APPLICATION OF SECRET SHARING TECHNIQRES ON CONFIDENTIAL FORENSIC INVESTIGATION

Shuhui Hou*, Siuming Yiu†, Tetsutaro Uehara‡ and Ryoichi Sasaki§
*Dept. of Information and Computing Science
University of Science and Technology Beijing, Beijing, China
Email: shuhui@ustb.edu.cn
†Dept. of Computer Science
The University of Hong Kong, Hong Kong, China
Email: smyiu@cs.hku.hk
‡Research Institute of Information Security
Wakayama, Japan
Email: uehara@riis.or.jp
§Graduate School of Science and Technology for Future Life
Tokyo Denki University, Tokyo, Japan
Email: sasaki@im.dendai.ac.jp

ABSTRACT

How to capture digital evidence is crucial for counteracting against the computer and cyber crimes. However, in a large sharing system (e.g. in a data center or even in a cloud system), cloning the whole set of data for investigation is not feasible and also may have conflict with privacy issues as most of the data is not relevant to the crime. The problem is how to retrieve the relevant information without the investigator knowing other irrelevant data while the server administrator does not know what the investigator is searching. To solve this problem, Hou et al. tried to model the problem as a secure keyword searching problem and proposed a few encryption-based schemes. While the schemes are theoretically sound, the efficiency is a concern. In this paper, we solve the same problem using secret sharing schemes and show that the efficiency can be greatly improved, thus making it practical. We demonstrate the advantages of our solution in terms of computational complexity analysis and experiments. The experimental results show that our solution is much faster in searching and processing the relevant data, so is superior to the existing work.

KEYWORDS

confidential forensic investigation, secret sharing, Chinese Remainder Theorem, application of cryptographic schemes, third-party neutral

1. INTRODUCTION

In recent years, computer-related crimes are becoming rampant and these crimes caused a lot of damages to business, the public and the government. Digital forensic investigation is crucial to counterattack against these crimes, which has relied on digital evidence collected from the storage device(s). However, with the development of computing technology (e.g., cloud system), evidence collection becomes difficult due to the huge volume of data and privacy issues. For example, the server under investigation may store the data from thousands or even more irrelevant users. Due to the privacy concern, the investigator may have no rights to access the irrelevant data especially as it may involve confidential information. Thus, it is not feasible for the investigator to clone a copy of all data from the server. A trivial solution is to ask the server administrator to retrieve only the data relevant to the crime (or the suspect) and hand this to the investigator. This simple solution also may not work since the investigator may not want the administrator to know what he is looking for due to some special crimes. In this paper, we focus on how to efficiently retrieve relevant data (i.e., evidential data) from a huge amount of data while the investigator does not know other irrelevant data and the server administrator does not know what the investigator is searching.

Under the assumption that the server administrator is willing to cooperate and search the relevant data/information for the investigator whenever a warrant is provided, Hou et al. [1], [2] is the

first to abstract the problem as a secure keyword searching problem and provides solutions to tackle the problem. The idea behind their solutions is as follows. The investigator specifies single or multiple keyword(s) based on investigation intent or investigation subject, encrypts and sends it (or them) to the server administrator; the administrator encrypts all the data files stored on the server (where each data file is represented as a set of words), searches for encrypted keyword(s) in the encrypted data files and returns the relevant data (i.e., the data files containing the keyword(s)) to the investigator. By searching the encrypted data files with encrypted keyword(s), the administrator has no idea of what keyword(s) the investigator is looking for; by performing investigation only on the relevant data files returned by the administrator, the investigator has no idea of other irrelevant data files (i.e., the data files without containing the keyword(s)). It should be pointed out that while not perfect, keyword searching is currently the most widely recognized culling method in the area of digital forensics and e-discovery.

The schemes for single keyword search on encrypted data [1] are based on homomorphic encryption and commutative encryption, and the schemes for multiple keyword search [2] are based on the protocol for privacy preserving set intersection. These schemes utilize encryption technology directly or indirectly, so the investigation efficiency is low due to the time consuming encryption and decryption procedures on large amount of data.

In this paper, we aim at improving the investigation efficiency. We propose to make use of (t, n)-threshold secret sharing schemes rather than encryption to realize the secure keyword searching/matching procedure. The high level idea is as follows. The data files managed by the server administrator are treated as a sequence of words. Each word and also each keyword given by the investigators are treated as secrets and are divided into n pieces of secret shares (or shadows). We employ a third-party neutral (e.g., technology experts) to match each word in a file with each keyword from the investigator, more precisely, the third-party neutral is matching the shares of each word to the shares of each keyword provided by the investigator. Once the shares of one word in a file matches the shares of a keyword, t shares of all remaining words of the same file will be forwarded to the investigator to reconstruct the whole file based on the principle of (t, n)-threshold secret sharing schemes. Note that the third-party neutral cannot learn anything about the keywords and the data files since he only knows the shares, not the original words. Its involvement in the matter is neutral and unbiased.

The reasons why we utilize (t, n)-threshold secret sharing schemes to improve confidential forensic investigation include: (1) in the cloud computing environment, there have been several data management services which are using secret sharing technology for managing/storing server data more securely (e.g., SecureCube/Secret Sharing of NRI SecureTechnologies, Ltd., Japan). It is reasonable to let the server administrator divide the data files into n pieces for protecting them from leaking; (2) the secret sharing technology can divide data files into separate files and store them in different physical locations. Each separate file becomes meaningless and the original data files cannot be reconstructed from any one separate file. Besides, the original data files can be reconstructed even if some of the separate files are unobtainable. On the contrary, it is possible for encryption-based schemes [1], [2] to recover the original data files only from their encrypted files. Also, the original data files cannot be recovered if the encrypted files become unobtainable due to various reasons. It should be pointed out the main contribution of the paper is not developing new cryptographic schemes, but making use of well-known schemes to solve a real application. We will illustrate the practicality of our proposed solution using experiments and show that our solution is much faster than the schemes given in [1], [2].

The rest of the paper is organized as follows. In Section 2, we introduce (t, n)-threshold secret sharing and Chinese Remainder Theorem which are necessary for understanding our solution. Section 3 addresses how our solution works based on the secret sharing schemes. The performance evaluation is conducted and experimental results are analyzed in Section 4. In Section 5, we conclude the paper and highlight some of the future work.

2. THRESHOLD SECRET SHARING SCHEMES

Definition 1. *Let $\mathcal{F}$ be a field. A (t,n)-threshold secret sharing scheme is a secret sharing scheme that can divide a secret $s\in\mathcal{F}$ into shares $\{s_1,s_2,\ldots,s_n\}\in\mathcal{F}$ so that $t \leq n$ and:*

1) *Given any set of t or more shares s_i, s can be reconstructed;*
2) *Any set of fewer than t shares gives no information about s.*

Modern cryptographic secret sharing, attributed to Shamir and Blakley, was first designed for key safeguarding, but has been applied far beyond its original intent. There exists a multitude of secret sharing schemes: Shamir's secret sharing scheme [3] uses curves and reconstructs the secret by polynomial interpolation; Blakley's scheme [4] reconstructs the secret by the intersection of hyperplanes; the Asmuth $\&$ Bloom secret sharing scheme [5] uses congruence classes to solve the secret sharing problem; the Mignotte's threshold secret sharing scheme [6] uses the Chinese Remainder Theorem to solve the problem of secret sharing.

In this paper, we adopt the Mignott's threshold secret sharing scheme to divide the data into pieces so as to protect the investigation subject and irrelevant data from unauthorized disclosing.

2.1. Chinese Remainder Theorem(CRT)

Theorem 1. *Given a set of simultaneous congruences*

$$x \equiv a_i \ (mod\ n_i) \tag{1}$$

for $i = 1, 2, \ldots, r$ and for which the n_i are pairwise relatively prime, the solution of the set of congruences is

$$x \equiv a_1 b_1 \frac{N}{n_1} + \cdots + a_r b_r \frac{N}{n_r} \ (mod\ N) \tag{2}$$

where $N = n_1 n_2 \cdots n_r$, and the b_i are determined from

$$b_i \frac{N}{n_i} \equiv 1 \ (mod\ n_i) \tag{3}$$

2.2. Mignotte's Threshold Secret Sharing Scheme

Along with the CRT, Mignotte's threshold secret sharing scheme uses special sequences of integers referred to as Mignotte sequences.

Definition 2. *Let n, t be positive integers, $n \geq 2$ and $2 \leq t \leq n$. An (t,n)-Mignotte sequence is a sequence of pairwise coprime positive integers $p_1<p_2<\ldots<p_n$ such that*

$$\prod_{i=0}^{t-2} p_{n-i} < \prod_{i=1}^{t} p_i. \tag{4}$$

The Mignotte's threshold secret sharing scheme works as follows:

- **Initialization**
  Choose a (t,n)-Mignotte sequence such that $\beta<s<\alpha$, where the secret s is chosen as a random integer, $\alpha = \prod_{i=1}^{t} p_i$ and $\beta = \prod_{i=0}^{t-2} p_{n-i}$;
- **Creating the shares**
  The secret shares s_i are computed by $s_i \equiv s \bmod p_i$, for all $1 \leq i \leq n$;
- **Reconstructing the secret**
  Given t distinct shares $s_{i_1},s_{i_2},\ldots,s_{i_t}$, where $i_1,i_2,\ldots,i_t$ are t numbers arbitrarily taken from $\{1,2,\ldots,n\}$. Based on the CRT, the secret s is reconstructed as the unique solution modulo $p_{i_1},p_{i_2},\ldots,p_{i_t}$ of the system

$$\begin{cases} x \equiv s_{i_1} \ mod\ p_{i_1} \\ \quad \vdots \\ x \equiv s_{i_t} \ mod\ p_{i_t} \end{cases} \tag{5}$$

3. CONFIDENTIAL FORENSIC INVESTIGATION BASED ON SECRET SHARING

3.1. Assumption and Notation

Given a shared server, relevant data (or evidential data) is stored together with the irrelevant data (some may involve confidential information or private information) on the server. Whenever a warrant can be provided, we assume that the server administrator is willing to cooperate and will not

hide some of the files from being searched. Besides, we assume that the administrator can check what data is extracted from the server when the data is presented as evidence in a court of law, that is, he can check if the server data is used in a secure manner.

For the brevity of description, we adopt the following notation. Based on the investigation intent or the investigation subject, the investigator will specify a set of keywords, denoted as $w^*=\{w_1^*, w_2^*, \ldots, w_u^*\}$, where each keyword w_i^* is assumed to be d-bit long; The data stored on the server is viewed as a set of documents, denoted as $\{W^1, W^2, \ldots, W^L\}$. Any one document $W^i \in \{W^1, W^2, \ldots, W^L\}$ consists of a sequence of words, denoted as $W^i=\{w_1^i, w_2^i, \ldots, w_v^i\}$ where every word is also assumed to be d-bit long. Although the keywords or words are of variable length, we can transform them into equal length by picking a fixed-size block like the work [7] where words that are too short or too long may be padded to a multiple of the block size with some pre-determined padding format. We also can use hash function to map the variable-length words into the fixed-length words. We also assume that w^* and W^i $(1 \leq i \leq L)$ come from the same domain (i.e., all their elements are d-bit long).

3.2. Details of Proposed Solution

Take "single keyword search" as an example, we describe how to realize confidential forensic investigation based on secret sharing (Figure 1).

We employ a third-party neutral (e.g., technology experts) to carry out the matching process. Utilizing third-party neutral will ensure safe handling of evidence since it is impartial and this impartiality presumptively aids in the evidence-finding process and administration of justice.

1) To manage or store the server data securely, the **administrator** uses (t, n)-threshold secret sharing schemes to divide the data into n pieces so that any one piece of data is meaningless and the original data cannot be reconstructed from any one piece of data. In detail,
   - He chooses n pairwise coprime positive integers $p_1, p_2, \ldots, p_n$ to construct a (t, n)-Mignotte sequence such that $\beta < w_l^i < \alpha$, where w_l^i denotes any word of any document $W^i \in \{W^1, W^2, \ldots, W^L\}$, $\alpha = \prod_{i=1}^{t} p_i$ and $\beta = \prod_{i=0}^{t-2} p_{n-i}$. As every word is d-bit long, we have $2^{d-1} \leq w_l^i < 2^d$.
   - To prevent the investigator from knowing the irrelevant data, he views each word w_l^i of the document W^i as secret and divides it into n secret shares by computing $w_{lk}^i \equiv w_l^i \bmod p_k$, for all $1 \leq k \leq n$;
   - He sets a unique id for the document W^i and marks its all secret shares w_{lk}^i $(1 \leq l \leq v, 1 \leq k \leq n)$.
   - As an investigator requests him to cooperate in investigation, he will send the n pairwise coprime positive integers $p_1, p_2, \ldots, p_n$ to the investigator, and provides third-party neutral (e.g., technology experts) all the marked w_{lk}^i.
2) For preventing the administrator and the third-party neutral from knowing the investigation subject, the **investigator** views the keyword w_i^* as secret and divides it into n secret shares by computing $w_{ik}^* \equiv w_i^* \bmod p_k$, for all $1 \leq k \leq n$; He randomly picks t distinct shares $w_{ii_1}^*, \ldots, w_{ii_t}^*$ of w_i^* ($i_1, i_2, \ldots, i_t$ are t numbers from $\{1, 2, \ldots, n\}$) and gives them to the third-party neutral.
3) Without knowing n pairwise coprime positive integers $p_1, p_2, \ldots, p_n$, the **third-party neutral** is in charge of matching the shares of w_i^* (which are from the investigator) with the shares of w_l^i (which are from the administrator). As shown in Figure 1, the **third-party neutral** will return t secret shares of each word of the document W^i to the investigator if there exists a word w_j^i which involves t out of n secret shares (in fact, all n shares will be the same) such that

$$\begin{cases} w_{ji_1}^i &= w_{ii_1}^* \\ &\vdots \\ w_{ji_t}^i &= w_{ii_t}^* \end{cases} \tag{6}$$

The third-party neutral cannot reconstruct the keywords and the documents without knowing n pairwise coprime positive integers, i.e., it will learn nothing about the keywords and the

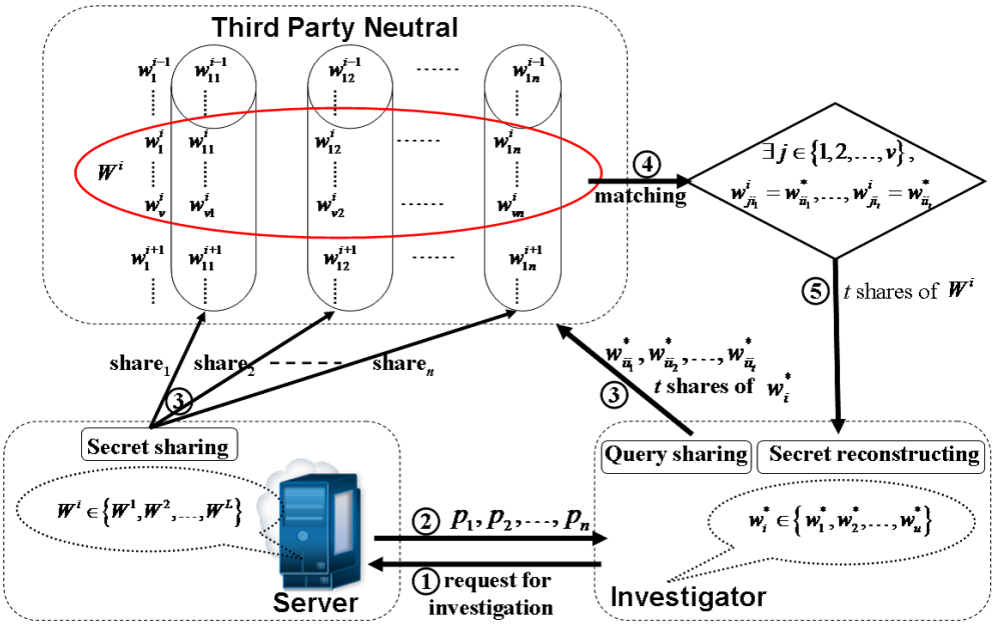


Figure 1. A Schematic Illustration of Single Keyword Search

documents. Furthermore, the third-party neutral separates the administrator from searching results so that the administrator has no idea of what the investigator is looking for, and the third-party neutral only returns the documents containing the matched keywords so that the investigator cannot know other irrelevant documents.

4) Based on the t secret shares of each word which are returned from the third-party neutral, the **investigator** can reconstruct each word and then reconstruct the whole document W^i such that W^i contains the keyword w_i^*. Finally, he can perform investigation on W^i for capturing evidence.

Based on the "single keyword search", "multiple keyword search" can be easily realized. The investigator provides the third-party neutral t secret shares of w^* (i.e., $w_{ii_1}^*,\ldots,w_{ii_t}^*$, $1\leq i\leq u$) and the third-party neutral matches them with the secret shares of the document W^i ($1\leq i\leq L$). The third-party neutral will return t secret shares of the document W^i which contains $w_{ii_1}^*,\ldots,w_{ii_t}^*$, $1\leq i\leq u$. Then, the investigator can reconstruct each word and the whole document which contains multiple keywords.

4. PERFORMANCE EVALUATION

4.1. Security Analysis

In our proposed scheme, the investigator can efficiently perform forensic investigation where he captures evidence only from the relevant data. The investigator cannot learn more than the search result and the server administrator cannot learn the investigation subject. In other words, the confidentiality of investigation subject (or keywords specified by the investigator) and privacy of irrelevant data (or documents without containing specified keywords) are protected. It is obvious that whether they can be completely protected relies on the security of secret sharing schemes. Without knowing the n pairwise coprime positive integers $p_1,p_2,\ldots,p_n$, the third-party neutral cannot reconstruct the specified keywords and the server data even he knows all the shares of data. Thus, no information leak occurs on the third-party neutral side.

4.2. Computational Complexity

For the sake of simplicity, we merely evaluate the computational complexity as one keyword w_i^* is input and one document W^i containing w_i^* is

output. We use the number of modular operations (MO), modular multiplications (MM), modular exponentiations (ME) and modular inversions (MI) to measure the computational complexity.

In our solution, the investigator needs n MO to compute the shares of keyword w_i^* and needs tv MI and tv MM to reconstruct the document W^i; the administrator needs nv MO to compute the shares of the document W^i; the third-party neutral needs t^2v comparison operations to perform searching and returning the relevant data W^i to the investigator.

To show the advantage of our solution, we also analyze the computational complexity of one existing work [1] which is based on Paillier cryptosystem. The investigator needs 2 ME and 1 MM to encrypt the keyword w_i^* and needs v ME, v MI and v MM to decrypt the document W^i; the administrator needs $2v$ ME and v MM to encrypt the the document W^i (we omit operations involved in the zero knowledge proof for the clarity).

We listed the above computational complexity in Table 1. ME and MI are usually more complex than MO and MM, so our solution is superior over the existing work in terms of computational complexity.

4.3. Experiments Evaluation

Taking the scheme based on Paillier cryptosystem as an example, we carry out experiments and show our solution has faster processing time than the existing work. We conducted the experiments on the Genuine Intel(R) CPU U7300, 1.30 GHz PC with 2 GB RAM, MATLAB 7 as integrated environment. We take a word document (consisting of 273 English words separated by spaces), randomly set the positions where the keyword appears five times, and use the average processing time to measure the efficiency. We set the following parameters in our experiment:

1) Proposed solution based on secret sharing
   Let $t = 3$, $n = 5$, $p_1 = 5$,$p_2 = 7$,$p_3 = 11$,$p_4 = 13$,$p_5 = 17$, then $\alpha = \prod_{i=1}^{t} p_i$=385 and $\beta = \prod_{i=0}^{t-2} p_{n-i}$=221; The "preprocessing" includes using MD5 hash function to transform the variable-length words to fixed-length words so that each word satisfies the condition of $(3, 5)$-threshold secret sharing.
2) Existing work based on Paillier cryptosystem
   The "preprocessing" includes using MD5 hash function to transform the variable-length words to fixed-length words so that each word satisfies the condition of Paillier cryptosystem, which is detailed below.
   - Key generation
     Let $p = 3$,$q = 11$, then n=pq=33; Let $g = 166$, compute λ=lcm($p-1$,$q-1$)=10 and μ=$(\Phi(g^{\lambda} mod \ \ n^2))^{-1} mod \ \ n$=2, where $\Phi(u) = \frac{u-1}{n}$. That is, the public key is (n, g)=$(33, 166)$ and the secret key is (λ, μ)=$(10, 2)$;
   - Encryption
     Let m (m<n and $m \in Z_n$) be plaintext, pick a random number $r \in Z_n^*$, the ciphertext c=$g^m r^n \ mod \ n^2$;
   - Decryption
     The plaintext m=$\Phi(c^{\lambda} mod \ n^2) \cdot \mu \ mod \ n$.

The experimental results are shown in Table 2, where dividing and reconstructing in our proposed solution are corresponding to the encryption and decryption procedures in existing work respectively, and "s" (i.e., "second") is the execution cputime. The processing time in our solution, especially the time on dividing the document into shares and searching are less than the processing time on encrypting the document and searching in existing work. Further, we carried out the above experiments on several larger size documents. As the document size is growing, the encryption time and searching time in existing work grow linearly while the time of secret dividing and searching time in our solution stay more or less the same (please refer to Figure 2). Thus, our solution is more scalable.

On the other hand, dividing data into n shares may result in more storage cost than encrypting the data, but it may not be a problem as it is not necessary to store the shares of all files before performing the matching. Once the matching is done for a file, it is no longer needed.

5. DISCUSSION AND CONCLUSIONS

Another advantage of our solution is to support multiple investigators efficiently. Generally, prosecution and defense attorneys are supposed to be

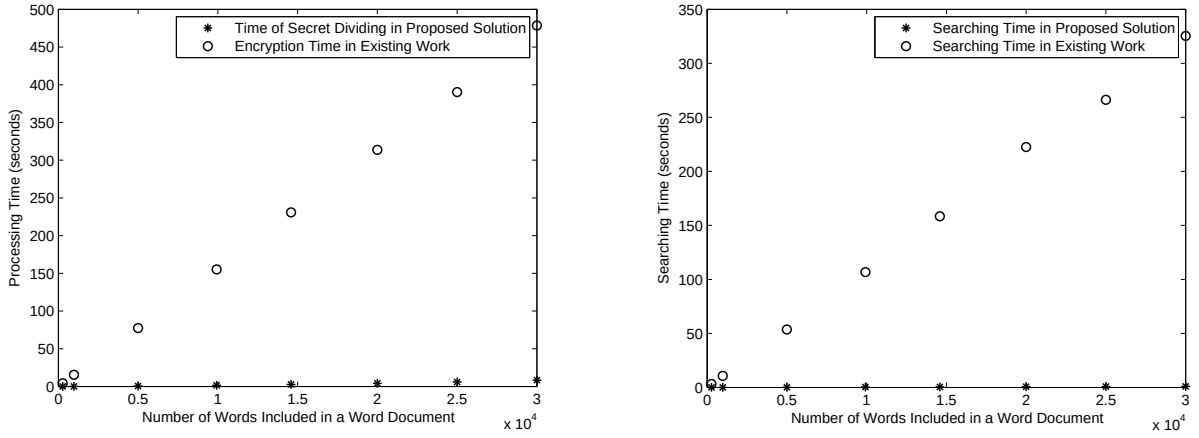
Table 1. Computational Complexity

	Investigator Side				Administrator Side		
	MO	MM	ME	MI	MO	MM	ME
Proposed Solution	n	tv	$-$	tv	nv	$-$	$-$
Existing Work	$-$	$1+v$	$2+v$	v	$-$	v	$2v$

"-": no computational complexity or the computational complexity can be neglected.

Table 2. Processing Time

	Preprocessing	Dividing/ Encryption	Reconstructing/ Decryption	Searching
Proposed Solution	1.6419 s	0.0351 s	0.1664 s	0.0312 s
Existing Work	1.6536 s	4.3719 s	2.3868 s	3.2604 s



(a)Secret Dividing/Encryption time

(b)Searching Time

Figure 2. Processing Time on Larger Size Documents

opposite parties in the criminal justice process. The prosecution attorney aims at getting a conviction and the defense attorney aims at finding facts that prove innocence. They stand in opposite positions so they have different requests during the procedure of investigation. In other words, there exist some cases where multiple investigators with different investigation intent need to perform investigation on the same data set.

Existing work [1], [2] cannot support multiple investigators *efficiently*, where the administrator has to re-encrypt the entire server data for each investigator with a different encryption scheme. In our proposed solution, it is easy to support multiple investigators. The administrator can cooperate with multiple investigators by sending the n pairwise coprime positive integers p_1,p_2,...,p_n of (t, n)-Mignotte sequence to each of them (e.g., prosecution and defense attorneys). The investigators can choose their own keywords to be sent to the third-party neutral. Of course, the third-party neutral may be able to discover that the same keyword is searched by different investigators. Since he does not know the exact keyword, this may not be a major concern.

To summarize, in this paper, we show that using a (t, n)-threshold secret sharing scheme, we can solve a real problem in computer forensic investigation. The experimental results show that our solution is superior over the existing work in terms of computational complexity and practical performance. In practice, we can replace MD5 by SHA-256 in the solution if we want to increase the security level. This will affect the pre-processing time (it is estimated to be about three times longer) but the other processing time stays the same.

For future work, this paper and also all previous work do not provide a total solution to solve the real forensic investigation problem yet. For example, in these schemes, the administrator has no way to make sure that the keywords provided by the investigator are all relevant to the crime case. And the schemes also cannot guarantee that all files kept by the administrator can be searched over. All these require more investigation. In this paper, we pick one of the threshold secret sharing schemes as illustration, other schemes may work equally well or even better. A better and specially designed solution for solving this real application is always desired. We are now in the process of applying this result to some real crime cases and the feasibility and efficiency of the solution in real cases will be further investigated.

ACKNOWLEDGMENT

This work is partially supported by “Heiwa Nakajima Foundation, Japan”, partially sponsored by “the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry” and partially sponsored by “the National Science Foundation of China (No.11201025, No.11101028)”.

REFERENCES

1 S. Hou, T. Uehara, S.M. Yiu, Lucas C.K. Hui, K.P. Chow: Privacy Preserving Confidential Forensic Investigation for Shared or Remote Servers. In: 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp.378-383 (2011).

2 S. Hou, T. Uehara, S.M. Yiu, Lucas C.K. Hui, K.P. Chow: Privacy Preserving Multiple Keyword Search for Confidential Investigation of Remote Forensics. In: 2011 Third International Conference on Multimedia Information Networking and Security, pp.595-599 (2011).

3 A. Shamir: How to share a secret. Commun. ACM, vol.22, No.11, pp.612-613 (1979).

4 G.R. Blakley: Safeguarding cryptographic keys. In: AFIPS Conference Proceedings, vol.48, pp.313-317 (1979).

5 C. Asmuth, J. Bloom: A modular approach to key safeguarding. IEEE Transactions on Information Theory, vol.29, No.2, pp.208-210 (1983).

6 M. Mignotte: How to share a secret?. In: Cryptography - Proceedings of the Workshop on Cryptography, Lecture Notes in Computer Science, vol.149, pp.371-375 (1983).

7 D. Song, D. Wagner, A. Perrig: Practical Techniques for Searches on Encrypted Data, In: Proceedings of IEEE Symposium on Security and Privacy 2000, pp.44-55(2000).