A Confidentiality Preserving Data Leaker Detection Model for Secure Sharing of Cloud Data using Integrated Techniques

Ishu Gupta
Department of Computer Applications
National Institute of Technology
Kurukshetra, India
ishugupta23@gmail.com

Ashutosh Kumar Singh
Department of Computer Applications
National Institute of Technology
Kurukshetra, India
ashutosh@nitkkr.ac.in

***Abstract*—Leakage of critical information through intentional or unintentional disclosure by the malicious entities to the aggrieved third party poses one of the most serious security hazards to various organizations. In the real world scenario, a data distributor has to share confidential data among various stakeholders. It comprises a number of threats in transferring the critical data, as it can be made public by any disgruntled employee or the indignant entity. Data Leakage Problem (DLP) has become a critical challenge and it keeps on increasing day by day. It is crucial to protect the critical data as it can be misused by any vicious entity. To resolve DLP, we present a generic Data Leaker Detection Model (DLDM), which identifies the malicious entity responsible for data leak. The proposed model ensures the data confidentiality and preserves the security of the sensitive information via applying an integration of cryptography, watermarking and hashing techniques. The results show that the computation times in the complete process are 2285.31 ms and 5628.84 ms when 200 documents of size 20MB are provided to the single user and distinct users respectively.**

***Index Terms*—Cloud computing, cryptography, data leakage, data privacy, watermarking, hashing, security, guilty client.**

I. Introduction

In the recent years, Internet and related technologies have grown rapidly. It offered unrivaled capability to access and redistribute the digital data [1], [2]. It is easy to copy huge amount of the digital data at almost no cost that can be transferred in very short time via the Internet [3]. The various organizations use this facility to enhance their capability by transferring data from one place to another, but it involves number of threats in transferring the confidential data as it can be made public by any malicious entity [4]–[6]. According to an exotic chronology of data breaches explored by Privacy Rights Clearinghouse (PRC), $11,583,442,497$ records have been breached in the united state alone from $9,094$ data breaches made public since 2005 [7]. It is not hard to speculate that it is just a fingertip of an iceberg as most of the times, the data leakage cases are not recorded due to concern of restrictive penalties and loss of customer trust [8]–[10]. In the year 2018, $10,674,189$ records have been breached in the health care, medical providers and medical insurance services organization alone and 340 breaches made public [11].

In the course of doing business, the critical organizational data have been shared among various supposedly trusted parties within or outside the organizational premises [12], [13]. In today's globally networked systems, sharing of data among multiple entities poses a threat to organizational confidentiality and individual privacy as any culprit employee can leak this data to unauthorized party advertently or unknowingly [14], [15]. Leakage of sensitive data in deliberate or undeliberate manner by the indignant entities and the venomous users influences one of the most grievous security threats to the organizations [16]. According to 2018 annual cost of data breach study conducted by Ponemon institute, the average size of data breaches, the average total cost of a data breach and the average cost for each compromised record have all enlarged beyond the 2017. This study states that the average consolidated cost of a data breach rose from $\$3.62$ million to $\$3.86$ million in the year 2018, which indicates an increase of 6.4% over the previous year. Also, the cost procured for each stolen or lost record consisting confidential information has increased by 4.8% from consolidated average of $\$141$ to $\$148$ and the average size of data has raised by 2.2% [17]. Because of these reasons, DLP is increasing day by day.

The number of compromised data records has reached up to 10 billion over the past five years and it keeps on raising as the number of users are increasing, also the malicious entity [18]. DLP has reached to a new dimension and it is consequential to protect the sensitive data from any unauthorized access [19], [20]. As the data is shared among multiple entities, thus it becomes difficult to identify the malicious entity who has leaked the confidential information [21]. There is a need of mechanism that can handle the DLP. In the proposed work, we present a model that solves DLP by identifying the malicious entity who has leaked the organization's sensitive data.

A. Our Contributions

In order to improve the security for sharing the data in cloud computing environment, we propose a model to detect the Guilty Client G_c who leaks the data provided by the Cloud Cl_{id}. The model handles the client's requests in dynamic

978-1-7281-1557-3/19/$31.00 © 2019 IEEE

manner where the requests are not fixed in advance. Our solution contributes in the following directions-

- To identify the guilty entity responsible for leaking the critical data to some third party unknowingly or advertantly.
- To secure the confidential information being transmitted and protect it from unauthorized use.
- To deal with an untrusted entity associated with transfer of data among three party cloud system, the model employ a combination of cryptography, watermarking and message authentication techniques.

We organize the rest of this paper as follows: Section II describes the definitions and preliminaries used in the paper and it introduces our proposed model DLDM. The detailed description of the embedding scheme for the purpose of hiding the secret information is given in Section III. Section IV provides the view of encryption and decryption schemes used in detail. Section V presents the identification of guilty entity, which is followed by performance evaluation in Section VI. We discuss the conclusion and future direction in section VII.

II. Data Leakage Detection Model

A. Definitions and Notation

1) Entities: Let the data Owner O_{id} owns the data $\mathbb{D}$ where $\mathbb{D} = \{D_1, D_2, \ldots, D_n\}$ is the valuable data stored in the Cloud Cl_{id} $(1 \leq id \leq p$ in various forms called as data items D_i $(1 \leq i \leq n)$ such that $\mathbb{D} = D_1 \cup D_2 \cup \ldots \cup D_n = \bigcup_{i=1}^{n} D_i$ is demanded by various clients denoted with $\mathbb{C} = \{C_1, C_2, \ldots, C_m\}$. The Client C_j $(1 \leq j \leq m)$ receives a subset of data items $Z_j \subseteq \mathbb{D}$ by sending request $R_q(D_i)$ to the server S_v.

The model can be represented as three party system denoted with $O_{id}, Cl_{id}, C\ell_{id}$ that refers to Owner ID, Cloud ID and Client ID respectively. O_{id} selects Cl_{id}; $id \in \{1, 2, \ldots, p\}$ to upload the D_i. The master server S_v provides the requested data items $Z_j \subseteq \mathbb{D}$ to various C_j $(1 \leq j \leq m)$, but does not want the data items to be leaked to any third party.

2) Guilty Client: The Client C_j is said to be guilty client G_c, if it shares at least one data item from its own allocated data set Z_j with target t that has been seized in possession of leaked dataset L_ψ.

Let S_v has provided the data item to various $C_j \; \forall \; j = \{1, 2, \ldots, m\}$ as per their request and later on it finds that a set $L_\psi \subseteq \mathbb{D}$ has been leaked then the client C_j is supposed to be G_c if it supplies one or more data item to set L_ψ.

B. Preliminaries

1) Hashing: A hash family is a tuple $(\mathcal{M}_\mathcal{X}, \mathcal{M}_\mathcal{Y}, \mathcal{H}_\mathcal{K}, \mathcal{H}_\mathcal{F})$ where $\mathcal{M}_\mathcal{X}$ is a set of possible messages that are not necessarily finite. $\mathcal{M}_\mathcal{Y}$ is a finite set of possible digests. $\mathcal{H}_\mathcal{K}$ is a finite set of possible keys for hashing. For each key $H_k \in \mathcal{H}_\mathcal{K}$, there exist a hash function $h_f \colon \mathcal{M}_\mathcal{X} \rightarrow \mathcal{M}_\mathcal{Y}$ in $\mathcal{H}_\mathcal{F}$. A pair (m_x, m_y) is called a valid pair under key H_k if $h_f(m_x) = m_y$ where $m_x \in \mathcal{M}_\mathcal{X}$ and $m_y \in \mathcal{M}_\mathcal{Y}$.

2) Watermarking: Let E_τ is the set of entities to be watermarked, $\mathcal{W}_\mathcal{K}$ is the set of keys for watermarking and $\mathsf{W}_\mathcal{M}$ is the set of all feasible watermarks includes the information that the owner wants to embed such that $\mathsf{W}_\mathcal{M} \subseteq \{0,1\}^+$ then the watermarking technique maps $\phi_e \colon E_\tau \times \mathsf{W}_\mathcal{M} \rightarrow E_\tau$ and $\delta_d \colon E_\tau \rightarrow \mathsf{W}_\mathcal{M}$ such that $\delta_d \phi_e(\mathcal{E}_\tau, \mathtt{W_m}) = \mathtt{W_m} \; \forall \; \mathcal{E}_\tau \in E_\tau$ and $\forall \; \mathtt{W_m} \in \mathsf{W}_\mathcal{M}$.

The symmetric watermarking technique consists of three function that can be defined as-

- The key generator function $K_{gen}(WM)$ generates a key $W_k \in \mathcal{W}_\mathcal{K}$ for the given security factor S^f such that $W_k \leftarrow K_{gen}(WM)$.
- The watermark embedding function ϕ_e takes the original entity $\mathcal{E}_\tau \in E_\tau$, the key $W_k \in \mathcal{W}_\mathcal{K}$ and the watermark $\mathtt{W_m} \in \mathsf{W}_\mathcal{M}$ as an input and generates watermarked entity $\mathcal{E}_\tau^*$ such that $\mathcal{E}_\tau^* = \phi_e(\mathcal{E}_\tau, W_k, \mathtt{W_m})$.
- The watermark detection function δ_d extracts the watermark $\mathtt{W_m^*}$ as an output by considering conceivably watermarked entity $E_\tau^* \in E_\tau$, the key $W_k \in \mathcal{W}_\mathcal{K}$ and the original entity $\mathcal{E}_\tau \in E_\tau$ as an input such that $\mathtt{W_m^*} = \delta_d(\mathcal{E}_\tau^*, W_k, \mathcal{E}_\tau)$.

C. Proposed Architecture

The symbols used in this paper is mentioned in Table I. To implement DLDM in the real environment, we make the following assumption regarding the stored $\mathbb{D}$ in the Cl_{id}.

Assumption 1: The data $\mathbb{D} = \{D_1, D_2, \ldots, D_n\}$ given to any client $\mathbb{C} = \{C_1, C_2, \ldots, C_m\}$ is the sensitive data.

This assumption specifies that the data $\mathbb{D}$ stored on Cloud Cl_{id} is sensitive if it gets leaked by some client C_j then the confidential information may fall into unauthorized hands and then it can be misused through unauthorized access.

Assumption 2: $\forall D_i \in L_\psi$, we assume the Cloud Cl_{id} and the data Owner O_{id} to be honest. By honesty in this case, we means that $\mathbb{D}$ is not leaked by these two parties i.e Cl_{id} and O_{id}. It is assumed that the owner O_{id} is primarily concerned about the privacy of the data $\mathbb{D}$ and thus can't leak its own data $\mathbb{D}$. We consider that data access policy will be honestly followed by Cl_{id} and hence will not leak $\mathbb{D}$.

In other words, suppose after giving data items to various clients C_j $(1 \leq j \leq m)$, it has been discovered that $L_\psi \subseteq \mathbb{D}$ has leaked and found at some unauthorized place possessed by target t. Then, we consider that O_{id} and Cl_{id} are not involved in leaking any $D_i \in L_\psi$.

Fig. 1 illustrates the conceptual framework of the proposed architecture. Client C_j sends the Request R_q to the Server S_v for the required Document D_i stored in the Cl_{id}. If D_i is

available in Database then S_v checks Information Management Table (IMT) to find whether C_j has already been registered with it. Otherwise, firstly C_j has to register with S_v then $C\ell_{id}$ is generated by S_v for C_j.

TABLE I: Symbols and their Description

Symbol	Description
n	Number of data items
m	Number of clients
$\mathbb{D}$	Set of data items
$\mathbb{C}$	Set of clients
D_i	i^{th} data item of $\mathbb{D}$
C_j	j^{th} client from $\mathbb{C}$
O_{id}	Data Owner
Cl_{id}	Set of cloud servers $Cl_1, Cl_2, \ldots, Cl_p$
$C\ell_{id}$	Client ID
S_v	Master server
Z_j	Set of data items allocated to client C_j
$R_q(D_i)$	Request for data item D_i
L_ψ	Set of data items that has leaked
G_c	Guilty client who leak the data items
K	Secret key for AES-128 Encryption
H_k	Hashing key for SHA-3
W_k	Watermarking key
S_k	Secret key for AES-256
P_k	Public key of RSA
PV_k	Private key of RSA
H	Hash value
$\partial^*(\varkappa)$	Encrypted value $\varkappa$
K_{gen}	Key generator
WMD	Watermarked Document
L_o	Organization's logo

In the proposed DLDM, S_v inserts the logo of the organization L_o to which C_j belongs, in D_i for embedding the information in it. S_v embeds $C\ell_{id}$ after encrypting it, in L_o by applying Digital Watermarking Technique (DWT) over it results into Watermarked document (WMD). This WMD is encrypted by applying Cryptography Technique (CT) over it and then Encrypted WMD (∂^*(WMD)) is passed to C_j. An entry is made to IMT by storing $C\ell_{id}$, Document-ID and hash value H generated for $C\ell_{id}$ of C_j which specifies the document having Document-ID as D_i has been allocated to C_j with client-ID as $C\ell_{idj}$. If C_j leaks D_i somewhere and later on it is found that D_i has been leaked, then detection technique is applied to find G_c who leaked the data and then G_c is identified by matching with $C\ell_{id}$ in IMT.

III. Embedding

In this phase, Digital Watermarking Technique (DWT) is used to embed and hide the information $I_m = \{C\ell_{id}, H\}$ in L_o. Firstly, $C\ell_{id}$ will be encrypted using CT. The aim of DLDM is to use a light weight CT instead of using heavy weight CT like RSA to reduce time complexity and cost in calculating the encrypted message. Therefore, we used Advanced Encryption Standard (AES-128) CT to encrypt $C\ell_{id}$ in combination with hashing technique (HT) for authentication purpose. The input of AES-128 are secret message $C\ell_{id}$ and

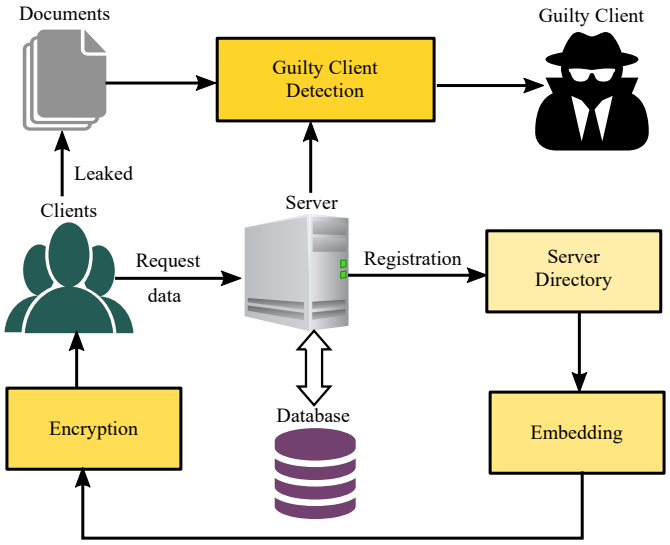

Fig. 1: Proposed architecture DLDM.

the secret key K of 128 bit and it produces cipher text $\partial^*(C\ell_{id})$ as output. In parallel to this Hash value H of $C\ell_{id}$ is calculated using Secure Hash Algorithm (SHA). We used SHA-3 512 for authenticity of the document D_i. If C_j make changes to $C\ell_{id}$ then its SHA-3 code will be changed and we can identify that something wrong has been happened with $C\ell_{id}$, which provides more security to the document D_i. Input of SHA-3 512 HT are the key $H_k \in \mathcal{H}_\mathcal{K}$ and $C\ell_{id}$ instead of $\partial^*(C\ell_{id})$. It will distract C_j and increase the security level of DLDM and produces the Hash value H of 512 bit as output.

Generated output $I_{m'} = \{\partial^*(C\ell_{id}), H\}$ from CT and HT are embedded into L_o in D_i using DWT. Input of DWT are D_i, H, $\partial^*(C\ell_{id})$ and $W_k \in \mathcal{W}_\mathcal{K}$ which produces WMD as output. One of the important thing of Watermark Embedding Process (WEP) is that the Hash H is embedded in L_o in reverse order to distract the Client C_j. It provides extra level of security to D_i. Other important thing about WEP is that firstly, we applied CT on $C\ell_{id}$ and then embedded in D_i. If we will directly embed $C\ell_{id}$ without encrypting it and its corresponding Hash value H in D_i, then C_j can change $C\ell_{id}$ and correspondingly H. Thus it provides significant security to D_i and it becomes difficult for C_j to predict the embedded $I_{m'}$ in D_i.

IV. Encryption/Decryption

To provide more security to the document, generated WMD is encrypted and then transmitted to the client C_j. The DLDM adopted a combinational CT to encrypt WMD. Although RSA is highly secure CT, but its encryption time, decryption time memory usage is high and it is expensive CT also. AES-256 is secure, faster, less expensive CT and utilizes less memory as compare to RSA. Therefore, instead of using RSA directly to encrypt/decrypt whole WMD, we applied a combinational approach that utilizes both AES-256 and RSA cryptography techniques. In the proposed DLDM, we encrypt/decrypt the whole WMD using AES-256 CT and then Secret Key S_k

used by AES-256 will be encrypted by RSA public key P_k and decrypted by RSA private key PV_k. The combinational approach provides the significant security to WMD as well as it is effective in term of encryption time, decryption time, memory utilization and cost. Thus, this approach has the benefit of fast file encryption/decryption and still requiring a non-shared private key to get access to the key needed to decrypt the files.

Let $\mathcal{D}$ is the set of documents to be encrypted, $\mathcal{S}_\mathcal{K}$ is the set of secret keys for symmetric cryptography, $\mathcal{P}_\mathcal{K}$ is the set of public keys and $\mathcal{PV}_\mathcal{K}$ is the set of private keys for asymmetric cryptography. Secret key $S_k \in \mathcal{S}_\mathcal{K}$ is generated by AES-256 key generator. RSA key generator generates the public and private key $P_k \in \mathcal{P}_\mathcal{K}$, $PV_k \in \mathcal{PV}_\mathcal{K}$ respectively. AES-256 encrypts WMD using S_k while S_k is encrypted by RSA using P_k and then encrypted WMD and secret key are transferred to client C_j where $j \in \{1, 2, \ldots, m\}$. After receiving the encrypted document, C_j decrypts it. For it, firstly encrypted S_k is decrypted by RSA using PV_k and after that AES-256 decrypts encrypted WMD using S_k. Finally, C_j obtains WMD.

V. Guilty Client Detection

After obtaining the document, let C_j has leaked WMD by some means. Later on, leak data set $L_\psi \subseteq \mathbb{D}$ is discovered at some unauthorized place. In Watermark Extraction Process (WEXP), embedded $I_{m'}$ is extracted from the discovered WMD by using DWT. Thus, we obtain H and $\partial^*(C\ell_{id})$ by extracting $I_{m'}$. Secret message $C\ell_{id}$ is obtained by decrypting $\partial^*(C\ell_{id})$ through AES-128 using key K and Hash $H^{'}$ is calculated by SHA-3 512 HT. S_v verifies the obtained $C\ell_{id}$ by matching it in the stored IMT and also verify the correctness of $C\ell_{id}$ by comparing H and $H^{'}$ which is calculated using $C\ell_{id}$. If the calculated $H^{'}$ is same as the embedded and stored H, this mean there is no change in $C\ell_{id}$ and it is proved as a correct message. In that case, guilty client G_c is the extracted $C\ell_{id}$. Algorithm 1 represents the pseudo code of the proposed approach.

Algorithm 1 Guilty Client G_c detection

Input: Request of data items by clients
Output: G_c

$Z_1 \leftarrow \phi, Z_2 \leftarrow \phi, \ldots, Z_m \leftarrow \phi$
while $!R_q$ **do**
    WMD $\leftarrow$ DWT('D_i')
    ∂^*(WMD) $\leftarrow$ WMD
    $Z_j \leftarrow Z_j \cup \{D_i\}$
end while
$I_{m'} \leftarrow$ WEXP('WMD', 'W_k', 'D_i')
//Extraction algorithm is applied on WMD
$C\ell_{id} \leftarrow$ AES-128('K', '$\partial^*(C\ell_{id})$')
$H^{'} \leftarrow$ SHA-3 512('H_k', '$C\ell_{id}$')
if $H = H^{'}$ **then**
    $G_c = C\ell_{id}$
end if

VI. Performance Evaluation

A. Experimental Setup & Benchmark

We performed the experiment on the machines running 32 bits kernel version 6.3 with an intel ®core ™i5-2600 CPU @3.4 GHZ having 8 GB RAM. We used the crypto ++ library [22] to implement the encryption and hashing scheme. We implemented the watermarking technique using matlab and conducted the experiment by considering different parameters with simulate data leakage problem to evaluate the performance. We used authenticated open data accessible on [23] to collect the dataset. The designed benchmark for DLDM is dealing with medical datasets as these datasets are universally alleged as most sensitive data in nature.

B. Results and Analysis

We considered mainly seven operations for the document D_i, requested by C_j that are: 1) Cloud DB search 2) $C\ell_{id}$ generation 3) $C\ell_{id}$ encryption 4) hashing on $C\ell_{id}$ 5) watermarking 6) document encryption 7) RSA encryption on AES-256 keys and then computed the execution time of each phase. In the first experiment, we considered a D_i of size 20 MB and find the computation time of each operation for it as shown in Fig. 2. It can be seen that the execution time for watermarking is maximum among all the operations. The next experiment is performed by sharing the single document with various number of users (N_U). We find the computation time of whole process by varying the size of documents. The results are represented in Table II. We can observe that the execution time increases as the number of users increase and it also raises with increment in the size of document.

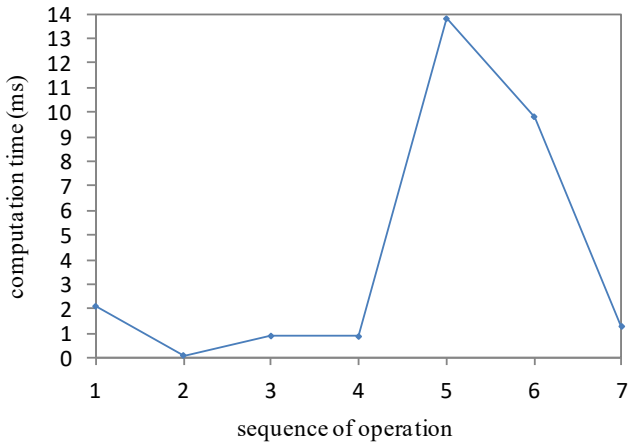

Fig. 2: Time taken by each operation for a document

In the next experiment, we used the varieties of documents of fixed size (20 MB) and compared the overall execution time when these documents are provided to the distinct users and single user as shown in Fig. 3. We observe, as the number of documents goes on increasing, the computation time difference for aforementioned cases also increases. The computation time in the complete process is less, when the documents are provided to the single user as compared to the other case. For

TABLE II: Computation Time w.r.t. Number of Users for different Document Sizes

N_U	Computation time (ms) Size of document				
	20MB	40MB	60MB	80MB	100MB
10	263.57	308.43	338.59	388.14	427.14
15	398.83	458.30	496.14	577.22	638.76
20	520.77	613.83	673.36	773.41	843.63
25	663.1	753.98	831.75	950.94	1068.47
30	782.22	908.85	1005.41	1158.43	1270.06
35	923.84	1063.75	1157.15	1339.99	1493.94
40	1046.45	1200.05	1333.6	1547.86	1688.46
45	1174.04	1370.95	1506.05	1703.28	1907.48
50	1316.31	1495.04	1646.47	1905.31	2096.47

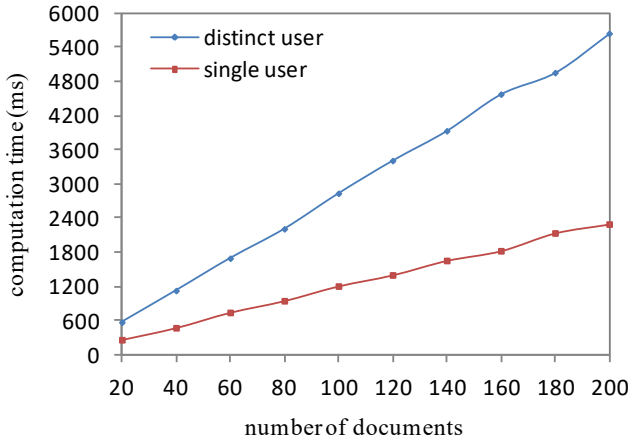

Fig. 3: Computation time comparison when varieties of document is provides to distinct user and single user

instance, the computation times are 2285.31 ms and 5628.84 ms, when 200 documents are provided to the single user and distinct users respectively. The reason for this can be explained as the execution time for most of the operation such as $C\ell_{id}$ generation, $C\ell_{id}$ encryption, hashing on $C\ell_{id}$, watermarking becomes constant in providing the varieties of documents to the single user.

VII. Conclusion and Future Directions

To unravel DLP, we presented a model DLDM that detects the malicious user who has leaked the confidential information at some unauthorized place through incidental exposure or intentional sabotage. To detect the guilty entity and to provide the security to the system, our model considers a vigorous combination of cryptography, hashing and watermarking techniques. The proposed model considers the documents of various types and handles the requests of the users in dynamic manner as in case of real world scenario. The results show that the proposed technique is computationally cost effective with regards to time. Our future work motivates the further investigation on the scheme for various possible scenarios.

References

[1] I. Gupta, N. Singh and A. K. Singh, "Layer-based Privacy and Security Architecture for Cloud Data Sharing", J. Commun. Software and Syst. (JCOMSS), vol. 15 no. 2, pp. 1-13, 2019.

[2] B. HAUER, "Data and Information Leakage Prevention Within the Scope of Information Security", IEEE Access: The Journal for rapid open access publishing, vol. 3, pp. 2554-2565, Dec. 2015.

[3] M. Backes, N. Grimm and A. Kate, "Data Lineage in Malicious Environments", IEEE Trans. Dependable Secure Comput., vol. 13, no. 2, pp. 178-191, Mar./Apr. 2016.

[4] I. Gupta and A. K. Singh, "Dynamic Threshold based Information Leaker Identification Scheme", Information Processing Letters, vol. 147, pp. 69-73, July 2019.

[5] I. Gupta and A. K. Singh, "A Probabilistic Approach for Guilty Agent Detection using Bigraph after Distribution of Sample Data", Procedia Comput. Sci., vol. 125, pp. 662-668, 2018.

[6] A. Harel, A. Shabtai, L. Rokach and Y. Elovici, "M-Score: A Misuseability Weight Measure", IEEE Trans. Dependable Secure Comput., vol. 9, no. 3, pp. 414-428, May/June 2012.

[7] Privacy Rights Clearinghouse (PRC) [Online]. Available: https://www.privacyrights.org/data-breaches, 2019.

[8] K. Kaur, I. Gupta and A. K. Singh, "Data Leakage Prevention: E-mail Protection via Gateway", J. Phys.: Conf. Ser., vol. 933, no. 1, IOP Publishing, 2018.

[9] I. Gupta and A. K. Singh, "A Probability based Model for Data Leakage Detection using Bigraph", 7th Int. Conf. Commun. and Network Security (ICCNS-2017), ACM, Tokyo, Japan, 2017.

[10] I. Gupta and A. K. Singh, "Malicious Agent Detection in Cloud Environment", Int. Conf. Knowledge Discovery in Sci. and Technology (ICKDST), Pune, India, 2019.

[11] Chronology of Data breaches [Online]. Available: https://www.privacyrights.org/data-breaches?title=\&org_type\%5B\%5D=258\&taxonomy_vocabulary_11_tid\%5B\%5D=2436.

[12] U. Arora, S. Verma, I. Gupta and A. K. Singh, "Implementing privacy using modified tree and map technique", 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA), pp. 1-5, IEEE, 2017.

[13] K. Kaur, I. Gupta and A. K. Singh, "A Comparative Evaluation of Data Leakage/Loss prevention Systems (DLPS)", in Proc. 4th Int. Conf. Comput. Sci. & Inform. Technology (CS & IT-CSCP), Dubai, UAE, pp. 87-95, 2017.

[14] K. N. Kaur, Divya, I. Gupta and A. K. Singh, "Digital Image Watermarking Using (2, 2) Visual Cryptography with DWT-SVD Based Watermarking", Computational Intelligence in Data Mining: Advances in Intelligent Systems and Computing, Springer, Singapore, vol 711, pp. 77-86, 2019.

[15] G. Batra, H. Singh, I. Gupta and A. K. Singh, "Best Fit Sharing and Power Aware (BFSPA) Algorithm for VM placement in cloud environment", 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA), pp. 1-4, IEEE, 2017.

[16] X. Shu, D. Yao and E. Bertino, "Privacy-Preserving Detection of Sensitive Data Exposure", IEEE Trans. Inf. Forensics Security, vol. 10, no. 5, pp. 1092-1103, May 2015.

[17] "2018 annual Cost of a Data Breach Study: Global Overview", conducted by Ponemon Institute and sponsored by IBM Security [Online]. Available: https://www.ibm.com/downloads/cas/861MNWN2.

[18] Gemalto's Data Breach Level Index [Online]. Available: https://betanews.com/2018/10/09/4-5-billion-records-compromised-2018/.

[19] K. Kaur, I. Gupta and A. K. Singh, "A Comparative Study of the Approach Provided for Preventing the Data Leakage", Int. J. Network Security & Applicat. (IJNSA), vol. 9, no. 5, pp. 21-33, Sept. 2017.

[20] X. Shu, J. Zhang, D. Yao and W.-C. Feng, "Fast Detection of Transformed Data Leaks", IEEE Trans. Inf. Forensics Security, vol. 11, no. 3, pp. 528-542, Mar. 2016.

[21] P. Papadimitriou and H. Garcia-Molina, "Data Leakage Detection", IEEE Trans. Knowl. Data ENG., vol. 23, no. 1, pp. 51-63, Jan. 2011.

[22] W. Dai, Crypto++ Library [Online]. Available: https://www.cryptopp.com

[23] Open Data. National Informatics Center (NIC) [Online]. Available: https://data.gov.in/catalogs.