ancestor for class C_k in the new hierarchy. The CA selects a new number S_k, and generates new polynomial functions for class C_k and its descendants classes. The CA securely transmits new polynomial functions to these affected classes.

6.4. User Level Dynamics

In this scheme, every class represents certain access privileges. Also, a group of users in a class can share a key if they belong to the same class. For example, all users in class C_j can compute the keys of class C_j and its descendant classes. Dynamic user operations deal with how a user can join in a class or leave from a class, and possible displacement from one class to a different class. They all require the class key to be changed after any user operation is completed so that the issue of backward secrecy and forward secrecy can be addressed. Specifically, our scheme can revoke a user from a class C_j. It is as quick and efficient as to join a user in the class C_j. Both operations require that the CA randomly select a new public parameter s_j for C_j and recompute a new polynomial function g_j by using the new s_j. Since the polynomial function g_j is newly produced, other polynomial functions and keys are also recomputed for the descendant classes of C_j. This will guarantee both backward secrecy and forward secrecy. The efficiency can be improved if backward secrecy or forward secrecy is not required. Another common user operation is to allow a user to move from one class C_j to another class C_k. Here, the CA will randomly choose two new public parameters s_j and sk for C_j and C_k so that new polynomial functions and keys are recomputed and transmitted to C_j, C_k and their descendants respectively. Thus, both backward secrecy and forward secrecy are guaranteed.

6.4.1. User Join

Every time if a single user wants to join a group the CA just allows the user to be added to the hierarchy and generates a private for that user by providing the corresponding group key. When a new user joins the hierarchy, it should be provided with a group key and there are no changes to be made on the user's key.

6.4.2. User Leave

When a user wants to leave from the hierarchy the CA change the group key by making changes on anyone of the following changing the polynomial or changing the value factor *P*.

7. Simulation Results

The system is developed using .NET and found to be secure and fast. The system takes care of USER level and class level dynamics. The large number of numbers prevents a possible guessing. e.g., for a eight parameter polynomial, 16! (*i.e.*, 10922789888000) combinations possible. Bursty leave and join operations also are possible and the system can be used for any hierarchy. The outputs are shown in **Figures 6-10**.

8. Performance Analysis

Performance and security: Each user u_i will receive

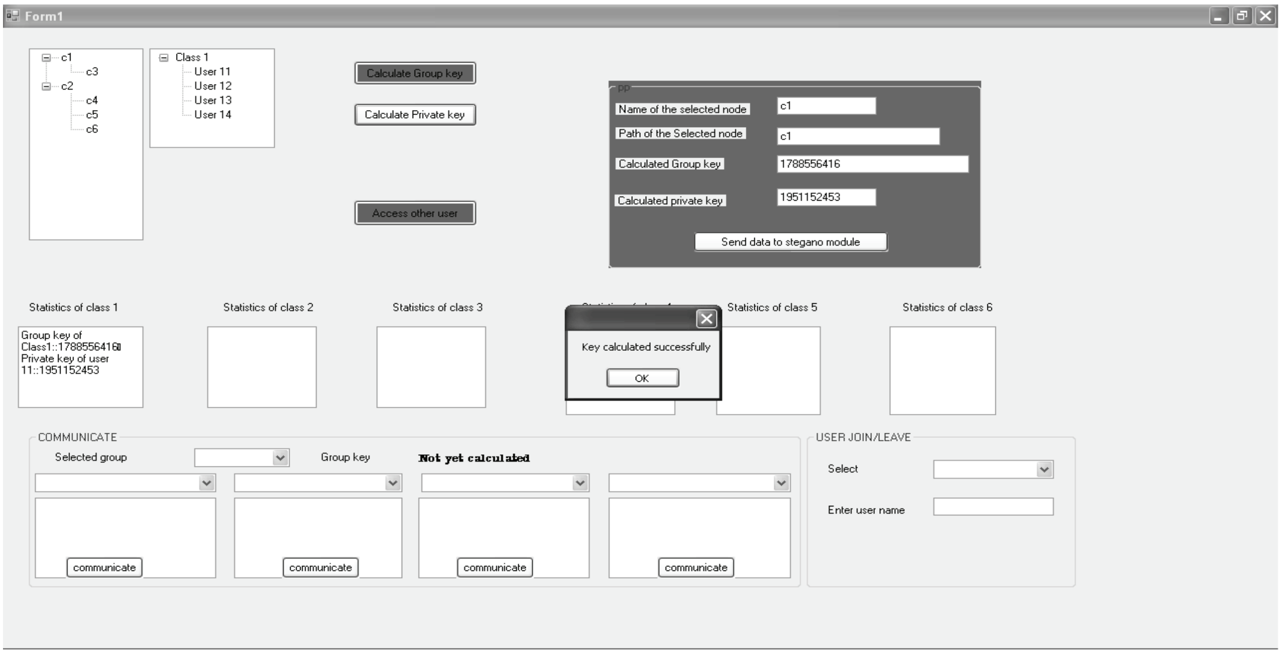
$$g_i = f\left(s_i, x_2, \ldots x_n\right) = g\left(x_2, \ldots x_n\right) = \sum_{i_2=0}^{i_2=w} \cdots \sum_{i_n=0}^{i_n=w} a_{i_2,\ldots i_n}, x_2^{i_2}, \ldots x_n^{i_n}$$

The time complexity for computing the group key is O (w^{2n}). An important measure for a secure group com munication scheme is the number of rekeying messages. Suppose that t users will be joining the group. The TA will send k and g_i to each of them respectively (2t messages) and broadcast one message to tell which users are joining. The total number of rekeying messages is O (2t). Suppose that t users are leaving the group. The TA only broadcasts one message to tell which users are leaving, thus the number of rekeying messages is O (1). Suppose that t users are joining and another v users are leaving the group, the total number of rekeying messages is still O (2t).

As for the security of the scheme, if w + 1 class collude, then they can Figure out the function f entirely. Therefore, the scheme is w-resilient. Moreover, if less than w + 1 classes collude, they cannot get any information about the key, *i.e.*, any value in the key space looks like a valid and equiprobable key to these colluding users. It follows that the scheme is unconditionally secure.

8.1. Memory

Each user will be able to calculate the key based on the



Name of the selected node: c1

Path of the Selected node: c1

Calculated Group key: 1788556416

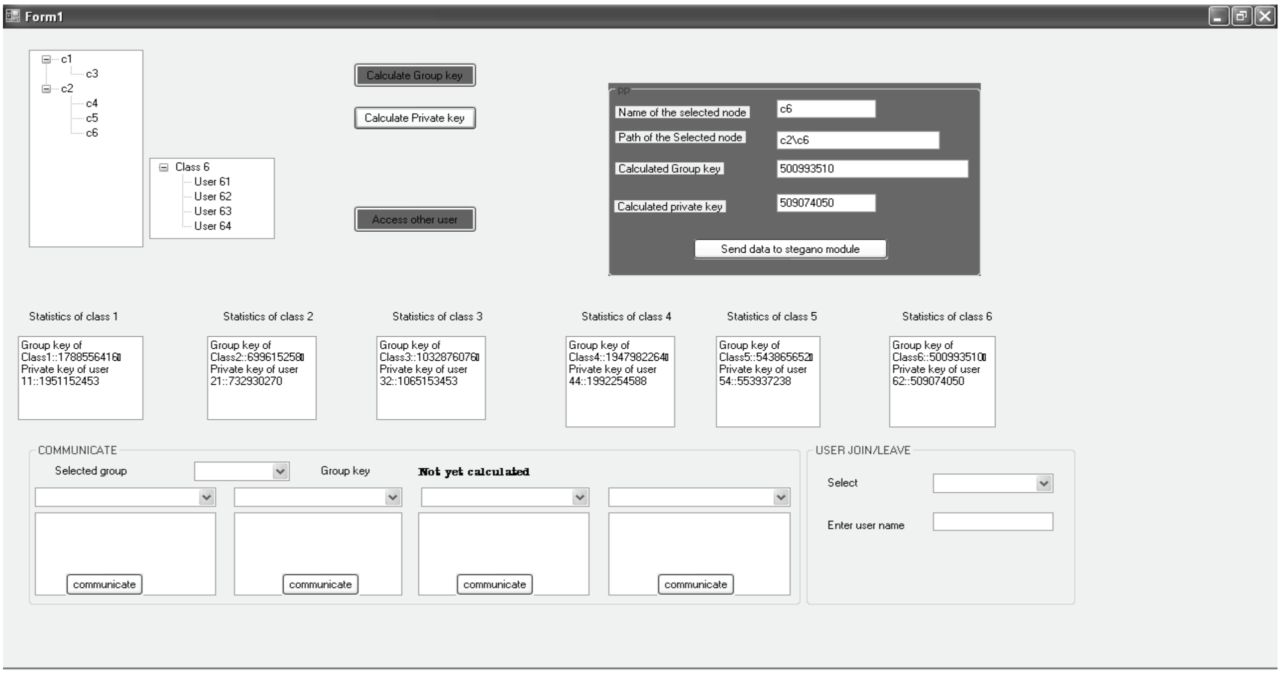
Calculated private key: 1951152453

Group key of Class1::1788556416
Private key of user 11::1951152453

Key calculated successfully

Not yet calculated

Figure 6. Users and classes.



Name of the selected node: c6

Path of the Selected node: c2\c6

Calculated Group key: 500993510

Calculated private key: 509074050

Statistics of class 1: Group key of Class1::1788556416 Private key of user 11::1951152453

Statistics of class 2: Group key of Class2::699615258 Private key of user 21::732930270

Statistics of class 3: Group key of Class3::1032876076 Private key of user 32::1065153453

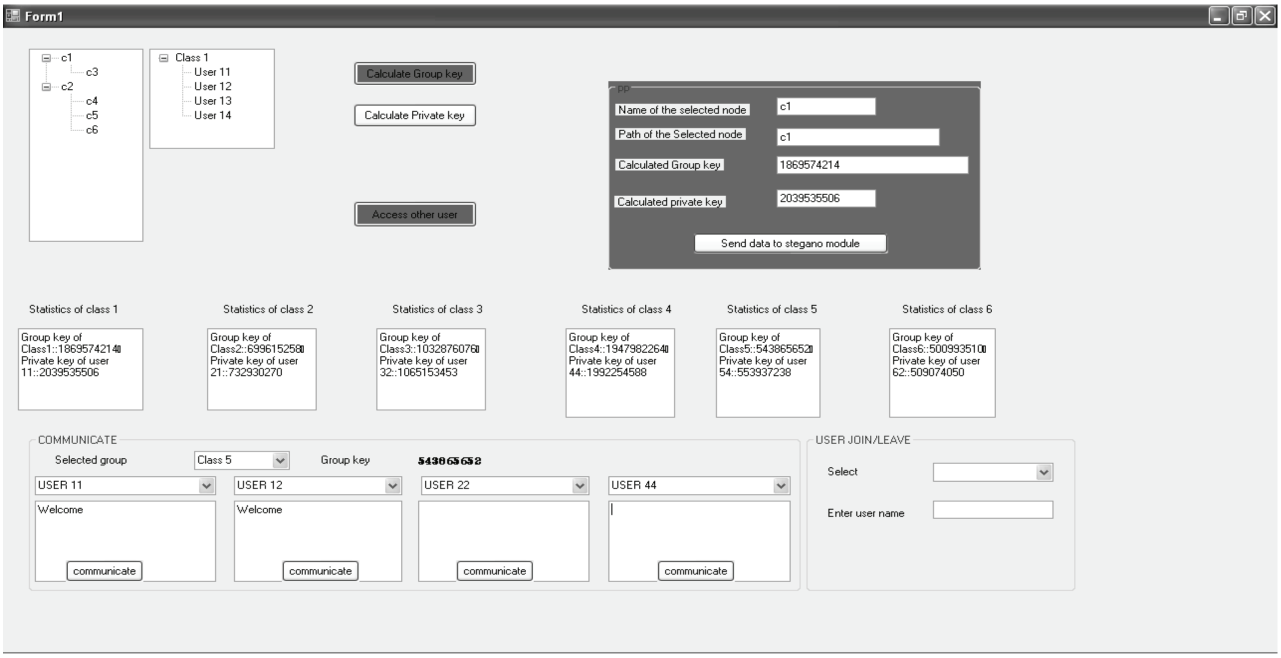
Statistics of class 4: Group key of Class4::1947982264 Private key of user 44::1992254588

Statistics of class 5: Group key of Class5::543865652 Private key of user 54::553937238

Statistics of class 6: Group key of Class6::500993510 Private key of user 62::509074050

Selected group: Group key: **Not yet calculated**

Figure 7. Key calculation for all classes.



Name of the selected node: c1

Path of the Selected node: c1

Calculated Group key: 1869574214

Calculated private key: 2039535506

Statistics of class 1: Group key of Class1::1869574214 Private key of user 11::2039535506

Statistics of class 2: Group key of Class2::699615258 Private key of user 21::732930270

Statistics of class 3: Group key of Class3::1032876076 Private key of user 32::1065153453

Statistics of class 4: Group key of Class4::1947982264 Private key of user 44::1992254588

Statistics of class 5: Group key of Class5::543865652 Private key of user 54::553937238

Statistics of class 6: Group key of Class6::500993510 Private key of user 62::509074050

Selected group: Class 5 Group key: **543865652**

USER 11: Welcome

USER 12: Welcome

USER 22

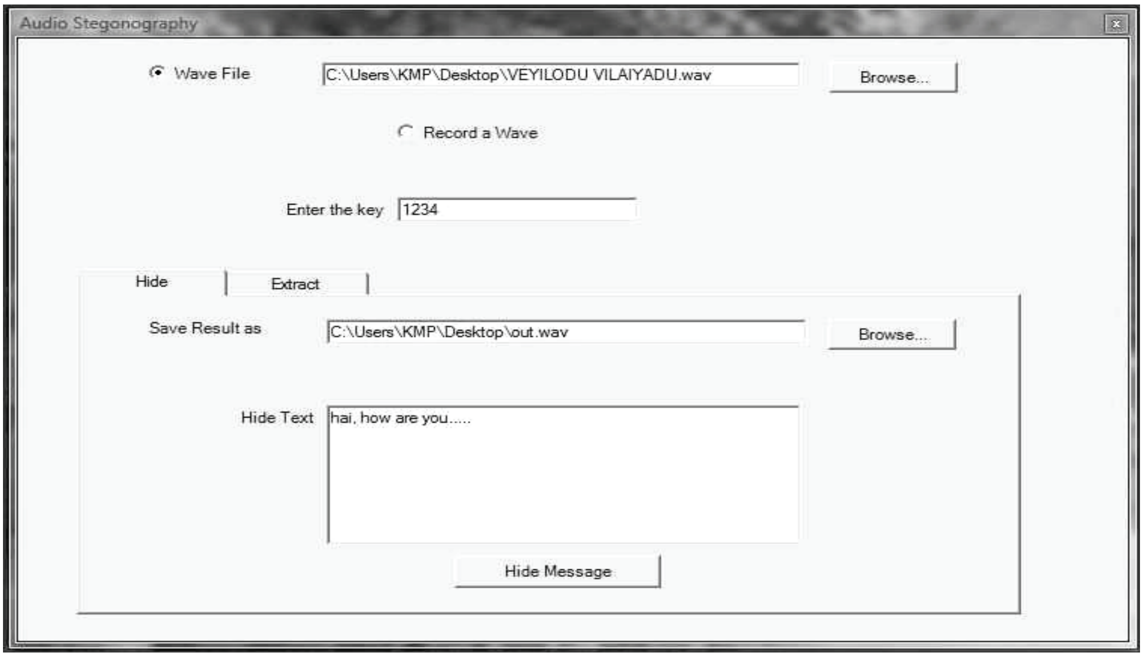
USER 44

Figure 8. Communication among same class.

polynomial and hence very less memory is used. All parameters are publicly available and using the same method keys of lower hierarchy can be derived by substituting the corresponding parameters as given by the derivation module. The steganography module does not involve any storage for storing the already recorded data as always data is recorded and subsequently sent to the receivers. The size of the cover medium does not increase because only two bits are used in each channel.

8.2. Computation Cost

When the user joins there is no need for recalculation because the recorded message has already been played. When a user leaves a group key is recalculated and given to the class. Private keys are generated from this. The


Audio Stegonography

● Wave File C:\Users\KMP\Desktop\VEYILODU VILAIYADU.wav

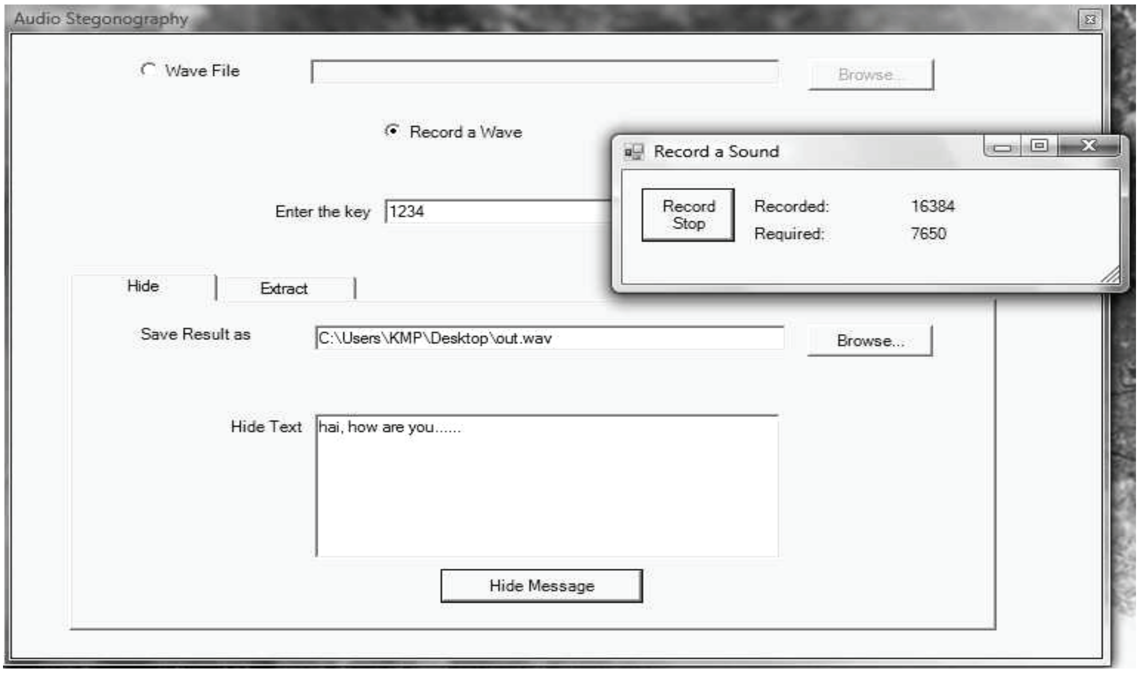
○ Record a Wave

Enter the key 1234

Save Result as C:\Users\KMP\Desktop\out.wav

Hide Text hai, how are you.....

Figure 9. Hiding the message.


Audio Stegonography

○ Wave File

● Record a Wave

Enter the key 1234

Record a Sound

Recorded: 16384

Required: 7650

Save Result as C:\Users\KMP\Desktop\out.wav

Hide Text hai, how are you......

Figure 10. Recording the message.

value of the new key involves not a change of parameters but a change of mod *P* value. Hence each class will be able to get a new polynomial value by passing the same parameters. Any left user will not be able to get the key. Only one key is used during creation of stego data. The higher class users need not remember the keys of all their descendant classes but rather using a simple scheme derive the exact parameters to generate the key. Hence the computation cost is reduced.

8.3. Communication Cost

The P value is changed by the Trusted Authority and when the users try to calculate the key the new key will be generated. The computation cost is reduced because, the class users are not bothered about the key transmission. Once the polynomial is given the users can calculate their own key.

8.4. Dynamics

Class joining, Class leaving, Dynamic Hierarchy and New user joining a class are all done by the trusted authority in a phased manner thereby allowing the scheme to scale to greater hierarchy. Additionally local messaging and service messages are taken care in this system.

8.5. No of Rekeys

To calculate the no of changes might be made on the user's key based on user join/leave. Key must be changed when a new user is being joined/left from the hierarchy In the Hierarchy "**Figure 11**" for analyzing the no. of rekeys there are 50 classes and Let every classes consists of four users each, then we can calculate an unique key for every class based on this group key, we can generate a private key for every single user.

Classes:
Totally fifty classes of eight levels

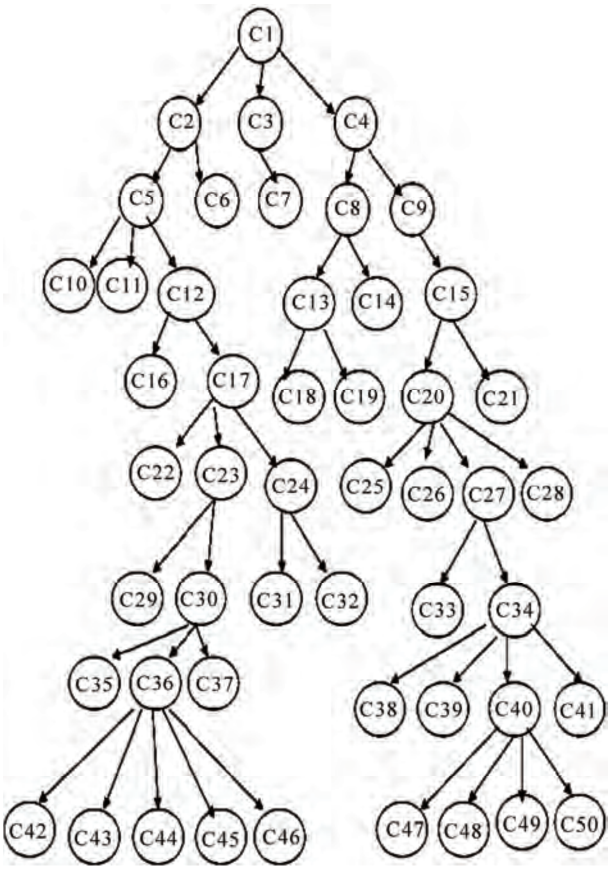


Figure 11. Hierarchy analyzed for calculating the no. of rekeys.

Users:
Every class is consists of 4 users
User joins:
Every time if a single user wants to join a group the CA just allows the user to be added to the hierarchy and generates a private for that user by providing the corresponding group key. The secrecy is maintained as the audio is real time.
User leave:
When a user wants to leave from the hierarchy the CA change the group key by making changes on anyone of the following Changing the polynomial, or by Changing the value factor P. The total no of key changes to be made = No. of Ancestor classes + 1 as shown in **Table 1**.

9. Conclusions and Future Work

9.1. Conclusions

Thus in this Paper, Design And Implementation Of Multilevel Access Control In Synchronized Audio To Audio Steganography Using Symmetric Polynomial Scheme is implemented successfully .The implementation results show that any type of hierarchy can be introduced and all dynamics can be done. For a 8 parameter polynomial that can have use 8 among the 16 values there are 16! combinations. Also, nodes can derive their descendants' keys without involvement of the CA once polynomial functions were distributed to them. In addition, the storage requirement and computation complexity at the CA are almost same as that at individual nodes, thus, the CA would not be a performance bottleneck and can deal with dynamic operations efficiently.

Table 1. No of rekeys for user leaving from a corresponding class.

Class 1: 1	Class 11:4	Class 21:5	Class 31:7	Class 41:8
Class 2:2	Class 12:4	Class 22:6	Class 32:7	Class 42:9
Class 3:2	Class 13:4	Class 23:6	Class 33:7	Class 43:9
Class 4:2	Class 14:4	Class 24:6	Class 34:7	Class 44:9
Class 5:3	Class 15:4	Class 25:6	Class 35:8	Class 45:9
Class 6:3	Class 16:5	Class 26:6	Class 36:8	Class 46:9
Class 7:3	Class 17:5	Class 27:6	Class 37:8	Class 47:9
Class 8:3	Class 18:5	Class 28:6	Class 38:8	Class 48:9
Class 9:3	Class 19:5	Class 29:7	Class 39:8	Class 49:9
Class 10:4	Class 20:5	Class 30:7	Class 40:8	Class 50:9

9.2. Future Work

1) The Trusted authority can still made secure by changing the value of the parameters.

2) A symmetric polynomial can be changed by the trusted authority.

3) Bit selection for steganography can be made by using some pseudo random generator.

10. References

[1] W. Stallings, Ed., "Network and Internetworking Security," Pearson Education Asia, Singapore, 2001.

[2] N. F. Johnson, Z. Duric and S. Jajodia, "Information Hiding Steganography and Watermarking-Attacks and Countermeasures," Kluwer Academic Publishers, Boston, 2001.

[3] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information Hiding - A Survey," *Proceedings of IEEE*, Vol. 87, No. 7, 1999, pp. 1062-1078.

[4] M. Hosei, "Acoustic Data Hiding Method Using Sub-Band Phase Shifting," *Technical Report of IEICE*, *EA*, Vol. 106, No. 205, 2006, pp. 7-11.

[5] M. Wu and B. D. Liu, "Multimedia Data Hiding," Springer-Verlag, New York, 2003.

[6] T. Aoki and N. Homma, "A Band Widening Technique for VoIP Speech Using Steganography Technology," *Report of IEICE*, *SP*, Vol. 106, No. 333, 2006, pp. 31-36.

[7] X. P. Huang, R. Kawashima, N. Segawa and Y. Abe, "The Real-Time Steganography Based on Audio-to-Audio Data Bit Stream," *Technical Report of IEICE, ISEC*, Vol. 106, No. 235, September 2006, pp. 15-22.

[8] X. P. Huang, R. Kawashima, N. Segawa and Y. Abe, "Design and Implementation of Synchronized Audio-to-Audio Steganography Scheme," *IEEE Explore*, 2008, pp. 331-334.

[9] S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," *ACM Transactions on Computer Systems*, Vol. 1, No. 3, March 1983, pp. 239-247.

[10] S. J. MacKinnon, P. D. Taylor, H. Meijer and S. G. Akl, "An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy," *IEEE Transactions on Computers*, Vol. 34, No. 9, September 1985, pp. 797-802.

[11] S. Chen, Y.-F. Chung and C.-S. Tian, "A Novel Key Management Scheme for Dynamic Access Control in a User Hierarchy," *COMPSAC*, September 2004, pp. 396-397.

[12] I. Ray, I. Ray and N. Narasimhamurthi, "A Cryptographic Solution to Implement Access Control in a Hierarchy and More," *SACMAT*'02: *Proceedings of the* 7*th ACM Symposium on Access Control Models and Technologies*, ACM Press, New York, 2002, pp. 65-73.

[13] R. S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," *Information Processing Letter*, Vol. 27, Vol. 2, February 1988, pp. 95-98.

[14] G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys," *Proceedings on Advances in Cryptology*: *CRYPTO*'89, *LNCS*, Vol. 435, 1989, pp. 316-322.

[15] M. L. Das, A. Saxena, V. P. Gulati and D. B. Phatak, "Hierarchical Key Management Scheme Using Polynomial Interpolation," *SIGOPS Operating Systems Review*, Vol. 39, No. 1, January 2005, pp. 40-47.

[16] L. Harn and H. Y. Lin, "A Cryptographic Key Generation Scheme for Multilevel Data Security," *Computers and Security*, Vol. 9, No. 6, October 1990, pp. 539-546.

[17] V. R. L. Shen and T.-S. Chen, "A Novel Key Management Scheme Based on Discrete Logarithms and Polynomial Interpolations," *Computers and Security*, Vol. 21, No. 2, March 2002, pp. 164-171.

[18] M.-S. Hwang, C.-H. Liu and J.-W. Lo, "An Efficient Key Assignment for Access Control in Large Partially Ordered Hierarchy," *Journal of Systems and Software*, February 2004.

[19] C. H. Lin, "Dynamic Key Management Scheme for Access Control in a Hierarchy," *Computer Communications*, Vol. 20, No. 15, December 1997, pp. 1381-1385.

[20] S. Zhong, "A Practical Key Management Scheme for Access Control in a User Hierarchy," *Computers and Security*, Vol. 21, No. 8, November 2002, pp. 750-759.

[21] X. Zou, B. Ramamurthy and S. Magliveras, "Chinese Remainder Theorem Based Hierarchical Access Control for Secure Group Communications," *Lecture Notes in Computer Science* (*LNCS*), Vol. 2229, November 2001, pp. 381-385.

[22] X. Zou, B. Ramamurthy and S. S. Magliveras, Eds., "Secure Group Communications over Data Networks," Springer, New York, October 2004.

Journal of Information Security, **2010, 1, 41-44**
doi:10.4236/jis.2010.11005 Published Online July 2010 (http://www.SciRP.org/journal/jis)

Game Theory Based Network Security

Yi Luo[1], Ferenc Szidarovszky[1], Youssif Al-Nashif[2], Salim Hariri[2]
[1]*Department of Systems and Industrial Engineering, University of Arizona, Tucson, USA*
[2]*Department of Electrical and Computer Engineering, University of Arizona, Tucson, USA*
E-mail: *Luo1@email.arizona.edu, szidar@sie.arizona.edu, {alnashif, hariri}@ece.arizona.edu*
Received March 9, 2010; *revised July* 9, 2010; *accepted July* 20, 2010

Abstract

The interactions between attackers and network administrator are modeled as a non-cooperative non-zero-sum dynamic game with incomplete information, which considers the uncertainty and the special properties of multi-stage attacks. The model is a *Fictitious Play* approach along a special game tree when the attacker is the leader and the administrator is the follower. Multi-objective optimization methodology is used to predict the attacker's best actions at each decision node. The administrator also keeps tracking the attacker's actions and updates his knowledge on the attacker's behavior and objectives after each detected attack, and uses it to update the prediction of the attacker's future actions. Instead of searching the entire game tree, appropriate time horizons are dynamically determined to reduce the size of the game tree, leading to a new, fast, adaptive learning algorithm. Numerical experiments show that our algorithm has a significant reduction in the damage of the network and it is also more efficient than other existing algorithms.

Keywords: Multi-Stage Attack, Dynamic Game, Multi-Objective Optimization, Adaptive Learning

1. Introduction

The increased dependence on networked applications and services makes network security an important research problem. Detection of intrusions and the protection of the networks against attacks is the central issue. Game theory is an appropriate methodology to model the interactions between attackers and network administrator and to determine the best countermeasure strategy against attacks. There are however some difficulties in directly applying classical game theory, since the attackers' strategies are uncertain, their steps are not instantaneous, the rules of the games might change in time, and so on. Therefore any game theory based methodology has to take these difficulties into account.

There are many types of intrusions. Multi-stage attacks are the most destructive and most difficult kinds for any defense system. They use intelligence to strategically compromise the targets in a planned sequence of actions, so the usual methodology designed to protect against single-stage attacks cannot be used.

Network intrusion response mechanisms have been intensively developed and studied in recent years. Several authors used Markov Games (MG) as a model and methodology. Lye and Wing [1] viewed the interactions between the attacker and the administrator as a two-player Markov game and modeled it by an intrusion graph. The recovery effort was considered as the cost of the response. The payoff for the attacker was defined by the amount of effort the administrator needed to spend in order to bring the network back to normal state. The equilibrium was obtained by using nonlinear programming and dynamic programming for infinite and finite horizon games, respectively. The main disadvantage of this approach is the huge size of the state space which makes extremely difficult to compute the equilibria. Shen *et al.* [2] used a piecewise linearized Markov game model with estimated beliefs of the possible cyber attack patterns obtained by data fusion and adaptive control. They also recognized that larger time-step horizons result in increased computation complexity. Another approach is based on Partially Observable Markov Decision Processes (POMDP) which results in more complex computation problems. Carin *et al.* [3] introduced the protection map and used reverse-engineering methodology to build an attack graph. Zhang and Ho [4] presented a model to characterize multi-stage collusive attacks in terms of key spatio-temporal properties. The attacker's behavior was modeled as a reward-directed partially observable Markov

The research presented in this paper is supported in part by National Science Foundation via grants numbers CNS-0615323 and IIP-0758579, and by Electronic Warfare Associates, Inc. via the grant number ewagsi-07-LS-0002, and it is conducted as part of the NSF Center for Autonomic Computing at University of Arizona.

decision process and the administrator was assisted by identifying the potential causal relationships between the different system vulnerabilities. This approach also suffered from serious computation difficulties because of the very large state space. Liu *et al.* [5] introduced a dynamic game approach based on modeling the Attacker Intent, Objectives and Strategies (AIOS) which resulted in a much smaller state space, so this approach is more efficient than the application of Markovian games. A similar concept was applied in Siever *et al.* [6] for the security of electric power transmission grids, when the goals of the attacker were used in formulating the attacker's game, which optimizes the difference of its reward and the amount of power delivered. The objective function of the defender is the sum of the amount of delivered power and a special reward function. The approach introduced by Luo *et al.* [7] was based on POMDP, however a reduced special game tree structure was used and a new stochastic multi-stage defense algorithm was developed.

All models and algorithms are based on assessing all damages and costs of the cyber attacks. The uncertainty in the knowledge of the network, its vulnerabilities, possible actions and counteractions, damages and costs, etc. make the mathematical modeling more complicated. In the economic literature this issue has been known and treated by the deterministic equivalent, which is a linear combination of the expected value (μ) and variance (σ^2) of a random outcome: $\mu - \alpha\sigma^2$, where α shows the level of willingness of the decision maker to take risk.

Almost all models of multi-stage attacks are based on special game trees. It is well-known from the game theory literature (see for example, Forgo *et al.*, [8]) that such games with full information always have at least one Nash equilibrium, which can be computed by using backward induction. This general result however cannot be used in computer network security, since the game tree and the possible strategies of the players are not completely known by all participants. The administrator and the attacker might believe in different game trees with different possible actions.

2. Consequence Modeling

We have adopted the approach given in Richardson and Chavez [9]. The consequence of any attack and any action during a multi-stage attack is based on the following six steps:

1) Define the categories of impact;

2) State the importance of each category relative to the others;

3) Define the measures of impact for each category;

4) Define the relationships between physical effects and impact measures;

5) Define the system and its users;

6) Define the events in terms of scales and network system impact.

Impact categories include and not restricted to economic, image, safety, security, intelligence, and privacy concerns. Their relative importance factors can be assessed by any one of the well known procedures from multi-criteria decision making (see for example, Szidarovszky *et al.*, [10]). A common approach of obtaining the weights is based on pair-wise comparisons, when all participants in the decision making process are asked to give relative importance factors for all pairs of categories. Then the results are summarized into a final set of importance weights either by averaging them or by using the Analytic Hierarchy Process (AHP). The measures of the impact in different categories are usually given in different units, and they can be combined by using multi-attribute utility theory or weighting method with normalized evaluations. Performance measures can be defined for the impact categories, and each performance measure can be divided into a set of constructed scales representing the amount of impact the physical consequences have on the network and its users including lost revenue, repair and/or replacement cost, damage by lost or stolen information, etc. Any actual attack has impacts on different categories with different levels. Using the consequence modeling tool, the overall consequence of the different types and scales of events on the system and its users can be assessed into one combined value. This value has to be computed at all states of the multistage attack and will be used in the game tree analysis.

3. Game Tree and Decision Nodes

Multi-stage attacks are represented by special game trees. **Figure 1** shows the first two interactions on a game tree. The attacker is the leader, the administrator is the follower. The root of the tree is the initial decision node of the attacker, and the possible initial moves of the attacker are represented by the arcs originating at the root. These actions might include attacking the server with different intensity levels, sending a virus to a group of customers, etc. At the end point of each arc the administrator has to

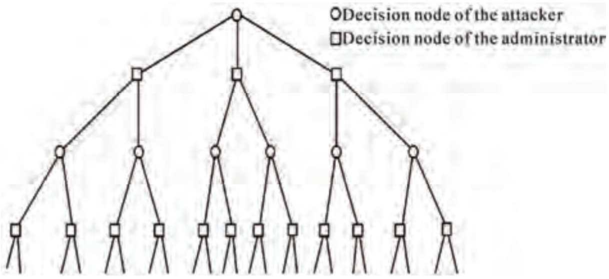

○Decision node of the attacker
□Decision node of the administrator

Figure 1. Special game tree.

respond, so they are its decision nodes. After the administrator's response the attacker makes the next move, and so on. This tree continuous until the intruder gives up the attack or reaches its goals. This tree can become very large and the payoff values at the decision nodes are uncertain, therefore the classic method, known as backward induction, cannot be used in this case.

4. Determining Optimal Responses

The algorithm to be described in this section is an on-line procedure, it provides the best response of the administrator at each of its decision nodes, when a multi-stage attack reaches that particular node of the game tree. So during an attack the algorithm can be used at each stage to find the best next move of the administrator starting just after the first action of the attacker and continuing until the end of the game.

Consider now a particular decision node of the administrator and the sub-game tree having this node as its root. The time horizon for this sub-game tree is obtained as follows. We have to check all end points of this sub-game tree where the attacker reaches its goals during smallest number of steps, so we select the shortest path with smallest number of arcs from the root to such end points. The length of the shortest path is the time horizon, and then all paths starting at the root will be considered only until this time horizon. The utility function of the attacker is then assessed at all endpoints of this truncated sub-game tree. The utility function is the linear combination of the expected payoff of the attacker and its variance as it was explained earlier. The risk taking coefficient of the attacker can be updated after each attack, since the administrator has estimates of the expectations and the variances of its utility values for all possible moves and also observes the actual move. The administrator then has to assess the probability distributions of the attacker's actions at all of its decision nodes. The probability values are computed based on the assessed utility values of the intruder as well as previous interactions with the attacker. First the probability values are computed proportionally to the utility values at the endpoints of the different arcs representing the next moves of the attacker, and if previous interactions provide relative frequencies, they are averaged with the computed probabilities. Using these probability distributions the expectation and the variance of the cumulative impact up to the time horizon for the administrator can be computed for each of its possible responses, and the corresponding utility values are obtained by combining expectations and variances with the risk acceptance coefficient of the administrator. The best response of the administrator is the arc which has its highest utility value.

The attacker makes the first move. At the end point of the corresponding arc the administrator has to respond. Using the above procedure the administrator finds its response. Then the attacker makes the next step, and the best next response of the administrator is obtained again by using the same algorithm with updated data based on the information obtained from the previous actions of the attacker. Then the attacker has the next move, the administrator responds by using the same algorithm, and so on until the game ends, which occurs when the attacker stops attacking by reaching its goals or giving up.

5. Numerical Example

Figure 2 shows a network structure. It is assumed that the HTTP server, Database 2, the FTP1 server and the information in the CEO are the vulnerable components in the network system, and access to the information in the CEO is the attacker's objective. It is also assumed that the CEO needs services provided by the HTTP server, Database 2 and the FTP1 server to do its jobs. The attacker can launch multi-stage attacks to obtain the information from the CEO in many different ways. Then the administrator can respond to it by selecting from a set of options, and so on, which leads to the game tree.

Next we assume that in addition to the sensitive data in the CEO the data in the Accounting is another vulnerability of the system. The attacker has now two objectives: the information in CEO and the data in Accounting. The Accounting also needs services provided by Database 2 and the HTTP server, etc. Our computer study assumed that the attacker always selects the action leading to maximal impact, and the administrator always selects its best action at its decision nodes by using one of the three tested algorithms.

We applied three methods to find the best responses of the administrator: One is a greedy algorithm (GA) in which the administrator completely blocks the traffic of

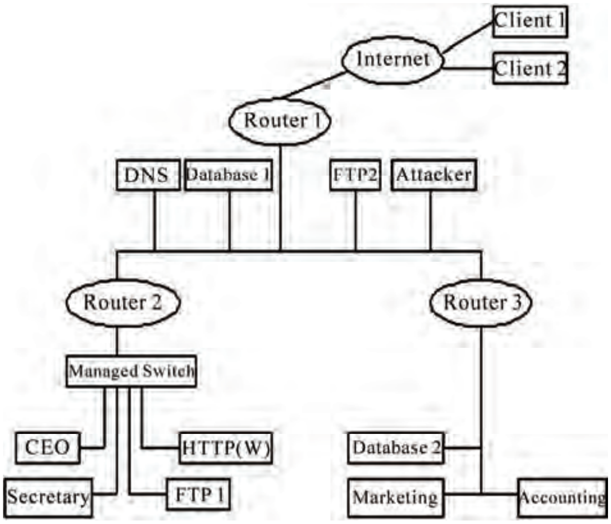


Figure 2. Network structure.

Table 1. The performances of the three algorithms.

The total losses of the system occurred during the life-cycle of the multi-stage attacks			Administrator		
			GA Algorithm	SO Algorithm	Our Algorithm
Attacker	Risk Seeking	Single Objective	4,694	3,608	2,781
		Two Objective	9,597	6,741	4,689
	Risk Neutral	Single Objective	4,694	3,176	2,252
		Two Objective	9,597	6,325	4,021

corresponding services on routers, firewall, or disconnect the machines using managed switches, etc. regardless of what kind of attack occurs or what is the intensity levels of the attack. Another algorithm is also myopic, single-interaction optimization algorithm (SO) in which the administrator tries to minimize the loss from the most current attack at each interaction without considering future interactions with the attacker. The third algorithm is the one we developed. The results are shown **Table 1**. Two types of attacks were assumed. The risk seeking attacker worried about only the expectation of the impact ($\alpha = 0$), while the risk neutral intruder selected a relatively high risk taking coefficient ($\alpha = 1$). The two scenarios refer to the cases of one or two objectives of the attacker. The last three columns indicate the three methods which were used for comparison. The numbers in the last three columns of the table show the total losses of the system with using different methods. Clearly our method resulted in the smallest overall losses in all cases, where the loss reduction was 41%, 51%, 52% and 58% in comparison to the Greedy Algorithm, and 23%, 30%, 29% and 36% in comparison to single-interaction optimization. In assessing the numerical values of the impacts in the consequence analysis, we used only economic impact. A more complex consequence analysis would not alter the main steps of the algorithms.

6. Conclusions

This paper introduced a multi-stage intrusion defense system, where the interactions between the attacker and the administrator are modeled as a two-player non-cooperative non-zero-sum dynamic game with incomplete information. The two players conduct *Fictitious Play* along the game tree, which can help the administrator to find quickly the best strategies to defend against attacks launched by different types of attackers. Our algorithm is an online procedure, which gives the most appropriate response of the administrator at any stage of the game. So it has to be repeated at all actual decision nodes of the administrator. Our algorithm is different than the usual methods based on decision trees, since at each step only a finite horizon is considered, instead of expected outcomes certain equivalents are used and the probabilities of the different arcs are continuously updated based on new information. In our numerical example our approach was compared to two other algorithms and the total network losses were compared. The loss reduction by using our approach varied between 23% and 58%. The performance of our algorithm is much better than that of other algorithms based on the results of our numerical experiments.

7. References

[1] K. Lye and J. Wing, “Game Strategies in Network Security,” *International Journal of Information Security,* Vol. 4, No. 1-2, 2005, pp. 71-86.

[2] D. Shen, G. Chen, E. Blasch and G. Tadda, “Adaptive Markov Game Theoretic Data Fusion Approach for Cyber Network Defense,” *IEEE Military Communications Conference* (*MILCOM* 2007), Orlando, 2007.

[3] L. Carin, G. Cybenko and J. Hughes, “Cybersecurity Strategies: The QuERIES Methoddology,” *Computer*, Vol. 41, No. 8, 2008, pp. 20-26.

[4] Z. Zhang and P. Ho, “Janus: A Dual-Purpose Analytical Model for Understanding, Characterizing and Countermining Multi-Stage Collusive Attacks in Enterprise Networks,” *Journal of Network and Computer Applications*, Vol. 32, No. 3, 2009, pp. 710-720.

[5] P. Liu and W. Zang, “Incentive-Based Modeling and Inference of Attack Intent, Objectives, and Strategies,” *CCS*’03, Washington, DC., 2003.

[6] W. M. Siever, A. Miller and D. R. Tauritz, “Blueprint for Iteratively Hardening Power Grids Employing Unified Power Flow Controllers,” *SoSE*’07, *IEEE International Conference on System of Systems Engineering*, Tampa, 2007.

[7] Y. Luo, F. Szidarovszky, Y. Al-Nashif and S. Hariri, “Game Tree Based Partially Observable Stochastic Game Model for Intrusion Defense Systems (IDS),” *IIE Annual Conference and EXPO* (*IERC* 2009), Miami, 2009.

[8] F. Forgo, J. Szep and F. Szidarovszky, “Introduction to the Theory of Games,” Kluwer Academic Publishers, Dordrecht, 1999.

[9] B. T. Richardson and L. Chavez, “National SCADA Test Bed Consequence Modeling Tool,” *Sandia National Laboratory Report, SAND*2008-6098, Albuquerque, 2008.

[10] F. Szidarovszky, M. Gershon and L. Duckstein, “Techniques for Multiobjective Decision Making in Systems Management,” Elsevier, Amsterdam, 1986.

Call for Papers

ISSN: 2153-1234 Vol.1 No.1 July 2010
Scientific Research
Journal of Information Security
ISSN: 2153-1234
www.scirp.org/journal/jis

Journal of Information Security

ISSN 2153-1234 (Print) ISSN 2153-1242 (Online)
http://www.scirp.org/journal/jis

JIS, a quarterly journal, publishes research and review articles in all important aspects of information security. Both experimental and theoretical papers are acceptable provided they report important findings, novel insights, or useful techniques in these areas.

Subject Coverage

JIS aims to provide a platform for scientists and academicians all over the world to promote, share, and discuss various new issues and developments in different areas of information security. All manuscripts submitted to JIS must be previously unpublished and may not be considered for publication elsewhere at any time during JIS's review period. Additionally, accepted ones will immediately appear online followed by printed in hard copy. The topics to be covered by Journal of Information Security include, but are not limited to:

- Access Control and Anonymity
- Anti-Virus and Anti-Worms
- Authentication and Authorization
- Biometric Security
- Data and System Integrity
- Database Security
- Distributed Systems Security
- Electronic Commerce Security
- Fraud Control
- Grid Security
- Information Hiding and Watermarking
- Intellectual Property Protection
- Intrusion Detection
- Key Management and Key Recovery
- Language-Based Security
- Network Security
- Operating System Security
- Security Models

We are also interested in short papers (letters) that clearly address a specific problem, and short survey or position papers that sketch the results or problems on a specific topic. Authors of selected short papers would be invited to write a regular paper on the same topic for future issues of the ***JIS***.

Notes for Intending Authors

Submitted papers should not have been previously published nor be currently under consideration for publication elsewhere. Paper submission will be handled electronically through the website. All papers are refereed through a peer review process. For more details about the submissions, please access the website.

Website and E-Mail

http://www.scirp.org/journal/jis **E-mail: jis@scirp.org**

TABLE OF CONTENTS

Volume 1 Number 1 **July 2010**