International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

An Encrypted And Searchable Audit Log

Komal D. Kate
Department of computer, Zeal Education Society's
Dyanganga College of Engineering, University of Pune
Pune, India

Prof. P. M. Mane
Department of computer, Zeal Education Society's
Dyanganga College of Engineering, University of Pune
Pune, India

***Abstract*—Audit log is important part of any software system. These audit logs may contain sensitive information, poses a threat to privacy and information security, so should be prohibited against any illegal reading and alteration or deletion. The best way to do this is encryption. The key challenges in an encrypted audit log are speed of log and search process, correctness of query and relevance of search results and log size. We describe an approach for constructing searchable encrypted audit logs which can be combined with any number of existing approaches for creating tamper-resistant logs. Even though previous searchable encryption schemes allow users to search encrypted data by keywords securely, these techniques only support exact keyword search and will fail if there are some spelling errors or if some morphological variants of words are used. In this paper, we provide the solution for fuzzy keyword search over encrypted data. K-grams are used to produce fuzzy results. Our technique for keyword search on encrypted data has wide application beyond searchable audit logs.**

Keywords—Encryption; Fuzzy keyword; K-gram

I. INTRODUCTION

Today most of Server software includes some logging mechanisms. The system log file contains actions that are logged by the operating system components. These actions are often determined by the operating system itself. System log files may contain information about device changes, device drivers, system changes, events, operations and more. System logs provide view inside current and past state of any type of complex system. Such a log should be secure to defend against malicious tampering, allow the current state of the system to be audited even when that system has been under active attack by malicious insiders or outsiders. Correctly designed secure audit logging mechanisms can detect unauthorized past activity, even when the person performing that action goes to great lengths to cover their tracks [1]. The presence of such logs can be used to enforce user to behaviour correctly, by holding users actions as recorded in the audit log. Such logs can be used in a wide variety of systems, a control system that logs the commands user issues, a database system that logs the queries a user makes. When an organization wishes to inspect past activity it will search the audit log for relevant information. Suppose, if a certain user was behaving improperly the organization might search for all actions performed by that particular user. If the organization wishes to see all actions perform by certain user, it might search for all log entries. But this approach cannot be useful, because it required searching all data, which is time consumable and not secure. For an audit log to be useful in practice, it is critical that it be efficiently searchable for keywords of interest. These keywords characterise the record. At the same time, the contents of an audit log can be considered to be sensitive information. If the log contains information about not only what query was made, but also what results were returned, then access to the audit log would imply effective access to the database, avoiding database access controls [1]. In general, audit log may contain sensitive information; this means that the contents of the audit log must be encrypted. However, this makes it extremely difficult to search. Using traditional techniques, searching the log would require decrypting every record. This approach has a disadvantage that, it requires decrypting the entire log data, regardless of for what information we are looking for; this opens opportunities for access to log records other than the ones relevant to the current investigation. In many applications, one would like to delegate the ability to decrypt audit logs to an entity or system with high levels of trust and assurance. It would be preferable to be able to selectively delegate the ability to search the log to parties with the means to process the data. The key challenge to building a successful, secure audit logging system is to simultaneously protect the integrity of the audit log, control access to contents, and maintain its usefulness by making it searchable.[1] This paper present, a design that allows a trusted party(*audit escrow agent),* to create *keyword search capabilities*, which allow *investigators* in possession of such capabilities to search for and decrypt entries matching a given keyword for an encrypted audit log. The escrow agent can allocate a capability to an investigator if he/she deems it appropriate. This paper present, a public key based cryptographic scheme that allows keyword searching on encrypted data by adapting Boneh and Franklin's [3] Identity-Based Encryption (IBE) scheme. In an IBE scheme, public keys can be arbitrary strings based on identity of particular user – *e.g.*, "bob@abc.com". Private keys are derived from public keys through use of a system-wide *master secret*. In our design, search keywords are used as IBE public keys, and the *master secret* is held by an authority trusted to issue keyword search capabilities for a given audit log, i.e. the *audit escrow agent* described above. In our design, the server generating audit log entries encrypts entries with the public keys corresponding to the keywords that are derived from those entries. The escrow agent, who holds the IBE master secret, can create a search capability for a given keyword as the private key , which is used by *investigator* to decrypt the entry for matching keyword only. To make search more useful K-gram based fuzzy keyword search is proposed.

II. CHARACTORISTICS OF AUDIT LOG

Three important properties of a secure audit log: those designed to prevent and detect tampering, and those designed to control data and search access.

IJERT

A. Tamper Resistance

A secure audit log must be *tamper resistant* – it make sure that only originator of the log can create valid entries and that once entries have been created, they cannot be altered. One cannot prevent an attacker who has compromised the system from altering what that system will put in future log entries. The goal of a secure audit log in such case is to make sure that he cannot alter existing log entries, and that any attempts to delete such existing entries will be detected. For some applications it may be enough to have the logging host "checkpoint" its state periodically – to copy its log data, or some function (*e.g.*, a signature) of its log data to another host, and simply be able to assure that no entries up till the most recent checkpoint have been deleted or altered[1].

B. Verifiability

A secure audit log must also be *verifiable* – it makes sure that all entries in the log are present and have not been altered. Audit logs can be *publicly verifiable* – verifiable by anyone who has authenticated public information. Or, they may require a *trusted verifier* – they can only be verified by a designated party holding one or more secrets.

C. Data acess control and searchability

Data in audit log may contain sensitive information, it must be encrypted. On other hand, one would like to be able to allow reasonable search access to all audit log entries (*e.g.*, all entries matching the keyword "John"). Delegation of capabilities is important so that an investigator can search and view entries of a narrow scope. For example, if Alice Smith wanted to investigate all entries related to her the audit escrow agent might give her the capability to search for all entries matching the keyword. Each entry must contain hash of previous entry to allow some recovery if any entry is missing and also provide verifiability. In case of delegation of authority, it must be impossible for adversary to learn the content of entries in the audit log that he should not have access.

Audit Log Components and Notation

The mechanisms can be applied to search the logged entry for any application and keywords used for search are modified to the application or system to log. Our audit log A consists of a series of individual *audit records*, R_0, R_1, . . ., R_n. Each record R_i contains:

1) $E_{K_i}(s_i)$, the encryption of the data to be logged under a key K_i. The string si consists of the database query to be logged, along with metadata such as the identity of the user who issued the query. Optionally, it could also contain the query results. In our system, the key Ki is chosen randomly for each log entry.

2) H(Ri−1), the hash of the previous record, to form a hash chain.

3) c_{w1} , c_{w2} ,c_{w3} , . . ., information about the keywords w_1, w_2, w_3, . . . that can be used for searching.

4) Verification information Vi. This is simply the hash to date of the current chain of audit records (i.e., H(Ri)). Vi must authenticate all of the other data in the audit record, including the keyword information c_{wn}.

To construct a searchable secure audit record *Ri*, the server first extracts keywords that characterize the record. These are the keywords that can be used to search for that record in the future. Then, it encrypts the audit log entry using the key *Ki*, producing the keyword search information c_{wn} in the process. Finally, the server constructs the verification data *Vi*. We periodically "checkpoint" the audit log by publishing the most recent verification value, *Vi*, to other trusted server, and it produces a publicly verifiable audit log.

Encryption schemes

In this section we present two cryptographic schemes for encryption and decryption, one is symmetric key cryptography and another one is asymmetric key cryptography. We first present symmetric key cryptography, in which same key can be used for encryption as well as decryption. Even though this scheme is secure against passive adversary, we find that the scheme is insecure against an adversary that is able to compromise an audit log server. The second scheme is asymmetric key cryptography, in which public key is used for encryption and private key is used for decryption, which address this issue.

D. Symmetric Key Cryptography

We describe a symmetric key based scheme for encrypting searchable audit log entries. Our method is derived from previous work on searching on encrypted data [7, 10].

Suppose there are *A* audit log servers. The audit escrow agent generates independent and uniformly random secrets S_1, . . ., S_A and gives S_i to the *i*th server.

Encryption: Suppose the server encrypt the log entry, s, along with keywords w_1, w_2, . . ., w_n. Let **flag** be a constant bit string of length *l*.

The server executes the following steps:

1) The server first chooses a arbitory symmetric encryption key, *K*, to be used only for entry s.

2) The server computes the encryption $E_K(s)$.

3) The server chooses a random bit string *f* of some fixed length. The random string *f* is uniformly independently drawn for each entry.

4) For *i* from 1 to *n* the server computes $a_i := H_S(w_i)$, $b_i := H_{a_i}(f)$, $c_i := b_i \oplus (\mathbf{flag}|K)$. In other words, the PRF is first keyed with secret *S* with given input w_i. The result a_i is then used to key the PRF with input *r* to give output b_i. The result b_i is then XORed with the concatenation of **flag** and the symmetric key *K* to give the output c_i.

5) The server writes $(E_K(s), f, c_1, c_2, \ldots, c_n)$ as the encrypted entry to the audit log.

Decryption: Suppose at some point of time investigator want to search an audit log, then he must request search capability to audit screw agent. If audit screw agent deems it appropriate, then he transfer search capability to investigator. The audit escrow agent constructs the search capability as

$$dw := (H_{S1}(w), \ldots, H_{SA}(w)).$$

We denote $d^j_w := H_{Sj}(w)$ as the search capability component corresponding to the jth server. Once given the capability, the investigator visits each audit log server. At the jth server, the investigator executes the following:

1) The investigator computes $p := H_{d^j_w}(f)$, where is the random string stored with the query.
2) For each ci in the entry, the investigator computes pci. If the first l bits of the result match **flag**, then the party extracts K t; otherwise, the computation is disregarded. If none of the results begin with **flag**, then the query does not with match a keyword, and the investigator moves to the next query.
3) If one of the results did match, the investigator uses the extracted K to decrypt $EK(s)$ to obtain s, the original audit log entry.

E. Asymmtric Key Cryptography

The limitations of the symmetric key based scheme suggest that an asymmetric key based scheme is necessary. We present an asymmetric key based scheme which is based on Identity-Based Encryption for creating encrypted and searchable log entries. Our scheme is based on the Identity-Based Encryption scheme of Boneh and Franklin [3].

Identity-Based Encryption: Identity based encryption is a type of public key encryption in which the public key of user is unique information about identity of user. If Alice want sent message to Bob, she uses a string that uniquely identifies a Bob - "bob@abc.com". That string is used as encryption key while sending encrypted message to Bob. Then Bob receive encrypted message. Escrow agent uses master secret key to generate private key. Bob authenticate to third party to obtain private key which can be derive from public key to decrypt the message. Some mathematical details are as follows:

IBE SETUP. To set up the system, one first selects large primes p and q, two groups G_1 and G_2 of order q, and an arbitrary generator $P_0 \in G_1$. One also picks an admissible bilinear map e: $G_1 \times G_1 \rightarrow G_2$ and two cryptographic hash functions H_1: $\{0, 1\} \in G_1$ and H_2: $G_2 \in \{0,1\}^n$. The master secret is a random value $s \in Z_q$, known only to the trusted escrow agent. The system parameters are

$M = (p, q, G_1, G_2, e, P_0, P_1)$, where $P_1 = sP_0$,

and are known by all parties.

IBE KEY GENERATION. To issue the private key corresponding to the public key w, the escrow agent uses the master secret s to compute dw: $= sH_1(w) \in G_1$.

IBE ENCRYPTION. To encrypt the plaintext $s \in \{0,1\}^n$ using a string w as the public key, one

1) Computes $Qw = H_1(w) \in G_1$,
2) Computes $gw = e(Qw, P_1)$,
3) Computes $c = (sP_0, m \oplus H_2(g^r w))$

Where a random $s \in Z_q$.

IBE DECRYPTION. To decrypt a ciphertext $c = (X,Y)$ using dw as the private key, one computes $s = X \oplus H_2(e(dw, Y))$.

Since e is a bilinear map, it follows that decryption operation is the inverse of the encryption operation. We refer the reader to [4, 7] for the details of this scheme.

Encryption: To encrypt the log server performs the following steps:

1) The server chooses arbitrary symmetric encryption key, K, which is used for that particular this entry.
2) The server encrypts the log entry using K, to get $EK(s)$.
3) For each keyword w_i, the server computes the Identity-Based Encryption ci of the string (**flag**|K) using w_i as the public key and M as the public parameters.
4) The server writes $E_K(s), c_1, c_2, \ldots, c_n$ as the entry to the audit log.

Decryption: suppose investigator want to search audit log and request a search capability, then audit escrow agent create capability dw as the Identity-Based Encryption private key corresponding to public key. Investigator execute following steps to decrypt each log entry:

1) For each c_i the investigator attempts to IBE-decrypt ci using the private key dw. If the prefix of the result matches **flag** then the investigator extracts K as the remainder of the result. If none of the results begin with **flag** then the log entry does not match and the investigator moves to the next log entry.
2) If one of the results did match, the capability holder may compute K to decrypt $E_K(s)$ to obtain original log entry.

F. Optimization For Asymmetric Schemes

A drawback of asymmetric scheme is performance overhead of using Identity Based Encryption, although to speed up our scheme we use optimization. We discuss three optimizations in this section.

PARING REUSE: The computation of gw only needs to be performed once per keyword. Subsequent Identity-Based Encryptions using w as the public key can reuse gw if it has already been computed for some other log entry. Encryption then only need to select a random r and following steps (3) and (4) of encryption (see explanation of Identity-Based Encryption above). This speed up encryption: over a set of log entries in which a keyword repeated k times, only one pairing operation is required.

INDEXING. Creating an index of keywords at periodic intervals in the log, instead of storing IBE encryptions with each log entry, saving is possible. Server execute following steps:

1) The server chooses random symmetric encryption keys, $K_1, \ldots, K_A$, for one-time use.
2) The server encrypts each log entry si using Ki, to get $E_{Ki}(si)$.
3) For each distinct keyword w_j, the server finds the indices $\{i_{j,1}, \ldots, i_{j,l(j)}\}$ for which wj is a keyword

IJERT

where $l(j)$ is the number of entries for which w_j is a keyword. (That is, w_j is a keyword in q_i exactly when $i \in \{i_{j,1}, \ldots, i_{j,l(j)}\}$.)

4) The server computes the Identity-Based Encryption c_j of the string $(\mathbf{flag}|i_{j,1}|Ki_{j,1}|\cdots|i_{j,l(j)}|Ki_{j,l(j)})$ using w as the public key and Q as the public parameters.

5) The server writes $E_{K1}(s_1), \ldots, E_{KA}(s_A), c_1, \ldots, c_u$ as the block and index to the audit log.

RANDOMNESS REUSE. An optimization for the decryption process. We perform an independent IBE encryption to creating the c_i corresponding to the keywords w_i for given log entry. However, it is possible to reuse an intermediate result of the IBE encryption process: we may save the value f chosen in step (3) of the encryption that produces c_1 to use in calculation of $c_2, \ldots, c_n$. As long as the w_i are distinct keywords, this reuse of the randomness produces results indistinguishable from the original method. This speeds up decryption, as only one pairing is needed for each distinct r chosen.

III. K-GRAM BASED FUZZY KEYWORD SEARCH

The previous schemes were presented for searching correct keyword. A type of search that will find matches even when users misspell words or enter in only partial words for the search, then it is called as fuzzy search. A concept of k-grams index is used, which is used to perform wildcard queries. K-grams is a sequence of k characters. For example, "sch", "cho", "hoo" and "ool" are all the 3-grams of the word "school". We use the character $ to denote the beginning or the end of a word. Thus, the set of 3-grams generated is: "$sc", "sch", "cho", "hoo", "ool" and "ol$". In a k-grams index, our dictionary contains all the k-grams of every word in the collection. In k-gram index, dictionary contains all k-grams of every word in collection then we create posting list of all the word in the collection.

For ex. Obj->Object->Objective.

A. Fuzzy Keyword Generation

Suppose user wants to search for certain record entry then he may enter certain keyword K to search the entry. The k-grams for keyword is created, called G(K). We construct k-gram index for the keywords that characterizes particular record. System searches k-gram index for every gram gi ϵ $G_k(K)$. If W is one of the words in our k-gram index that contain the gram gi, we use the Jaccard coefficient to measure the similarity of the word K and the word W.

$$(|A \cap B|/|A \cup B|)$$

If W is equal to K, W will have the highest Jaccard coefficient value compare to the other words in the index. If the Jaccard coefficient of W(λw) is greater than our threshold value, λ_{min}, i.e.

$$\lambda w = \frac{|Aw \cap Bk|}{|Aw \cup Bk|} > \lambda min$$

We add W to our fuzzy keyword set Fk={F_{k1}, F_{k2}, ..., F_{kn}}.

B. Weighted Ranking Algorithm

Once users entered their search query, system will generate the fuzzy keyword for the keyword entered and calculate the weight of word. U_k be the pre-defined weight of K. The weight of the fuzzy keyword $F_{kj} \epsilon F_k$ is

$$F_{kj} = \lambda_{kj} * U_k$$

The weight of word F_{kj} in fuzzy keyword set F_k is multiplication of the predefined weight of the word U with jaccord coeffient of word Fk. Once we finish with calculating the weight for each keyword, then we sort them in ascending order according to weight and return to user.

IV. CONCLUSION

A natural approach is adopted to protect the audit log, which is done in such a way that log still searchable. Asymmetric IDE-based key scheme for searching on encrypted data is used. Privileged *audit escrow agents* can create search capabilities that allow their possessor to search the audit log for records matching certain keywords without leaning extra information about other record. Work is relies on efficiently searching audit log with legitimate access to the information. Adapted novel way for multiple fuzzy keyword ranked search over encrypted data by utilizing some advanced techniques (such as k-gram) are used to generate a search-efficient index.

REFERENCES

[1] B. Waters, D. Balfanz, G. Durfee, and D. Smetters, "Building an encrypted and searchable audit log", Network and Distributed System Security Symposium (NDSS 2009), pp. 205V214, 2009.

[2] W. Zhou, L. Liu, H. Jing, C. Zhang, S. Yao and S. Wang, "K-Gram Based Fuzzy Keyword Search over Encrypted Cloud Computing," Journal of Software Engineering and Applications, Vol. 6 No. 1, 2013, pp. 29-32. doi: 10.4236/jsea.2013.61004.

[3] D. Boneh and M. Franklin.,"Identity-based encryption from the Weil pairing", In *Proc. CRYPTO 01, pages 213–229.* Springer-Verlag, 2001. LNCS 2139.

[4] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano.,"Searchable public key encryption",See http://eprint.iacr.org/2003/195/.

[5] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," Proceedings of IEEE INFOCOM'10 Mini- Conference, San Diego, March 2010.

[6] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval,"Key-privacy in public-key encryption", *Lecture Notes in Computer Science, 2248, 2001.*

[7] E.-J. Goh. "Building secure indexes for searching efficiently on encrypted compressed data", See http://eprint.iacr.org/2003/216/.

[8] B. Schneier and J. Kelsey,"Cryptographic support for secure logs on untrusted machines", In *Proceedings of the 7th USENIX Security Symposium, pages 53–62. USENIX* Press, 1998.

[9] B. Schneier and J. Kelsey,"Minimizing bandwidth for remote access to cryptographically protected audit logs",In *Web Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection. USENIX Press,* 1999.

[10] D. X. Song, D. Wagner, and A. Perrig,"Practical techniques for searches on encrypted data",In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000

[11] B. Schneier and J. Kelsey,"Secure audit logs to support computer forensics,"ACM Transactions on Information and System Security (TISSEC), 2(2):159–176, 1999.

IJERT