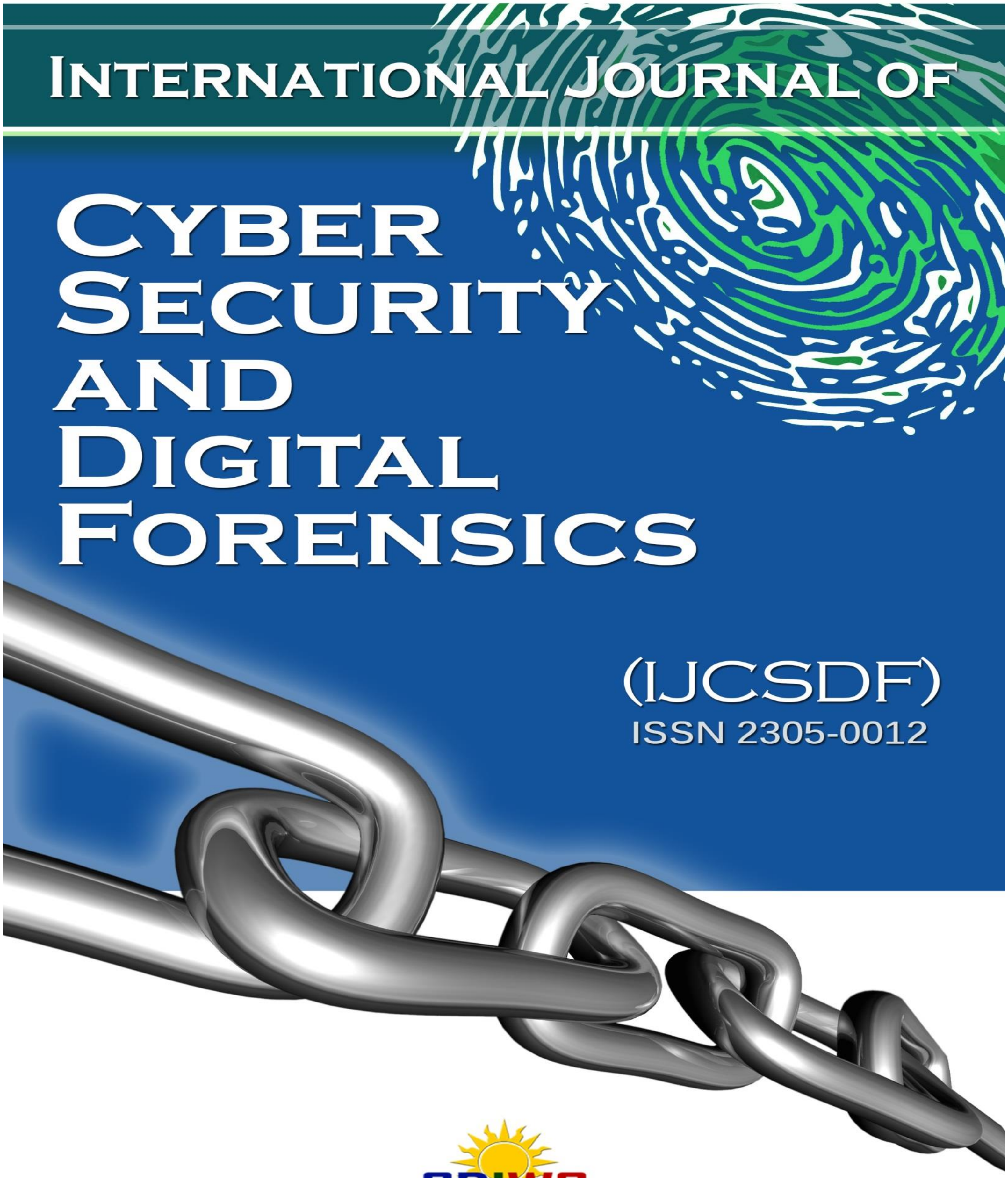
(IJCSDF) Vol. 4, No. 2

International Journal of Cyber Security and Digital Forensics

(IJCSDF)

ISSN 2305-0012



SDIWC

www.sdiwc.net

ISSN: 2305-0012 (online)

CONTENTS

ORIGINAL ARTICLES

International Journal of Cyber-Security and Digital Forensics (IJCSDF) 4(2): 327-338
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2305-0012)

Development and Evaluation of a Continuity Operation Plan Support System for an Information Technology System

Ichiro Matsunaga,
Tokyo Denki University, Japan
5 Senjyu-Asahicho, Adachi-ku, Tokyo 120-8551, Japan
matsunaga@isl.im.dendai.ac.jp

Ryoichi Sasaki,
Tokyo Denki University, Japan
5 Senjyu-Asahicho, Adachi-ku, Tokyo 120-8551, Japan
sasaki@im.dendai.ac.jp

ABSTRACT

Because organizations need to be able to deliver products and services at acceptable predefined levels, even in the wake of disruptive incidents, the number of enterprises implementing business continuity management (BCM) systems is increasing. However, during business continuity risk planning, difficulties are often encountered when attempting to determine the most appropriate BCM to implement. To solve this problem, we have developed a BCM implementation method for use in combination with event tree analysis (ETA), which is used to secure safety in engineering, and the program-evaluation-and-review-technique (PERT), which is used in operations research, and which also includes a function for showing analysis results in an easy-to-understand manner. We have also developed a system to support the above method and have confirmed the utility of the developed system by applying it to a small example.

KEYWORDS

Business continuity, Risk assessment, Information technology, ETA, PERT

1 INTRODUCTION

Because organizations need to be able to deliver products and services at acceptable predefined levels, even in the wake of disruptive incidents, the number of enterprises implementing business continuity management (BCM) is increasing recently [1] [2]. However, during business continuity risk planning, difficulties are often encountered when attempting to determine the most appropriate BCM measures to implement because the following characteristics need to be considered in order to determine the most optimal measures. Note that, hereinafter, the word “measure” refers to an action implemented to decrease a risk.

1. Measures need to consider both incident prevention and business continuity damage reduction should incidents occur.
2. There are numerous measures which are effective only in limited situations. The overall effect of a combination of measures will not be a simple summation of the effects of each measure.
3. Even if risk attenuation measures are taken, it may not be possible to completely remove risk.

To cope with 1, it is necessary to analyze changes in both the probability and downtime. To cope with 2, it is necessary to envision the various consequences that can result from an incident. To cope with 3, it is necessary to estimate the effects of each measure combination. Finally, to cope with 4, it is necessary to discuss the residual risks which remain after taking measures.

To deal with these various characteristics we developed a BCM implementation method for use in combination with event tree analysis (ETA) [3], [10], which is used in safety engineering, and the program-evaluation-and-review-technique (PERT) [4], which is used in operations research, and which also includes a function to show the analysis result in an easy-to-understand manner (Fig. 1).

The proposed method makes it possible to analyze proposed changes in both the probability and downtime. In addition, by displaying the residual risk visually, it is possible to cope with the fourth characteristic mentioned above. In this paper, we report on the development of a system designed to support the above method and the result of a small example performed to confirm its efficacy.

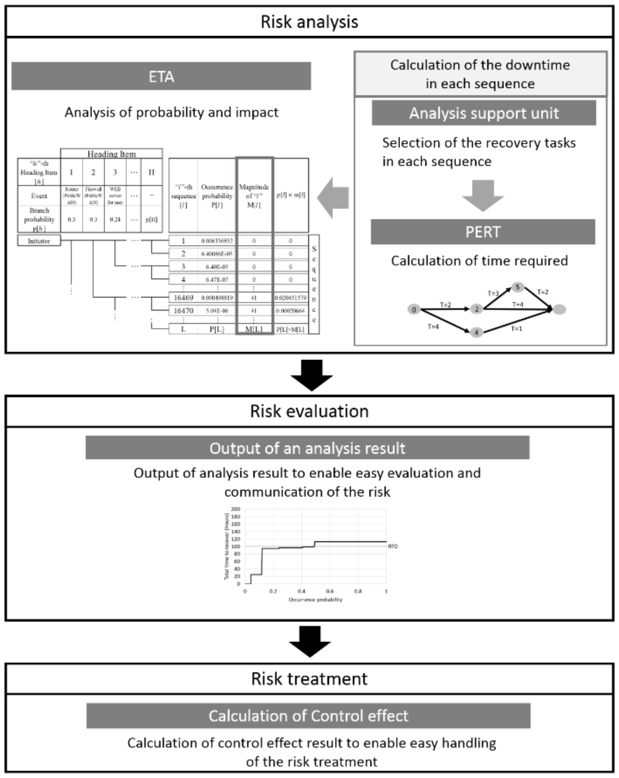

Risk analysis
ETA
Analysis of probability and impact
Calculation of the downtime in each sequence
Analysis support unit
Selection of the recovery tasks in each sequence
PERT
Calculation of time required
Risk evaluation
Output of an analysis result
Output of analysis result to enable easy evaluation and communication of the risk
Risk treatment
Calculation of Control effect
Calculation of control effect result to enable easy handling of the risk treatment

Figure 1. Outline of proposed method

2 RELATED WORKS

We found numerous studies related to the term “business continuity” by using the keyword search function of Google Scholar and the IEEE Xplore Digital Library, including the following:

(1) Studies on BCM frameworks. [5], [6]
(2) A study on methods for collecting business continuity information in order to implement a BCM framework [7]
(3) A study on effective measures in business continuity [8]
(4) A study on standards for prioritizing risks [9]

However, very few detailed studies have examined methods of determining appropriate incident response measures based on risk analysis. [5] In addition, no studies could be identified that examined the different response sequences which could result from initial incidents. Another novel aspect of our research is that it is the first risk analysis method to be used in combination with ETA and PERT.

3 APPLICATION PROCEDURES

For BCM, it is necessary to manage risks which can bring business to a standstill. As is noted in ISO22301 section 8.2.3 [2], this process can be carried out in compliance with ISO31000 [10]. Accordingly, our proposed method is applied in accordance with the risk-management process described in ISO31000. Use of the proposed method makes it easy to implement risk analysis, risk evaluation, and risk treatment. Figure 2 shows a flowchart of the application procedure in line with a risk management process that illustrates the scope of the proposed method. Explanations for the main processes are provided below:

1. Establishing context

The objective of this process is to determine, based on a business impact analysis, which critical systems have the highest priorities, the recovery time objective (RTO), and the minimum business continuity objective for an information system. The RTO is the elapsed time from the occurrence of an incident to the time when a product, service, and/or business activity is resumed, or when resources are recovered. In this paper, the RTO is used as the business continuity risk criteria.

2. Risk identification

The objective of this process is to identify risks by examining an event in relation to the critical system and the consequences that will result from such an event. Risks are not limited solely to natural disasters; they can also be the result of a device breakdown, wiring breaks, or some other such cause. Our proposed method uses these identified risks as ETA initiators.

3. Risk analysis

The objective of this process is to determine the magnitude level of a particular risk. This level is expressed in terms of its combined likelihood and consequences. The likelihood is defined as the probability of the event occurring. The use of the proposed method makes it easy to carry out this process.

4. Risk evaluation

The objective of this process is to determine whether the risk magnitude is acceptable. The proposed method makes it easy to see the result of a risk analysis.

5. Risk treatment

The objective of this process is to evaluate both the risk being treated and the measure being implemented in order to achieve the RTO. By calculating the effects of a measure from the risk analysis result, it is easy to determine the risk treatment required.

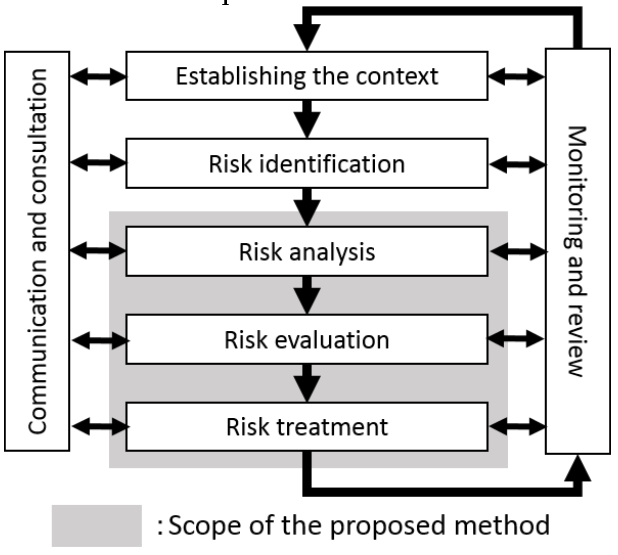


Figure 2. Risk management process

4 PROPOSED METHOD

We have developed a BCM implementation method for use in combination with ETA and PERT. This method can consider changes in both the probability and downtime, and confirm residual risk. The remainder of this section will be as follows: Sections 4.1 to 4.4 describe the analysis method used for determining risk magnitude; Section 4.5 describes the calculation method used to determine the effect of measures achieved by using the analysis result; Section 4.6 describes the method used to confirm the residual risk.

4.1 Risk value

Hereinafter, risk is expressed by multiplying risk probability by risk magnitude. In the proposed method, the risk value is calculated using the following equation:

$$\mathbf{R} = \mathbf{P} \times \mathbf{M} \tag{1}$$

R: Value of risk
P: Occurrence probability
M: Magnitude of damage

4.2 ETA

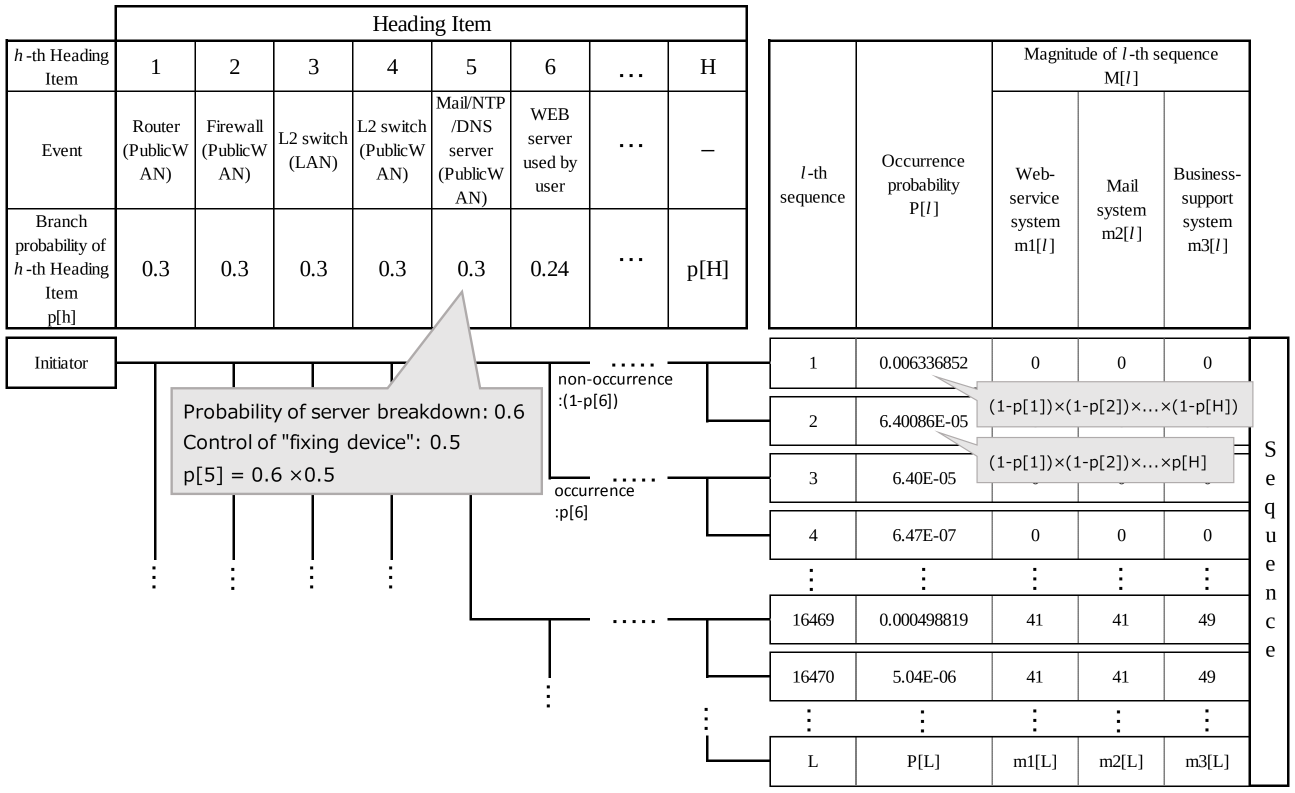


Figure 3. Event tree

An ETA can be used to cover all sequences and can also easily calculate occurrence probability by setting branch probability, which represents the occurrence probability of the heading item. To calculate a risk via ETA, it is necessary to determine the following information:

1. Heading item
2. Heading item probability
3. Damage magnitude in each sequence

Here, the heading item is a factor of a branching sequence (Fig. 3). In the proposed method, the heading item is whether a device has broken down.

Total risk can be calculated using ETA as follows:

$$\mathbf{R} = \sum_{l=1}^{\mathbf{L}} \mathbf{P}[l] \times \mathbf{M}[l] \quad (2)$$

$$\mathbf{P}[l] = \prod_{h=1}^{\mathbf{H}} \mathbf{P}'[h] \quad (3)$$

$$\mathbf{P}'[h] = ((1 - p[h])(1 - y[h]) + p[h] * y[h]) \quad (4)$$

$$y[h] = \begin{Bmatrix} 1: h - \text{th heading item unfolds in horizontal derection} \\ 0: h - \text{th heading item unfolds in vertical derection} \end{Bmatrix} \quad (5)$$

R:	Value of risk
L:	Number of sequences
l:	*l*-th sequence
P[*l*]:	Occurrence probability of *l*-th sequence
M[*l*]:	Magnitude of damage for *l*-th sequence
H:	Number of heading item
h:	*h*-th heading item
p[*h*]:	Branch probability of *h*-th heading item

4.3 PERT

PERT is a method used to analyze the time needed to complete the total project (Fig. 4). In the proposed method, the downtime of the *l*-th sequence calculated via PERT is introduced as a magnitude of damage for the *l*-th sequence. Here, the downtime, which also refers to the total recovery time for the *l*-th sequence of the ETA, can be calculated using equation 6.

$$\mathbf{M}[l] = \mathrm{PERT}(\mathrm{S}[l], \mathrm{D}[l][q] : q = 1,2,---, \mathrm{Q}_{[l]}) \quad (6)$$

M[*l*]:	Downtime (total time required for recovery) used in *l*-th sequence of ETA
PERT:	Function to calculate downtime using PERT chart [4]
S[*l*]:	Structure of PERT chart used in *l*-th sequence of ETA
D[*l*][*q*]:	Duration of *q*-th activity used in *l*-th sequence of ETA
Q[*l*]:	Number of activities in PERT chart used in the *l*-th sequence of ETA

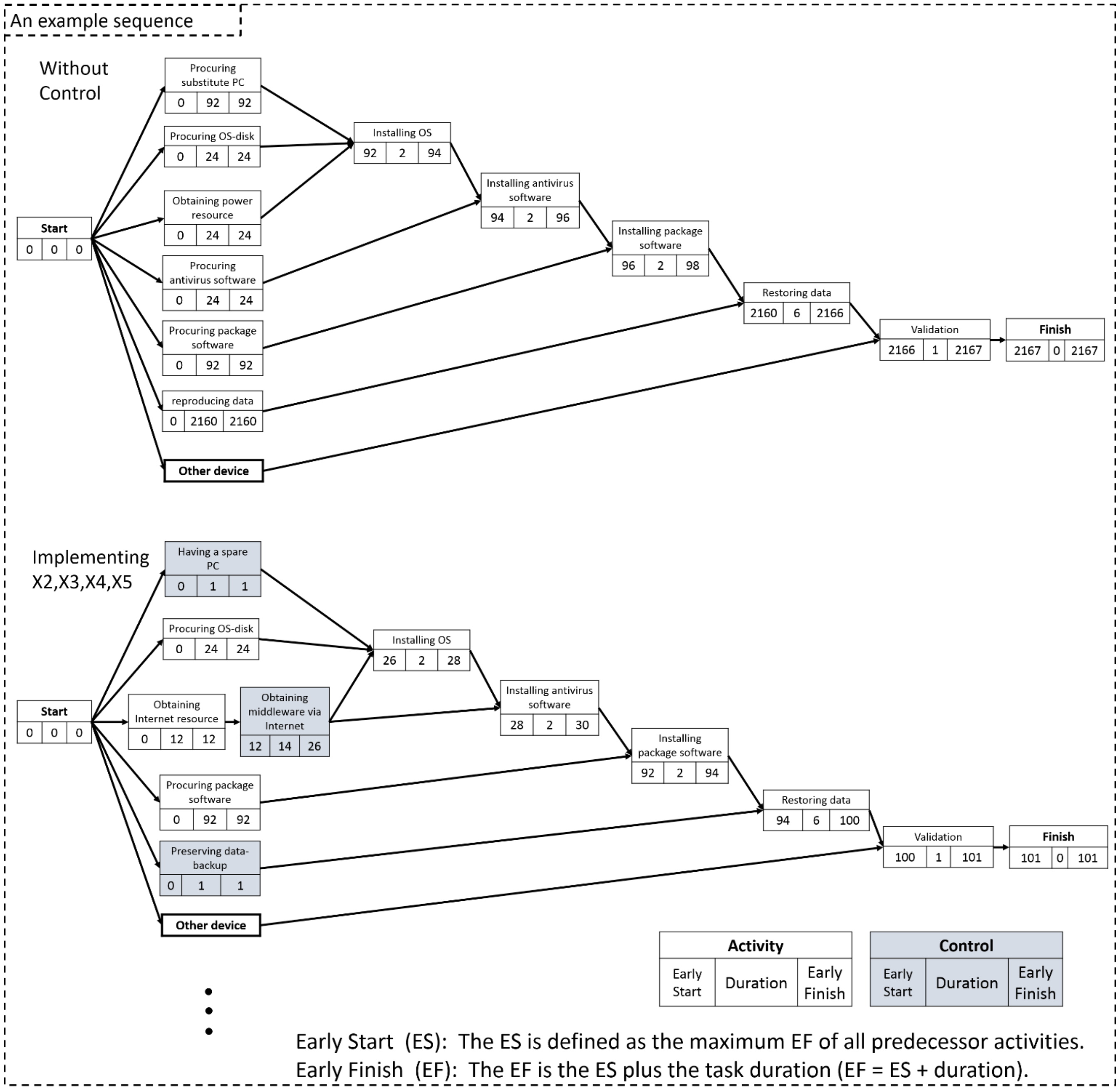

An example sequence
Without Control
Start 0 0 0
Procuring substitute PC 0 92 92
Procuring OS-disk 0 24 24
Obtaining power resource 0 24 24
Procuring antivirus software 0 24 24
Procuring package software 0 92 92
reproducing data 0 2160 2160
Other device
Installing OS 92 2 94
Installing antivirus software 94 2 96
Installing package software 96 2 98
Restoring data 2160 6 2166
Validation 2166 1 2167
Finish 2167 0 2167
Implementing X2,X3,X4,X5
Start 0 0 0
Having a spare PC 0 1 1
Procuring OS-disk 0 24 24
Obtaining Internet resource 0 12 12
Obtaining middleware via Internet 12 14 26
Procuring package software 0 92 92
Preserving data-backup 0 1 1
Other device
Installing OS 26 2 28
Installing antivirus software 28 2 30
Installing package software 92 2 94
Restoring data 94 6 100
Validation 100 1 101
Finish 101 0 101
Activity: Early Start | Duration | Early Finish
Control: Early Start | Duration | Early Finish
Early Start (ES): The ES is defined as the maximum EF of all predecessor activities.
Early Finish (EF): The EF is the ES plus the task duration (EF = ES + duration).

Figure 4. PERT

4.4 Risk evaluation via proposed method

The risk which is calculated in the above subsection is an expected downtime value. By comparing this with the RTO, we can determine whether that risk is acceptable. Specifically, if the calculated risk is less than the RTO, the risk is deemed acceptable.

4.5 Risk treatment via proposed method

In order to determine the optimal combination of measures, we must first identify the effectiveness of those measures. The effect of a measure is identified as the incremental difference in the calculated risk between the case without the measure and the case with the measure (Eqs. 7-9):

$$\mathbf{R}_0 = \sum_{l=1}^{\mathbf{L_0}} \mathbf{P}_0[l] \times \mathbf{M}_0[l] \tag{7}$$

$$\mathbf{R}_i = \sum_{l=1}^{\mathbf{L}_i} \mathbf{P}_i[l] \times \mathbf{M}_i[l] \tag{8}$$

$$\mathbf{E}_i = \mathbf{R}_0 - \mathbf{R}_i \tag{9}$$

R_0:	Risk value without measure
L_0:	Number of sequences without measure
L*i*:	Number of sequences when adopting *i*-th measure combination
l:	*l*-th sequence
$P_0[l]$:	Occurrence probability of *l*-th sequence without measure
$M_0[l]$:	Downtime of the *l*-th sequence in ETA without measure
i:	*i*-th measure combination
R_i:	Risk value when adopting *i*-th measure combination
$P_i[l]$:	Occurrence probability of *l*-th sequence in ETA when adopting *i*-th measure combination
$M_i[l]$:	Downtime of *l*-th sequence in ETA when adopting *i*-th measure combination
E_i:	Effect of adopting *i*-th measure combination

Here, $P_i[l]$ is calculated as follows:

$$\mathbf{P}_i[l] = \prod_{h=1}^{\mathbf{H}i} \mathbf{P}_i'[h] \tag{10}$$

$$\mathbf{P}_i'[h] = ((1 - \mathrm{p}_i[h])(1 - y[h]) + \mathrm{p}_i[h] * y[h]) \tag{11}$$

$$\mathrm{p}_i[h] = \mathrm{p}_i'[h] \times \mathrm{xp}_i[h] \tag{12}$$

H_i:	Number of heading item when adopting *i*-th measure combination
$p_i[h]$:	Branch probability of *h*-th heading item when adopting *i*-th measure combination
$p_i'[h]$:	Probability of device breakdown when adopting *i*-th measure combination
$xp_i[h]$:	Effect of measures to reduce *h*-th occurrence probability when adopting *i*-th measure combination

$$y[h] = \begin{Bmatrix} 1: h - \text{th heading item unfolds in} \\ \text{horizontal derection} \\ 0: h - \text{th heading item unfolds in} \\ \text{vertical derection} \end{Bmatrix} \tag{13}$$

Here, M*i*[*l*] is calculated as follows:

$$\mathbf{M}_i[l] = \mathrm{PERT}(\mathrm{S}_i[l], \mathrm{D}_i[l][q]: q = 1,2, - - -, Q_i[l]) \tag{14}$$

PERT:	Function used to calculate downtime using PERT chart
$S_i[l]$:	Structure of PERT chart used in *l*-th sequence of ETA when adopting *i*-th measure combination
$D_i[l]\,[q]$:	Duration of *q*-th activity used in *l*-th sequence of ETA when adopting *i*-th measure combination
$Q_i[l]$:	Number of activities in PERT chart used in *l*-th sequence of ETA when adopting *i*-th measure combination

4.6 Cumulative probability distribution graph

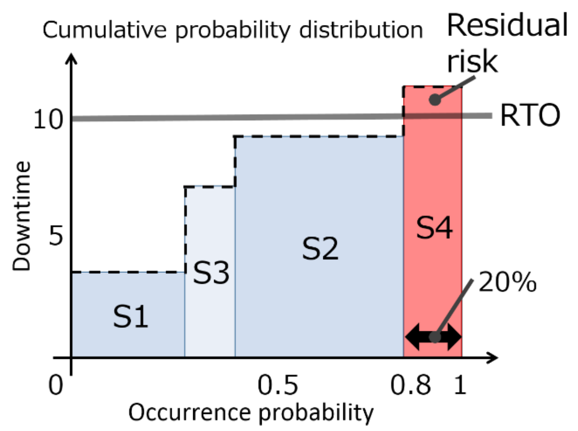


Figure 5. Cumulative probability distribution graph

Since it is often impossible to completely remove a risk, it is necessary to discuss the remaining residual risk after the implementation of a measure. To facilitate this, we developed a graph for presenting the relationship between the downtime and the occurrence probability (hereinafter referred to as the cumulative probability distribution graph) (Fig. 5). The use of this graph makes it possible to visually express residual risk. The process used to create this graph is as follows (see Fig. 6):

1. The sequences in ETA for the *i*-th measure combination and without measures are sorted according to the small order of the downtime value
2. The cumulative probability is calculated by adding the occurrence probability with its relation to total downtime
3. A graph is made from the relationship between the downtime and the cumulative probability

5 TRIAL APPLICATION

The proposed method was applied to a small example. To apply the method effectively, we developed a support system by adding a function to Excel. Table 1 shows the support system development and operating environments.

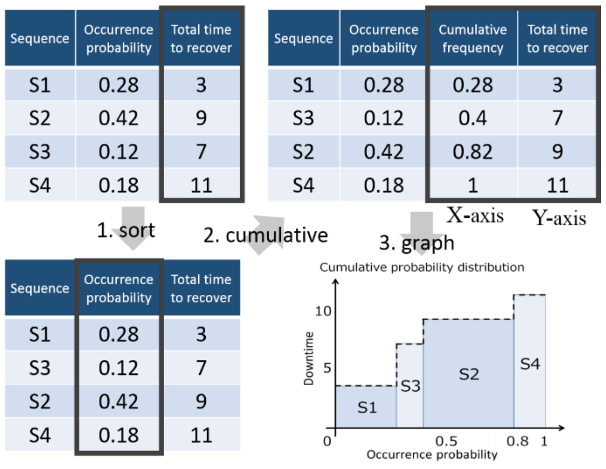
In this trial application, we calculated the values of R*i* and E*i* shown in Eqs. (10) and (11). In addition, we obtained the cumulative probability distribution graph for when the *i*-th measure combination was adopted.

Table 1. Development environment and movement environment

OS for development	Microsoft Windows 7 Professional Service Pack 1
Development language	Microsoft Visual Basic for Applications 7.1
Operating environment	Microsoft Excel 2013 (15.0.4641.1000) 64 bit

5.1 Evaluation model

In this trial application, we applied the proposed method to a Web-service system, a mail system, and a business-support system. These systems were assumed to be installed in the same room of a municipal city hall. The Web service system is designed to support a help line that deals with various application forms. It is very important and its RTO is assumed to be 50 hours.



Sequence	Occurrence probability	Total time to recover
S1	0.28	3
S2	0.42	9
S3	0.12	7
S4	0.18	11

Sequence	Occurrence probability	Total time to recover
S1	0.28	3
S3	0.12	7
S2	0.42	9
S4	0.18	11

Sequence	Occurrence probability	Cumulative frequency	Total time to recover
S1	0.28	0.28	3
S3	0.12	0.4	7
S2	0.42	0.82	9
S4	0.18	1	11

Figure 6. Cumulative probability distribution graph construction process

The mail and business support systems must be operational for the employees to perform their duties. Their RTO is assumed to be 100 hours. Table 2 shows the structure of the examined systems and the RTO. Figure 7 shows the system configuration of the application model. The implemented measures that are already in place and operating are shown in Table 3.

As the initiator of a BCM-related incident, the business continuity planning guidelines published by the Japanese government recommends assuming the occurrence of an earthquake measuring six on the seven-point Japanese scale [11], [12]. Accordingly, each data set was set up based on those guidelines. Table 4 shows the index used for setting the time required for recovery tasks. Table 5 shows

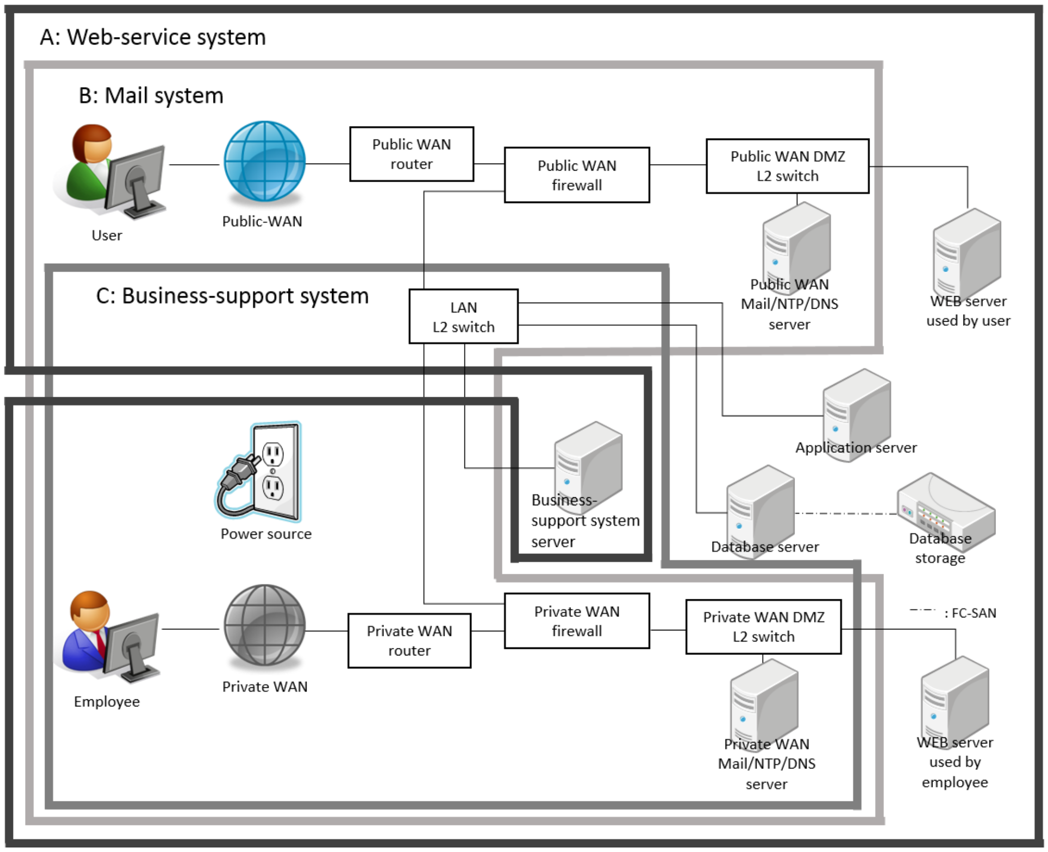


Figure 7. System configuration

the index for setting failure probability. The application process is as follows:

First, we calculate the current downtime. Second, we list an alternative measure for a system that has a high treatment priority. Third, we estimate the values of R*i* and E*i* and obtain the cumulative probability distribution graph for when the *i*-th measure combination is adopted using the proposed method. Finally, we decide the correct measure combination taking into consideration not only the values of R*i* and E*i*, by which we obtain the cumulative probability distribution graph, but also the cost and the derivative risk.

Table 2. Structure of treated systems and RTO

ID	System	RTO (hours)
A	Web-service system	50
B	Mail system	100
C	Business-support system	100

Table 3. Implemented measure

ID	Measure	Time required (hours)	Failure probability
Y1	Preserving data backup	1	0.01
Y2	Obtaining middleware via Internet	14	-
Y3	Setting reference of network device	1	0.01
Y4	Clustering device	-	0.8
Y5	Load balancer	-	0.8

Table 4. Index of setting time required for recovery

Recovery task	Time required (hours)
Procuring common device	24
Procuring technical device	92
Procuring legacy device	2160
Installation	2

Setting	4
Restoration	6
Validation	1
Obtaining power resource	24
Obtaining Internet resource	12

Table 5. Index of setting failure probability

Device or measure	Failure probability
Constitution device	0.6
Clustering device or load balancer	0.8
Fixing device	0.5

5.3 First Experiment

The objective of our first experiment was to calculate the risk for each system in order to determine whether the risk magnitude is acceptable. In addition, the achievement rate of RTO was confirmed via the cumulative probability distribution graph.

Table 6. Calculated risk for each system

ID	System	Calculated risk R_0 (Hours)	RTO (hours)
A	Web-service system	119.13	50
B	Mail system	95.61	100
C	Business-support system	73.52	100

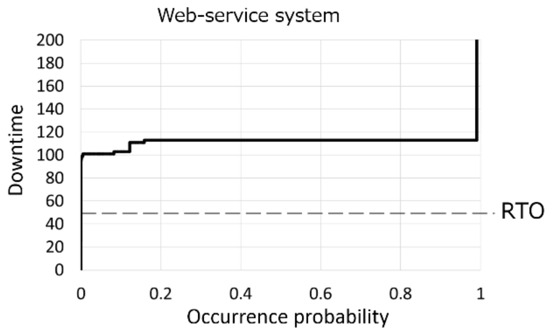


Figure 8. Cumulative probability distribution for web-service system

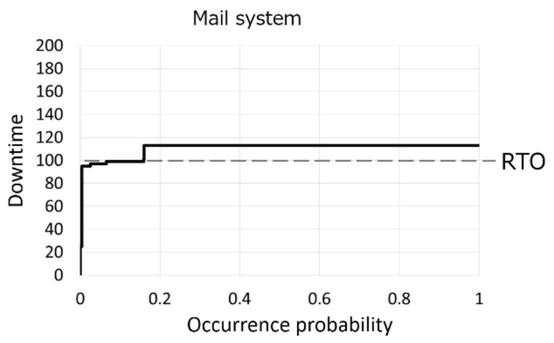


Figure 9. Cumulative probability distribution for mail system

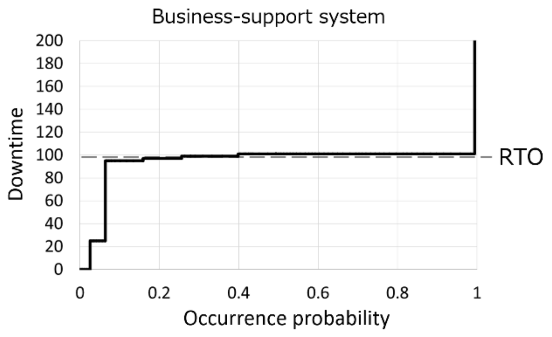


Figure 10. Cumulative probability distribution for business-support system

5.4 First Experiment Result

Table 6 shows the calculated risk R_0 calculated in Eq. 9 for the three systems. Figs. 8-10 show the cumulative probability distribution for the three systems. As can be seen in the table, the value of R_0 for all systems is more than the RTO. The mail and business-support systems could obtain sufficient effects by introducing small-scale measures because the RTO is exceeded by a very small amount. However, the Web-service system must consider the adoption of large-scale measures because the RTO is exceeded by a very large amount.

5.5 Second Experiment

The objective of the second experiment was to consider a measure that would reduce risks for the Web-service system. Two measure alternatives were created, as shown in Table 7. X1 indicates a measure in which spare devices are prepared (without setup) in a remote location. X2 indicates a measure in which those spare devices are prepared in a local location. X3 indicates a measure in which the device is fixed in place with bolts, screws, etc.

X1 and X2 are the measures implemented to reduce the downtime, and X3 is the measure implemented to prevent incident occurrence. This measure changes the occurrence probability.

Table 7. Measure alternatives

ID	Measure alternatives	Time required (hours)	Failure probability	Cost ($)
X1	Preparing a spares in remote location.	48	0.01	6,250
X2	Preparing spares in nearby location.	1	0.01	2,500
X3	Fixing the device in place	-	0.5	500

5.6 Result of Second Experiment

Table 8 shows the measure effects and the calculated risk when a measure is implemented. Fig. 11 shows the cumulative probability distribution.

Even though the summation of the X1 and X3 effects gives 92.81, the effect of X1+X3 can be calculated as 82.07. Conversely, even though the summation of the X2 and X3 effects is 57.26, the effect of X2+X3 can be calculated as 66.73.

Table 8 shows that although the most effective individual measure is X1, the most effective combination of measures is the selection of all three. In addition, when the cost constraints are $7,000, for example, we can understand that the optimal combination is X1+X3 (see Fig. 12).

From these results, we confirmed the basic effectiveness of the proposed method.

Table 8. Measure effect

ID	Measure effect (hours)	Calculated risk (hours)	Cost ($)
X1	81.07	49.81	6,250
X2	45.52	85.36	2,500
X3	11.74	119.13	500
X1+X3	82.07	48.81	6,750
X1+X2	86.4	44.48	8,750
X2+X3	66.73	64.15	3,000
X1+X2+X3	92.12	38.76	9,250

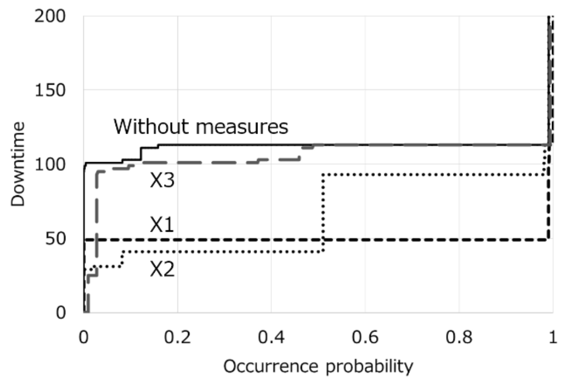


Figure 11. Cumulative probability distribution showing the effect of implementing measures

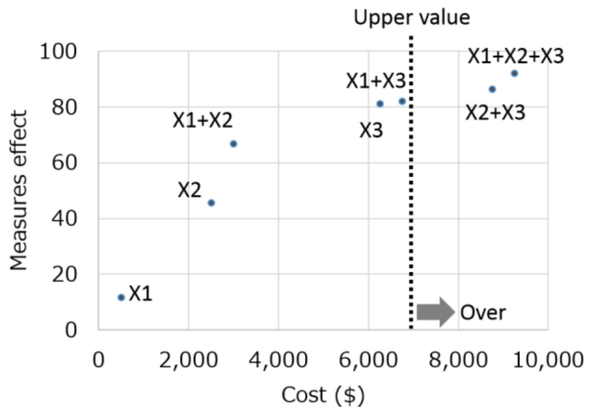


Figure 12. Constraints of Cost

6 Conclusions

In this paper, we proposed a BCM implementation method for use in combination with ETA and PERT that considers changes in both the probability and downtime. In addition, we provide an intuitively understandable display of analysis results and the residual risks.

Furthermore, we have also developed a system to support the above method and confirmed its usefulness by applying the BCM implementation method to a small example.

From these results, we confirmed the basic effectiveness of the proposed method.

However, since the number of applied cases is very limited, it will be necessary to apply the proposed method to other cases as future work.

REFERENCES

[1] I. Nakajima, the explanation of requirements in ISO 22301:2012, Japanese Standards Association,2013.

[2] ISO22301:2012 Societal security – Business continuity management systems – Requirements

[3] T. Matsuoka, N.Mitomo and Y.Matsukura, “Application of probabilistic safety assessment method and its marine areas: As an example the Titanic accident,” International Conference on Probabilistic Safety Assessment and Management,2000/12/1.

[4] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a Technique for Research and Development Program Evaluation," Operations Research, vol. 7, pp. 646-669, 1959.

[5] Y. Asnar and P. Giorgini, “Analyzing Business Continuity through a Multi-layers Model,” in Business Process Management. vol. 5240, M. Dumas, M. Reichert, and M.-C. Shan, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 212-227.

[6] M. Samejima and H. Yajima, “IT risk management framework for business continuity by change analysis of information system,” in Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on, 2012, pp. 1670-1674.

[7] U. Winkler, M. Fritzsche, W. Gilani, and A. Marshall, “A Model-Driven Framework for Process-Centric Business Continuity Management,” in Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the, 2010, pp. 248-252.

[8] K. Saleem, S. Luis, Y. Deng, S.-C. Chen, V. Hristidis, and T. Li, “Towards a business continuity information network for rapid disaster recovery,” presented at the Proceedings of the 2008 international conference on Digital government research, Montreal, Canada, 2008.

[9] H. Miller and K. Engemann, “Using Analytical Methods in Business Continuity Planning,” in Modeling and Simulation in Engineering, Economics and Management. vol. 115, K. Engemann, A. Gil-Lafuente, and J. Merigó, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 2-12.

[10] ISO 31000:2009 Risk management – Principles and guidelines.

[11] Ministry of Internal affairs and communications , Guidelines for business continuity planning of the ICT sector in the local governments, http://warp.ndl.go.jp/info:ndljp/pid/283520/www.soumu.go.jp/s-news/2008/080821_3.html,2008/08.

[12] Cabinet office of Japan government, Business Continuity Guideline second edition, http://www.bousai.go.jp/kyoiku/kigyou/keizoku/pdf/guideline02.pdf, 2009/11.

International Journal of Cyber-Security and Digital Forensics (IJCSDF) 4(2): 339-353
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2305-0012)

Analysis of Slow Read DoS Attack and Countermeasures on Web servers

Junhan Park, Keisuke Iwai, Hidema Tanaka and Takakazu Kurokawa
National Defense Academy of Japan
1-10-20 Hashirimizu, Yokosuka-Shi, Kanagawa-Ken, 239 -8686, Japan
junhanp78@gmail.com, {iwai, hidema, kuro}@nda.ac.jp

ABSTRACT

The ideas and techniques of DoS (Denial of Service) and DDoS (Distributed DoS) Attack strategies become more effective and more complex. In this paper, we focus on a Slow Read DoS Attack which is one of the sophisticated DoS attack techniques. This technique prolongs time to read the response from the Web server, although an attacker sends a legitimate HTTP request. When an attacker sends many such legitimate requests, he can keep many open connections to Web server and eventually cause DoS situation. In this paper, we analyze the effectiveness of Slow Read DoS Attack using the virtual network environment. As the result, we can find that Slow Read DoS Attack by a single attacker can be prevented by adequate security settings of Web server and applying countermeasure such as ModSecurity. However, from the analysis of Slow Read DoS Attack technique, we can also find that these countermeasures are not effective against distributed Slow Read DoS Attack (Slow Read Distributed DoS Attack) which is proposed in this paper.

KEYWORDS

Slow Read DoS Attack, Web Server Security, Slowhttptest, Apache, ModSecurity

1 INTRODUCTION

DoS (Denial of Service) attacks are evolved and consolidated as severe security threats to network service. Earlier DoS attacks use flood-based high-bandwidth approach and exploit the resource of network and transport protocol layers. Since the strategies of DoS attacks are simple, they can be prevented by filtering the source IP address. In order to break through this countermeasure, DDoS (Distributed DoS) attacks deliver a DoS attack by many attackers. However, there is a problem using high-bandwidth in DoS and DDoS attacks, and many improvements are proposed. One of such improvements is low-bandwidth approach. The latest attack method using such approach is low bit-rate type which exploits vulnerabilities of the application layer protocols to accomplish DoS attacks [1]. In this paper, we focus on the technique called Slow Read DoS Attack that has been designed by Sergey Shekyan [2]. This attack is that an attacker basically sends a legitimate HTTP request to target Web server and then very slowly reads the response. If an attacker sends many legitimate requests in this strategy, the Web server quickly reaches its maximum capacity and becomes unavailable for new connections by other legitimate clients. Moreover, it is very hard to detect if countermeasures do not monitor the network layer, because those requests are indistinguishable from other legitimate clients [3].

In Japan, some actual attacks using Slow Read DoS attack to the real on-line systems are reported but there are no open documents because they have clients' sensitive issues. In many cases, the targets are stock trade sites and on-line banking systems and damages of chance loss have caused [4]. In addition, there is a speculation for which Slow Read DoS Attack was used by Anonymous [5] to attack the site of JASRAC (Japanese Society for Rights of Authors, Composers and Publishers) for a protest against copyright protection in September, 2012 [6]. Thus the strategy of Slow Read DoS Attack evolves into the attack techniques with the actual damage.

In this paper, we analyze the effectiveness of Slow Read DoS Attack using the virtual network environment. We adopt *slowhttptest* as a general Slow Read DoS Attack scenario. It is freeware and available at [7]. We set the target Web server using *Apache* which is most popular one [8], [9]. From our analysis, we found that there is the limitation of effectiveness of attack by a single attacker, and it is determined by the setup of Timeout parameter in Web server. And we also discovered the improvement attack technique using collusion attack scenario. As the result, we propose a new attack technique “Slow Read DDoS Attack (Slow Read Distributed DoS Attack).” Furthermore, we analyze the effectiveness of Slow Read DoS Attack to the Web server with *ModSecurity* which limits huge number of connections from the same source IP address [10]. As the result, we confirmed that Slow Read DoS Attack can be prevented by it completely. However, when we use our new attack technique, we can ignore the security setting of Web server with ModSecurity and we can succeed in attacking. We conclude that although there is a function which such countermeasure makes time length of attack success status restrict, an attack cannot be prevented fundamentally. In finally, we discussed the problems to solve for improvement of attack strategies and countermeasures against our proposed attack method.

2 SLOW READ DOS ATTACK

2.1 Outline of Slow Read DoS Attack

The Slow Read DoS Attack is one of Slow HTTP attacks. Slow HTTP attacks do not aim at the network layer like DoS and DDoS attacks, but exploit the application layer. If a HTTP request is not complete, or if the transfer rate is very low, the Web server keeps its resources busy waiting for the rest of data. Slow HTTP attacks are based on this fact. Figure 1 shows the outline of Slow Read DoS attack. Thus when the Web server keeps too many resources busy, this situation becomes like DoS attacks. To realize this malicious condition, the attacker can take following two types of techniques.

1) The technique of sending request slowly
2) The technique of reading response slowly

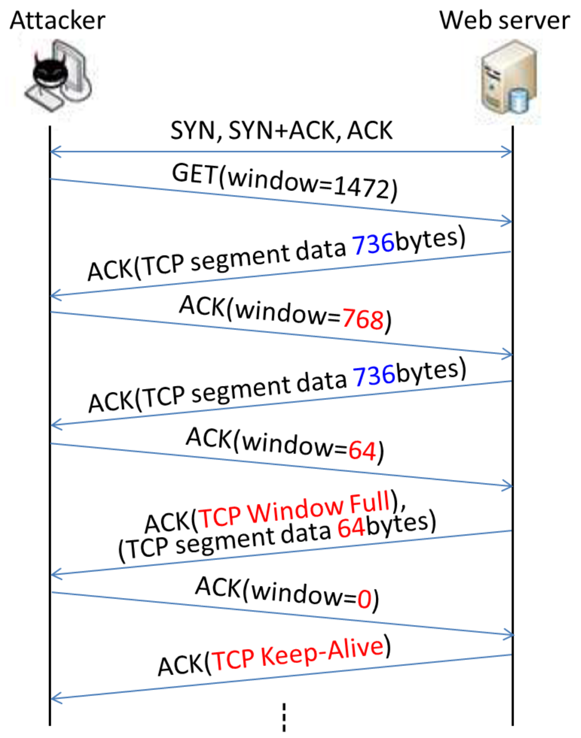


Figure 1. Outline of Slow Read DoS attack (packet flow)

Type 1) is well-known technique and Slowloris (also known as Slow Headers or Slow HTTP GET) or Slow HTTP POST attacks are famous. The Slow Read DoS Attack is categorized into type 2) and this is the latest technique. In this paper, we focus on type 2) strategy and technique of Slow Read DoS attack.

2.2 Basic Strategy

An attacker can deliver the Slow Read DoS attack by exploiting the flow control of TCP. Figure 1 also shows an example packet flow between an attacker and a target Web server in Slow Read DoS Attack procedure which can be captured by the network monitoring tool such as *Wireshark* [11]. First, the attacker sends a legitimate request after 3-way-handshake. After that, the attacker advertises the window size smaller than usual to make the HTTP response operation slow down. If the attacker advertises window size with zero, the Web server will stop sending data with keeping the connection. As the result, the attacker succeeds in Web server making resource waste.

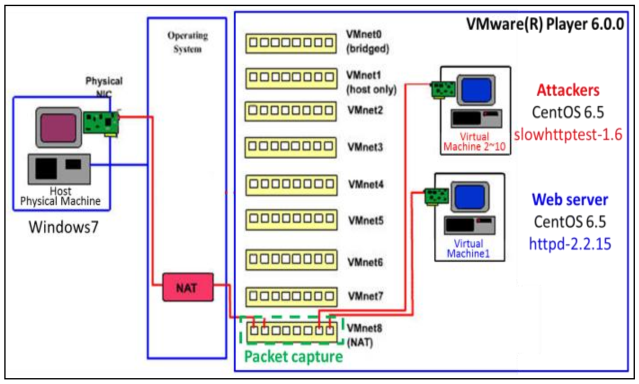


Figure 2. Experiment Environment [12]

Table 1. Directives of httpd.conf

Directive	Value
Timeout	60 (sec)
KeepAlive	Off

Table 2. Prefork MPM

Directive	Value
StartServers	8
MinSpareServers	5
MaxSpareServers	20
ServerLimit	256
MaxClients	256
MaxRequestPerChild	4000

Table 3. Parameters of slowhttptest

Parameter	Value
Number of attack connections	500
Receive window range	8-16 (byte)
Pipeline factor	1
Read rate from receive buffer	5 (byte/sec)
Connections Rate	50 (connections/sec)
Timeout for probe connection	10 (sec)
Using proxy	no proxy

3 PRELIMINARY

3.1 Experiment Environment

Figure 2 shows our experiment environment. We set 100 KB of Web page and use the Wireshark to observe packets between the attacker and Web server. Table 1 shows the default directives of “httpd.conf” which controls client connections to the Web server which is the attack target. Table 2 shows the default configuration of “prefork MPM (Multi-Processing Module)” which controls the generation of child processes when connections are established. Table 3 shows the parameters of slowhttptest which is a test tool of Slow Read DoS Attack.

Table 4. Parameters of Slow Read DoS Attack for each Experiment

Number of Experiment	Timeout	MC/SL
1	100 (sec)	200
2	200 (sec)	200
3	100 (sec)	300
4	200 (sec)	300
5	100 (sec)	600
6	200 (sec)	600
7	10 (sec)	200
8	10 (sec)	300

3.2 Max Client and Timeout

When Slow Read DoS Attack is delivered, service unavailable state of Web server is detected when the number of attack connections is reached the limit of Max Clients (MC in the following) in Table 2. When the total number of attack connection is larger than MC, there are no child processes to use. As the result, the Web server becomes service unavailable state. However, if the time of Timeout (Table 1) passes, the attack connection is disconnected compulsorily, and it will return to service available state. Thus, the Timeout parameter has a function which controls connection between them. And the value of MC controls the number of connections between the clients and the Web server.

3.3 Settings of Experiments

We fix the parameters of attacker as shown in Table 3 and set 8 types of target Web server as shown in Table 4. Note that parameter MC/SL means the value of MC and SL (Server Limit). We set them equal value because the setting of SL is not larger than the value of MC.

3.4 Definition of Attack Success

When conducting the Slow Read DoS Attack ex-

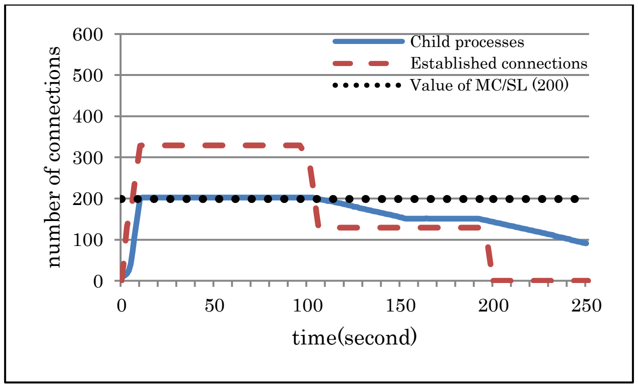


Figure 3. Experiment 1(Timeout 100, MC/SL 200)

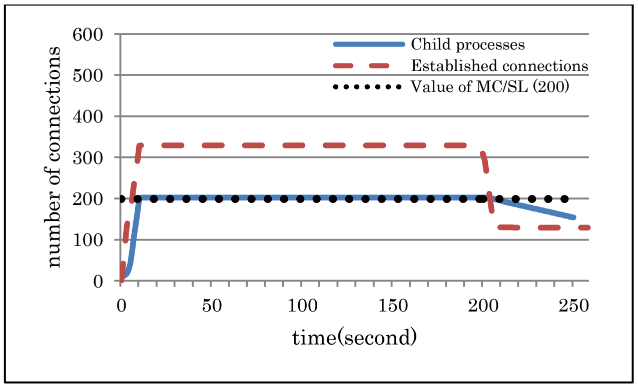


Figure 4. Experiment 2(Timeout 200, MC/SL 200)

periments, we define status of attack as follows.

- Attack success: Unacceptable new legitimate connections.
- Attack failure: Acceptable new legitimate connections.

"Attack success" means that the number of generated child processes is larger than or equal to the value of MC/SL. On the other hand, "Attack failure" means that the number of generated child processes is less than it. The effectiveness of attack success is estimated by time length (second) of maintaining service unavailable state.

4 EXPERIMENTS AND RESULTS

In the followings, we show the results of experiments as the graphs which show time transaction of the total number of child processes of Web server (blue line), and the total number of established connections of TCP which is connected to port #80 (red dashed line) after starting an attack at 0 second. And black dotted line shows the number of MC/SL. When the blue line is over the black dotted line, it is attack success.

4.1 Experiment 1 and 2

Figure 3 shows the result of Experiment 1. The total number of child processes reaches 200 which is the maximum value of MC/SL after 11 seconds. And we can succeed in the attack. However, by the setup of Timeout, the attack connections begin to be disconnected after 103 seconds, and it returned to the service available state. Therefore, the attacker is able to maintain the attack success status for 92 seconds. Though MC/SL is set to 200, the total number of established connections of TCP reaches 328. The reason for this situation is that these 128 (=328-200) of attack connections are contained by 3-way-handshake of TCP and treated as pending connections. By the setup of Timeout, the total number of established connections of TCP begins to decrease after 97 seconds. On the same time, pending connections are processed, but also disconnected after 193 seconds. Although the attack itself is ended after 10 (=500 (Total number of Attack connections) / 50 (Connections Rate)) seconds, the attack connections are processed until 193 seconds. As the result of Experiment 1, we can conclude that the attack succeeded between 11 and 103 seconds.

Figure 4 shows the result of Experiment 2. We set the value of Timeout to double (200 seconds) compared with Experiment 1. We can find that the attack success status is between 11 and 203 seconds, and the length is almost double of Experiment 1.

4.2 Experiment 3 and 4

We set the value of Timeout of Experiment 3 and 4 equal to Experiment 1 and 2 respectively, but are increased MC/SL to 300. Figure 5 and 6 shows the result of Experiment 3 and 4 respectively. From these results, they have the same tendencies as Experiment 1 and 2 but we can find that the number of child processes and the number of established connections of TCP are increased by the value of MC/SL. In Experiment 1 ~ 4, we used

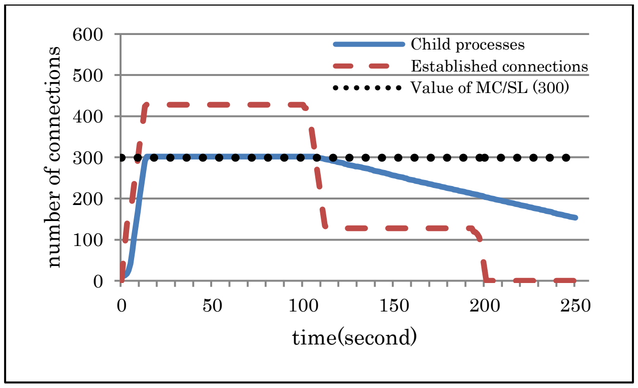


Figure 5. Experiment 3(Timeout 100, MC/SL 300)

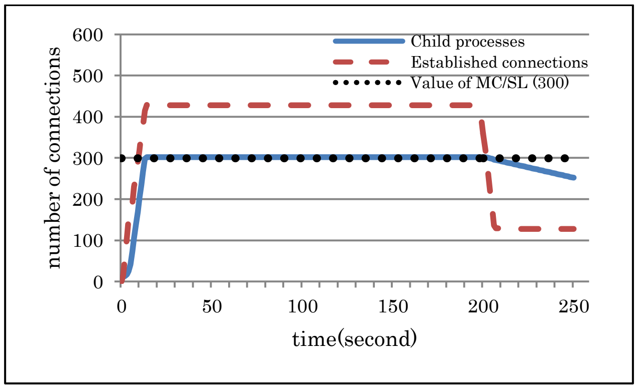


Figure 6. Experiment 4(Timeout 200, MC/SL 300)

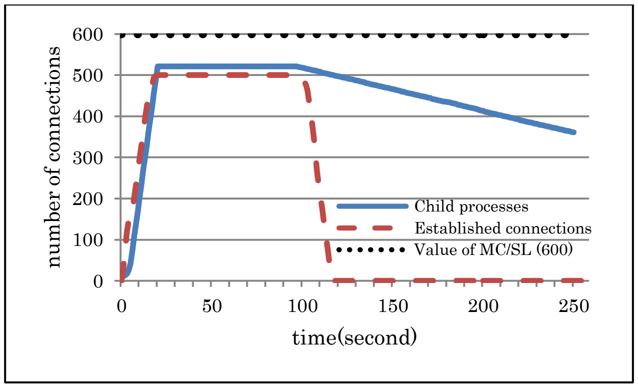


Figure 7. Experiment 5(Timeout 100, MC/SL 600)

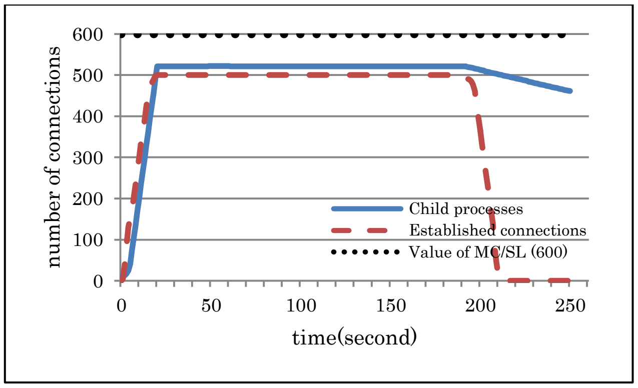


Figure 8. Experiment 6(Timeout 200, MC/SL 600)

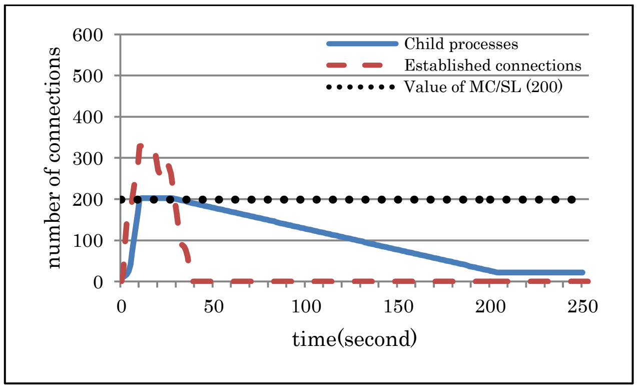


Figure 9. Experiment 7(Timeout 10, MC/SL 200)

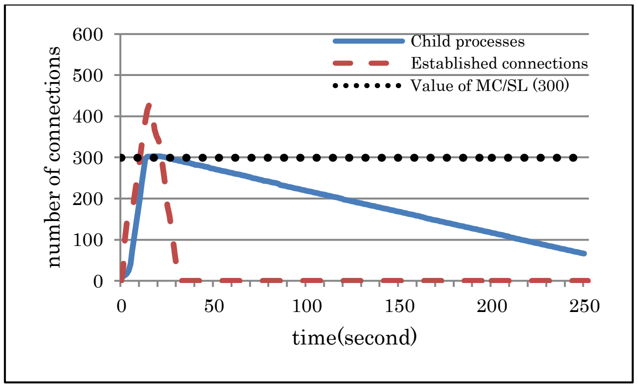


Figure 10. Experiment 8(Timeout 10, MC/SL 300)

500 of attack connections which are enough larger than MC/SL (300). Therefore, we can conclude that under such condition, the time length of attack success status is decided by the value of Timeout.

4.3 Experiment 5 and 6

We set the value of Timeout of Experiment 5 and 6 equal to Experiment 1 and 2 respectively, but are increased MC/SL to 600. Figure 7 and 8 show the result of Experiment 5 and 6 respectively. Naturally, we cannot succeed in the attack, because The Web server has enough resource against the total number of attack connections.

We can find two characteristics from these results. One is that the setup of Timeout also works after 98 seconds. Another is that we can find 520 of child processes, though the total number of attack connections is 500. This is because the number of MaxSpareServers is added as shown in Table 2.

4.4 Experiment 7 and 8

We set the value of MC/SL of Experiment 7 and 8 equal to Experiment 1 and 3 respectively, but are decreased Timeout to 10. Figure 9 and 10 show the result of Experiment 7 and 8 respectively. We can succeed in the attack. However, the attacker is able to maintain the attack success status for only 17 seconds in Experiment 7, and 8 seconds in Experiment 8, because the value of Timeout is extremely short. We can find that the value of Timeout is the main factor which determines the time length of attack success status.

4.5 Consideration

In order to analyze the relationship between the attack success status and the Web server parameters; Timeout and MC/SL, we conducted the attack simulations under the condition shown in Table 3 and 4. As the result, we can find following three factors that have determined attack success status.

1. The value of Timeout of Web server.
2. The value of MC/SL of Web server.
3. The total number of attack connections.

In general, the setup of Timeout is recommended for 10 seconds or more [13]. If Timeout is set short, Slow Read DoS attack can be prevented, but QoS (Quality of Service) is also reduced remarkably. If MC/SL is set large, Slow Read DoS Attack can be prevented, but there are limitations of resource and specification of the Web server. The number of attack connections is more effective factor, if it can be set huge comparing with Timeout and MC/SL. However, it depends on the attacker's cost. In addition, the attack connections from the same source IP address are easy to be detected. From these three reasons, we can conclude that the effectiveness of a single attacker's Slow Read DoS Attack is restrictive.

5 SLOW READ DDOS ATTACK

5.1 Outline of Proposal Technique

Since the attack connection is compulsorily disconnected by passing the Timeout, the attack success status of Web server returns to service available state. From the result of Experiment 8, we found that the countermeasure with 10 seconds of Timeout is effective against Slow Read DoS Attack which is parameterized as Table 3. Moreover, if the total number of attack connections is less than MC/SL, the attack cannot succeed. So, the effectiveness of attack by a single attacker is small as described in section 4.5. However, if another attacker sends new attack connections before previous attack connections are disconnected, it will be expected that the time length of attack success status can be maintained efficiently longer. Thus, we consider the scenario with which two or more attackers collude. In this paper, we call this attack technique "Slow Read DDoS Attack (Slow Read Distributed DoS Attack)."

Table 5. Variables

Variable	Explanation
t_0	Value of Timeout
t_z	Finish time of sending attack connections ($t_z = N / C$)
N	Total number of attack connections
C	Number of attack connections which send per second
K	Number of disconnected connections per second after t_0
M	Total number of connections that Web server can process (MC/SL)
$A_{(t)}$	Total number of attack connections which the attacker sent at time

5.2 Conditions for Attack Success Status

From the results of experiments shown in section 4, we can deduce the conditions of successful attack. Table 5 shows the variables. Let A be the total number of attack connections which connected to Web server. Then the condition of $M \leq A$ is necessary condition for successful attack, where M denotes the value of MC/SL. There are two cases of calculations for A under the conditions of t_0 and t_z. In the case of $t_0 \geq t_z$,

$$A(t) = C \times t \qquad (1)$$

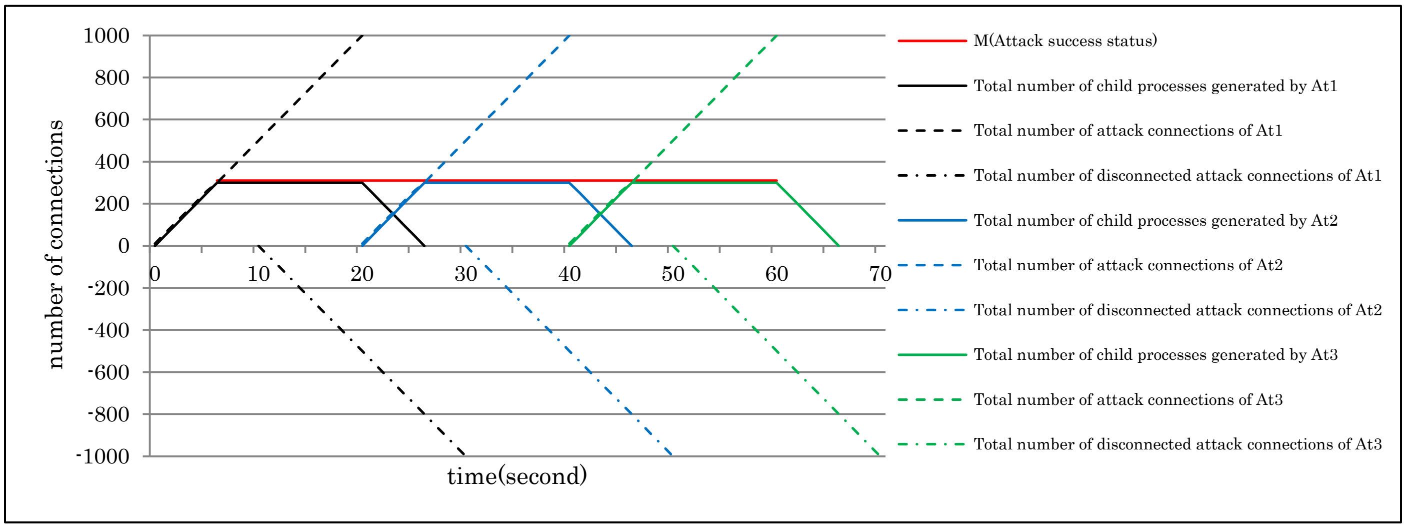


Figure 10. Attack Diagram

where C denotes the number of attack connection per second. In the case of $t_0 < t_z$,

$$A(t) = \begin{cases} C \times t & (t < t_0) \\ C \times t - K \times (t - t_0) & (t \geq t_0) \end{cases} \qquad (2)$$

where t $(t \geq t_0)$ denotes time progress after attack starts $(t = 0)$. Eq. (1) is in the case that Timeout t_0 is enough large. Therefore, even if the attack finished at t_z by a single attacker, it will maintain the attack success status until t_0. However, since Timeout t_0 is shorter than t_z, in the case of Eq. (2), it is difficult to maintain the attack success status by a single attacker (see Experiment 8). Previous attack connections will be disconnected after Timeout t_0. This situation causes the limitation of attack effectiveness. In order to solve this problem, we use colluded attack scenario with some attackers.

The basic idea for maintaining attack success status is that following attacker begins to send new attack connections before former attacker's t_z. Therefore the colluded attackers can maintain attack success status by repeating it. Let us consider N attackers $At_1, At_2, \ldots, At_N$. Attacker At_n $(2 \leq n \leq N)$ begins his attack at ta_n (sec), which is calculated as follows.

Table 6. Parameters of Attack Simulation

Parameter		Value
Apache Web server	Timeout	10 (sec)
	ServerLimit	300
	MaxClients	300
Attackers	Connections Rate	50
	Number of attack connections	1,000

$$ta_n = \sum_{i=1}^{n-1} \frac{N_i}{C_i} \qquad (n \geq 2) \qquad (3)$$

where C_i and N_i denote the values of C and N which attacker At_i set respectively. Note that $ta_1 = 0$ (sec). For example, Figure 10 shows a theoretical attack diagram of the Slow Read DDoS Attack by three attackers deriving from Eq. (3).

We assumed that the target Web server is the same as Experiment 8, because it has the most resistant against Slow Read DoS Attack. The almost setting of attacker is the same as Section 4. However we set to $N = 1{,}000$, in order to hold the condition $t_0 < t_z$. As the results, we set the value of parameters to $C_1 = C_2 = C_3 = 50$, $N_1 = N_2 = N_3 = 1{,}000$, $t_0 = 10$, $t_z = 20$ and $M = 300$. Thus, we can deduce the attack diagram (Figure 10) of Slow Read DDoS Attack using these parameters. And from these settings, we can exp-

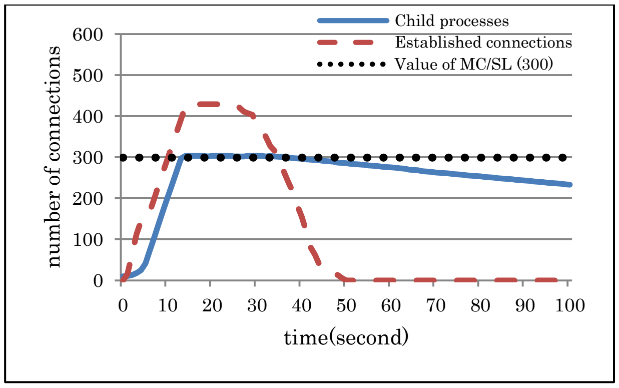


Figure 11. Simulation 1 (Attack result by At_1)

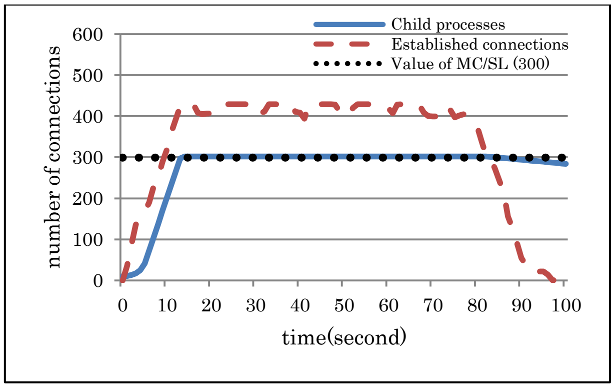


Figure 12. Simulation 2 (Attack result by At_1~At_3)

ect that attack success status can be maintained from 6 to 60 seconds. The parameters are summarized in Table 6.

6 ATTACK SIMULATIONS OF SLOW READ DDOS ATTACK

6.1 Outline of Attack Simulations

We set the parameters of attackers and Web server as Table 6. Before the Slow Read DDoS Attack simulation, in order to analyze the effectiveness using huge number of attack connections described in section 5.2, we conduct an attack simulation by a single attacker with $N = 1{,}000$ (Simulation 1). Next, we simulate the Slow Read DDoS Attack by three attackers following the attack diagram shown in Figure 10 (Simulation 2). From the attack diagram, each attacker's attack start time is set to $ta_1 = 0$ (sec), $ta_2 = 20$ (sec) and $ta_3 = 40$ (sec) using Eq. (3).

6.2 Results of Attack Simulations

Figure 11 shows the result of Simulation 1. We can find longer attack success status (20 seconds) than the result of Experiment 8 (8 seconds). We can confirm that increasing N is adequate for improving attack effect.

Figure 12 shows the result of Simulation 2. We can find that attack success status was maintained for 68 seconds (14~82 seconds). It is longer than the theoretical attack diagram shown in Figure 10 (54 seconds). From this result, we can conclude that the countermeasure with short Timeout do not work against our new attack technique.

6.3 Consideration

From the result of Simulation 1, we can consider that pending connections were also increased and they were newly processed even if Timeout passes. Because the total number of attack connections (N) was increased to 1,000 from 500, attack success status was maintained longer than Experiment 8.

We define following "attack rate" to evaluate the effectiveness of attack.

$$\text{attack rate} = \frac{\text{time length of attack success status}}{\text{number of attackers}} \quad \text{(sec/attacker)} \quad (4)$$

In Simulation 1, attack rate is 20.0 (sec/attacker). On the other hand, attack rate of Simulation 2 is 22.7 (sec/attacker). As the result, we can find that the Slow Read DDoS Attack is more efficient and lower-cost attack than Simulation 1.

In Simulation 2, we conducted the Slow Read DDoS Attack by three attackers. As the result, attack success status was maintained longer than Simulation 1. Therefore, we can consider that attack success status can be maintained for longer time, if three attackers attack repeatedly. We can find two facts between the attack diagram (Figure 10) and the result of Simulation 2 (Figure 12).

1. Time lag of child processes generation

From the result of attack shown in Figure 12, we can find that the attack starts to success after 14 seconds. On the other hand, from Figure 10, we expected after 6 seconds. Therefore there is 8 seconds of time lag. The reason is that we did not consider the delay time of generation of child processes which Web server generates. By predicting this time lag in advance, we will be able to set the attack start time to success exactly.

2. Influence of pending connections for establishment.

Pending connections will be generated, when the total number of attack connections (N) are set more than MC/SL. And they will be newly processed to establishment of connect after the time decided Timeout parameter for the former attack connections is passes. For this reason, we can consider that the pending connections can extend the time length of attack success status. Thus, it was extended longer than the theoretical attack diagram (Figure 10).

Therefore, there are two new problems for the improvement of our proposal attack.

1. Analysis of the child process generation.
2. Analysis of the pending connection processing.

If we can solve these problems, we can realize more precise and lower-cost attack.

7 MODSECURITY AND ITS EFFECTIVENESS

7.1 ModSecurity

"ModSecurity" is one of WAF (Web Application Firewall) which supports Apache HTTP Server, IIS and NGINX. It supplies real-time web application monitoring, logging, and access control. In this research, we use OWASP (Open Web Application Security Project) ModSecurity CRS (Core Rule Set) to control ModSecurity by setting configurable rule sets. OWASP ModSecurity CRS is distributed by Trustwave's SpiderLabs. The CRS provides configurable security rules such as follows [14].

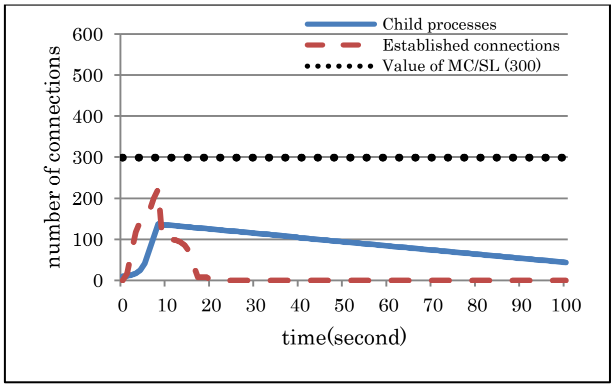


Figure 13. Experiment 9 (Timeout 10, MC/SL 300, Connection limit 100, Attack by At_1)

- HTTP Protection
- Real-time Blacklist Lookups
- HTTP Denial of Service Protections
- Common Web Attack Protection, etc.

In order to analyze the effectiveness of Web server with ModSecurity against the Slow Read DoS Attack, we focus on "HTTP Denial of Service Protections" to limit the number of connection from the same source IP address.

7.2 Outline of Experiment

We set the attacker's parameters as same as Table 3 in sections 3.1. And we set the target Web server same as Experiment 8; Timeout 10 and MC/SL 300. In addition, we use ModSecurity to limit the number of connections up to 100 from the same IP address. First, we conduct an experiment to analyze the effectiveness of Slow Read DoS Attack by a single attacker (Experiment 9). Next, we check the effectiveness of our proposal technique Slow Read DDoS Attack (Experiment 10) shown in section 5 with the same settings as Simulation 2.

7.3 Results of Experiments

Figure 13 shows the result of Experiment 9. We can see that ModSecurity functions after 8 seconds and the number of established connections are dramatically decreased. We can confirm that after ModSecurity starts, it can react immediately to the increase in attack connections. But since there is time lag of its starting, the Web server allows many generations of child processes which are

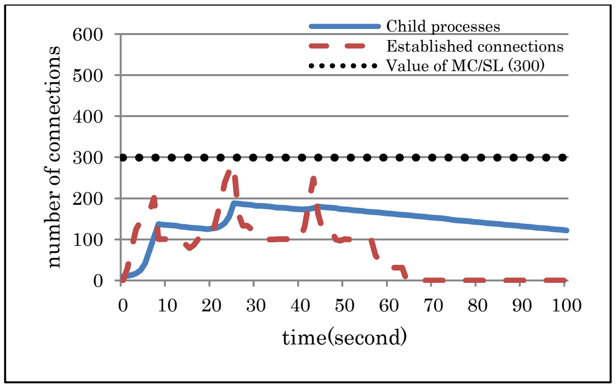


Figure 14. Experiment 10 (Timeout 10, MC/SL 300, Connection limit 100, Attack by At_1~At_3)

more than the limitation number. As the result, 138 child processes are generated. Since the reduction in child process follows the setup of Timeout, in the case of Experiment 9, child processes decrease at rate 1 (process/sec). In the settings of Experiment 9, we cannot succeed in the attack at all from above reasons. Therefore, we can conclude that ModSecurity has enough effectiveness against a simple Slow Read DoS Attack scenario.

Figure 14 shows the result of Experiment 10. We can see that the attack also did not succeed at all with same reasons described above. From the time transaction of total number of established connections, we can see that ModSecurity has enough effectiveness in the same way of Experiment 9. In addition, by the time lag of ModSecurity starting, the total number of child processes increased more than 100 at 8 seconds and 25 seconds. They are in a situation as the purpose of the Slow Read DDoS Attack. However, in the sense of increasing the number of child process, contribution of At_3 is smaller than At_2. This is because it is set off against reduction in child processes generated by At_1 and At_2. Therefore we can expect that the attack can be succeeded if the interval of each attacker's start time (ta_n) is closer than estimation using Eq. (3).

To confirm our assumption, we did two types of experiments with heuristic attack setup. They are Condition 1 with ta_1 = 0 (sec), ta_2= 10 (sec), ta_3= 20 (sec), and Condition 2 with ta_1 = 0 (sec), ta_2= 5 (sec), ta_3= 10 (sec).

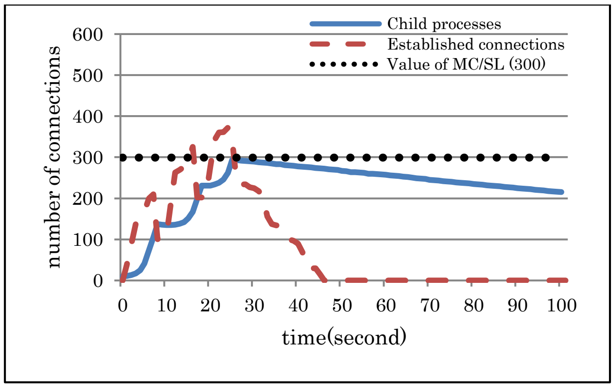


Figure 15. Condition 1 ($ta_1 = 0, ta_2 = 10, ta_3 = 20$)

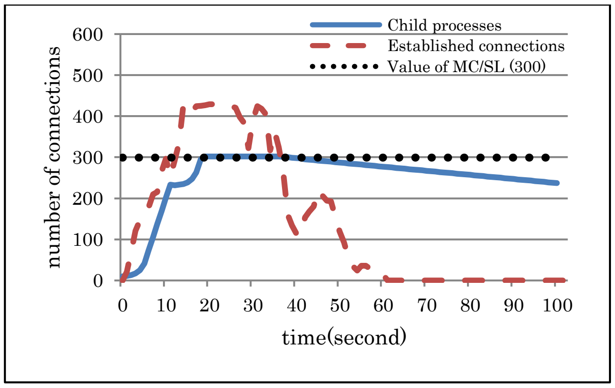


Figure 16. Condition 2 ($ta_1 = 0, ta_2 = 5, ta_3 = 10$)

Figure 15 shows the result of Condition 1. From this result, we can confirm that the total number of child processes did the monotone increase without influence of reduction of Timeout and disconnecting of attack connections by ModSecurity. As the result, the total number of child processes reached 293 at 25 seconds. Unfortunately, it has not yet resulted in the attack success.

Figure 16 shows the result of Condition 2. We can find that attack success status is maintained for 17 seconds (19~36 seconds) and attack rate was 5.7 (sec/attacker). Therefore it shows that our assumption was adequate. However, the attack diagram was derived heuristically from our attack experiments. Development of theoretical derivation of attack diagram against Web server with ModSecurity is our future work. We can see that the total number of established connections is increased at 31 seconds and 45 seconds in Figure 16. We can consider that this is because the pending connections which were waiting for establishment were newly processed as shown in the subject 2 of section 6.3.

As the results, we can conclude that the security setting of Web server which applying the short value of Timeout with ModSecurity has enough effectiveness against the Slow Read DoS Attack and simple Slow Read DDoS Attack. However, from the results of Condition 1 and 2, we can expect that some techniques can lose effectiveness of these countermeasures. The heuristic type of the improvement technique is already shown above. The algorithmic type which uses many colluded attack groups is shown in section 8.

8 IMPROVEMENT OF SLOW READ DDOS ATTACK

8.1 Attack Strategy against ModSecurity

In this section, we consider how to deceive the function of ModSecirity. In a simple way, we take a technique reducing the number of connections from the same IP address by increasing the number of attackers. The advantage of this technique is able to predict the effectiveness of attack easily by applying the technique shown in section 5.2 to attackers who are member of the same group. This is more positive than heuristic type as shown in previous section. On the other hand, if the setup of limitation number in ModSecurity CRS is unknown, the number of attackers is not decided. So, it is necessary to perform the attacks such as Experiment 9 in previous and predict the number of limitation. And it is easy to collect many attackers whose IP address is unique, if we use botnet which is popular in DDoS attack [15]. So, it is easily considered from the above discussions to satisfy the condition for the effective improved Slow Read DDoS Attack.

In the followings, we assume that the settings of Web server are known to attacker and it is the same as Experiment 9 in section 7.2. Since the value of limitation number in ModSecurity is 100, the maximum number of attack connections which one attacker can generate is 100. And since MC/SL = 300, one group needs to consist of three attackers (300/100=3) at least. Therefore, the following composition is the minimum attack unit.

- Attack Group 1 (Atg_1): ($\mathrm{At}_{11}, \mathrm{At}_{12}, \mathrm{At}_{13}$)
- Attack Group 2 (Atg_2): ($\mathrm{At}_{21}, \mathrm{At}_{22}, \mathrm{At}_{23}$)
- Attack Group 3 (Atg_3): ($\mathrm{At}_{31}, \mathrm{At}_{32}, \mathrm{At}_{33}$)

Total: 9 attackers.

Thus, minimum attack unit can be easily constituted using the information of limitation number of ModSecurity and the value of MC/SL. When many attackers can be prepared rather than minimum, it is obvious that more efficient attack can be performed.

Each attack group attacks according to the attack diagram which is shown in figure 10. And the attackers of each group conduct simultaneously. In other words, in order to ignore the function of ModSecurity, we increase the number of attackers to attack at the same time. And in order to maintain the attack success status for a long time, we have to increase other attack groups whose IP address are different previous attackers.

8.2 Outline of Attack Simulations

We set the parameters of Web server and attacker as Experiment 9. We assumed that three attack groups and one group consist of three attackers as shown in section 8.1. In the following attack simulations, our purpose is to analyze the total number of attack connection $\mathrm{N_i}$ of each attacker for successful attack. Therefore each group's attack start time follows the attack diagram shown in Figure 10. So, $\mathrm{tg}_1 = 0$ (sec), $\mathrm{tg}_2 = 20$ (sec) and $\mathrm{tg}_3 = 40$ (sec), where $\mathrm{tg_n}$ denotes attack start time of attack group n. We conducted two types of attack simulations; Simulation 3 and Simulation 4.

Simulation 3 is for the minimum attack unit. From Simulation 2 (see Figure 12), we obtained the successful attack result near theoretical estimation under the condition of $\mathrm{N_i} = 1{,}000$. Since 1,000 of attack connections from one group are necessary, 340 of attack connections per attacker are assigned. So, in Simulation 3, we set the attack condition

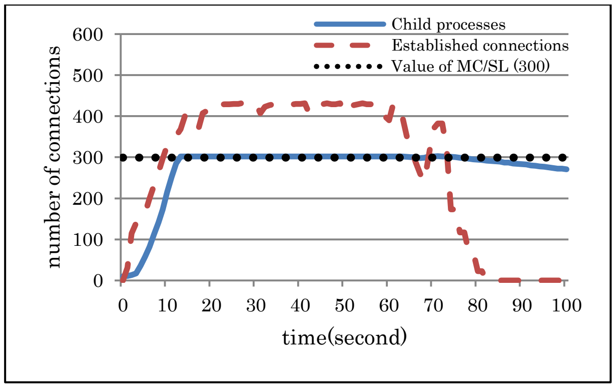


Figure 17. Simulation 3 ($N_{11}\sim N_{33}$ = 340)

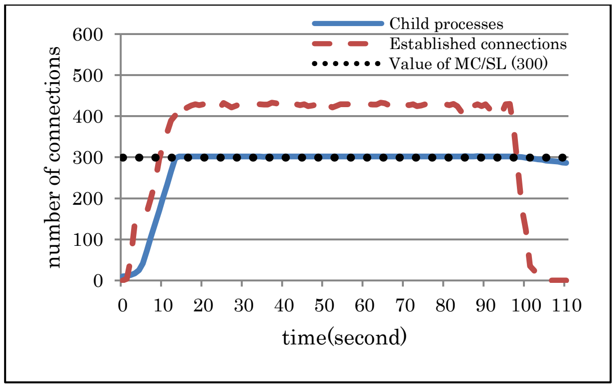


Figure 18. Simulation 4 ($N_{11}\sim N_{33}$ = 1,000)

with $N_{11}\sim N_{33} = 340$ to expect the effectiveness such as Condition 1 shown in section 7.3 (Figure 15).

Simulation 4 denotes a generous attack condition. The result of Simulation 2 shows that the condition of $N_i = 1{,}000$ is not high cost for each attacker. The purpose of this simulation is to analyze the effectiveness of generation of pending connections, and its influence for attack result. So, in Simulation 4, we set the attack condition with $N_{11}\sim N_{33} = 1{,}000$.

8.3 Results of Simulations

Figure 17 shows the result of Simulation 3. We can find that the attack success status was maintained for total 56 seconds (13~65 seconds and 69~73 seconds). From 66 seconds, the number of child processes became lesser than 300 because of Timeout and ModSecurity, and changed to attack failure status. After 69 seconds, it returned to attack success status because of the effectiveness of Atg_3's established connections. However, as the same situation as Condition 1 shown in Figure 15, the contribution of Atg_3 is lesser than Atg_2. So, it maintained attack success status for only four seconds. As the result, we can judge that Simulation 3 succeeded in the attack.

Figure18 shows the result of Simulation 4. On the whole, this simulation succeeded in the attack for 83 seconds. If we compare with Simulation 2, assuming one group to be one attacker, this result means that Simulation 4 used three times as many attack connections in Simulation 2. From the view point of this way, the condition of Simulation 4 is not efficient than Simulation 2, although there is a countermeasure of ModSecurity. Therefore it is thought that there is a method of determining more effective N and this is our future work.

As the results, we can confirm that the countermeasure which limits the number of connections from the same source IP address can be ignored by our improved new attack technique.

8.4 Consideration

From the results of Simulation 3 and 4, the attack rate of Simulation 3 is 6.2 (sec/attacker), and of Simulation 4 is 9.2 (sec/attacker). Comparing with Simulation 2, the attack cost is rose by applying ModSecurity on Web server. In this point, there is effectiveness as a countermeasure. However, ModSecurity cannot prevent the attack at all and “HTTP Denial of Service Protections” has not enough effectiveness against improved Slow Read DDoS Attack.

As shown in section 8.1, in order to hold attack success status longer than Simulation 4, we need to prepare other attack groups whose IP addresses are unique each other. On the other hand, if ModSecurity is not used, two attackers can maintain in the attack success status forever by attacking by turns in theoretically. If ModSecurity is used, even if we attack by improved Slow Read DDoS Attack with botnet, the time of attack success status is limited. This is the significant point of ModSecurity.

Also in Simulation 3 and 4, we can conclude that the pending connections are important factor. We already pointed out in section 6.3, analysis of the pending connections processing is useful for improvement of attack technique or defense strategy.

9 DISCUSSIONS

The important feature of Slow Read DoS Attack is that the target Web server is not down. From our attack simulations, it is clear from the fact that the target Web server certainly returned to service available state after the attack. Conversely, it is difficult for administrators to detect the attack. Especially, since legitimate requests are sent, it is expected that the signature type IDS is impossible to detect the attack.

In our attack simulations, in order to attack strategy simply, the following conditions were given.

1. The value of C (the number of attack connections which send per second) is fixed to 50.
2. The window size is fixed to zero.

When the target Web server is Apache, the maximum number of generated child processes per second is 32 [16]. Thus, under the condition of C=50, 18 of attack connections are processed per second as pending connections. Since child process is valid within the period decided by Timeout, if we can control the rate of generating pending connections by C, we could develop more effective attack method controlling the pending connections.

In our attack scenario, the attacker advertises "window size = 0." However, this is not necessary action in actual Slow Read DoS Attack. In fact, it is easy to be detected as malicious action. So, the value of window size should be set to a minimum necessary. As a result, although the effectiveness of attack is inferior to our simulations, the risk of attack detection will become small. In addition, the technique to which the value of window size is changed flexibly in a session is also considered. The evaluation to "Adaptive Slow Read DoS Attack" which changes the value of C and window size is our future work.

We also showed that improved Slow Read DDoS Attack is effective even if the target Web server uses secure modules. This attack requires the systematized attacker group. However, in the latest cyber-attack and cyber-crime, the executions by the systematized group are general cases. We should recognize that this attack scenario is a real threat.

As already we described above, the construction of IDS against Slow Read DoS and Slow Read DDoS Attack is very difficult. However, it is thought that the behaviour of adaptive attack shown above has some characteristics. So, an anomaly type IDS may be able to be constructed by discovering such features. In other way, TCP acceleration will prevent Slow Read DoS Attack from foundation. TCP acceleration is technique to achieve better throughput between client and server using TCP tuning [17]. There is a method to decide the value of window size based on the real-time measurement of packet arrival time [18], and it can be an effective countermeasure against our proposal attack method. This is also our future work.

10 CONCLUSIONS

In this paper, we analyzed the effectiveness of Slow Read DoS Attack by virtual network environment. From results, we concluded that the attack by a single attacker is not so efficient. However, we can derive the improvement of Slow Read DoS Attack and develop Slow Read DDoS attack. And we derived the attack diagram which maximizes the effectiveness of Slow Read DDoS Attack. We confirmed it by computer simulations. In addition, we conducted the attack simulations against the Web server with secure module, ModSecurity. ModSecurity can limit the length of attack success status however, attack itself cannot be prevented. As the result, we succeeded in improving the attack technique whose effectiveness is same level in the case of attack against the target Web server without secure modules. We summarized our discussions and future works in section 9. And we concluded that the analysis of generation of pending connections and the time lag of starting security modules or child process are important factor to improve the attack technique and to develop countermeasures.

Note that our attack simulations are done in only one-hop virtual environment in order to be easy to analyze (see Figure 2). Therefore, the attack will affect differ in actual internet environment. Specifically, we have to add followings as attack factors.

- Existence of intermediate servers and routers.
- Existence of legitimate users who has already connected.

Since the existence of intermediate servers and routers affects the communication speed, we should adjust the value of C (The number of attack connections which send per second) and N (The total number of attack connections) for successful attack. Depending on the number of already connected by general users, our attack may be successful with lesser cost. The analysis of more actually based threat is also our future work.

11 REFERENCES

1. Cambiaso Enrico, Papaleo Gianluca, Chiola Giovanni and Aiello Maurizio, “Slow DoS attacks: definition and categorisation,” Int. J. Trust Management in Computing and Communications, Vol. 1, No. 3/4, pp. 300-319 (2013).
2. Sergey Shekyan, “Are you ready for slow reading?,” QUALYS BLOG, Available: https://community.qualys.com/blogs/securitylabs/2012/01/05/slow-read (Last access: Dec. 12, 2014)
3. Kelly Jackson Higgins, “New Denial-Of-Service Attack Cripples Web Server By Reading Slowly,” InformationsWeek DarkReading (Internet article), Available: http://www.darkreading.com/attacks-breaches/new-denial-of-service-attack-cripples-we/232301367 (Last access: Dec. 12, 2014)
4. Comments of Mizuho Research Institute for our presentation “Anlysis of Slow Read DoS Attack and countermeasure” at Japanese domestic conference, Computer Security Symposium (2014).
5. Anonymous Operation Japan, twitter account “@OP-japan.”
6. Japanese bulletin board, 2ch.net.
7. Sergey Shekyan, “slowhttptest,” Available: https://code.google.com/p/slowhttptest (Last access: Dec. 12, 2014)
8. W3Techs, “Most popular web servers,” Available: http://w3techs.com (Last access: Dec. 12, 2014)
9. Apache, Available: http://httpd.apache.org (Last access: Dec. 12, 2014)
10. ModSecurity, Available: http://www.modsecurity.org (Last access: Dec. 12, 2014)
11. Wireshark, Available: https://www.wireshark.org (Last access: Dec. 12, 2014)
12. ExtremeTech, “Virtual Machines & VMware Part II,” Available: http://www.extremetech.com/computing/72268-virtual-machines-vmware-part-ii (Last access: Dec. 12, 2014)
13. Noopy Zang, “Tuning of Apache (in Korean blog),” Available: http://openlife.tistory.com/340 (Last access: Dec. 12, 2014)
14. OWASP ModSecurity Core Rule Set Project, Available: https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project (Last access: Dec. 12, 2014)
15. Esraa Alomari, B. B. Gupta, Shankar Karuppayah, “Botnet-based Distributed Denail of Service(DDoS) Attacks on Web Servers,” Classification and Art, Int. Journal of Computer Applications (0975-8887), Vol. 49, No.7, pp. 24-32 (2012).
16. Apache Performance Tuning, Available: http://httpd.apache.org/docs/trunk/en/misc/perf-tuning.html (Last access: Dec. 12, 2014)
17. Sameer Ladiwala, Ramaswamy Ramaswamy, Tilman Wolf, “Transparent TCP acceleration,” Computer Communications 32, pp. 691-702 (2009).
18. Takashi Isobe, Naoki Tanida, Yugi Oishi and Ken-ichi Yoshida, “TCP Acceleration Technology for Cloud Computing: Algorithm, Performance, Evaluation in Real Network,” Proceedings of the 2014 International Conference on Advanced Technologies for Communication, pp. 714-719 (2014).
19. Evan Damon, Julian Dale, Evaristo Laron, Jens Mache, Nathan Land and Richard Weiss, “Hands-On Denial of Service Lab Exercises Using Slowloris and RUDY,” Proceedings of the 2012 Information Security Curriculum Development Conference, pp. 21-29 (2012).
20. Cambiaso Enrico, Papaleo Gianluca, Chiola Giovanni and Aiello Maurizio, “Taxonomy of Slow DoS Attacks to Web Applications,” International Conference on Security in Computer Networks and Distributed Systems 2012, Communications in Computer and Information Science 335, pp. 195-204 (2012).
21. Hiroshi Kurakami, “The advanced DDoS attack and countermeasure (in Japanese),” IPSJ (Information Processing Society of Japan) Magazine, Vol.54, No.5, pp. 475-480 (2013).
22. Jonathan Lemon, “Resisting SYN flood DoS attacks with a SYN cache,” Proceedings of the BSD Conference 2002 on BSD Conference (2002).

23. Junhan Park, Keisuke Iwai, Hidema Tanaka and Takakazu Kurokawa, "Study of Slow Read DoS Attack on Web server (in Japanese)," Proceeding of the 31st Symposium on Cryptography and Information Security, 2C1-4 (2014).
24. Junhan Park, Keisuke Iwai, Hidema Tanaka and Takakazu Kurokawa, "Analysis of Slow Read DoS Attack," 9th ACM Symposium on Information Computer and Communications Security, Poster Session (2014).
25. Junhan Park, Keisuke Iwai, Hidema Tanaka and Takakazu Kurokawa, "Analysis of Slow Read DoS Attack and Countermeasures (in Japanese)," Proceeding of the Computer Security Symposium, pp. 354-361 (2014).
26. Junhan Park, Keisuke Iwai, Hidema Tanaka and Takakazu Kurokawa, "Analysis of Slow Read DoS Attack," Proceeding of the International Symposium on Information Theory and Its Applications, pp. 60-64 (2014).
27. Junhan Park, Keisuke Iwai, Hidema Tanaka and Takakazu Kurokawa, "Analysis of Slow Read DoS Attack and Countermeasures," Proceeding of the International Conference on Cyber-Crime Investigation and Cyber Security, pp. 37-49 (2014).
28. Stephen Specht and Ruby Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures," 2004 International Workshop on Security in Parallel and Distributed Systems, pp. 543-550 (2004).
29. Takeshi Yatagai, Takamasa Isohara and Iwao Sasase, "Dectection Technique of HTTP-GET Flood Attack Based on Analysis of Page Access Behavior (in Japanese)," IPSJ (Information Processing Society of Japan) SIG Technical Report, 2007-CSEC-37, pp. 33-38 (2007).
30. Ha.ckers, Slowloris HTTP DoS, Available: http://ha.ckers.org/slowloris (Last access: Dec. 12, 2014)
31. KISA (Korea Internet Security Agency), "The response guide against DDoS attack (in Korean)," Technical Report, KrCERT-TR-2012-002, Available: http://www.krcert.or.kr (Last access: Dec. 12, 2014)
32. Ronen Kenig, "Why Low & Slow DDoS Application Attacks are Difficult to Mitigate (blog)," Available: http://blog.radware.com/security/2013/06/why-low-slow-ddosattacks-are-difficult-to-mitigate (Last access: Dec. 12, 2014)
33. VMware Player, Available: http://www.vmware.com/jp/products/player (Last access: Dec. 12, 2014)

International Journal of Cyber-Security and Digital Forensics (IJCSDF) 4(2): 354-379
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2305-0012)

Characterising File Sharing and its Protection, the Insider Threat Case

Rakan Alsowail and Ian Mackie
School of Engineering and Informatics,
University of Sussex, Falmer, Brighton, BN1 9RH, United Kingdom
ra216@sussex.ac.uk, i.mackie@sussex.ac.uk

ABSTRACT

File sharing has become an indispensable part of our daily lives. Some of the shared files are sensitive, thus, their confidentially, integrity and availability should be protected. This paper investigates the protection requirements and the activities of file sharing from the perspective of the insider threat problem. It addresses three fundamental questions to the design of a protection mechanism against insider misuses: who is the insider, what are the insider misuses, and how the activity of file sharing can be performed. This paper proposes a new approach for classifying the insider threat problem into different categories and then focuses on one category that is related to file sharing. It characterises the protection required by the shared files against different types of insiders misuses and characterises the activity file sharing based on two factors: how files can be propagated and how they can be accessed after their propagation.

KEYWORDS

Insider threat, security, information security, file sharing, data confidentiality, file dissemination.

1 INTRODUCTION

File sharing has been a topic of interest in computer science right from the beginning–ever since files were created. The prevalence of file sharing activity nowadays is attributed to the existence of variety of methods that simplified such activity to be performed. These methods have gone through several stages until they reached the maturity at the present time to become fundamentals to any Internet user. At the beginning, no actual storage medias existed, where the only way to transfer information from one computer to another is to type them manually. Later on, the first magnetic storage media emerged which could contain data, however, moving around these magnetic storage were very difficult [1]. The first time file sharing became an easy task to perform was in 1971 when the 8-inch floppy disk was developed by IBM [2, 3, 4, 5]. However, spreading of files went slowly as the files had to be moved physically from one place to another. Users were able to share files online by utilising their phone lines in 1978 when the the first online bulletin board system (BBS) emerged. This was followed by various methods of sharing such as Usenet in 1979, FTP in 1985, Napster in 1999. From 2000 up to the present time, a wide variety of peer-to-peer file sharing system emerged such as Gnutella, eDonkey 2000, Kazaa, BitTorrent as well as web-based file sharing services such as Dropbox, GoogleDocs, youSENDit, Streamfile, Wikisend, 4shared and social networking sites such as Facebook, YouTube, Instagram and Flickr.

The existence of these methods nowadays encouraged more people to share files with each other. Some of the shared files might be confidential content that needs to be protected. Such confidential content raised the awareness of people to the security concept *share but protect*. Depending on the nature of the content, the shared files might need to be protected against unauthorised disclosure, modification, or withholding. Generally speaking, such protections stem from three distinct fields of security due to the fact that data can be in three different states. First, data can be stored, and protecting it is the main concern of a field named Perimeter security which prevents attacks on data stored inside a trusted internal network. Second, data can be in transit, and protecting it is the main concern of a field named Communication security which prevents attacks on data transmitted over a network. Third, data can be in use, and protecting it is the main concern of a field named Insider security which prevents attacks on data by those who have authorised access.

According to the 2011 CyberSecurity Watch Survey, conducted by the U.S. Secret Service, the CERT Insider Threat Center, CSO Magazine, and Deloitte [6], 58% of the attacks are caused by outsiders (those unauthorised to access network systems or data) while 21% of the attacks are caused by insiders (those authorised to access network systems and data), and 21% from unknown sources. Even though the percentage of insider attacks is less than the external attacks, the consequences of insider attacks can be more severe. The survey indicated that 33% of the respondents consider insider attacks to be more costly and damaging. Consequently, insider attacks merit the same attention as external attacks.

Protecting the shared files from the perspective of Insider security is a challenging problem. It has always been recognised that preventing policy violation by authorised users is more challenging than those who are not. Authorised users have access privileges that make it hard to prevent or detect policy violation. Additionally, providing a mechanism to protect the shared files from insiders requires an investigation into three fundamental questions which we address in this paper.

- First: What is the insider problem?

The problem with the insider security literature is that there is no a widely accepted definition of what is an insider and there is no a clear distinction between insiders and outsiders. What is considered an insider for someone might be an outsider for someone else. Therefore, protecting the shared files from insiders without knowing who is the insider is meaningless. Bishop and Gates [7] pointed out that there exist many definitions of insider and insider threat in the literature that complicated the research in insider threats as one solution to the insider problem might not be applicable to another insider problem. Also, Hunker [8] stated that although there exists a large body of work in the literature to address the insider threat problem, little progress has been made due to the absence of clear answers to fundamental questions such as "What is an insider threat". We believe that the insider problem should be classified into several categories which can be defined, studied and solved independently, and later combined to solve the problem as a whole.

- Second: What are the insider misuses?

Defining the insider problem and the insider precisely is the first step towards protecting the shared files from insiders. What more important is identifying the misuses that can be performed by insiders. Misuses are actions taken by the insider which violate the confidentiality, integrity or availability of a particular asset. by knowing the misuses that the insider can perform on the shared files, we can derive the different types of protections that are required to protect the shared files.

- Third: How the activity of file sharing can be performed?

While the first two questions are related to the insider security, this question is related to the activity of file sharing. However, similar to the insider problem, the activity of file sharing is not clearly identified. Most of the research on file sharing is focused on specific domains and applications while little research studied file sharing more broadly [9]. The term *file sharing* is rarely defined in the literature, and if defined, it is tailored to a specific method of sharing. One exception is the study by Whalen et al. [9] who defined file sharing as "the activity of making specified file(s) available to an individual or group, with the option of granting specific right (e.g., ability to view, edit, delete) over those files". However, a general characterisation of how the activity of file sharing can performed is currently missing.

Protecting the shared files is a topic that have been studied from two different fields with different interests, namely, information sharing and security. The former focuses on facilitating information sharing and provides sharing tools that are suitable for various tasks of sharing but not secure. The latter focuses on securing information sharing and provides sharing tools that are secure but not suitable for every task of sharing. Considering both fields will help us to design a protection mechanism that will not only protect the shared files from insiders but also allow owners of files to share their files as desired. Therefore, in addition to identifying the different types of insider misuses, we investigate how the activity of file sharing can be performed.

In this paper we study the protection requirements and the activities of file sharing from the perspective of the insider threat problem by providing answers to the above three questions. The rest of this paper is structured as follows. In Section 2 we review the literature and related work on insider threat and file sharing, respectively. In Section 3, we propose a new approach for classifying the insider threat problem and focus on one category that is related to file sharing. In Section 4, we give our first contribution to characterising the protection required by the shared files against different types of insiders. In Section 5, we give our second contribution to characterising file sharing based on two factors: how files are propagated and accessed after their propagation. In Section 6, we define a framework to classify the activity of file sharing. Finally, in Section 7, we conclude the paper with our future work.

2 RELATED WORK

2.1 Insider Security

Several definitions of insider and insider threat exist in the literature. Some authors have focused on the trust relationship when defining the term insider. For instance, RAND report [10] defined the insider as "an already trusted person with access to sensitive information and information systems". Bishop [11] defined the insider as "a trusted entity that is given the power to violate one or more rules in a given security policy". Other authors have focused on the abuse of given access privileges. For instance, Chinchani et al. [12] defined the insider as "legitimate users who abuse their privileges". CERT report [13] defined the insider as "individuals who were, or previously had been, authorised to use the information systems they eventually employed to perpetrate harm". Others defined the insider very broadly. For instance, Predd et al. [14] defined the insider as "someone with legitimate access to an organisation's computers and networks". RAND report [10] defined the insider again as "anyone with access, privilege, or knowledge of information system and services". The former definition might include masqueraders who stole the credential of a legitimate user to get access to the computer or the network. The latter definition eliminates the need of trust and includes those who have knowledge of the system or the service even if they do not have access privileges. In 2008, a cross-disciplinary workshop on "Countering Insider Threats" [15] concluded that

> "an insider is a person that has been legitimately empowered with the right to access, represent, or decide about one or more assets of the organisation's structure"

With regard to insider threat, Predd et al. [14] defined insider threat as "an insider's action that puts an organisation or its resources at risk". RAND report [10] defined it as "malevolent (or possibly inadvertent) actions by an already trusted person with access to sensitive information and information systems". Hunker and Probst [16] defined it as "an insider threat is [posed by] an individual with privileges who misuses them or whose access results in misuse". The CERT Insider Threat Center's current definition of insider threats as follows:

> "A malicious insider threat to an organisation is a current or former employee, contractor, or other business partner who has or had authorised access to an organisation's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organisation's information or information systems". [17]

Due to the differences and contradictory definition of insider and insider threats that complicated the problem to be solved, many authors are urging the community to establish a framework or taxonomy for distinguishing among different types of insider threats [15, 16]. They mentioned that each determining factor for an insider can be used for a taxonomy, for example based on distinctions between: Malicious and accidental threats; Doing something intentionally (for malice, or good reasons which nonetheless may result in damage) versus events that occur accidentally; Obvious and stealthy acts; Acts by masqueraders (e.g, an individual with a stolen password), traitors (malicious legitimate users) and naive or accidental use that results in harm; A combination of factors such as access types; aim or intentionality or reason for misuse; level of technical and the system consequences of insiders threats.

Bellovin [18] identified three different types of insider attack which are misuse of access, defence bypass, and access control failure. He stated that access control failure attacks can be prevented by purely technical means, while the other two attacks require combination of technical and non-technical means. Hunker and Probst [16] identified three different approaches, which current works in the field revolve around, to solve the insider threat problem. These approaches are technical approach, socio-technical approach, and sociological approach. The authors noted that technical approaches are focused on policy languages, access control and monitoring, while socio-technical approaches are focused on policy, monitoring and profiling, prediction, forensics and response work. Sociological approaches are focused on motivation, organisational culture, human factors and privacy and legal aspects. Silowash et al. [19] analysed cases of insider threat from the CERT insider threat database, which contains more than 700 cases of insider threat, and observed that malicious insider activities can be classified into four classes as follows.

- IT sabotage: an insider's use of IT to direct specific harm at an organisation or an individual. Example of this are destroying critical data, planting logical bomb to delete data at critical times, etc.
- Theft of IP: an insider's use of IT to steal IP from the organisation. This category includes industrial espionage involving outsiders. Examples of usually stolen IP assets are proprietary software, business plans, product details, and customer information.
- Fraud: an insider's use of IT for the unauthorised modification, addition, or deletion of an organisation's data (not programs or systems) for personal gain, or theft of information that leads to an identity crime (e.g., identity theft or credit card fraud).
- Miscellaneous: cases in which the insider's activity was not for IP theft, fraud, or IT sabotage.

2.2 File Sharing

A wide variety of file sharing methods exist, and they differ from one another in the way that they allow users to control the what, how, and with whom to share [20]. Various studies have been conducted to investigate these properties.

Olson et al. [21, 22] conducted a pilot study and a more formal survey to explore preferences for general information sharing by investigating what information people are willing to share, and with whom. Their findings indicated that people willingness to share is different from one another and it depends on who they are sharing the information with, therefore, a one-size-fits-all permissions structure for sharing is inappropriate. They found that people deal with particular types of information similarly when assessing whether or not to share it with others (example classes include work email and telephone number, pregnancy, health information, email content, credit card number). Also, they found that people deal with particular types of individuals similarly when assessing whether or not to share information with them (example classes include spouse, manager, trusted coworker, the public and competitors). The authors believe that their findings can provide guidance to the design of access control and interfaces.

Voida et al. [20] conducted a survey and follow-up interviews at medium-sized research organisation to explore users' current practices and needs around file sharing. The authors stated that the understanding of what, with whom and how of sharing will lead us to understand users current sharing practices of file sharing. The result of their study indicated that almost third of the respondents shared files with groups or classes of individuals, and in many cases these classes mapped directly onto categories identified by Olson et al. [21, 22]. Also, their survey respondents reported sharing files at work regularly with an average of 7 individuals or groups. With respect to the types of files are shared, their respondent reported 34 different types of files or electronic information they share, which range from business documents and paper drafts to music, ideas, schedules, and TV shows. In terms of how the sharing is taking place, they found that Email is the most common method used for sharing files by their respondents (43% of all responses), followed by shared network folder (16%), followed by posting content to a website (11%).

Their findings indicated that there are three main classes of difficulties and breakdowns that people encounter in sharing, which are: forgetting what file had been shared with who; difficulties in selecting a sharing method with desired features that was also available to all sharing participants; and problem in knowing when new content was made available. They mentioned that their respondents usually fall back to use the most universal method, which is Email, in order to share their files when they are uncertain about the tools available to their intended recipients. Based on their findings, they identified a number of critical characteristics of file sharing methods including universality, addressing, visibility, notification, and the differentiation between push- and pull-oriented sharing. They developed a prototype of a set of user interface features called a sharing palette, providing a platform for exploration and experimentation with new modalities of sharing.

Whalen et al. [23] conducted an online survey and follow up interviews at a medium-sized industrial research laboratory to address the issue of users' experience of file sharing and access control by gathering information on how and why people share files; the types of information shared; and how, when and why people limit access to those files. The results of the survey showed that email attachments were the most commonly used method for sharing files (98% of all responses), followed by network files sharing (55%), followed by commercial content management system (25%) and removable media (25%). Also their result indicated that 37% of respondents protect their shared files from friends and colleagues, and the methods used for restricting access to their sensitive files are: passwords; permissions/access control lists; physical controls (e.g., safeguard in office or on person); encryption ; obscurity (e.g, given files innocuous names, hidden directories); and deleting/relocating sensitive files. Based on the results of the study, the authors suggest guidelines to improve methods for appropriate content protection.

In another study, Whalen et al. [9] conducted a web-based survey at a medium-size university to investigate the fundamental issues regarding how files are shared and the difficulties encountered when managing files in collaborative environments. From the result of their survey, they found that file sharing is a common activity, with over 70% of respondents share professional and personal files at least once per week . The file sharing methods used by their respondent are email attachments, physical devices (e.g., USB token, CD), networks file share, instant messenger (e.g., MSN, Yahoo), Web server (e.g., webpage, wiki), peer-to-peer (e.g., KaZaa), file copy protocol (e.g., scp,ftp). The most commonly-used file sharing method by their respondents is Email (42.7%) followed by network file share (14.7%) followed by peer-to-peer and file copy protocol (10.3%). This corresponds with the findings of Voida et al. [20] and their previous study [23]. Their results show that there are a number of positive and negative factors that have an impact on peoples choice of file sharing methods. The positive factors are: the convenience and the ease of use of the method, the widespread availability of the method in order to reach all recipients, and the suitability of the method to the organisation or task at hand. The negative factors are: the limit on file space or file size, lack of access control or security features and the inability to reach all recipients. Also, the result show that the majority of respondents share files between two and four groups, and 80% of respondents have sensitive files. These sensitive files were shared as the results indicated that 44% of respondents shared sensitive professional files and 11% of respondents shared sensitive personal files such as financial or medical information. The authors found that people utilise various methods to control access to their sensitive files, some are technical (passwords, permissions) and others are socially-controlled such as hiding files.

Unlike the study of Voida et al. [20] and Whalen et al. [23, 9] which focused on subjects within a single organisation, all of whom had access to similar, established file sharing methods, Dalal et al. [24] conducted in-depth interviews with respondents across various domains in their homes, home offices, or in cafes where people worked to examine how file sharing and access controls are used, not used or circumvented in order to get work done. The result of their study show that 80% of respondents shared files with overseas collaborators or clients in Europe and the Asia-Pacific region and 100% shared files with colleagues across the US. Their results showed differences between personal and professional sharing as they found that people in professional sharing concentrate on sharing files that are related to project work, such as shared documents included technical specification, meeting minutes and action items, proposals, reports. On the other hand, they found that people in personal sharing concentrate on sharing their experiences with others, and the content being shared (primarily multimedia) relational in nature, such as sharing photographs with with family members who live overseas. Email was used by all the respondents of their survey and 80% of them used various social software such as wiki, blogs, social networking sites (including MySpace and Facebook), public websites for sharing images and multimedia files (including Flickr, YouTube), and online forums and games. Moreover, their respondents made distinctions between two type of sharing which are sharing with oneself and sharing with others. Sharing files with oneself is very useful as it allows people to synchronise their activities regardless of their location, accessibility, or what devices are at hand. They found that USB drives and email are convenience and preferred methods for sharing with oneself. Analysing their results, they derived a set of design criteria for more effective file sharing system [24].

In contrast to previous studies which have focused on asking users themselves to report on how they share and protect files, Smetters and Good [25] conducted an automated survey of access control in a medium-sized corporation to collect behavioural data over time by analysing digital record of actual user behaviour as they believe that users' self-descriptions of their own behaviour can be incomplete or inaccurate. They used automated data mining to examine how users in a medium-sized corporation utilise two common access control features: the definition of access control groups, and the permissions settings, or ACLs, that users set on folders and documents.

They found that access control polices which are applied by users to their content are quite complex. Based on the results of their study, they derived a number of suggestions for the design of both access control systems themselves, and the interfaces used to manage them [25].

Mazurek et al. [26] conducted semi-structured interviews with 33 non-technical computer users in 15 households to examine the current access control attitudes, needs, and practices of home users when they share files inside and outside their homes. They found that people utilise a wide range of measures to restrict access to their files, some of them are standard access-control tools while others are ad-hoc tools. These tools are the same as those reported in [23] which are user accounts, password, encryption, limiting physical access to devices, and hide and delete sensitive files. They found that people have complex policies that ever-changing over time which are inadequately addressed in current file sharing and access control methods; a finding supported by Olson et al. [21, 22], Whalen et al. [23, 9], and Voida et al. [20]. Based on the results of their study, they have generated several guidelines for developers of access control systems aimed at home users.

Hart et al. [27] surveyed 23 blogging and social networking sites such as Blogger, Facebook, Flickr, YouTube, and MySpace to determine what access control and privacy features are currently available. They found that a lot of content-sharing sites provide primitive access control mechanisms which make a file entirely private or public while others allow more flexible control by offering private/friends/public access control model. The authors asserted that these models failed to support people's needs, and thus, proposed a method of access control for content-sharing sites that specify access control polices in terms of the content being mediated. Whalen et al. [28] pointed out that a potential solution for file sharing problems, such as exposing sensitive files accidentally, is to provide the user with clear information about file sharing settings and activities. Therefore, they explored existing research on awareness in collaborative environments, and used it to develop a framework for file sharing awareness. The authors used this awareness framework to develop a prototype for a file manager that facilitates file sharing by making sharing activity and settings more visible to the user.

Table 1 summarises the results of the above studies of file sharing with respect to answering the following fundamental questions: with whom the file is shared, what type of file is shared, how the file is shared and protected. The previous studies investigated these questions in details and provided valuable answers which could lead to better design of file sharing methods and access control models. However, the question of how the file is shared has been answered improperly. They merely answered the question of how people share their files by enumerating the methods of sharing files that people utilised. Such answers are applicable to the question of what methods people utilise to share their files rather than how the files are shared as we believe that the files can be shared in different ways using the same method.

3 CLASSIFYING the INSIDER PROBLEM

By surveying the previous work of insider security, we argue that the insider problem is significant and that no single definition can encompass the problem as a whole, which most researchers attempt to do. In the literature, insiders have always been defined and differentiated from outsiders by either being inside the network perimeter, trusted, authorised, or knowledgeable of the information system. Definitions based on these factors are either ambiguous or insufficient. To make progress and find a solution to the insider problem, we suggest that the problem should be classified into several categories which can be defined, studied and solved independently and which later can be combined to solve the problem as a whole. There are three factors which play an important role in classifying the insider problem which are: the type of activity that deals with an asset in an organisation; the type of asset that

Table 1: Summary of previous studies on file sharing

	With whom the file is shared	What type of file is shared	How the file is shared	How the file is protected
Olson et al. [21]	-The public, co-workers, managers and trusted co-workers, family and spouse.	-Email content, credit card number, transgression, work related documents, work email and desk phone number.	-	-
Voida et al. [20]	-Similar to Olson et. al.- With an average of 7 individuals or group	-34 different types of files e.g. business documents, paper drafts, music, ideas, schedules, and TV show	-Email (43%), shared network folders (16%) and posting content to a web site (11%)	-
Whalen et al. [9]	-Over 69% shared with two to four groups such as friends, family, research group, general public and colleagues. -25% shared with five to twenty groups.	-Only focused on sensitive files, such as email, personal financial or medical information, professional data or documents of an organisation, professional data or documents governed by law.	-Email (42%), shared network folders (14.7%), peer-to-peer program (10.3%) and file copy protocol (10.3%)	Various methods to control access to their sensitive files, some are technical (passwords, permissions) and others are socially-controlled such as hiding files.
Whalen et al. [23]	-	-	-Email (98%), shared network folder (55%), commercial content management systems (25%) and portable devices (25%)	Passwords; permissions/ access control lists; physical controls (e.g., safeguard in office or on person); encryption; obscurity (e.g., given files innocuous names, hidden directories); and deleting/relocating sensitive files.
Dalal et al. [24]	-With employees in professional sharing -With friends and family in personal sharing.	-In professional sharing: revolve around project work such as technical specifications, meeting minutes, and action items, proposals, reports.-In personal sharing: revolve around multimedia relational in nature such photograph and video.	Email (100%), - 80% used a wide variety of social software, such as wikis, blogs, social networking sites (including MySpace and Facebook) hosted services (such as Yahoo! Briefcase) public websites for sharing image and multimedia files (including Flickr and YouTube) and online forums and games.	-
Mazurek et al. [26]	-Family, friends, co-workers and strangers.	-Music, photo, video, private documents, school work, work files, and other personal documents.	-	-User accounts, password, encryption, limiting physical access to devices, and hide and delete sensitive files.

needs to be protected; and the type of attack that targeted the asset.

The activity. The activities are identified by the organisation for its partners, contractors, and employees to perform a particular job and might be different from one organisation to another. The activity will differentiate insiders from outsiders as an insider will be a person who is a legitimately given an activity by an organisation to perform a particular job. Therefore, the activity will lead to identifying who is the insider and what the insider is doing. The type of activity that insiders perform in an organisation are various and organisation-specific. Examples of activities that are given to insiders are file sharing, updating customer information, installing software to organisation's devices, setting up organisation's network, provisioning authorisation credentials to organisation's employees, etc.

The asset. The assets that need to be protected are identified by an organisation based on a clear description of activities in the organisation, such that each activity will involve one or more assets to deal with. For example, if an activity in an organisation is employees sharing files with each other, the asset will be the file being shared which contains sensitive information. Another examples of activities and assets are an IT administrator who provisioning authorisation credentials to an organisation's employees where the asset here is the authorisation credential, a software developer who writes software scripts to an organisation computer where the asset can be the software itself or the computers that run the scripts, a network administrator who sets up the organisation's network and maintains it where the asset is the network.

Generally, the assets can be of three types which are the network which connects devices together, the devices which contains the data, or the data itself.

The attack. The attacks that targeted the asset can be generally of three types which are availability attacks, confidentiality attacks and integrity attacks, each of which can be performed in different ways which might require either physical security or IT security. Choosing which type of attacks to prevent is determined by the type of protection required for the chosen asset. For instance, if the asset is the network which needs to be available all the time, availability attacks should be prevented. On the other hand, if the asset is data that needs to be secret, confidentiality attacks should be prevented and so on. Therefore, the asset will determine which type of attacks should be prevented.

Based on these three factors, we can define the insider precisely as a person who is legitimately given an activity by an organisation to deal with the organisation's assets, and define the insider problem as particular types of attacks that performed by insiders on particular types of assets of an organisation during particular types of activities. Therefore, we can classify the insider problem into several categories based on these three factors such that each particular type of attack by insiders on a particular type of asset of an organisation during a particular type of an activity will result in a unique class of the insider problem which can be defined, studied and solved independently. For example, one class of the insider problem is preventing confidentiality attacks on sensitive files by employees when they share them with each other. Another class might be preventing availability attacks on an organisation's network by IT administrators when they maintain it, or preventing integrity attacks on customers information by employees when they update them etc.

Our concern in this paper is not to classify the insider problem thoroughly, rather we have provided an approach for such classification. However, we are interested in one class of the insider problem which is related to file sharing. The activity in our class is file sharing, the asset is the files being shared, and the attacks we are concerned with are confidentiality and integrity attacks. Thus, we define our class of the insider problem as preventing confidentiality and integrity attacks on sensitive files when employees share them with each other.

Since file sharing is not only an activity that is performed by an organisation's employees but also it is an activity that can be performed among friends, family members, or colleagues, we will look at this class of the insider problem from broader perspective to include any individuals performing such activity. In other words, the insider in our class will be the recipients whether those recipients are employees, friends, or family members.

4 PROTECTING SHARED FILES

Although we defined our class of the insider problem in the previous section, the attacks we are concerned with (i.e. confidentiality and integrity attacks) are still vague. These attacks can be performed in different ways, which in turn requires different types of protections. Claiming that a particular protection mechanism can protect the confidentiality of the files is not enough. Instead, one should claim that a particular protection mechanism can protect the confidentiality of the files under specific kinds of attacks. Therefore, In order to protect the confidentiality and the integrity of the shared files from the insiders (i.e. recipients), the different attacks and misuses that affect the confidentiality and the integrity of shared files must be identified.

Generally, protection of the shared files can be realised from two different angles: protecting the shared files while in transit, and protecting the shared files when they are received by the recipients. In this section we characterise the protections required by the shared files against different types attacks and misuses that are performed during the activity of file sharing.

4.1 Protecting the Shared Files in Transit

This type of protection prevents attacks on the file while it is transferred from the owner to the recipients. We divided these attacks into confidentiality attacks and integrity attacks as follows:

Confidentiality attacks. These attacks lead to the disclosure of the shared files to unauthorised users and can performed in two ways. First, someone eavesdrops or monitors the communication between the owner and the recipient to obtain knowledge about the files. We refer to such attacker *Interceptor*. Second, someone pretends to be the original recipient to deceive the owner and obtain the files. We refer to such attacker *Masquerader*. Therefore, there should be two types of protections to prevent unauthorised disclosure of the shared files in transit as follows. Protecting the confidentiality of files from interceptor and protecting the confidentiality of files from Masquerader.

Integrity attacks. These attacks lead to unauthorised modification to the shared files by unauthorised users. The attacker in such attacks pretends to be the original owner to deceive the recipient by sending them files as if they were originated by the original owner. These files can either be an entirely new files or modified version of the original files. We refer to such attacker *Masquerader*. Therefore, there should be one type of protection to prevent unauthorised modification of the shared files in transit which is protecting the integrity of files from Masquerader.

4.2 Protecting the Shared Files at the Recipient

This type of protection prevents misuses on the file after it has been received by legitimate recipients. These misuses can affect the confidentiality and integrity of the files. Such misuses can be committed by three different entities which are *Malicious* recipients, *Naive* recipients or *Masqueraders*. *Malicious* recipients are untrusted legitimate recipients who deliberately misuse the shared files. *Naive* recipients are trusted recipients who accidentally misuse the shared files. *Masqueraders* are unauthorised users who accidentally acquire a device of a trusted legitimate user which contains the shared files and misuse these files. Therefore, misuses can be deliberate which are committed by *Malicious* or accidental which are committed by *Naive* or *Masqueraders*.

Protection against *Malicious* and *Naive* recipients are different from protection against *Masquerader*. *Malicious* and *Naive* recipients are already allowed to view the files, therefore, confidentiality of the files is achieved by not allowing them to redistribute the files to unauthorised users. Also, they may or may not be allowed to modify the files, therefore, integrity of the files is achieved by not allowing them to modify it in an unauthorised manner. On the other hand, *Masqueraders* are unauthorised users, therefore, confidentiality is achieved by not allowing them to view or redistribute the files, and integrity is achieved by not allowing them to modify the files.

Moreover, protection against *Naive* recipients is different from protection against *Malicious* recipients. The former is trusted to not redistribute or modify the files in an unauthorised manner, while the latter is untrusted and might strive to circumvent any protection to misuse the files. Therefore, we divided misuses which can be committed by the three entities broadly into confidentiality misuses and integrity misuses as follows:

Confidentiality misuses. Confidentiality misuses are those misuses which lead to the disclosure of the shared files to unauthorised users and which can be done in two ways. First, the shared file can be copied and sent to an unauthorised user through a file sharing method. Second, the device of a legitimate recipient which contains the shared file can be acquired by an unauthorised user, which we refer here to as a Masquerader, who discloses the shared files.

In the first case, the file can be redistributed in three ways. First, the file can be redistributed accidentally by a Naive legitimate recipient. Second, the file can be redistributed deliberately by a Malicious legitimate recipient. Third, the file can be redistributed accidentally by a Masquerader who found a device of a legitimate recipient unattended. In the second case, the file can be disclosed to Masqueraders in two ways. First, an unauthorised user steals the device of a Naive legitimate recipient. Second, a Malicious recipient lends his device to an authorised user.

Therefore, there should be five different types of protections to prevent unauthorised disclosure of the shared files at the recipients as follows. Protecting the confidentiality of files from accidental redistribution by Naive; protecting the confidentiality of files from accidental redistribution by Masquerader; protecting the confidentiality of files from deliberate redistributions by Malicious; protecting the confidentiality of files from accidental disclosure by Naive to Masqueraders; protecting the confidentiality of files from deliberate disclosure by Malicious to Masqueraders. Since the last two types of protection have similar impact which is disclosing the file to Masqueraders, we refer to them as protecting the confidentiality of files from accidental or deliberate disclosure to Masqueraders.

Integrity misuses. Integrity misuses are those misuses which lead to unauthorised modification to the shared files. Such unauthorised modification can be either modifying the shared files that do not allow any modification or modifying the shared file, that allowing partial modification, in an unauthorised manner. In both cases, the file can be modified in three ways. First, the file can be modified accidentally by a Naive legitimate recipient. Second, the file can be modified deliberately by a Malicious legitimate recipient. Third, the file can be modified accidentally by a Masquerader who found a device of a legitimate recipient unattended.

Therefore, there should be three different types of protections to prevent unauthorised modification of the shared files at the recipients as follows. Protecting the integrity of files from accidental modification by Naive; protecting the integrity of files from accidental modification by Masqueraders; protecting the integrity of files from deliberate modification by Malicious.

Below we classify the aforementioned protections into two types which are protection of files in transit and protection of the files at the recipients.

Protection of files in transit: this can be further divided into confidentiality protection and integrity protection.

- Confidentiality protection
  - Protecting the confidentiality of files in transit from *interceptor*
  - Protecting the confidentiality of files in transit from *Masquerader*
- Integrity protection
  - Protecting the integrity of files in transit from *Masquerader*

Protection of files at the recipients: this can be further divided into protection against accidental misuses when sharing with trusted recipient and protection against deliberate misuses when sharing with untrusted recipient.

Accidental misuse: this can be further divided into accidental misuse of confidentiality and accidental misuse of integrity.

- Accidental misuse of confidentiality:
  - Protecting the confidentiality of files at the recipients from accidental redistribution by *Naive*
  - Protecting the confidentiality of files at the recipients from accidental redistribution by *Masquerader*
  - Protecting the confidentiality of files at the recipients from accidental disclosure to *Masquerader*
- Accidental misuse of integrity:
  - Protecting the integrity of files at the recipients from accidental modification by *Naive*
  - Protecting the integrity of files at the recipients from accidental modification by *Masquerader*

Deliberate misuse: this can be further divided into deliberate misuse of confidentiality and deliberate misuse of integrity

- Deliberate misuse of confidentiality:
  - Protecting the confidentiality of files at the recipients from deliberate redistribution by *Malicious*
  - Protecting the confidentiality of files at the recipients from deliberate disclosure to *Masquerader*
- Deliberate misuse of integrity:
  - Protecting the integrity of files at the recipients from deliberate modification by *Malicious*

4.3 Summary

Figure 1 illustrates eleven types of protections that might be required to protect the files in transit and at the recipients. Protections of files in transit are concerned with preventing external attacks while protections of files at the recipients are concerned with preventing insider attacks.

The characterisation of the protections required by the shared files at the recipients, illustrates the different ways of how files can be misused by different types of insiders. This characterisation makes it clear which type of insider misuse needs to be prevented in a particular sharing scenario. For instance, misuses by Masqueraders need not to be prevented if the machine containing the file resides in a locked room where unauthorised users cannot access. Also, deliberate misuses by Malicious insiders need not to be prevented if the file is shared with trusted recipients. A major advantage of this characterisation is the avoidance of the chaos exists in the literature with respect to distinguishing insider attacks from external attacks, and between insiders attacks themselves. We listed different protections requirements to prevent different insiders misuses so that one can select the desired protection requirements for a particular sharing scenario and develop a mechanism to enforce it.

From our point of view, in order to protect a shared file against insiders attacks, it should only be shared with trusted insiders. Protecting the shared files from untrusted insiders who might strive to circumvent the protection is a dilemma for two reasons. First, each system has its own vulnerabilities and there is no system without vulnerabilities. Research efforts have proven that there is no system 100% secure against all deliberate attacks or misuses [29]. A brief look at the approach taken to protect commercial content, justifies this principle. Commercial content is protected by the use of Digital Rights Management systems that dictate how the content must be used by each individual. Although these systems are in place to protect commercial content, the content can still be obtained illegally in unprotected form. Second, the easiest way to circumvent any protection system used to protect confidential files is by exploiting the analog hole. All digital content must eventually be converted to human-perceptible form, known as the analog form, to be consumed by users. Once the digital content is converted to analog form, it will be in an unprotected form, and thus, it will be susceptible to unauthorised uses [30].

However, sharing a file with trusted insiders without any protection in place is risky. Even if insiders are trusted to not violate the content policy deliberately, there is a chance of accidental violation. According to a survey conducted by Infosecurity Europe and PwC on 1,402 UK companies, 36% of the worst security breaches in the year were caused by inadvertent human error [31]. Also, AngloSec conducted a survey on 197 network, security, and compliance professionals, and found that the greatest security concern is employees accidentally jeopardising security through data leaks or similar errors [32]. Therefore, there should be appropriate level of protection that prevents accidental misuses on the shared files by trusted insiders who do not misuse the files intentionally.

5 CHARACTERISING FILE SHARING

Although the different types of misuses on the shared files and the protection requirements are identified in the previous section, the activity of file sharing is still ambiguous. Some people conceive the activity of file sharing is to send an email attachment, while others conceive it as to make files available to others through peer-to-peer networks. Designing a mechanism that provides the various types of protection without taking into account how the activity of file sharing is performed, might be useless. This is pointed out by previous studies, where they showed that some people might avoid secure methods of file sharing and utilise insecure methods because it is more suitable for the task of sharing, although security is a concern for them. For instance, employees in organisations might be forced to utilise particular sharing methods because they are secure. However, since these methods have been built with only security in mind, they might not be suitable for the task of sharing that employees need to get their job done. Hence, employees usually tend to utilise other sharing methods that might be insecure to avoid obstacles found in secure methods, and hence, putting organisation's confidential files at risk. To avoid such problem, the different ways of performing the activity of file sharing should be considered when designing a protection mechanism, so that the protection mechanism will not only protect the shared files but also allow various task of sharing to be performed.

The activity of file sharing is performed by individuals for various purposes (e.g. professional or personal). The purpose of performing the activity of sharing makes it obvious to whom the files should be shared with (e.g. family, friends, colleagues, or anyone), which type of file to be shared (e.g. music, photo, video, business documents, etc.), and which method of sharing to be utilised that is possibly the most satisfied to the sharing purposes (e.g. secure, convenient, available to everyone etc.). These factors are discussed in literature and summarised in Table 1.

However, there two factors that are clearly affected by the purposes of sharing and which are overlooked by previous studies. These factors are file propagation and access which can be different based on the sharing purpose. To the best of our knowledge, we are not aware of any work that characterises the activity of file sharing based on these two factors. Therefore, In this section we characterise the activity of file sharing based on how files can be propagated and how files can be accessed after their propagation.

5.1 How files are propagated

5.1.1 Publish vs. Share:

Files can be propagated in two main ways depending on their sensitivity. Confidential files are only released to selected individuals while non-confidential files are released to everyone. Available file sharing methods can either allow people to share files with selected individuals (suitable for confidential files) or allow people to share files with everyone (suitable for non-confidential files). A few file sharing methods provide both options. We will use the term *share* to refer to a file that is released to selected individuals and the term *publish* to a file that is released to everyone. Publishing or sharing files can be performed in different scenarios. Therefore, we use the following terminology to characterise the different ways of how files are shared and published.

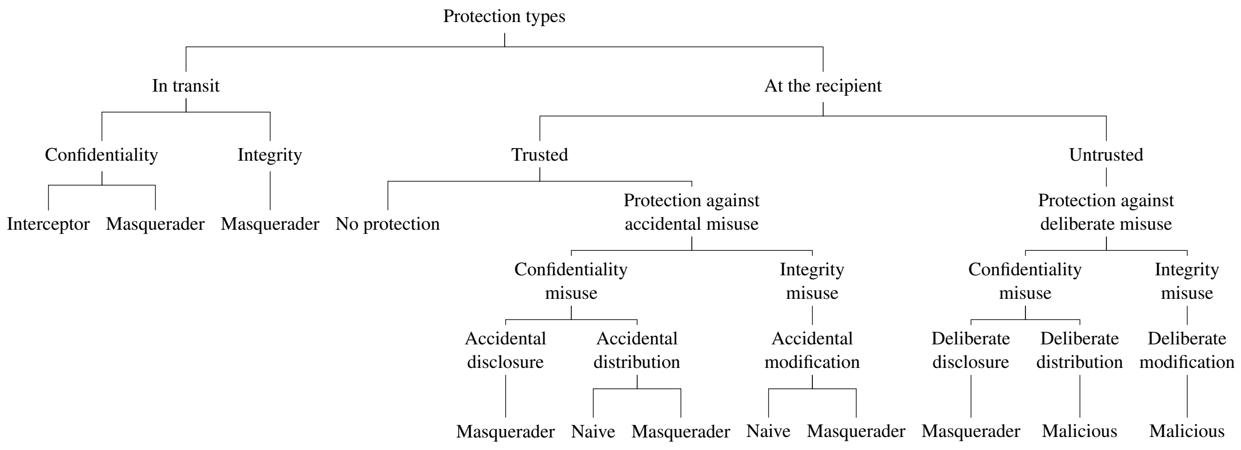


Figure 1: Types of protection of the shared files

Terminology:

- $\bar{O}$: a particular owner of files who might or might not be known in advance.
- $\bar{InG}$: a set of owners of files whom their numbers and identities are known in advance and share their files with each other.
- $\bar{G}$: a set of owners of files whom their numbers and identities are known in advance and do not share their files with each other.
- $\bar{M}$: a set of owners of files whom their number and identities are not known in advance and do not share their files with each other.
- O: a particular recipient who is known in advance.
- G: a set of recipients whom their number and identities are known in advance and whom receive the same copies of the shared files.
- M: a set of recipients whom their numbers and identities are not known and whom receive the same copies of the shared files.

In general, files can be released either to O, G or M. However, the received files by the recipients who can be O, G and or M, might belong to $\bar{O}$, $\bar{InG}$, $\bar{G}$ or $\bar{M}$. Therefore, including $\bar{InG}$ as a category of sharing we have 11 different categories that can describe all the possible ways of how files are shared or published as described below.

- $\bar{O} \rightarrow O$ (OneToOne): This describes a situation when a particular owner of files wants to share his files with a particular recipient who is known in advance. For example, Alice wants to share her file only with Bob but no one else.
- $\bar{O} \rightarrow G$ (OneToGroup): This describes a situation when a particular owner of files wants to share his files with a set of recipients whom their numbers and identities are known in advance, and whom receive the same copies of the shared files. For example, Alice wants to share her file only with her colleagues Bob, Carol, and Dave but no one else.
- $\bar{O} \rightarrow M$ (OneToMany): This describes a situation when a particular owner of files wants to share his files with a set of recipients whom their numbers and identities are not known, and whom receive the same copies of the shared files. For example, Alice wants to share her file with everyone on the internet regardless of whom they are.
- $\bar{InG}$ (InGroup): This describes a situation when owners of files whom their numbers and identities are known in advance want to share their files with each other. For example, Alice, Bob, and Carol want to share their files only with each other but no one else.
- $\bar{InG} \rightarrow O$ (InGroupToOne): This describes a situation when a set of owners of files whom their numbers and identities are known in advance and share their files with each other, want to share their shared files with a particular recipient who is known in

advance. For example, Alice, Bob and Carol who are sharing their files with each other want to share these shared files only with their colleague Dave but no one else.

- $I\bar{n}G \rightarrow G$ (InGroupToGroup): This describes a situation when a set of owners of files whom their numbers and identities are known in advance and share their files with each other, want to share their shared files with a set of recipients whom their numbers and identities are known in advance, and whom receive the same copies of the shared files. For example, Alice, Bob and Carol who are sharing their files with each other want to share these shared files only with their colleagues in the same department but no one else.
- $I\bar{n}G \rightarrow M$ (InGroupToMany): This describes a situation when a set of owners of files whom their numbers and identities are known in advance and share their files with each other, want to share their shared files with a set of recipients whom their numbers and identities are not known and whom receive the same copies of the shared files. For example, Alice, Bob and Carol who are sharing their files with each other want to share these shared files with everyone on the internet regardless of whom they are.
- $\bar{G} \rightarrow O$ (GroupToOne): This describe a situation when a set of owners of files whom their numbers and identities are known in advance and whom do not share their files with each other, want to share their files with a particular recipient who is known in advance. For example, Alice, Bob and Carol who work in the same company want to share their files only with Dave who is their employer but not with each other or anyone else.
- $\bar{G} \rightarrow G$ (GroupToGroup): This describe a situation when a set of owners of files whom their numbers and identity are known in advance and whom do not share their files with each other, want to share their files with a set of recipients whom their numbers and identities are known in advance, and whom receive the same copies of the shared files. For example, Alice, Bob and Carol who work in the same company want to share their files only with employees of the HR department but not with each other or anyone else.
- $\bar{M} \rightarrow O$ (ManyToOne): This describes a situation when a set of owners of files whom their numbers and identities are not known in advance and whom do not share their files with each other, want to share their files with a particular recipient who is known in advance. For example, applicants to a particular job want to share their documents files only with Alice who is the employer but no one else.
- $\bar{M} \rightarrow G$ (ManyToGroup): This describes a situation when a set of owners of files whom their numbers and identities are not known in advance and whom do not share their files with each other, want to share their files with a set of recipients whom their numbers and identities are known in advance, and whom receive the same copies of the shared files. For example, applicants to a particular job want to share their documents files only with Alice, Bob and Carol, who are the employees responsible for recruiting new staff, but no one else.

Figure 2 illustrates these categories and classifies them to either publish or share. Note that we excluded situations that do not make sense such as $M' \rightarrow M$ and $G' \rightarrow M$, since any of the owners can be of the recipients and vise versa.

5.1.2 Static vs. Dynamic vs. Transfer mode

In any of the categories of files propagation described above, files can be moved from an owner to a recipient differently. For instance, the original file can be moved physically as an object in real world, leaving no copies behind, or a copy of the original file can be moved to the recipient. In the latter case, the moved copy can be either a dynamic or a static. Below we describe each one of them.

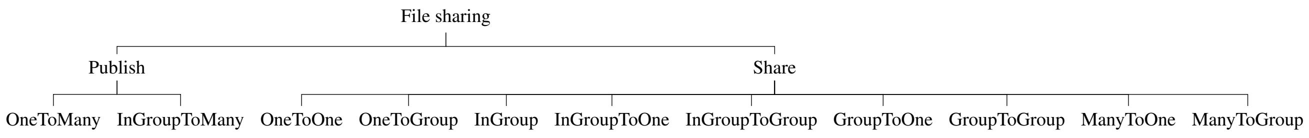


Figure 2: How files can be published and shared

Publishing or sharing in Static Mode: Publishing or sharing in a static mode describes a scenario where independent copies of the original file are moved from the owner to the recipients. Any changes made to the copies of the original file by the recipients or to the original file by the owner do not reflect on one another. It is useful when the owner of the file does not want to receive a new version of the published or shared files from the recipients or update the copies that the recipients have. An example of a method that allows sharing in a static mode is an email attachment where neither the owner nor the recipients can observe changes made on the shared files by others.

Publishing or sharing in Dynamic Mode: Publishing or sharing in a dynamic mode describes a scenario where copies of the original file that are linked to the original file are moved from the owner to the recipients. Therefore, any changes made to the copies of the original file by the recipients or to the original file by the owner do reflect on one another. It is adequate for a collaborative project where a group of members may work on a set of documents collectively. An example of a method that allows sharing in a dynamic mode is Dropbox where a file can be shared and updated by the owner or the recipients such that both can observe changes made to the shared files.

Publishing or sharing in Transfer Mode: Publishing or sharing in a transfer mode describes a scenario where the original file is moved, leaving no copies behind, from the owner to the recipients. The file is treated as a real world object that cannot exist at two places at the same time. Hence, in this mode, releasing a file to more than one recipient, requires the file to be held by one recipient at a time. We are not aware of any method meets this mode of publishing or sharing.

5.1.3 Distributed memory vs. Shared memory

Files can be moved from the owner to the recipients directly to their devices or indirectly to a location where recipients can access (e.g. server). We refer to the former as sharing or publishing in distributed memory (DM), and the latter as sharing or publishing in shared memory (SM). A file that is shared or published in DM, will be stored in each recipient device, allowing them to access the file when they are off-line. On the other hand, a file that is shared or published in SM, will be stored in a central location where recipients must access each time they need to access the file. Thus, unlike DM, a file in a SM requires the recipients to be online to get access to the file.

SM best suited for a file that is shared or published in a dynamic mode. Since the owner and the recipients have access to the same copy of a file, changes made to the file will be observed by others without the need to move copies of the file with the new changes to others. An example of a file that is published in a dynamic mode in SM is Wikipedia page. Also, SM can be suitable for transfer mode, such that a file in the central location can be accessed only by one recipient, who the file is transferred to, at a time. However, sharing or publishing in a static mode is not suitable for SM, particularly when the recipients are allowed to change the files. This because recipients of a file that is published or shared in a static mode are each intended to have a independent copy of the original file that can be changed without affecting the copies that other recipients have. Hence, this cannot be achieved in SM, since all recipients access the same copy of the file simultaneously.

On the other hand, DM can be suitable for all sharing or publishing modes (i.e. static, dynamic, and transfer). In static mode, independent copies of the original files are moved to the recipients devices, while in transfer mode, the original file is moved to one recipient device at a time. In case of a dynamic mode, copies of the original files are also moved to the recipients devices, however, the moved copies are linked to the original file, so that any changes made on them will be communicated to other copies.

Table 2 illustrates 33 types of files propagation. Each cell in the table marked with letter T indicates a way of propagating a file. For instance, OneToOne sharing can be performed in static (DM), dynamic (DM or SM) or transfer (DM or SM) mode. In other words, an independent copy of the original file can moved to one particular recipient device (static DM), a linked copy to the original file can be moved to one particular recipient device (dynamic DM) or a copy of the original file is moved to a location where one particular recipient can access (dynamic SM), or the original file is moved to one particular recipient device rather than a copy (transfer DM), or moved to a location where one particular recipient can access (transfer SM).

5.2 How files are accessed:

Once files are propagated, recipients need to access them. There are only two types of access that the recipients might need which are read and write access. The former allows them to read the file while the latter allows them to append or remove content from that file. However, an owner of a file might want to restrict these types of access based on the sharing or publishing purpose. For instance, the owner might want the recipients to: *a)* read but not edit the file. *b)* not read but edit the file by appending new content only. *c)* read and edit the file by appending or removing content. *d)* not read and not edit the file but just hold it (e.g. cloud storage providers). We refer to these access types as ReadOnly, WriteOnly, ReadWrite, and NoReadOrWrite, respectively.

Table 2: Types of propagation

Types of propagation	Static (DM)	Dynamic (DM or SM)	Transfer (DM or SM)
OneToOne	T	T	T
OneToGroup	T	T	T
OneToMany	T	T	T
InGroup	T	T	T
InGroupToOne	T	T	T
InGroupToGroup	T	T	T
InGroupToMany	T	T	T
GroupToOne	T	T	T
GroupToGroup	T	T	T
ManyToOne	T	T	T
ManyToGroup	T	T	T

Additionally, an owner of a file might find these types of access not restrictive enough for some sharing or publishing purposes. For instance, the owner might know that the recipients need only to read or edit the file *a)* for a limited number of times (e.g. only once). *b)* for a limited period of time (e.g. for three days starting from 1/9/2014). *c)* on a specific time (e.g. only Monday from 9am - 3pm). *d)* at a specific location (e.g. only in London). Therefore, these can be used as restrictions to further control the different access types mentioned above.

Table 3: Types of access

Types of access	Ln [1]	Lp [2]	St [3]	Sl [4]
ReadOnly	T	T	T	T
WriteOnly	T	T	T	T
ReadWrite	T	T	T	T
NoReadOrWrite	F	T	F	T

Table 3 illustrate 14 types of access the recipients might have. Each cell marked with letter T indi-

[1]Limited number of times
[2]Limited period of time
[3]Specific time
[4]Specific location

Table 4: Types of files propagation and access

How Files are		ReadOnly				WriteOnly				ReadWrite				NoReadOrWrite			
propagated and accessed		Ln	Lp	St	Sl	Ln	Lp	St	Sl	Ln	Lp	St	Sl	Ln	Lp	St	Sl
Share	Static	T	T	T	T	F	F	F	F	T	T	T	T	F	T	F	T
	Dynamic	T	T	T	T	T	T	T	T	T	T	T	T	F	T	F	T
	Ttransfer	T	T	T	T	F	F	F	F	T	T	T	T	F	T	F	T
Publish	Static	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
	Dynamic	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F
	Transfer	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F

cates a useful type of access that owners of files might want the recipients to have. Table 2 shows two unuseful types of access which are (NoReadOrWrite,Ln) and (NoReadOrWrite,St). This because restricting NoReadOrWrite access for a limited number of time or for a specific time does not make sense. However, the other two restrictions on NoReadOrWrite access (i.e. Lp and Sl) might be useful for some scenarios of sharing. For instance, an owner might want to share files with a cloud storage providers provided that the files are kept in the provider servers that are located at a particular geographical area. Another owner might want the files to be kept at the provider servers until a particular point of time after which the provider will not be authorised to keep the files in the servers.

It should be noted that recipients can only have one type of access, however, various restrictions can be used to restrict that type of access. For instance, an owner might want the recipients to have the following type of access: (ReadOnly, Lp, Sl) which allows the recipients to read the file for a limited period of time and at specific location. These two restrictions should be satisfied in order for the recipients to read the file. Also, there is a difference between having no type of access at all and having NoReadOrWrite type of access. The former disallows holding the file, while the latter allows holding the file but not reading or editing it.

Table 4 combines the different types of files propagation and access and identifies the useful combinations of these types. Each letter T in the table identifies a useful a way of propagating and accessing a file by the recipients. The term *Share* and *Publish* can be replaced with any of the categories of files propagation depicted in Figure 1. As shown in the table, not all types of access are suitable for all types of files propagation (i.e. not all combinations of files propagations types and access types are applicable). For instance, it is not sensible for the recipients to have WriteOnly type of access for a file that is shared or published in a static or transfer mode. Although the recipients will be able to add content to the file, no one can observe this content. Also, it is not useful to publish a file with NoReadOrWrite type of access, since there is no need to release the file to everyone and not allowing them to read it or edit it.

6 TAXONOMY BASED on the CHARACTERISATION of FILE SHARING

Based on the characterisation of the activity of file sharing discussed in the previous section, we define a framework that can be used to classify the activity of file sharing in a systematic way. This framework is shown in Figure 3, will help to classify the activity of file sharing by distinguishing how files are propagated to and accessed by the recipients. Below is a brief description of the proposed framework.

The framework has a tree-based structure, where each level represents either a way of files propagation or files access. Paths of the tree are numbered. Therefore, specifying the path number for each level of the tree starting from the root down-

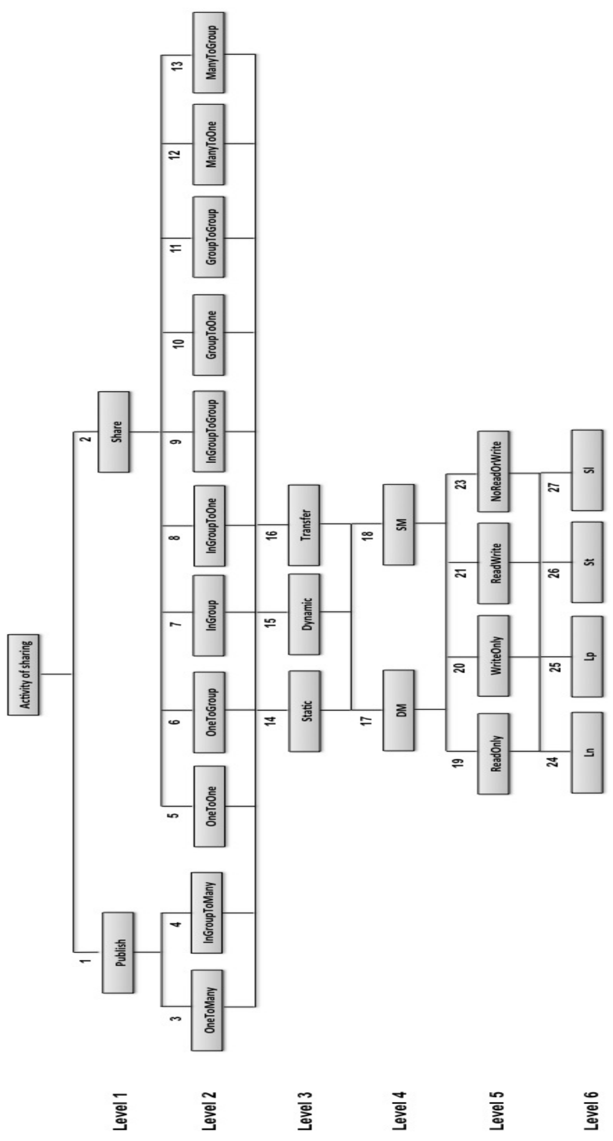


Figure 3: Framework for classifying the activity of file sharing

wards, will result in a unique class of the activity of file sharing. The first four levels after the root (i.e. paths from 1-18) represents types of files propagation, while the last two levels (i.e. paths 19-27) represents types of files access. At each level of the framework a unique choice has to be made. In this way every class of file sharing will form a single path in the tree. However, there is one exception: namely level six,"restriction over access types". Any class of file sharing can utilise none or multiple restrictions (e.g. Ln and Lp at the same time) over one type of access (e.g. ReadOnly) as described in the previous section.

Due to space limitations and to avoid redundant branches, not the entire tree is drawn. For instance, level two has eleven types to choose from, two types belong to path one, and nine types belong to path two. Each of these types has the same three possibilities for level three (i.e. Static, Dynamic and Transfer). Hence, at level three there are eleven identical groups of the three possible values. Therefore, to avoid using redundant branches, the types of level three are written once and can be used by all types of level two. On the other hand, each type of level one might have different possibilities of level five, and also from level three to six, each level might have different possibilities of its subsequent level. However, for simplicity and to save space, the maximum possibilities that each level should have is specified while table1, 2 and 3 can be utilised to help exclude those unuseful classes of sharing. For example, as shown in table 3, NoReadOrWrite access is not suitable for publishing. Hence, if the path 1 is chosen, regardless of the paths chosen for level 2, 3, and 4, path 23 in level 5 should be excluded. Otherwise, this class will be unuseful. Another example is when path 23 is chosen in level, paths 24 and 26 should be excluded as restricting the number of time and specific time on NoReadOrWrite access is unuseful.

6.1 Utilising the taxonomy of file sharing

The framework, depicted in Figure 3, can be utilised in two ways. First, the framework can be applied to classify the activity of file sharing, by showing different classes of how owners might want their files to be propagated and accessed for different sharing scenarios. Second, it can be applied to classify available file sharing methods, by showing which method provides which class of the sharing activity. Below two examples are discussed to illustrate how the framework can be utilised.

Example 1: classifying the activities of file sharing in an organisation. Alice has a start-up company consists of several departments which are Human Resource, Marketing, Production, Finance. Each department contains several employees. Employees within the same department and between different departments need to share files with each other to get their job done. Therefore, Alice wants to define how the activity of file sharing should be performed among employees.

Alice knows the Marketing department is responsible for dealing with customers. The Marketing department should send surveys to customers, however, Alice wants the surveys that should be sent to customers to be approved by the Manager of the department who is then responsible to move copies of the surveys to customers devices, so that customers can access them on their devices to read and edit them and move these copies back to the department if they are willing to do so. Hence, Alice specified the following class of file sharing for this department: 1-3-14-17-21.

Also, Alice knows that employees of the Production department, each should write a report and share it with other employees in the same department, so that each will be aware of others work and able to modify other reports in case mistakes are found. Alice wants employees to view and edit others reports when they are in their offices and during working hours. Hence, Alice specified the following class of file sharing for this department: 2-7-15-18-21-(26 + 27).

With respect to the Finance department, Alice knows that employees of this department write reports that should be viewed by employees of the Human Resource department in order for them to make a decision for recruiting new employees. However, Alice wants these reports to be approved by the Manager of the department who then is responsible to move copies of the reports to the company's server where employees of the Human Resource department can access. This will allow these reports to be updated by the Manager of the Finance department while employees of the Human Resource department will be able to view up to date reports. In addition, Alice wants employees of the Human Resource department to view these reports for a limited period of time and during working hours. Hence, Alice specified the following class of file sharing for this department: 2-6-15-18-19(25 + 26).

Finally, Alice who owns the company, needs to view a monthly reports written by each department manager. Alice does not want any manager to view reports written by other managers. Therefore, Alice specified the following class of file sharing between the managers and herself as follows: 2-10-14-17-19.

Example 2: Classifying file sharing methods There are various methods of file sharing exist today. Some of them have been designed merely for sharing files such as File Hosting Services, FTP, and Peer-to-peer file sharing, while others include file sharing as an added feature to their main purposes such as Emails and Social Networking Sites. Table 5 classifies some of the most popular file sharing methods based on the taxonomy described in the previous section. Each cell in the table shows which path the sharing method can take in each level of the framework. Below is a brief discussion of the classified methods in the table.

Email email is considered the most commonly-used method for sharing files. Although there is a few drawbacks of sharing files via an email such as limitation on file size, it is still popular method for sharing files at the present due to certain features. These features are the ease of use, the widespread availability and the suitability to various tasks. Almost anyone uses a computer owns an email account, and knows how to use it. Therefore, by using an email to share files, the user will avoid all the difficulties associated with other methods of file sharing such as ensuring that all recipients have the same method to be able to share the files or ensuring that all recipients know how to use the method of sharing especially if the method is quite complex and difficult to learn. Examples of emails are Hotmail,Yahoo, and Gmail.

- Level 1: since email requires an owner of a

Table 5: Classifying file sharing methods

File sharing methods	Emails	Peer-to-peer file sharing	Anonymous FTP	None-anonymous FTP	Cloud-storage services
Level 1	2	1	1	2	1,2
Level 2	5, 6, 7, 8, 9, 10, 11, 12, 13	3	3, 4	5, 6, 7, 8, 9	3, 4, 5, 6, 7, 8, 9, 10, 11
Level 3	14	14	14	14	14, 15
Level 4	17	17	17	17	17, 17
Level 5	21	21	21	21	19, 21
Level 6	-	-	-	-	-

file to enter the emails addresses of the recipients, which means that recipients should be known in advance, it is only suitable for sharing rather than publishing (i.e. path 2 in Figure 3).

- Level 2: email allows an owner of a file to share the file with a particular person or with a group of people, hence, files can be shared as OneToOne or OneToGroup. A group of owners can share their files with each other by email, as well as sharing their shared files with another person or group. Therefore, files can also be shared as InGroup, InGroupToOne and InGroupToGroup. Also, email allows a group of owners who do not share their files with each other to share their files with a particular person or a group of people. This group might or might not be known in advance to the recipients. Therefore, files can be shared as GroupToOne, GroupToGroup, ManyToOne, ManyToGroup. Hence, all paths (i.e. 5,6,7,8,9,10,11,12,13) of level two are applicable for sharing files by emails.
- Level3: email allows an owner of a file to only send a copy of the file to the recipient rather than the file itself. The copy received by the recipient is not linked to the original, therefore, any changes to the copy by the recipient will not be reflected on the original file. Hence, email allows sharing files in static mode only (i.e. path 14 in Figure 3).
- Level 4: since the copy that is sent to the recipient must be stored at the recipient device in order to be accessed, email allows sharing files in distributed memory rather than shared memory (i.e. path 17 in Figure 3).
- Level 5: email provides only one type of access to the recipients which is ReadWrite which allows the recipients to read and edit the received files (i.e. path 21 in Figure 3).
- Level 6: email provides no restrictions on the type of access the recipients have.

Peer-to-peer file sharing Peer-to-peer (P2P) file sharing applications have gained much attention in recent years. As its name suggests, P2P file sharing applications utilise P2P network. Unlike client-server network, P2P network consists of multiple computers (nodes) that are able to act as client and server in the same time. For instance, a node in a P2P network can send request to another node in the network while responding to requests from other nodes. Therefore, in P2P file sharing applications, files are not uploaded to a central server, instead, they are scattered across users devices which each of which can act as a client and server simultaneously. Examples of P2P file sharing applications are Napster, LimeWire, Shareaza, Kazaa,and BitTorrent.

- Level 1: P2P file sharing requires an owner of a file to use a P2P client to register the file to P2P network. Once the file is registered to the network, other users who use clients that connect them to the same network will be able to search and download that file. Therefore, it is suitable for publishing rather than sharing.
- Level 2: P2P file sharing allows an owner of a file to share the file with everyone on the network, therefore, files can be published only as OneToMany.
- Level 3: P2P file sharing allows an owner

of a file to publish an independent copy of the file to the recipients. Hence, it allows publishing in static mode.

- Level 4: Since the sent copies to the recipients will be stored at their devices in order to be accessed, P2P file sharing allows publishing files in distributed memory rather than shared memory.
- Level 5: P2P file sharing provides only one type of access to the recipients which is ReadWrite which allows the recipients to read and edit the received files.
- Level 6: P2P file sharing provides no restrictions on the type of access the recipients have.

Anonymous FTP Anonymous FTP allows anonymous access to the uploaded files on the FTP sever to anyone with an FTP client or even through a web browser. Most anonymous FTP servers allow anonymous users to download files from the server but no one can update the directory except the owner of the directory.

- Level 1: Since the uploaded files to the server are publicly available to be accessed by anyone with an FTP client, anonymous FTP is suitable for publishing rather than sharing.
- Level 2: Anonymous FTP allows a particular owner of files to publish the files to everyone, or a group of owners of files, who share their files with each other, to publish their files with everyone. Therefore, files can be published only as OneToMany or InGroupToMany.
- Level 3: Anonymous FTP allows an owner of a file to only publish an independent copy of the file to the recipients. Hence, it allows publishing in static mode.
- Level 4: Since the sent copies to the recipients must be stored at their devices in order to be accessed, Anonymous FTP allows publishing files in distributed memory rather than shared memory.
- Level 5: Anonymous FTP provides only one type of access to the recipients which is ReadWrite which allows the recipients to read and edit the received files.
- Level 6: Anonymous FTP provides no restrictions on the type of access the recipients have.

None-anonymous FTP Unlike anonymous FTP, non-anonymous FTP does not allow anonymous access to the uploaded files on the server. Users accessing non-anonymous FTP server will be prompted for a unique username and password which will be used as a basis for making decision whether to allow or deny the user access to the files. As a result, the owner of the directory can specify different operations that can be performed by each user such as view, update, delete, and execute a file in the directory. The differences between anonymous FTP and non-anonymous FTP is only in level 1 and 2.

- Level 1: Unlike anonymous FTP, since the users in non-anonymous FTP are prompted for a unique username and password, which means that not anyone can access the files, non-anonymous FTP is suitable for sharing rather than publishing.
- Level 2: Non-anonymous FTP allows an owner of a file to share the file with a particular person or a group. Also, it allows a group of owners of files to share their files with each other as well as sharing their shared files with a particular person or group. Therefore, files in non-anonymous FTP can be shared as OneToOne, OneToGroup, InGroup, InGroupToOne, InGroupToGroup.

Cloud-storage services Cloud-storage services allow the users to create storage accounts to store their files. Users are able to perform several operations on their storage accounts such as upload, download, delete, and share files. These operations can be performed by the users in two ways. First, through a web browser from any device. Second, through a proprietary software client that installed into their devices.

Cloud-storage services offer a synchronisation service, which means operations on a storage account made through a browser will be reflected in the installed client of that account and vice versa. Also, users who own several devices (e.g., laptop, tablet, smartphone) can install a client into each device to synchronise the files stored in their storage accounts across their devices. Examples of cloud-storage services are Dropbox, Google Drive, Microsoft's Skydrive.

- Level 1: Cloud-storage services allow users to share their files either with users subscribed to the same service or with users form the outside. Sharing files with other users subscribed to the same service requires an owner of a file to select a person or a group of people from the same service to share the file with and specify the operations that they can perform on the shared file (e.g., read and write). Since the file will be released only to users from the same services and to whom the owner has selected, it is suitable for sharing. On the other hand, sharing files with other users that are not subscribed to the same service, requires the owner of the file to generate URL for that file and distribute the URL to others. The URL can be distributed to everyone (e.g. posted in a public forum) or to a person or a group (e.g. via email). Therefore, Cloud-storage services are suitable for publishing and sharing.
- Level 2: Cloud storage services allow an owner of a file or group of owners of files who are sharing their files with each other to publish their file to every one. Also, it allows an owner of a file to share his file with a particular person or group and a group of owners to share their files with each other. Also, it allows a group of people who do not share files with each other to share their files with a particular person or group. Therefore, files in cloud storage providers can be published or shared as OneToMany, InGroupToMany, OneToOne, OneToGroup, InGorup, InGorupToOne, InGroupToGroup, GroupToOne, and GroupToGorup.
- Level 3: Cloud storage services allow an owner of a file to publish or share an independent or linked copies of the file to the recipients. Hence, it allows publishing and sharing and static and dynamic mode.
- Level 4: The published or shared copies of the files must be stored at the recipients devices to be accessed. Therefore, Cloud-storage services allow publishing and sharing files in distributed memory.
- Level 5: Cloud-storage services allow the recipients to have two types of access which are ReadOnly and ReadWrite.
- Level 6: Cloud-storage services provide no restrictions on the type of access the recipients have.

6.2 Summary

We used the characterisation of the activity of file sharing in Section 5 to define a framework that can classify all possible ways of performing the activity of file sharing. The framework depicted in Figure 3, can be used to specify different policies for different sharing scenarios to meet the protection requirements of the shared files discussed in Section 4.2. For example, levels 5 and 6 of the framework are concerned with Read and Write operations and is useful to specify policies to protect the file against accidental modification by Naive or Masquerader and accidental disclosure to Masquerader. To protect the file from accidental modification by Naive, the file should be shared with ReadOnly or NoReadOrWrite type of access, so that Write operation cannot be performed on the file. To protect the file from accidental disclosure to and accidental modification by a Masquerader, ReadOnly, WriteOnly, and ReadWrite types of access should be more restricted. For example, they can be restricted to be exercised only on a specific time (e.g. working hours) or location (e.g. office building), so that whether the device of a legitimate user is stolen from home or found unattended outside working hours, Read and Write operations cannot be performed.

One the other hand, levels 1 to 4 of the framework are concerned with Send operation and is useful to specify policies to protect the file against accidental redistribution by Naive or Masquerader. For example, Send operation on a class of sharing such as share-OneToOne-Static-DM, can only be performed successfully if the file is sent to only one user who is either the owner of the file or an authorised recipient, and the copy to be sent is not linked to the original file, and must be moved to the recipient device. Therefore, any attempt to send the file to a group of people or to one user who is neither the owner nor the authorised recipient will fail.

Since the framework can classify all possible ways of sharing files, having a mechanism that enforces all classes of the framework will ensure that files will not only be protected against insider misuses but also files will be shared as their owners desired.

7 CONCLUSION and FUTURE WORK

This paper has studied one category of the insider threat problem that is concerned with file sharing. In particular, protecting the shared files against insider misuses. In this paper we investigated three fundamental questions to the design of a protection mechanism against insiders misuses. Since the insider problem is not well-defined in the literature and the insider is not clearly identified, we have proposed a classification to the insider threat problem and defined the insider and the insider threat problem precisely. Having defined the insider problem and identified the insider precisely, is the first step towards protecting the shared files against insiders. More importantly is identifying misuses that insiders might perform on the shared files. We have looked at the different insiders misuses on the shared files and characterised the protection requirements of the shared files against them. However, in order to provide a useful protection mechanism that will not interfere with the activity of file sharing, we need to consider how the activity of file sharing can be performed. Otherwise, such protection mechanism might be abandoned because it is not suitable for various task of sharing. Therefore, we have characterised the activity file sharing, motivated from the development of extensive use-case scenarios. From the characterisation we have defined a framework that can classify all possible ways of how the activity of file sharing can be performed. Such framework can be used to specify different policies for different sharing scenario to meet the protection requirements of the shared files.

We are currently working to develop a mechanism to enforce the different protections types against accidental misuses on the shared files (See Figure 2). Since software is the major cause of many breaches in security, a promising approach to create a secure software is to write it in a typesafe programming language. Therefore, we take a type-based approach to enforce security policies which is a language-based technique to provide security in programs. Generally, system security requirements can be divided into two concerns which are access control and information flow control. The former places restrictions on the release of the resources, while the latter on its propagation. Access control requirement can be specified by our characterisation of how files can be accessed while information flow control can be specified by our characterisation of how files can propagated. Next, we formalise our approach using a type system to formally analyse access control and information flow whereby our characterisation of file sharing are represented as security type annotations and access control and information flow polices are enforced through type checking to prevent accidental misuses.

REFERENCES

[1] S. Christensen. Introduction to file sharing services: An it-forensic examination of p2p clients, Accessed on [30/11/2014]. URL `http://www.yumpu.com/en/document/view/15379746/introduction-to-file-sharing-services-cacpdf`.

[2] IBM. The floppy disk, Accessed on [30/11/2014]. URL `http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/floppy/transform/`.

[3] M. S. Smith. The history of file sharing: Where did it begin?, Accessed on [30/11/2014]. URL `http://www.brighthub.com/computing/smb-security/articles/67395.aspx`.

[4] C. Nistor. File sharing - history, Accessed on [30/11/2014]. URL `http://www.pctips3000.com/file-sharing-history/`.

[5] Wikipedia. Timeline of file sharing, Accessed on [[30/11/2014]. URL `http://en.wikipedia.org/wiki/Timeline_of_file_sharing`.

[6] Software Engineering Institute. 2011 CyberSecurity Watch Survey. Software Engineering Institute, Carnegie Mellon University, 2011.

[7] Matt Bishop and Carrie Gates. Defining the insider threat. In *In Proceedings of the 2008 Cyber Security and Information Infrastructure Research Workshop*, 2008.

[8] J. Hunker. Taking Stock and Looking Forward - An Outsider's Perspective on the Insider Threat. In S. J. Stolfo, S. M. Bellovin, A. Keromytis, S. Hershkop, S. W. Smith, and S. Sinclair, editors, *Insider Attack and Cyber Security - Beyond the Hacker*, Advances in Information Security. Springer, 2008.

[9] T. Whalen, E. Toms, and J. Blustein. File sharing and group information management. Workshop on Personal Information Management (PIM 2008), 2008.

[10] R. Anderson and R. Brackney. Understanding the insider threat. In *Proceedings of a March 2004 Workshop. Prepared for the Advanced Research and Development Activity (ARDA). `http://www.rand.org/publications/CF/CF196`*, 2004.

[11] M. Bishop. Position: "insider" is relative. In *Proceedings of the 2005 workshop on New security paradigms*, NSPW '05, pages 77–78, 2005.

[12] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya. Towards a theory of insider threat assessment. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, DSN '05, pages 108–117. IEEE Computer Society, Washington, DC, USA, 2005.

[13] K. Michelle and E. Kowalski. Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. May.

[14] J. Predd, S. L. Pfleeger, J. Hunker, and C. Bulford. Insiders behaving badly. *IEEE Security & Privacy*, 6 (4):66–70, 2008.

[15] C. W. Probst, J. Hunker, M. Bishop, and D. Gollmann. 08302 summary – countering insider threats. In Matt Bishop, Dieter Gollmann, Jeffrey Hunke, and Christian W. Probst, editors, *Countering Insider Threats*, number 08302 in Dagstuhl Seminar Proceedings. Dagstuhl, Germany, 2008. ISSN 1862-4405.

[16] J. Hunker and C. W. Probst. Insiders and insider threats: An overview of definitions and mitigation techniques. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(1):4–27, 2011.

[17] CERT. The CERT Insider Threat Center @ONLINE, April 2013. URL `http://www.cert.org`.

[18] S. M. Bellovin. The Insider Attack Problem Nature and Scope. In S. J. Stolfo, S. M. Bellovin, A. Keromytis, S. Hershkop, S. W. Smith, and S. Sinclair, editors, *Insider Attack and Cyber Security - Beyond the Hacker*, Advances in Information Security. Springer, 2008.

[19] G. Silowash, D. Cappelli, A. Moore, R. Trzeciak, T. J. Shimeall, and L. Flynn. Common Sense Guide to Mitigating Insider Threats 4th Edition, December 2012.

[20] Stephen Voida, W. Keith Edwards, Mark W. Newman, Rebecca E. Grinter, and Nicolas Ducheneaut. Share and share alike: exploring the user interface affordances of file sharing. In Rebecca E. Grinter, Tom Rodden, Paul M. Aoki, Edward Cutrell, Robin Jeffries, and Gary M. Olson, editors, *CHI*, pages 221–230. ACM, 2006. ISBN 1-59593-372-7.

[21] J. S. Olson, J. Grudin, and E. Horvitz. A study of preferences for sharing and privacy. In *Proceedings of CHI 05*, pages 1985–1988. ACM Press, 2005.

[22] J. S. Olson, J. Grudin, and E. Horvitz. Toward understanding preferences for sharing and privacy. MSR Technical Report 2004–138, 2004.

[23] T. Whalen, D. Smetters, and E. F. Churchill. User experiences with sharing and access control. In *In CHI 06: CHI 06 extended abstracts on Human factors in computing systems*, pages 1517–1522. ACM Press, 2006.

[24] B. Dalal, L. Nelson, D. Smetters, N. Good, and A Elliot. Ad-hoc guesting: when exceptions are the rule. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, UPSEC'08, pages 9:1–9:5, 2008.

[25] D. K. Smetters and N. Good. How users use access control. SOUPS '09. ACM.

[26] M. L. Mazurek, J. P. Arsenault, J. Bresee, N. Gupta, I. Ion, C. Johns, D. Lee, Y. Liang, J. Olsen, B. Salmon, R. Shay, K. Vaniea, L. Bauer, L. F. Cranor, G. R.

Ganger, and M. K. Reiter. Access control for home data sharing: Attitudes, needs and practices. In *CHI 2010: Conference on Human Factors in Computing Systems*, CHI '10, pages 645–654. ACM, New York, NY, USA, 2010.

[27] M. Hart, R. Johnson, and A. Stent. More content-less control: Access control in the web 2.0. *Control*, pages 1–3, 2006.

[28] T. Whalen, E. G. Toms, and J. Blustein. Information displays for managing shared files. In *Proceedings of the 2nd ACM Symposium on Computer Human Interaction for Management of Information Technology*, CHiMiT '08, pages 5:1–5:10. ACM, New York, NY, USA, 2008.

[29] K. Scarfone and P. Mell. The common configuration scoring system (ccss): Metrics for software security configuration vulnerabilities. Technical Report 7502, National Institute of Standards and Technology, December 2010.

[30] S. Haber, B. Horne, J. Pato, T. Sander, and R. E. Tarjan. If piracy is the problem, is drm the answer? In E. Becker, W. Buhse, D. Gnnewig, and N. Rump, editors, *Digital Rights Management*, volume 2770 of *LNCS*, pages 224–233. Springer, 2003. ISBN 3-540-40465-1.

[31] Infosecurity Europe and PwC. 2013 information security breaches survey. Technical report, April 2013.

[32] AlgoSec. The state of network security 2013: Attitudes and opinions, 2013.

International Journal of Cyber-Security and Digital Forensics (IJCSDF) 4(2): 380-389
The Society of Digital Information and Wireless Communications, 2015 (ISSN: 2305-0012)

A Secure Non-Repudiable General Proxy Signature

Samaneh Mashhadi[1] and Maryam Abdi

[1]Department of Mathematics, Iran University of Science & Technology, Narmak, Tehran, 1684613114, Iran
smashhadi@iust.ac.ir
ma.abdi88@gmail.com

ABSTRACT

A (t,n) threshold proxy signature scheme allows any t or more proxy signers to cooperatively sign messages on behalf of an original signer, but $t-1$ or fewer proxy signers cannot. In this paper, in order to adapt to some practical applications, by a modification of Hu threshold proxy signature, we propose a novel secure variation of proxy signature scheme called general proxy signature scheme. This scheme has some advantages over the threshold scheme, in the sense that the subsets of proxy group that allow to sign messages on behalf of an original signer, are not necessarily defined according to their cardinality and it becomes this proposed concept more attractive and practical when the members of a proxy group do not all have the same power or influence. Also, it is more efficient. This scheme is proved to enjoy security against the existing attacks.

KEYWORDS

Cryptology, Proxy signature, Threshold proxy signature, Access structure, General secret sharing scheme.

1 INTRODUCTION

A proxy signature scheme is a variation of the ordinary digital signature scheme [1, 3] which enables a proxy signer to generate signatures on behalf of an original signer. So far, many proxy signature schemes were discussed [9-16,19-21].
The multi-proxy signature scheme was first proposed in 2000 [12]. In a multi-proxy signature scheme, an original signer could authorize a proxy group as his proxy agent. Then only the cooperation of all the signers in the proxy group can generate the proxy signatures on behalf of the original signer. It can be regarded as a special case of a (t,n) threshold proxy signature scheme for $t=n$. In a (t,n) threshold proxy signature scheme any t or more proxy signers can cooperatively sign on behalf an original signer, but $t-1$ or fewer proxy signers cannot [9-16,19-21].
The schemes mentioned above, consider threshold access structures. That is, a proxy signature could be generated only if a certain number of proxy signers of the proxy group participate. However, there are situations in which the members of a proxy group do not all have the same power or the same probability to be dishonest. In these cases, access structures more general than the threshold ones must be considered.
To bridge this gap, in this paper, according to the idea of general secret sharing scheme, we show that the standard (t,n) threshold proxy signature schemes can be adapted to run in a more general scenario. A general proxy signature scheme can be applied in situations where an original signer wants to delegate some digital signing rights or capabilities to another one. For example, the central office of a bank can delegate to a branch office the capability to sign some kinds of documents (loans, mortgages) on behalf of the bank, whose secret key is distributed among members of the central office. The delegated capability will be distributed among members of the branch office. In real situations, these members do not all have the same power or influence within the office. Perhaps the policy of the bank is that a loan can be granted and signed by a branch office in the name of the bank only if the general manager and two other members of the branch office, or else any four members, agree.

This is the motivation in order to consider structures that are more general than the threshold ones mainly considered until now.

In a general proxy signature scheme, an original signer is allowed to delegate his/her signing power to a proxy group of proxy signers for shared signing responsibility. Then any authorized subset of the proxy group can collaborate to generate a valid signature on behalf of the original signer, but non-qualified subsets of participants cannot.

As an application of these protocols, we design a general proxy signature scheme, based on the (t,n) threshold proxy signature scheme proposed in [10]. There are various proposals of (t,n) threshold proxy signature scheme. We consider HZ scheme [10] because it has two main advantages over many previous (t,n) threshold proxy signature schemes:

1. Can resist the public-key substitute attack successfully by Zero-Knowledge Proof;
2. Is more efficient and secure than many other proxy signature schemes in terms of computational complexity.

Furthermore, new general protocol proposed in this paper, has two main advantages over the threshold one:

1. Is more general than the threshold case, in the sense that the authorized subsets of proxy signers are not necessary defined according to their cardinality. Therefore, it can apply in both general and threshold cases;
2. Computational complexity can be lowered.

Also, it inherits its security from the security of HZ scheme. All these properties, make our scheme more complete than the previous proposals of (t,n) threshold proxy signature schemes.

The rest of this paper is organized as follows: Definitions and notations of a general secret sharing are presented in Section 2. In Section 3, we briefly review HZ scheme [10]. We describe the new scheme in Section 5. The security properties and the performance of the proposed scheme are discussed in Sections 6 and 7, respectively. Finally, we draw our conclusions in Section 8.

2 PRELIMINARY

General secret sharing schemes are methods designed to split a secret among a group of participants in such a way that the secret can be reconstructed only by specified groups of participant (called authorized sets) while unauthorized groups of participants cannot do so [17].

Let $P = \{P_1, P_2, \dots, P_n\}$ be a set of elements called participants and let 2^P denote the set of all subsets of P. Suppose that Γ is a non-empty subset of 2^P. Then the closure of Γ, denoted by $cl\,(\Gamma)$, is the set

$$cl\,(\Gamma) = \{C \subseteq P \mid \exists B \in \Gamma,\ such that B \subseteq C\}. \quad (1)$$

We call Γ an *access structure* over P if it satisfies the monotone ascending property:

For any $A \in \Gamma\,, A' \in 2^P, A \subseteq A'$ implies $A' \in \Gamma$.

Obviously, if Γ is an access structure, then $\Gamma = cl\,(\Gamma)$ holds. The elements in Γ are usually called the *authorized sets*, and the elements in $\Delta = 2^P \backslash \Gamma$ are called the *unauthorized sets*. Δ is monotonically decreasing, that is:

For any $A \in \Delta, B \in 2^P,\ B \subseteq A$ implies $B \in \Delta$.

It is obvious that $\Gamma \cap \Delta = \emptyset$. Furthermore, $B \in \Gamma$ is a *minimum authorized subset* of Γ if $A \notin \Gamma$ whenever $A \subsetneq B$. The set of all minimal authorized subsets of Γ is denoted by Γ_0 and is called the *basis* of Γ. It is obvious that $\Gamma = cl\,(\Gamma_0)$. Actually, Γ and Γ_0 can be uniquely determined by each other.

For example, let $P = \{P_1, P_2, \dots, P_5\}$, and $\Gamma_0 = \{\{P_1, P_3\}, \{P_1, P_2, P_4\}\}$. Then,

$$\Gamma = \Gamma_0 \bigcup \{\{P_1,P_2,P_3\},\{P_1,P_3,P_4\},\{P_1,P_3,P_5\},\{P_1,P_2,P_3,P_4\}, ,\{P_1,P_2,P_3,P_5\},\{P_1,P_2,P_4,P_5\},P\}. \quad (2)$$

A particular class of general secret sharing schemes is that of (t,n) threshold schemes which were introduced independently by Blakley [2] and Shamir [18]. In a (t, n) threshold secret sharing scheme, any t or more than t participants can reconstruct the secret, but fewer than t participant cannot [4-8]. Therefore, (t, n) threshold access structure consists of all subsets of P with at least t out of n participants. That is, $\Gamma_0 = \{A \subseteq P| \ |A| = t\}$.

3 BRIEFLY REVIEW OF HZ SCHEME

In this section, we briefly review the HZ scheme [10]. HZ scheme is an improvement of Yang's threshold proxy signature scheme [20]. In [10] the authors show that Yang's scheme is not secure against the frame and public-key substitute attacks. Both of these schemes consider threshold structures; that is any subset of at least t proxy signers can compute a valid signature whereas $t-1$ or fewer proxy signatures cannot. HZ scheme is more efficient and secure than many other proxy signature schemes in terms of computational complexity. It consists of four phases: The initialization phase, the proxy share generation phase, the proxy signature generation phase and the proxy signature verification phase.

3.1 INITIALIZATION PHASE

Let p be a large prime, q a large prime factor of $p-1, g$ a generator in $\mathbb{Z}_p^*$ of order q, and $h(\cdot)$ a secure one-way hash function. The parameters (p, q, g) are public. Suppose that P_0 be the original signer and $P = \{P_1, P_2, \dots, P_n\}$ the proxy group of n proxy signers. Let m_w be a warrant which records the identities of the original signer and proxy signers of the proxy group, the parameters of t, n, and the valid delegation time, etc. Let ASID denote the identities of the actual proxy signers. The certificate authority CA requires the original signer P_0 and proxy signers $P_i (1 \le i \le n)$ offer the zero-knowledge proof of its private key about its public key as follows:

1. CA randomly chooses $x \in \mathbb{Z}_q^*$, computes $X = g^x \text{mod} p$, and then sends X to P_0 and each P_i.
2. P_0 chooses $x_0 \in \mathbb{Z}_q^*$, computes $y_0 = g^{x_0} \text{mod} p$, $X_0 = X^{x_0} \text{mod} p$, and send X_0 to CA.
3. P_i chooses $x_i \in \mathbb{Z}_q^*$, computes $y_i = g^{x_i}$, $X_i = X^{x_i} \text{mod} p$, and then each P_i send X_i to CA.
4. CA checks whether the qualities $y_0^x = X_0$ and $y_i^x = X_i \text{mod} p$ hold or not; if it does, CA accepts their certification; otherwise CA refuses.

3.2 PROXY SHARE GENERATION PHASE

In this phase the original signer P_0 executes the following steps to generate the proxy key σ:

1. P_0 randomly chooses an integer $a \in \mathbb{Z}_q^*$ and computes $A = g^a \text{mod} p$.
2. P_0 computes $\sigma = x_0 h(m_w, A) + aA \text{mod} q$ as the proxy group's key.

Then P_0 performs a (t, n)-verifiable secret sharing scheme to share the proxy key σ among n proxy signers in P.

1. P_0 chooses a $t-1$ degree polynomial in $\mathbb{Z}_q$

$$f(x) = \sigma + b_1 x + b_2 x^2 + \cdots + b_{t-1} x^{t-1} \text{mod} p \qquad (3)$$

where $b_i \in \mathbb{Z}_q^* (i = 1, 2, \dots, t-1)$. Then P_0 obtains each proxy signer P_i's public key y_i and computes $Y_i = f(y_i) \text{mod} p (1 \le i \le n)$ as each proxy signer P_i's secret shadow. He/she computes $C_i = g^{Y_i} \text{mod} p$ for $(1 \le i \le n)$ and $D_j = g^{b_j} \text{mod} p$ for $(1 \le j \le t-1)$.

2. P_0 sends (y_i, Y_i, m_w, A) to each proxy signer P_i via a secure channel and broadcasts D_j for $(1 \le j \le t-1)$, and C_i for $(1 \le i \le n)$.

After each proxy signer P_i receiving (y_i, Y_i, m_w, A) they check whether the following equality holds or not:

$$g^{Yi} = A^A y_0^{h(m_w, A)} \prod_{l=1}^{t-1} D_l^{y_i{}^l} \text{mod} p. \qquad (4)$$

If it does, each P_i accepts this proxy share; otherwise they refuse.

3.3 PROXY SIGNATURE GENERATION PHASE

Without loss of generality, let $D = \{P_1, P_2, \dots, P_t\}$ be the t actual proxy group, who want to cooperatively generate a proxy signature, m the message to be signed, and B the designated clerk.

1. Each $P_i \in D$ chooses random $a_i \in \mathbb{Z}_q^*$, computes and broadcasts $r_i = g^{a_i} \bmod p$.
2. After receiving $r_j (1 \le j \le t,\ j \ne i)$, each P_i computes $R = \prod_{j=1}^{t} r_j \bmod p$ and his partial signature

$$s_i = a_i R + (Y_i W_i + x_i) h(R, m, ASID, m_w) \bmod q,$$

where

$$W_i = \prod_{j=1 i \ne j}^{t} \frac{y_j}{y_j - y_i}. \quad (5)$$

Then, each P_i sends s_i to the designated clerk B.

2. After receiving s_i, B checks whether the following equation holds or not:

$$g^{s_i} = r_i^R (C_i^{W_i} y_i)^{h(R,m,ASID)} \bmod p. \quad (6)$$

If it does, B computes $S = \sum_{i=1}^{t} s_i$. Thus $(R, S, A, m_w, ASID)$ is the threshold proxy signature of the message m.

3.4 PROXY SIGNATURE VERIFICATION PHASE

When the verifier receives $(R, S, A, m_w, ASID)$, he check whether the following equality holds or not:

$$g^S = R^R (y_0^{h(m_w, A)} A^A \prod_{i=1}^{t} y_i)^{h(R,m,ASID)} \bmod p. \quad (7)$$

If it does, $(R, S, A, m_w, ASID)$ will be the valid proxy signature of the message m.

4 NEW PROPOSED SCHEME

In this section, we introduce a general proxy signature scheme based on the HZ threshold proxy signature scheme [5]. We consider general secret sharing instead of threshold one. That is, those subsets of proxy signers, authorized to sign messages, are not necessarily defined according to their cardinality. This new protocol has two main advantages over the threshold one:

1. Is more general than the threshold case, in the sense that the authorized subsets of proxy signers are not necessary defined according to their cardinality. Therefore, it can apply in both general and threshold cases;
2. Computational complexity can be lowered.

Also, it inherits its security from the security of HZ scheme. All these properties, make our scheme more complete than the previous proposals of (t, n) threshold proxy signature schemes. The new scheme can also be divided into four phases: The initialization, proxy share generation, proxy signature generation and proxy signature verification.

4.1 INITIALIZATION PHASE

The system parameters p, q, g, h are the same as those in Section 3.1. Suppose that P_0 be the original signer, and $P = \{P_1, P_2, \dots, P_n\}$ be the proxy group of n proxy signers. In our scheme, P_0 determines access structure of scheme Γ and basis $\Gamma_0 = \{F_1, F_2, \dots, F_m\}$, s.t., $F_i \subseteq P$ for $1 \le i \le m$. Let m_w be a warrant which records important information such as the identities of the original signer and proxy signers, Γ_0, the valid delegation time, etc. Let IDF_j and IDO, respectively, denote the identities of the proxy signers in minimal qualified subsets F_j and the identity of original signer. Similar to HZ scheme, CA requires the original signer P_0 and proxy signers $P_i (P_i \in P)$ offering the zero-knowledge proof of its private key about its public key as follows:

1. CA randomly chooses $x \in \mathbb{Z}_q^*$, computes $X = g^x \bmod p$, and then sends X to P_0 and each P_i.
2. P_0 chooses $x_0 \in \mathbb{Z}_q^*$, computes $y_0 = g^{x_0} \bmod p$, $X_0 = X^{x_0} \bmod p$, and send X_0 to CA.
3. P_i chooses $x_i \in \mathbb{Z}_q^*$, computes $y_i = g^{x_i}$, $X_i = X^{x_i} \bmod p$, and then each P_i send X_i to CA.
4. CA checks whether the quality $y_0^x = X_0$ and $y_i^x = X_i \bmod p$ hold or not, if it does, CA accepts their certification; otherwise CA refuses.

4.2 PROXY SHARE GENERATION PHASE

In this phase the original signer P_0 executes the following steps to generate the proxy key σ:

1. P_0 randomly chooses an integer $a \in \mathbb{Z}_q^*$ and computes $A = g^a \text{mod} p$
2. P_0 computes $\sigma = x_0 h(m_w, A, IDO) + aA \text{mod} q$ as the proxy group's key.

Then P_0 performs a general verifiable secret sharing scheme to share the proxy key σ among n proxy signers in P, according to access structure Γ. Therefore, P_0 for every minimal qualified subsets $F_j \in \Gamma_0$, executes the following steps:

1. Assume that $F_j = \{P_i\}_{i \in I_j}$ and $|F_j| = t_j . P_0$ randomly chooses a $t_j - 1$ degree polynomial in $\mathbb{Z}_q$

$$f_j(x) = \sigma + b_{j1}x + \cdots + b_{jt_j-1}x^{t_j-1} \text{mod} p \qquad (8)$$

where $b_{jk} \in \mathbb{Z}_q^*(k = 1,2, \dots, t_j - 1)$. Then P_0 obtains each proxy signers' public key y_i and computes $Y_{ji} = f_j(y_i) \text{mod} p (i \in I_j)$ as P_i's secret shadow. P_0 computes $C_{ji} = g^{Y_{ji}} \text{mod} p \text{for}\ i \in I_j$ and $D_{jk} = g^{b_{jk}} \text{mod} p$ for $1 \le k \le t_j - 1$.

2. P_0 sends (y_i, Y_{ji}, m_w, A) to each proxy signer $P_i \in F_j$ via a secure channel and broadcasts D_{jk} for $1 \le k \le t_j - 1$, and C_{ji} for $i \in I_j$.

After broadcasting (y_i, Y_{ji}, m_w, A), each proxy signer $P_i \in F_j$ can check whether the following equality holds or not

$$g^{Yji} = A^A y_0^{h(m_w, A, IDO)} \prod_{k=1}^{t_{j-1}} D_{jk}^{{y_i}^k} \text{mod} p. \qquad (9)$$

If it does, each P_i accepts this proxy share, otherwise they refuse.

4.3 PROXY SIGNATURE GENERATION PHASE

Let a qualified set of proxy signers $E \in \Gamma$ want cooperatively to generate a proxy signature. Let B be the designated clerk and m the message to be signed.

1. The members of E agree on a minimal qualified subset $F_j \subseteq A$. Assume that $F_j = \{P_1, P_2, \dots, P_{t_j}\}$.
2. Each $P_i \in F_j$ chooses random $a_{ji} \in \mathbb{Z}_q^*$, computes and broadcasts $r_{ji} = g^{a_{ji}} \text{mod} p$.
3. After receiving $r_{jk} (1 \le k \le t_j,\ k \ne i)$, each P_i computes

$$R_j = \prod_{k=1}^{t_j} r_{jk} \text{mod} p, \qquad (10)$$

and his partial signature

$$s_{ji} = a_{ji} R_j + (Y_{ji} W_i + x_i) h(R_j, m, IDF_j, IDO, m_w) \qquad (11)$$

Where

$$W_i = \prod_{l=1 i \ne l}^{t_j} \frac{y_l}{y_l - y_i} \qquad (12)$$
.

Then, each P_i sends s_{ji} to the designated clerk B.

3. After receiving s_{ji}, B checks whether the following equation holds or not

$$g^{s_{ji}} = r_{ji}^{R_j} (C_{ji}^{W_i} y_i)^{h(R_j, m, IDF_j, IDO, m_w)} \text{mod} p. \ (13)$$

If it does, B computes $S = \sum_{i=1}^{t_j} s_{ji}$. Thus (R_j, S, A, m_w, IDF_j) is the general proxy signature of the message m.

4.4 PROXY SIGNATURE VERIFICATION PHASE

When the verifier receives (R_j, S, A, m_w, IDF_j), he check whether the following equality holds or not:

$$g^S = R_j^{R_j} (y_0^{h(m_w, A, IDO)} A^A \prod_{i=1}^{t_j} y_i)^{h(R_j, m, IDF_j, IDO, m_w)}. \ (14)$$

If it does, (R_j, S, A, m_w, IDF_j) is the valid proxy signature of the message m. This is because

$$
\begin{aligned}
g^S &= g^{\sum_{i=1}^{t_j} a_{ji}R_j + (Y_{ji}W_i + x_i)h(R_j,m,IDF_j,IDO,m_w)} \\
&= (\prod_{i=1}^{t_j} g^{a_{ji}})^{R_j} (g^{\sum_{i=1}^{t_j} Y_{ji}W_i} \prod_{i=1}^{t_j} g^{x_i})^{h(R_j,m,IDF_j,IDO,m_w)} \\
&= R_j^{R_j} (g^{\sigma} \prod_{i=1}^{t_j} y_i)^{h(R_j,m,IDF_j,IDO,m_w)} \\
&= R_j^{R_j} (y_0^{h(m_w,A,IDO)} A^A \prod_{i=1}^{t_j} y_i)^{h(R_j,m,IDF_j,IDO,m_w)}. \quad (15)
\end{aligned}
$$

5 SECURITY ANALYSIS OF THE PROPOSED SCHEME

Our scheme is based on the HZ scheme, and inherits its security from the security of this scheme. In this section, we present a security analysis of our scheme. Before examining the security of our improved scheme, we give the following lemma and theorems.

Lemma 1 If $\prod_{i=1}^{t_j} y_i = g^c \bmod p$, and $A' = g^t = ((\prod_{i=1}^{t_j} y_i)^{-1} g^{\alpha})^{A'^{-1}} \bmod p$, then $A'^{-1}(-c+\alpha) = t \bmod q$.

Proof. Indeed, we have

$$
\begin{aligned}
g^t = A' &= ((\prod_{i=1}^{t_j} y_i)^{-1} g^{\alpha})^{A'^{-1}} \bmod p \\
&= (g^{-c+\alpha})^{A'^{-1}} \bmod p \\
&= g^{A'^{-1}(-c+\alpha)} \bmod p. \quad (16)
\end{aligned}
$$

Thus,

$$A'^{-1}(-c+\alpha) = t \bmod q. \quad \blacksquare$$

Theorem 2 $(R'_j, S', A', m_w, IDF_j)$ given by

$$
\begin{aligned}
R'_j &= g^{\beta} \bmod p, \\
A' &= ((\prod_{i=1}^{t_j} y_i)^{-1} g^{\alpha})^{A'^{-1}} \bmod p, \quad and \\
S' &= [\alpha + x_0 h(m_w, A', IDO)] h(R'_j, m, IDF_j, IDO, m_w) \\
&\quad + \beta R'_j \quad (17)
\end{aligned}
$$

is a valid proxy signature.

Proof. It is a valid proxy signature of the message m because

$$
\begin{aligned}
g^{S'} &= g^{\beta R_{j'} + [\alpha + x_0 h(m_w, A', IDO)] h(R_{j'}, m, IDF_j, IDO, m_w)} \\
&= g^{\beta R_{j'}} (g^{x_0 h(m_w, A', IDO)} \\
&\quad (\prod_{i=1}^{t_j} y_i)^{-1} g^{\alpha} \prod_{i=1}^{t_j} y_i)^{h(R'_j, m, IDF_j, IDO, m_w)} \\
&= (y_0^{h(m_w, A', IDO)} A'^{A'} \prod_{i=1}^{t_j} y_i)^{h(R'_j, m, IDF_j, IDO, m_w)} \\
&\quad R'^{R'_j}_j. \quad (18)
\end{aligned}
$$

$\blacksquare$

Theorem 3 If $y_1 = g^a y_0^{-h(m_w, A', IDO)} A'^{-A'} (\prod_{i=2}^{t_j} y_i)^{-1}$, and $F_j = \{P_1, P_2, \dots, P_{t_j}\}$, then $(R'_j, S', A', m_w, IDF_j)$ given by

$$
\begin{aligned}
R'_j &= g^b \bmod p, \quad and \\
S' &= bR_{j'} + ah(R_{j'}, m, IDF_j, IDO, m_w) \bmod q,
\end{aligned} \quad (19)
$$

is a valid proxy signature.

Proof. It is a valid proxy signature of the message m because

$$
\begin{aligned}
g^{S'} &= g^{bR_{j'} + ah(R_{j'}, m, IDF_j, IDO, m_w)} \\
&= g^{bR'_j} g^{ah(R_{j'}, m, IDF_j, IDO, m_w)} \\
&= g^{bR'_j} (y_0^{h(m_w, A', IDO)} A'^{A'} g^a y_0^{-h(m_w, A', IDO)} \\
&\quad A'^{-A'} (\prod_{i=2}^{t_j} y_i)^{-1} \prod_{i=2}^{t_j} y_i)^{h(R'_j, m, IDF_j, IDO, m_w)} \\
&= (y_0^{h(m_w, A', IDO)} A'^{A'} y_1 \prod_{i=2}^{t_j} y_i)^{h(R'_j, m, IDF_j, IDO, m_w)} \\
&\quad g^{bR'_j} \\
&= (y_0^{h(m_w, A', IDO)} A'^{A'} \prod_{i=1}^{t_j} y_i)^{h(R'_j, m, IDF_j, IDO, m_w)} \\
&\quad R'^{R'_j}_j \bmod p. \quad (20)
\end{aligned}
$$

$\blacksquare$

Now, we examine the security of our scheme.

1. It can be seen that an adversary cannot derive the P_0's private key x_0 from $X_0 = X^{x_0} \bmod p$ based on the DLP assumption. Similarly, the adversary cannot derive any P_i's private key x_i from $X_i = X^{x_i} \bmod p$.
2. Consider the scenario of a *frame attack* that a malicious original signer P_0 wants to forge a valid general proxy signature

$(R'_j, S', A', m_w, IDF_j)$ for his arbitrary chosen message m' and claim dishonestly that it is generated by a minimal qualified subset $F_j = \{P_1, P_2, \dots, P_{t_j}\}$. Let IDF_j be the identities of F_j.

For this purpose, P_0 can choose random integers $\alpha, \beta \in \mathbb{Z}_q^*$ and computes $R'_j = g^{\beta} \mathrm{mod} p$. Now, according to Theorem 2, P_0 should determine

$$A' = ((\prod_{i=1}^{t_j} y_i)^{-1} g^{\alpha})^{A'^{-1}} \mathrm{mod} p, \qquad (21)$$

And

$$S' = \beta R'_j + [\alpha + x_0 h(m_w, A')] h(R'_j, m, IDF_j, IDO, m_w). \qquad (22)$$

However, according to Lemma 1, P_0 should solve the discrete logarithms $\prod_{i=1}^{t_j} y_i = g^c \mathrm{mod} p$ and $A' = g^t \mathrm{mod} p$ in order to compute A'. Thus P_0 cannot forge a valid general proxy signature of any message m', which generated by F_j.

3. Consider the scenario of a *public-key substitute attack.* Without loss of generality, suppose that a malicious proxy signer $P_1 \in F_j$ s.t. $\{P_1\} \notin \Gamma_0$ tries to forge a general proxy signature scheme of a message m'. For this purpose, P_1 chooses random $a, b, A' \in \mathbb{Z}_q^*$ and according to Theorem 3 computes

$$R'_j = g^b \mathrm{mod} p,$$
$$S' = bR'_j + ah(R'_j, m', IDF_j, IDO, m_w), \ \ and$$
$$y_{1'} = g^a y_0^{-h(m_w, A', IDO)} A'^{-A'} (\prod_{i=2}^{t_j} y_i)^{-1}. \qquad (23)$$

Then he wants CA to replace his public key with the above y'_1; the certificate authority, CA, again asks P_1 for the Zero-Knowledge Proof of his private key x'_1 associated to new public key y'_1; but P_1 cannot obtain x'_1, s.t. $y'_1 = g^{x'_1} \mathrm{mod} p$, because of the difficulty of solving discrete logarithm problem. Hence, P_1 cannot again perform Zero-Knowledge Proof with CA when he changes his public key.

Similarly, the original signer P_0 or any proxy signer P_i can't forge a valid threshold proxy signature by the public-key substitute attack.

4. Consider the scenario of a *collusion attack* made by proxy signers. There are two cases.

   - Firstly, assume that proxy signers in a minimal qualified subset F_j want to conspire to sign a message m, and claim dishonestly that it is signed by another minimal qualified subsets $F_k, (j \neq k)$. Each proxy signer in F_j shows his secret shadow Y_{ji}. Next, they cooperatively reconstruct the secret polynomial function $f_j(x)$, while they can't obtain $Y_{ki} = f_k(y_i)$ for $P_i \in F_k$ since $f_k(x) \neq f_j(x)$. Also they can't obtain Y_{ki} and x_i from $C_{ki} = g^{Y_{ki}} \mathrm{mod} p$ and $y_i = g^{x_i} \mathrm{mod} p$, respectively because of the difficult problem of solving discrete logarithm. Therefore, they can't derive each P_i's partial signature

$$s_{ki} = a_{ki} R_k + (Y_{ki} W_i + x_i) h(R_k, m, IDF_k, IDO, m_w). \ (24)$$

   Therefore, proxy signers in F_j are unable to achieve collusion attack successfully.

   - Secondly, assume that proxy signers in an unauthorized set U of the proxy group conspire to derive σ and s_{kl} for $P_l \in F_k \backslash U$. They have to reconstruct a polynomial function $f_k(x)$ and compute $\sigma = f_k(0)$. But the secret polynomial function $f_k(x)$ can only be reconstructed by at least t_k secret shadows Y_{ki}, and they cannot obtain Y_{kl} from $C_{kl} = g^{Y_{kl}} \mathrm{mod} p$ based on the DLP assumption. Thus, they are unable to compute secret polynomial function $f_k(x)$ and σ. Also they can't obtain Y_{kl} and x_l from $C_{kl} = g^{Y_{kl}} \mathrm{mod} p$ and $y_l = g^{x_l} \mathrm{mod} p$ for $P_l \in F_k$ because of the difficult problem of solving discrete logarithm. Therefore, they are unable to derive each P_l's partial signature

$$s_{kl} = a_{kl} R_k + (Y_{kl} W_l + x_l) h(R_k, m, IDF_k, IDO, m_w). \ (25)$$

   Then, our new scheme can resist the collusion attack made by any unauthorized set of proxy signers.

5. In new scheme, (m, IDF_j, IDO, m_w) is a part of hash function $h(\cdot)$ in individual signature

$$s_{ji} = a_{ji}R_j + (Y_{ji}W_i + x_i)h(R_j, m, IDF_j, IDO, m_w). \quad (26)$$

Thus, after intercepting a valid proxy signature (R_j, S, A, m_w, IDF_j), it is impossible for anyone to replace $(m, IDF_j, IDO, m_w, \sigma)$ by another $(m', IDF_{j'}, IDO', m_{w'}, \sigma')$, and in the same time the following equality holds:

$$h(R, m, IDF_j, IDO, m_w) = h(R', m', IDF_{j'}, IDO', m_{w'}). \quad (27)$$

This is because $h(\cdot)$ is a collision resistant hash function.

6. At last, in our new scheme, there are the designated warrant m_w, the identity of the actual original signer's IDO, and the identities of the actual proxy signer's IDF_j. Furthermore, all the verifiable equalities consist of m_w, IDF_j, IDO, As discussed in (5), we know that our scheme can resist adaptively chosen warrant attack. Thus, the verifier can be convinced of the warrant m_w to be published by the original signer, and m_w records the stipulated period of this proxy, which provides the time constraint.

From what have been analyzed above, we are fully certain that the requirements of a secure non-repudiable general proxy signature scheme are fulfilled in our scheme.

As discussed in analysis of Attack 1, the adversary cannot compute the original signer's private key from public information. Therefore, in our scheme, the original signer's private key can always be kept secret and used repeatedly.

From analysis of Attacks 3, 4 and 5, we know that only the owner of x_i can generate the partial signature s_{ji} and as discussed in analysis of Attack 2, the adversary cannot compute x_i from the public information. Therefore, the property of proxy protection is fulfilled in our scheme.

From analysis of Attacks 3, 4 and 5, we know that the general proxy signature can be only generated by any authorized subset of the proxy group. Furthermore, from analysis of Attack 6, we know that our scheme can resist warrant attack. Therefore, the adversary cannot change Γ and the basis Γ_0. Hence, our scheme satisfies the property of unforgeability.

As discussed above, the partial proxy signature s_{ji} can be only generated by the proxy signer P_i, and the proxy signature can be only generated by any authorized subset of the proxy group. On the other hand, the verifier can be convinced of the original signer's agreement on the signed message, since only the owner of x_0 can generate σ from m_w. Therefore, our scheme follows the property of nonrepudiation. The proxy group cannot repudiate the proxy signatures they created, and the original signer cannot deny having delegated the power of signing to the proxy group.

As discussed in analysis of Attack 6, the verifier can be convinced of the warrant m_w is published by the original signer. Furthermore, m_w records the stipulated period of this proxy, which provides the property of time constraint.

From IDF_j and IDO, the verifier can notice who the actual signers are. Furthermore, as discussed in analysis of Attack 6, it is impossible for anyone to replace $(m, IDF_j, IDO, m_w, \sigma)$ by another $(m, IDF_{j'}, IDO', m_{w'}, \sigma')$. Hence, our scheme satisfies the property of known signers.

As it is mentioned before, HZ scheme [10] is an improvement of Yang scheme [20]. We have compared security of our scheme with these two previous schemes and summarized the result in Table 1.

Table 1. security comparison of threshold schemes with proposed scheme

Security features	Yang [20]	HZ [10]	Our scheme
Scheme can resist frame attacks.	No	Yes	Yes
Scheme can resist public-key substitute attacks.	No	Yes	Yes
Scheme can resist collusion attacks.	Yes	Yes	Yes
Scheme satisfies the property of proxy protection.	No	Yes	Yes
Scheme satisfies the property of unforgeability.	No	Yes	Yes

Scheme satisfies the property of non-repudiation.	No	Yes	Yes
At least t proxy signers can generate valid proxy signature.	Yes	Yes	No
Any minimal qualified subsets can generate valid proxy signature.	Yes	Yes	Yes
Each user determines his private and public keys.	Yes	Yes	Yes
CA can derive the original signer's private key.	No	No	No
CA can derive any proxy signer's private key.	No	No	No

6 PERFORMANCE

In this section, in terms of computational complexity, we compare the new general proxy signature scheme with (t,n) threshold proxy signature scheme proposed in [10] and summarize the result in Table 2. For convenience, the following notations are used to analyze the computational complexity.
T_e the time for one exponentiation computation.
T_m the time for one modular multiplication computation.
T_H the time for hash function computation.
T_i the time for one inverse computation.
Also, we consider $\Gamma_0=\{F_1,F_2,\dots,F_m\}$, and $|F_j|=t_j$.
From Table 2, we can see that proxy signature generation and verification of general scheme are more efficient than threshold scheme if $t_j<t$, i.e., the cardinal of minimal qualified subset F_j, that wants to sign the message m, is less than the threshold t. Also, proxy share generation in new scheme is more efficient than that in HZ scheme if $4n+t<\sum_{j=1}^{m}5t_j$, $n(t+1)<\sum_{j=1}^{m}t_j(t_j+1)$. Therefore, general protocol not only have the merits of threshold one, but also can reduce computation costs.

Table 2. comparison of computational complexity HZ scheme with proposed scheme

Schemes	Share generation	Signature generation	Signature verification
HZ scheme [10]	$(4n+t+1)T_e+(nt+n+2)T_m+T_H$	$4tT_e+(3t+3)T_m+2T_H$	$5T_e+(t+2)T_m+2T_H$
Proposed scheme	$\left(\sum_{j=1}^{m}5t_j\right)T_e+\left(\sum_{j=1}^{m}t_j^2\right)T_m+T_H$	$4t_jT_e+(3t_j+3)T_m+2T_H$	$5T_e+(t_j+2)T_m+2T_H$

7 CONCLUSIONS

The previous proxy signatures have been based on threshold secret sharing schemes [9-16,19-21], in the sense that any t or more proxy signers can sign a message on behalf of the original signer. However, in the real world, all of the proxy signers in a proxy group do not have necessarily the same power or influence. For this reason, we design a novel proxy signature scheme based on general secret sharing scheme. This fact that we consider a scenario which is more general than the threshold one with fewer computational complexity, makes it more complete than the previous proposals of (t,n) threshold proxy signature schemes.

8 REFERENCES

1. Yassin, A. A., Neima, H. Z., Hashim, H. Sh.: Security and Integrity of Data in Cloud Computing Based on Feature Extraction of Handwriting Signature, International Journal of Cyber-Security and Digital Forensics 3, pp. 93--105. (2014).
2. Blakley, G. R.: Safeguarding cryptographic keys. The National Computer Conference 1979, AFIPS 48, pp. 313--317. (1979).
3. Bernadette, C. F., Ocampo, D., Mari, L. T., Castillo, D., Alberto, M., Gomez, N.: Automated signature creator for a signature based intrusion detection system with network attack detection capabilities (pancakes), International Journal of Cyber-Security and Digital Forensics 2, pp. 25--35. (2014).
4. Dehkordi, M. H., Mashhadi, S.: New efficient and practical verifiable multi-secret sharing schemes Information Sciences 178, pp. 2262--2274. (2008).
5. Dehkordi, M. H., Mashhadi, S.: Verifiable secret sharing schemes based on non-homogeneous linear recursions and elliptic curves, Computer Communications 31, pp. 1777--1784. (2008).
6. Hadian, M., Mashhadi, S.: A new threshold multi-secret sharing scheme, Amirkabir Journal of Science and Technology 68, pp. 45--50. (2008).
7. Hadian, M., Mashhadi, S.: Two verifiable multi secret sharing schemes based on nonhomogeneous linear

recursion and LFSR public-key cryptosystem, Information Sciences 294, pp. 31--40. (2015).
8. Hadian, M., Mashhadi, S.: An efficient threshold verifiable multi-secret sharing. Computer Standards & Interfaces 30, pp. 187--190. (2008).
9. Hsu C., Wu T., Wu T.: New nonrepudiable threshold proxy scheme with known signers. The Journal of Systems and Software 58, pp. 119--124. (2001).
10. Hu, J., Zhang, J.: Cryptanalysis and improvement of a threshold proxy signature scheme. Computer Standards & Interfaces 31, pp. 169--173. (2009).
11. Huang, H. F., Chang, C. C.: A novel efficient (t, n) threshold proxy signature scheme. Information Sciences 176 (10), pp. 1338--1349. (2006).
12. Huang, S., Shi, C.: A simple multi-proxy signature scheme. Proceedings of the 10th National Conference on Information Security, Hualien, Taiwan, ROC, pp. 134–138. (2000).
13. Mambo, M., Usuda, K., Okamoto, E.: Proxy signature: delegation of the power to sign messages. IEICE Trans Fund Electron, Commun Comput Sci E79-A,(9), pp. 1338--1353. (1996).
14. Mashhadi, S.: A novel non-repudiable threshold proxy signature scheme with known signers'. International Journal of Network Security 15, pp. 231-236. (2013).
15. Mashhadi, S.: A novel secure self-proxy signature scheme, International Journal of Network Security 14, pp. 83-87. (2012).
16. Mashhadi, S.: Analysis of frame attack on Hsu et al.'s non-repudiable threshold multi-proxy multi-signature scheme with shared verification, Scientia Iranica 19, pp. 674–679. (2012).
17. Santis, A. D., Masucci, B.: New results on non-perfect sharing of multiple secrets. The Journal of Systems and Software 80, pp. 216--223. (2007).
18. Shamir, A.: How to share a secret. Communications of the ACM 22, pp. 612--613. (1979).
19. Shao, J., Cao, Z., Lu, R.: Improvement of Yang et al.'s threshold proxy signature scheme. The Journal of Systems and Software 80, pp. 17--177. (2007).
20. Yang, C. Y., Tzeng, S. F., Hwang, M.S.: On the efficiency of nonrepudiable threshold proxy signature scheme with known signers. The Journal of Systems and Software 73, pp. 507--514. (2004)
21. Zhang, K.: Threshold proxy signature schemes. in: 1997 Information Security Workshop, pp. 191—197. (1997).

Anti-Forensics: A Practitioner Perspective

Richard de Beer, Adrie Stander and Jean-Paul Van Belle
Department of Information Systems, University of Cape Town
Private Bag, Rondebosch, 7701, South Africa
Adrie.Stander@uct.ac.za
Jean-Paul.VanBelle@uct.ac.za

ABSTRACT

With the increase in cybercrime, digital evidence is becoming an integral part of the judicial system. Digital evidence is to be found everywhere from computers, to mobile phones, ATMs and surveillance cameras, and it is hard to imagine a crime that does not contain any element of digital evidence. It is however not simple to extract such evidence and present it to court in such a way that there is no uncertainty that it was not changed in any way. Thus the responsibility placed on a Digital Forensics (DF) practitioner to present usable evidence to a court is increasing fast. In some respects, however, it is relatively easy to get rid of digital evidence or to hide it. Many tools exist for cybercrime criminals to prevent DF practitioners from getting their hands on information of probative value. Such tools and methods known as Anti-Forensics (AF).

The purpose of this study is to identify the abilities of DF practitioners to identify the use of AF in their active investigations. The research model used, attempts to identify all the factors and constructs of AF that impacts on investigations. This model was then used to develop a survey instrument to gather empirical data from South African DFs.

The research has shown that whilst South African DF practitioners perceive DF as having an impact on their investigations, they also perceive electronic evidence as forming only part of the evidence presented to court, and that even if most of the usable evidence of lost, some will generally remain.

It was also found that while most DF practitioners in South Africa are well versed only in the more commonly known AF techniques. They do not rate their abilities on more complex techniques well. Finally, most DF practitioners appear not to actively attempt to identify AF techniques as part of their investigations. This combined with a lack of understanding of more complex AF techniques could leave South African DF practitioners exposed by missing important evidence due to lack of technical proficiency.

KEYWORDS

Digital forensics; anti-forensic tools; anti-forensic methods; digital forensic evidence; South Africa.

1 INTRODUCTION

Forensics as a scientific discipline is the process whereby science is used to investigate artefacts or transfer of evidence and interpret its relevance to an investigation [1]. The goal of the DF practitioner is the collection and analysis of digital evidence with a view towards presenting such evidence in a court of law or other legal proceeding. Key to the success of this process is the probative value of the collected evidence [2].

Anti-forensics (AF) involves the use of methods specifically designed prevent the use of scientific methods and tools to collect and analyse forensic artefacts for use in court proceedings. This is mainly aimed at the destruction or hiding of evidence. (Harris, 2006).

Due to the ever-increasing frequency of AF tool use, greater vigilance by DF investigators will be required to ensure the integrity of investigative results [3]. The use of anti-forensics is a difficult issue to overcome and digital forensics investigators can expect these techniques and tools to become much more sophisticated and also more widespread as suspects become more aware of the techniques and tools used by digital forensic investigators. It appears that this is a growing problem and that investigators will have to ascertain that they stay informed about the latest anti-forensic techniques and countermeasures.

All research into this phenomenon up to this point has come out of the developed world, with the USA, Europe and Australia leading the way. The 2012 Verizon data breach report states that one third of all DF investigations undertaken by Verizon are affected by AF [4]. No similar research has been conducted in South Africa, and as such the current scope of the AF problem is South Africa is not known. Due to the risks inherently posed by AF and the fact the electronic evidence in South African law is still in its infancy, a real risk exists that DF practitioners in South Africa are either not aware of the AF practices being used or are incorrectly identifying information that indicates the use of AF practices or applications, and are negatively impacting court rulings based on the acceptance of digital evidence

This research aims to establish to what extent the use of AF by the subjects of DF investigations has affected the ability of DF practitioners to complete such investigations successfully in the South African context. In this instance success is defined as the existence of digital evidence of sufficient probative value to ensure the presentation of admissible evidence in a court or other legal proceeding.

The motivation for this research is the furtherance of the knowledge of AF practices, techniques and applications for the benefit of practicing DF investigators and aims to provide a basis for further research into this phenomenon with a view to expanding the academic knowledge in this area.

2 LITERATURE REVIEW

2.1 Digital Forensics and Digital Evidence

Digital forensics as a discipline is aimed at identification, collection and analysis of digital evidence following an attack [5]. The purpose is to determine the identity of the attacker or suspect (who), their actions (what), when their actions were taken, how they perpetrated the attack or crime and their possible motivations (why).

Courts of law base the adjudication of all cases on evidence presented. In examining AF and its effect on the punctiliousness of digital evidence presented, it is first necessary to examine digital evidence closely. Evidence can be defined as anything presented to logically prove or disprove an issue at hand in a judicial case [6]. Digital evidence is information of legal probative value that is stored or transmitted in electronic form [3]. Digital evidence is similar to traditional evidence in that it contains information that is used to confirm or refute a hypothesis placed before the court or legal proceeding [7].The only difference is that such information is stored digitally. As such the quality of such evidence remains a critical factor as with any other case.

The proliferation of electronic devices has meant that digital evidence can be relevant to any case and not just for computer crimes. By its very nature digital evidence is fragile. Incorrect handling, examination or intentional destruction or modification can alter digital evidence to a point where it is no longer usable. The greatest challenge to using digital evidence in court is the fact that manipulation or alteration of the evidence can be achieved very easily without leaving any indication of such actions [5].

Whilst digital evidence is valuable as a source of evidence in any variety of investigations, it also introduces a new level of complexity that could potentially confound digital forensic investigators [3].

2.2 Anti-forensics

Anti-forensics (AF) in the digital realm is the process of removal or obfuscation of digital forensic artefacts with the aim of invalidating digital forensic investigations [8]. Typically, one or more of the following strategies are used: data hiding, data destruction, trail obfuscation, data contraception, data fabrication, file system attacks [9]. AF aim to remove all traces of a digital event, invalidate the data or increase the complexity of the investigation, remove evidence of its own use, or generally cast doubt on the investigation

[10]. The various AF methods are discussed below in more detail.

Data hiding

Data hiding refers to the practice of storing data where it is unlikely to be found, or employing the method of security through obscurity [9]. Simple methods such as extension renaming or signature editing exist, but these are generally easily identified by most current forensic software. In data communications, data hiding refers to the art of adding an obscure message signal in a host signal without any perceived distortion of the host signal. This composite signal is typically referred to as the 'stego' signal, and employs a different communication scheme than normal data communications [11].

One of the simplest and most effective methods of data hiding is Steganography. Whilst the practice of hidden writing has been around for millennia, the ability to hide any form of digital data within another carrier file poses a difficult challenge for digital forensic investigators [8]. In addition to its versatility to hide any data, Steganography is also very hard to detect. At the moment the only Open Source tool that effectively detects data hidden by modern steganography tools is StegDetect by www.outguess.org [12].

Some steganography algorithms hide the information in such a way that it is impossible to recover such information without knowing the key to the algorithm. Whilst that may sound like cryptography, it is accomplished simultaneously with the cloaking of the information in a masquerade file, and as such, is still steganography [13]. The most obvious difference between cryptography and steganography is that cryptography essentially hides data by disguising it as completely random data which is sometimes referred to as random noise. Stenographic algorithms are generally not trivial to break, even if the examiner has learned that there is hidden data to be discovered, which is often not simple to achieve in the first place.

Encryption

Encryption is simply a process of protecting data by using an algorithm to scramble the data and make it either intelligible or undetectable unless a key is used to decrypt the data [14]. Encryption has been used since ancient times in one way or another to protect against the interception of messages [15]. Encryption is used in many facets of digital data storage and transmission. When seen in the context of AF data-hiding, encryption tools provides the user, who are attempting to thwart the efforts of the DF investigator, with an extremely powerful tool.

Open-source encryption software is becoming more mainstream. Software such as TrueCrypt even offers the ability to hide one encrypted volume within another. TrueCrypt is a cross-platform encryption tool that uses 'On the fly encryption (OTFE)' to encrypt and decrypt files as they are accessed, and makes all data within the encrypted area available as soon as the decryption key is entered [16].

The most popular forms of data storage encryption include the encryption of a virtual or physical disk or partition and system encryption whereby the system (boot) files are encrypted.

Network traffic can also be quite easily encrypted using standard protocols such as SSL (secure sockets layer), SSH (Secure shell) or TLS (transport layer security). Whilst these protocols were developed as security protocols for the legitimate protection of information transmitted over either a public or private network, they can be used by criminals to transmit data securely.

Program packers

Program packers such as Armadillo and UPX are used to encrypt and/or compress an attack program and then incorporate the file in a new 'packed' file that is wrapped with a suitable extractor. When the seemingly innocuous process is run the packed attack application is then run simultaneously.

Hiding data in system areas

There are methods available to hide information in areas reserved as system space or file slack (area between the end of the logical file data and the end of the cluster). One such tool is 'Slacker', by the Metasploit Project [17]. Creators of this project claim that Slacker is the first ever tool that allows you to hide files within the slack space of the NTFS file system.

In addition, custom attacks created using software exploit frameworks such as the Metasploit framework are generally delivered using payloads created using tools such as ‘Msfpayload”, which allow the creator to create custom file signatures that will not be detected by forensic signature analysis.

More advanced tools such as ‘FragFS’ exist that have the ability to store information in the NTFS file system’s Master file table ($MFT file) [9].

Rootkits represent a malicious method of data hiding. Such programs allow attackers undetected administrative access to a computer [18]. In this instance the attacker can affect multiple states on the infected computer, such as executing programs, logging keystrokes or even storing data.

Rootkits are mostly installed on computers through the binding of a malicious program to a seemingly harmless one. An example of this is where a user downloads an MP3 or e-book from a file sharing site, and once the user runs the file they inadvertently install the rootkit on their computer. To further confound the issue, many rootkits are self-healing, and will automatically reinstate themselves if deleted or uninstalled. An example of this is the Computrace client that consists of both an application agent and a persistence module.

Data destruction

The destruction of data by wiping of files is a commonly used AF method which has been used for a long time. For the cybercriminal the wisest course of action is to simply remove all traces that anything untoward took place [19]. A simple delete essentially leaves the data intact. Though not visible to the general computer user, such data is easily recoverable using data recovery or forensic tools.

By using any one of a number of freely available data wiping tools (Including, Eraser, PGP etc.) the user is able to securely delete files by overwriting the clusters occupied by those files with random data, any number of times, according to existing standards such as Guttmann (35 times) and DoD standard 5220.22-M (US DoD, 1995) (7 times) [10]. Recovery of such securely deleted data is normally not possible [8].

Other tools also exist that focus on securely removing artefacts that pertain to activities of the user, such as internet history, file access, file downloads, peer to peer networking and Internet Messaging. A good example of a tool such is this is CCleaner by Piriform which is available as a free download.

Tools such as these perform a secure delete of the artefacts mentioned above to ensure that such remnants are not recoverable, after deletion. Such tools can also quite easily be configured to securely wipe all hard drive free space, including slack space, either manually or automatically at scheduled intervals.

In addition to data destruction by wiping or overwriting, there are also more drastic measures that cybercriminals sometimes revert to. These are degaussing the drive – sweeping the drive with a powerful magnet, thereby rendering the data unstable – or the physical destruction of the storage media.

Trail obfuscation

Trail obfuscation follows three basic methods. The first has the aim of obscuring required information from the would-be investigator. This is achieved by either replacing relevant information with false information (such as IP address spoofing) or using third parties to act as proxies of the source data in order to remove all traces of the origin of the data from the transmission at the destination (such as mail anonymizing services).

The second form of obfuscation involves altering the data associated with forensic artefacts by altering metadata such as date and time stamps.

Finally, trail obfuscation can also take the form of log deletion or modification in order to hide log entries that would identify the identity or action of the perpetrator. Securely wiping or modifying log files can be achieved by using freely available tools such as Touch [8].

Data contraception

Data contraception, also referred to as evidence source avoidance, is the process whereby the perpetrator uses software and methods that have been designed to leave no traces on the host operating system. A number of different methods can be used [20]:

- Portable applications – these applications do not install any files on the host computer (e.g. TrueCrypt and FTK imager lite).
- Live distros – these are fully functional operating systems from bootable devices such as CD’s or Flash drives. As all functions run in memory, no traces are left on the local hard drive, as the local hard drive is in fact not even required (e.g. Windows CE or BartPE) [18].
- Syscall proxying – A local system call or function is proxied to another system to complete.
- Remote library injection – Information (typically a Dynamic Link Library) is inserted directly in RAM of the host leaving no traces on the hard drive.
- DKOM (Direct Kernel Manipulation) – The process whereby the memory space utilized by kernel objects are penetrated and used by other inserted processes.
- Utilizing ‘in-private’ browsing on web browsers such as Mozilla Firefox will keep all cache and history in memory and will not write any information to disk for later analysis.

Data fabrication

This practice, also referred to as evidence counterfeiting, is very similar to some of the practices followed for trail obfuscation as discussed elsewhere in this document. In Windows, the Modified, Accessed and Created dates are referred to as the MAC information.

Modifying the MAC information on the computer serves both the purpose of obfuscating the original data and also can be seen as fabricating data [18].

Another data fabrication practice that is employed, is the creation of excessive amounts of data of a certain type in order to side-track and investigation to the point where cost-effectiveness becomes the deciding factor in the continuance of the investigation.

File system attacks

When an attack of sufficient severity is launched on a file system, it might inhibit the ability of a forensic application to make sense of the data contained therein.

An example of such sabotage would be to damage the master file table of an NTFS file system to such an extent that a forensic analysis of the logical drive is unable to extrude any meaningful data.

2.3 Trace Evidence of Anti-forensic Tool Use

As with many other computer applications, the actual use of an AF tool to remove forensic artefacts might in itself leave trace evidence.

Steganography is probably one of the most difficult methods of data hiding to detect and decipher and combines the art of hiding data from human perception and cryptography [9].

The DF practitioner must therefore first develop the knowledge or suspicion that steganographic data hiding is present in any given case, and then the practitioner has to establish which files out of all data that falls within the scope of the investigation could be affected. As there could potentially be hundreds of thousands of files on a suspect computer this is often difficult.

In this instance the detection of files affected by Steganography will often depend on the intuition and experience of the DF examiner. In the event that Steganography is suspected and the DF practitioner has identified files to target, the practitioner may use methods similar to the one below to confirm the presence of steganography and to attempt to decipher the contents.

Encryption is often quite obvious to detect. An example of this may be where a DF practitioner has attempted to image a suspect computer and is not able to access any of the data on the drive due to full disk encryption. In such examples the DF practitioner will not be able to access any of the data on the suspect drive unless such a password is provided, and it is often more practical to attempt to extract the password from the suspect using legal means than it would be to use technology.

Some forms of encryption may not be as easy to detect, such as a virtual encrypted disk. In this instance the DF practitioner will have to rely on other methods of detection such as identifying suspiciously large files and files with an unknown or no file signature. However, even in the event of detecting an encrypted volume, the challenge of decrypting such a volume still remains. Without the passkey required to decrypt and mount such a volume it is near on impossible to decrypt the information contained therein [15].

Attempting to break the password of the volume using any number of traditional methods such as brute force attacks, dictionary attacks or rainbow tables may take years to achieve [18]. In this instance the most viable option is often to attack the human factor. Often it is simply not possible to decrypt an encrypted volume using current technology and the only means available in such a case is the use of a password.

Most forensic software will detect signatures of commercial *file packers*. The forensic investigator should also be aware that such file packers typically have file signatures that can be searched for manually. An example of the UPX file packer a manual keyword search can be completed for the search term ‘UPX’, which will identify executable files that have been packed using this packer.

When using a tool such as slacker to *hide information in system areas*, the forensic investigator will likely have no specific indication that such an indication has occurred as the only change to the file assigned to legitimately occupy that sector, will be a change in the ‘date modified’ field. AF users who is thorough in their attempt at data hiding will then quite simply, use another tool such as ‘Timestomp’ to change that metadata attribute back to a value that would not arouse suspicion [21].

The data written to the slack area is not encrypted, but the metadata of the files written in slack space is encrypted. This is intended to complicate efforts to locate a list of files created using Slacker by doing metadata analysis. The most likely indication of the use of slacker will be the discovery of the Slacker executable on the suspect drive or in the event of volatile memory analysis or virtual memory analysis, a keyword search may indicate that the program had been loaded into memory. In the absence of such trace evidence the DF practitioner may still find information relating to the case in slack space by making use of a simple keyword search. Whilst this information may relate to the case, it will not provide any proof that a product such as Slacker was used to place it there.

Most anti-virus software claims to be able to detect *rootkits*. The reality is unfortunate that most Anti-virus software may stop a computer from being infected with new rootkits, but established rootkits will often remain undetected [9].

This does not mean that the DF practitioner need not run an antivirus scan on the acquired forensic image, as this remains a possible tool for detecting rootkits and viruses and can be useful in the event that the suspect decides to pursue a ‘Trojan- defence’.

When assessing the ability to detect rootkits from a forensic analysis perspective, it is important to look at two basic modes of forensic operation. These are live incident response i.e. volatile and non-volatile data analysis on a live computer, and dead' forensics, the analysis of static system data

Data destruction is often time consuming and very often the applications used do not destroy all data, as advertised. In addition to these factors, there will very often be indications that data destruction applications have been used on a computer [22].

Finally, commonly used *trail obfuscation tools* often leave certain artefacts behind e.g. by changing time/date stamps in only one system area but not others.

3 RESEARCH METHODOLOGY

3.1 Research Question and Hypotheses

The core research question is whether *the use of AF is affecting the ability of South African DF practitioners to complete DF investigations*?

To answer the main research question the research aimed to find answers to the sub-questions listed below:

- What AF tools are being used, and how they are being used? Not only did the research aim to determine which applications are used for Anti forensic purposes, but also how suspects use such software or tools. In addition, the research looked at the success rate of such tools in achieving the intended purpose and if any artefacts remained after the use of a particular tool. This research will identify which of the available AF tools are being used and for what purpose.
- Are South African DF practitioners able to identify AF tool use by the artefacts that such tools leave behind? As with many other computer applications, the actual use of an AF tool to remove forensic artefacts, will in itself leave trace evidence.
- How prevalent is AF tool use in the South African environment? This research will attempt to establish whether digital forensic practitioners experience anti forensic tool use on a regular basis.

3.2 Research Methodology and Proposed Model

A positivistic research philosophy was adopted for this research. A cross-sectional time-horizon is used, as the study will aim to understand the status of the situation at a particular point in time. The research used a deductive approach. To that end, a research model was developed as is shown in figure 1.

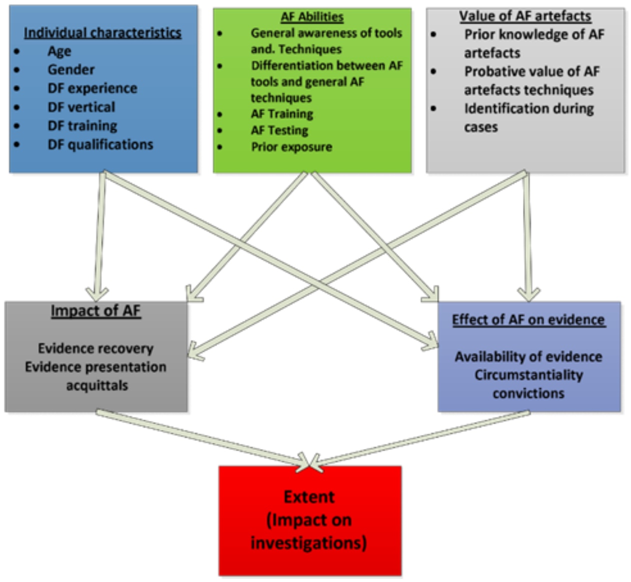


Figure 1. Proposed research model

Some of the constructs warrant further explanation. Under the Individual Characteristics, "DF" refers to the Digital Forensics experience and skills, including DF experience, formal DF training, any qualifications obtained and the industry environment in which they are deployed (DF *vertical*: civil, criminal or corporate environment).

The AF abilities refer to specific Anti-Forensic tool and method abilities. The value of AF artefacts refers to the knowledge of and ability to identify AF artefacts (i.e. evidence left behind by AF tools) including the knowledge of artefacts left behind by AF tools and techniques, the ability to identify AF tools by their artefacts and knowledge of the evidentiary value of the artefacts left by AF use.

The impact of AF is measured by evidence recovery (the DF practitioner's ability to recover useable evidence), evidence presentation (the DF practitioner's ability to present evidence of probative value) and acquittals (where AF leads to acquittals). The Impact of AF as part of all evidence refers to the impact AF has on electronic evidence when seen as part of the entire case and all other types of evidence presented and convictions refers to the impact of AF on convictions when seen as part of the entire case and all other types of evidence presented.

The sample size needed to test the model statistically, far exceeded the number of digital forensics practitioners in South Africa, which made testing the multiple relationships simultaneously impossible.

Instead, the model served as a guide for the development of constructs to include in the questionnaire, generating descriptive statistics and doing limited inferential statistical tests such as construct correlations and ANOVA.

3.3 Research Instrument and Sampling Approach

Collection of data was by means of a survey to assess the state of events as experienced by DF practitioners. A mostly original survey instrument was created. Due to the large number of constructs, usually only one question (test item) was formulated for each of the constructs.

South African DF practitioners were targeted to include practitioners that operate in criminal, civil, and corporate environments. The aim was to include practitioners that deal with evidence in traditional, mobile and internet / e-commerce forensics. Thus a probability-sample using the stratified random-sampling technique was used in order to identify forensic practitioners functioning across the strata. Figure 2 illustrates the employment sectors of the respondents.

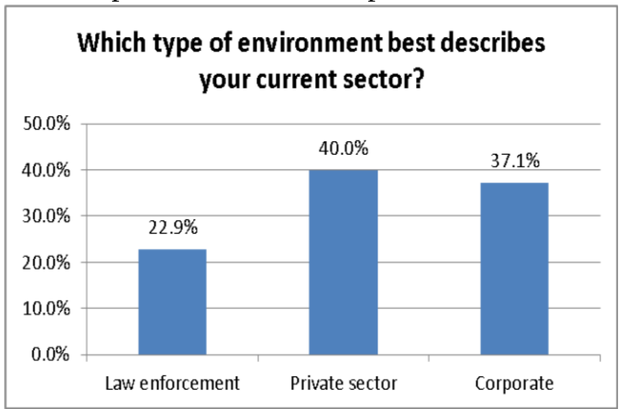


Figure 2. Employments sectors

A limitation is the relatively small size of the South African DF fraternity. The researchers have access to a large proportion of the South African digital forensics practitioners, since due to the small size of the group, practitioners tend to meet on a regular basis and it was felt that a representative sample was obtained.

No personal interviews were conducted and no personal information was collected about any of the practitioners or their places of employment.

4 DATA ANALYSIS

4.1 Demographic Profile of Respondents:

The majority (85.7%) of respondents were male and most fall in the 31-40 year age range. These sample characteristics are fairly representative of the DF practitioner community in South Africa.

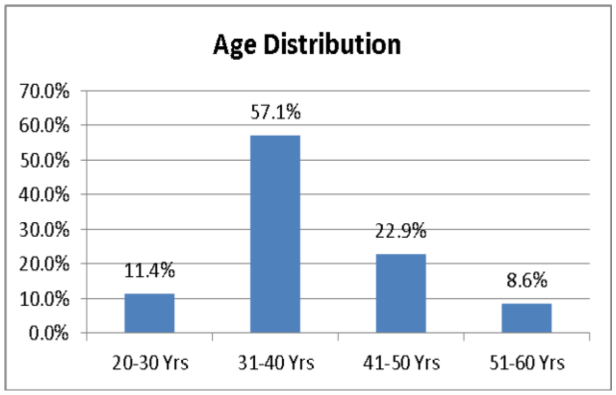


Figure 3. Age distribution of respondents

Most DF practitioners were practicing although nine respondents (25.7%) indicated that they were not practicing DF investigators. Six of these non-practicing AF investigators (67%) are in the corporate environment. It is possible that the 9 (25.7%) survey participants have branched into other areas of their organization, some even into managerial positions.

A wide spread of experience in DF was borne out by the respondents. In line with the stratified random-sampling method a satisfactory spread of employment sectors was achieved.

Table 1. DF qualifications

Education and training level	Yes	%(Yes)	No	%(No)
Do you have a Digital forensics qualification? (EnCe, ACE etc.)	3	33.33%	6	66.67%
Do you have a tertiary qualification?	8	88.89%	1	11.11%
Have you completed any Anti-forensic tool testing?	2	22.22%	7	77.78%
Have you had any Anti-forensics training?	5	55.56%	4	44.44%
Have you had any formal digital forensics training?	8	88.89%	1	11.11%

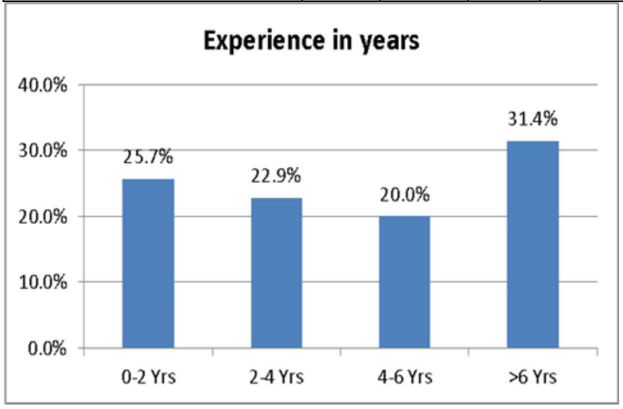


Figure 4. Experience.

Whilst most respondents (89%) indicated that they had received formal DF training, only a few respondents followed such training up with a qualification. However, on the other hand, 89% of them had a tertiary qualification.

More than half (56%) of respondents claim to have had training in AF techniques and tools. The same respondents who have completed AF training have also tested AF tools and techniques.

Most respondents rate their knowledge of AF tools and techniques as average to good but their prior exposure to AF was lower as can be seen in tables 2a &b. Very few respondents rate their prior exposure to AF as excellent.

Tables 2a & 2b. Knowledge of AF tools and techniques and prior exposure to AF

How would you rate your knowledge of anti-forensics tools and techniques?	%
Excellent	5.7%
Good	42.9%
Average	37.1%
Poor	11.4%
None	2.9%

How would you rate your prior exposure to anti-forensics?	%
Poor	34.3%
Average	28.6%
Good	28.6%
Excellent	8.6%

4.2 Interesting Correlations between Independent Variables

Possible correlations between demographic variables and other independent variables were investigated. For instance, there was a significant difference between the sectors in which an FP was employed and their investigation environments. FPs employed in the law-enforcement and private sectors tend to investigate computer/networks, internet/e-commerce whereas those in corporate environments tend to limit their investigations mainly to computer/network forensics and, to a lesser extent, internet/e-commerce (Chi-square χ^2 value=7.79, DF=2, p-value = 0.0203).

However, there was no statistical evidence to suggest that a difference exists between the knowledge, qualifications and training in Anti-Forensic investigators between the 3 employment sectors, or that they were exposed to different types of AF threats. Not surprisingly, there is a statistically significant correlation between completed AF tools training and a respondent's own rating of knowledge of anti-forensics tools and techniques (χ^2 value=8.241, DF=1, p-value=0.0041). In fact, respondents with AF tools Training completed are 8.4 times more likely to rate their knowledge of AF tools and techniques as "Good/Excellent".

Formal Forensics training, Digital Forensics Qualification and Tertiary Qualification show no association (or relationship) with respondent's prior exposure to AF. However, specific AF training" has an association (or relationship) with prior exposure to AF (χ^2 value=4.61, DF=1, p-value=0.0318) and persons with AF training are 4 times more likely to rate their prior exposure to AF as GOOD. Even more so, "specific AF tools testing" has an association with prior exposure to AF (χ^2 value=12.61, DF=1, p-value=0.0004) with those who have completed AF tools testing are 18.7 times more likely to have GOOD prior AF exposure.

This shows that *generic* formal digital forensics training is deficient in imparting AF-specific skills or knowledge. This should be remedied by addressing the curricula of (generic) forensics training courses.

On the whole, respondents seem to be familiar with the more common AF techniques i.e. data hiding and data destruction. The twelve (34%) respondents that rate their prior AF exposure as "poor" have hardly any familiarity with the AF techniques of data contraception, trail obfuscation, data fabrication and file system attacks, although half of them were exposed to data hiding and data destruction.

These respondents are familiar with an average of only 1.2 AF techniques whereas respondents who rate their prior exposure as average to excellent have an exposure to an average of 2.9 techniques.

Table 3. Familiarity with AF techniques.

Which Anti forensic techniques are you most familiar with?	%No	%Yes
Data Hiding	28.6%	71.4%
Data destruction	31.4%	68.6%
Trail obfuscation	68.6%	31.4%
Data fabrication	77.1%	22.9%
File System attacks	77.1%	22.9%
Data contraception	85.7%	14.3%
None of the above	88.6%	11.4%

As with AF techniques, most respondents appear to be familiar with the more commonly known AF tools such as data wiping and encryption.

Table 4. Familiarity with AF tools (all respondents)

Familiarity with AF Tools	No	%No	Yes	%Yes
Data wiping and history removal (CCleaner, Eraser etc.)	7	20.0%	28	80.0%
Encryption tools (Truecrypt etc.)	8	22.9%	27	77.1%
Steganography tools (Quickstego etc.)	21	60.0%	14	40.0%
Timestomp (by Metasploit)	25	71.4%	10	28.6%
Rootkits	31	88.6%	4	11.4%
Transmogrify (by Metasploit)	32	91.4%	3	8.6%
The complete A-Z of open source tools out there.	34	97.1%	1	2.9%

No relationship (association) were found between respondent's rating of the probative value of artefacts that AF tools leave behind, and their familiarity with AF tools. (Chi-Square tests performed). Neither is there a relationship (association) between respondent's rating of their efforts to actively identify anti-forensics as part of investigations, and their familiarity with AF tools. (Chi-Square tests performed)

The following graph summarizes respondent's view of AF techniques that most affected their cases. The more common techniques appear to be more prevalent, however respondents previously indicated that they are less familiar with the more complex techniques such as data contraception, which may indicate that such techniques are not identified.

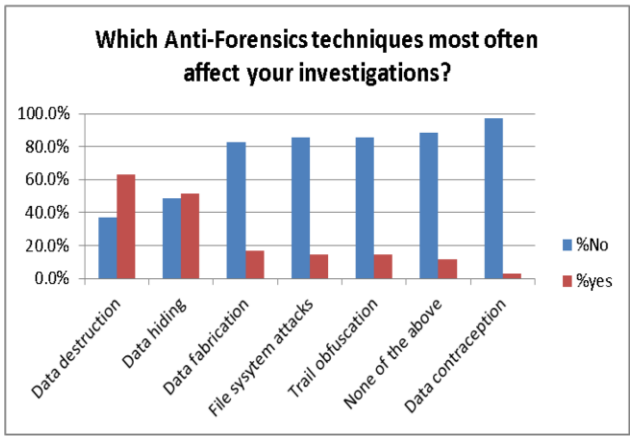


Figure 5. AF Techniques that most affect investigations

As with the AF techniques, respondents feel that the commonest AF tools are most prevalent during their investigations. Data destruction and Data-hiding are the most common AF techniques that respondents experience. Data contraception, Trail obfuscation, System file attacks and data fabrication are the least seen AF techniques.

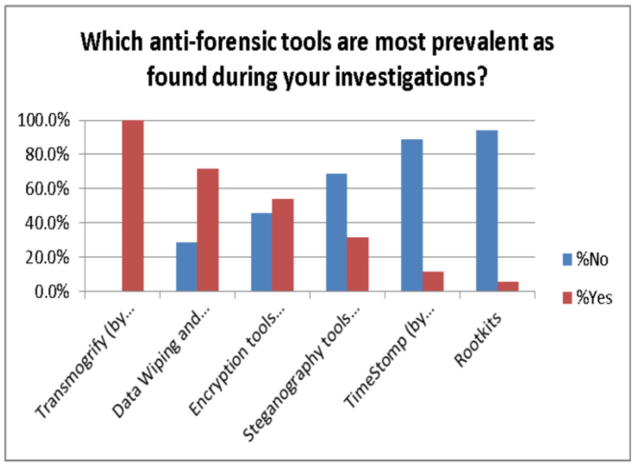


Figure 6. AF tools most prevalent in investigations

4.3 Respondents Rating of Their AF Abilities

Respondents were asked to rate their own Anti-Forensics abilities by means of three questions: their own knowledge of AF, their prior exposure to AF and their ability to counteract the use of AF tools. The 3 items were considered as a single construct as they logically represent respondent's abilities to investigate AF.

The items were subjected to a reliability analysis to determine the internal consistency between items to measure a single construct. The overall Cronbach Alpha was 0.825 which is considered a very good reliability. A mean score of approximate 2.8 for this construct was achieved

The influence of the biographical profile of respondents, employment sector, and type of forensic investigation participation, education and training level was tested using ANOVA. In the absence of normality, a non-parametric test (Mann-Witney U test for 2 groups and Kruskal-Wallis for more than 2 groups) was performed. If the variances of the scores of the groups are unequal, the result of the Welch test is reported.

The male respondents rate themselves significantly more highly in AF ability than their female counterparts (Welch test p = 0.0361). The age groups 31-40 and 41-50 years also rated themselves significantly higher than the two other (smaller) age groups i.e. those younger than 31 or above 50 years old (Welch test p = 0.0088).

Respondents who are practicing digital forensic investigators rate themselves significantly higher than those who are currently not practicing (Welch test p = 0.0385). Finally, respondents with more than 2 years of experience rate their AF abilities higher than the group with 2 or less years' experience (p = 0.0355)

However, there was *no* significant correlation between the AF ability of the respondent and:

the sector in which employed, the type of forensic participation or the respondent's training and education level.

4.4 The Impact of Anti-Forensics upon Investigations

Figure 7 summarizes the responses to the questions related to the impact of AF on investigations.

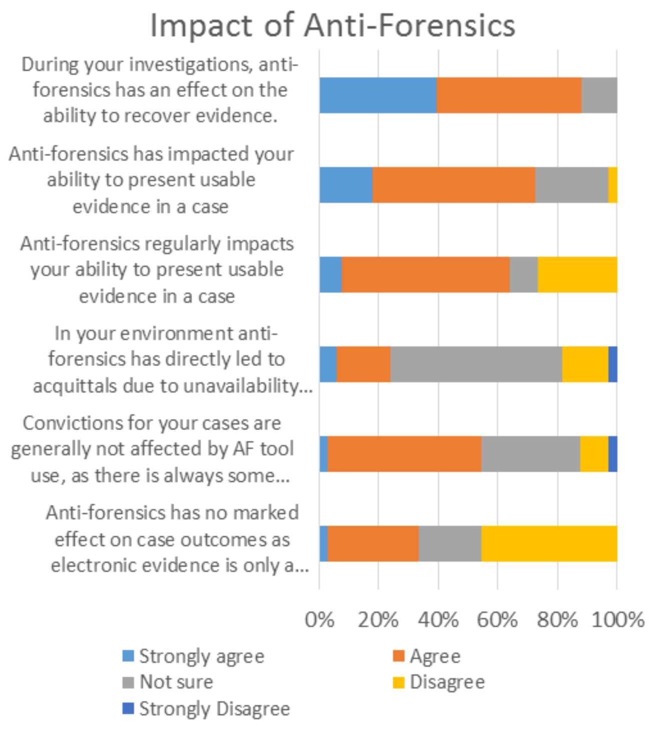


Figure 7. Impact of Anti-Forensics

Factor analysis was performed on the above test items to see if the five statements can be summarized into meaningful constructs. This was done using Principal Component extraction and Varimax (orthogonal) rotation techniques.

The factor analysis yielded 2 underlying constructs (factor loadings > 0.4), although item 4 ("Anti-forensics regularly impacts your ability to present usable evidence in a case") is ambiguous as it loads strongly upon both factors.

Two new constructs were created: the *Effect of AF on evidence*, which was composed of four test items (Cronbach Alpha of the resulting construct was 0.8433) and the *Impact of AF*.

The influence of: Biographical profile of respondents, Employment sector, Type of forensic investigation participation, Education and training level, Knowledge of AF tools and techniques (own rating), Prior exposure to AF (own rating), Familiarity with various AF techniques, Exposure to various AF Tools upon the constructs (Impact of AF and Effect of AF upon evidence) was investigated using Analysis of Variance (ANOVA) since the mean scores derived from the Likert scale statements can be viewed as continuous variables and their distributions were found to be roughly normally distributed.

Table 5. Construct creation after factor analysis.

Test Item (Question)	Factor 1	Factor 2
During your investigations, anti-forensics has an effect on the ability to recover evidence.	0.9132	0.0582
Anti-forensics has impacted your ability to present usable evidence in a case	0.8727	0.2216
In your environment anti-forensics has directly led to acquittals due to unavailability of evidence	0.7331	0.3592
Anti-forensics regularly impacts your ability to present usable evidence in a case	0.5583	0.5489
Convictions for your cases are generally not affected by AF tool use, as there is always some usable digital evidence.	0.3488	0.6433
Anti-forensics has no marked effect on case outcomes as electronic evidence is only a portion of the evidence presented to court or legal proceeding.	0.0301	0.9433

However, only one factor was found to exert a significant influence and that was one particular type of forensic participation that significantly influences the mean score of the Impact of AF, namely *mobile device investigations* (χ^2 statistic=4.26, DF=1, p-value =0.039). It is possible that this is an artefact of the data or small sample size.

5 DISCUSSION AND CONCLUSION

The main purpose of this research was to establish the impact of AF on DF investigations in a South African context.

Contextual factors and individual characteristics did *not* show a significant influence of respondent's rating of their knowledge of AF tools and techniques. More surprisingly, neither the respondent's familiarity with the various AF tools and techniques, nor their exposure to AF appeared to affect score for Effect of AF upon evidence or their score for impact of AF.

Perhaps contradictorily, although practitioners rate the value of AF artefacts highly, they don't make an effort to identify them as part of their investigations, as can be seen in table 6a & b below.

Table 6a & 6b. Probative value of AF artefacts and relative effort used to identify AF.

Rating of probative value of artefacts that AF tools leave behind	N	Mean
Critically valuable	6	2.61
Very valuable	12	2.44
Valuable	14	3.02
Moderately valuable	2	4.33
Useless	1	4
Rating of efforts to actively identify anti-forensics as part of investigations	**N**	**Mean**
Always	4	2.91
Often	12	2.38
Sometimes	13	2.97
Rarely	4	3.16
Never	2	4.16

The concern is that investigators who do not place a high value on AF artefacts and do not employ significant efforts to identify AF as part of their investigations could potentially be overlooking evidence of AF use, or indeed the opportunity to recover usable evidence.

Whilst the current research attempted to understand the impact of Anti-forensics on evidence, a troubling trend emerged as part of this research. Whilst most respondents (89.9%) list their knowledge of AF tools and techniques as average to good, it appears that that exposure is limited to certain types of AF tools and techniques.

From the findings, it is clear that investigators are quite familiar with the more common AF techniques such as data hiding and data destruction, but much less familiar with the more complex techniques such as data contraception amongst others.

Correspondingly, respondents were much more familiar with the tools related to data wiping and encryption than they are with the more complex tools such as steganography, rootkits and data contraception. This trend is also supported by the finding that most respondents feel that their cases are affected mainly by the more common AF techniques such as data destruction and data hiding.

The risk is that DF investigators are not properly versed in the more complex techniques and tools, and as such does not have the skills to identify the use of such tools and may be ignorant of their presence and effect on the cases that they participate in. This combined with the results that show a lack of active identification of AF as part of investigations, due to the low value placed on AF artefacts leaves a potential gap for usable evidence of probative value to be overlooked, as the AF technique or tool used to destroy or misrepresent that evidence is not identified.

The main contributions of this paper to the body of academic knowledge is the application of a model-driven empirical investigation as well as a number of findings which partly collaborate and partly extend current knowledge. For the practitioner community, it is clear that additional effort and resources must be deployed to inform and educate practitioners about common anti-forensic tools and activities.

Apart from further empirical research to establish comparative baselines in other countries or, longitudinally, one or two years into the future, we also strongly recommend additional in-depth research into the advanced AF techniques and tools to establish the actual effect of these advanced techniques in investigations.

6 REFERENCES

[1] R. Harris, "Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem," Digital Investigation, 2006, 3(Supplement 1), pp. 44-49.

[2] M. Pollitt, "Applying traditional forensic taxonomy to digital forensics" in Advances in Digital Forensics IV (pp. 17-26), New York: Springer, 2008.

[3] E. Casey, Handbook of digital forensics and investigation (Non Trans.), San Diego, California: Elsevier, 2010.

[4] Verizon, "Data breach investigations report", Retrieved 05/14, 2012, from http://www.verizonbusiness.com/resources/reports/rp_data-breachinvestigations-report-2012_en_xg.pdf

[5] E. Casey, Digital evidence and computer crime. Burlington: Elsevier, 2004.

[6] K.M. Hess, Criminal investigation (9th ed.). New York: Delmar, Cengage Learning, 2009.

[7] B. Carrier, File system forensic analysis. Addison Wesley Professional, 2005.

[8] G.C. Kessler, "Anti-forensics and the digital investigator," paper presented at the Proceedings of the 5th Australian Digital Forensics Conference, 2007, 1(1) 5. Retrieved from http://scissec.scis.ecu.edu.au/proceedings/2007/forensics/00_Forensics2007_Complete_Proceedings.pdf

[9] B. Blunden, The rootkit arsenal escape and evasion is the dark corners of the system, Wordware Publishing, 2009.

[10] D. Forte, "Dealing with forensic software vulnerabilities: Is anti-forensics a real danger?" Network Security, 2008(12), 18.

[11] H. Sencar, M. Rankukar, & A. Akansu, Data hiding fundamentals and applications. San Diego, California: Elsevier, 2005.

[12] A. Philipp, D. Cowen, D., & C. Davis, C., Hacking exposed computer forensics, second edition: Computer forensics secrets & solutions (Second Ed.) New York: McGraw-Hill Osborne Media, 2009.

[13] P. Wayner, Disappearing cryptography, second edition: Information hiding: Steganography & watermarking. The Morgan Kaufmann series in software engineering and programming, New York: Morgan Kaufman, 2003.

[14] J. Pan, H. Huang, L. Jain, & W. Fang, Intelligent multimedia data hiding. New York: Springer, 2007.

[15] H/ Nemadi, & L. Yang, Applied cryptography for cybersecurity and defense: Information encryption and cyphering. Hershey, PA: Information Science Reference, 2011.

[16] L. Roy, "Lockdown: Secure your files with TrueCrypt", Makeuseof.com.

[17] D.O. Kennedy, J. Gorman, & M. Aharoni, Metasploit: The penetration tester's guide. No Starch Press, 2011.

[18] D. Behr, "Anti-forensics – what it is what it does what you need to know," New Jersey Lawyer Magazine, 2008, 255, 4-5.

[19] D. Forte, & R. Power, "A tour through the realm of anti-forensics," Computer Fraud & Security, 2007(6), 18-20.

[20] M. Kedziora, "Anti-forensics overview," retrieved 16/02, 2011, from http://www.forensics-research.com

[21] D. Maynor, Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research, San Diego, California: Elsevier, 2011.

[22] M. Geiger, "Counter-forensic tools: Analysis and data recovery," 18th Annual FIRST Conference, Maltimore, Maryland, 25-30 June 2006.