On the Confidential Auditing of Distributed Computing Systems

Yiping Shen, Jyh-Charn Liu, Wei Zhao
{ypshen, liu, zhao}@cs.tamu.edu
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

August 4, 2003

Abstract

In this paper, we propose a confidential logging and auditing service for distributed information systems. We propose a cluster-based TTP (trusted third party) architecture for event log auditing services, so that no single TTP node can have the full knowledge of the logs, and thus no single node can misuse log information without being detected. On the basis of a relaxed form of secure distributed computing paradigms, one can implement confidential auditing service so that the auditor can retrieve certain aggregated system information e.g., number of transactions, total of volumes, event traces, etc., without having to access the full log data. Similar to the peer relationship of routers to provide global network routing services, the mutually supported, mutually monitored cluster TTP architecture allows independent systems to collaborate in network-wide auditing without compromising their private information.

Key words - Auditing, commutative cryptography, secure distributed computing, cluster, anonymity, authenticity
1. INTRODUCTION

Logging and auditing are effective tools for security management of distributed information systems. By keeping track of messages between interacting parties, one would have much better abilities to implement critical system functions in a fashion similar to the global collaboration of Internet routing and phone switching. Despite its obvious benefits for security management, distributed logging and auditing (DLA) has not been widely adopted because of privacy concerns. Unlike network routing information, which can easily be separated from business data, unveiling of system logs could lead to business loss or litigation. To make DLA useful, e.g., auditing of transactions multiple independent sources, distributed event correlation for intrusion detection, etc., the system must guarantee the privacy of system owners.

DLA could be performed by computing nodes directly, albeit it is difficult to guarantee the privacy and integrity of the (auditing) operations. Event logs are usually kept by the information systems within close control of the administrators. It is very unlikely that any organization would allow some third parties to examine its event logs without close scrutiny of its own administrators. This process makes auditing a very slow, labor intensive, and inaccurate process. Our aim is to design a TTP cluster architecture to provide logging and auditing services, so that one can examine global events without having to unveil details of individual systems. We note that, an unsupervised TTP can easily manipulate the system and commit illegal acts for its own benefit. By requiring nodes in the TTP-cluster to interact in certain ways to perform any useful functions, i.e., a secure distributed computation or multiparty private computation system, we prevent a single node from cheating.

Many auditing services have been developed for policy compliance, operational procedures, and service quality, etc. [1]. The auditor uses random samples to prevent or detect fraudulence or errors. One major application of auditing technology is for real-time intrusion detections [2]-[5] by analyzing the audit trail of user activities. Strous [6] made suggestions on the requirement of auditing of electronic commerce systems. In [7], the notion of secret counting was proposed to audit the system statistics, such as the number of specific services that have been used, the number of records located in each search, etc., without having to unveil the privacy of library patrons. In [8], a “confidential auditing control” approach is discussed, where timestamping, multiparty private computations, etc. were mentioned without much discussion of the details of the auditing functions.

DLA requires that participants to engage in collaborative computing, without unveiling total local information to others. To meet this requirement, the notion of multiparty private computation is an effective tool that has extensively been studied in the literature: $n$ different participants $P_i$ with secrets $x_i$ collectively compute $f(x_1, ..., x_n)=w$ such that every participant gets the result $w$ while keeping $x_i$ private [9]. For example, the millionaire protocol [10] allows two parties to determine who is richer without revealing their wealth. A bitwise AND and NOT protocol using oblivious transfer was proposed in [11]. An XOR and AND protocol based on secure blobs was proposed in [12]. A protocol for linear operators ($+, *$) to simulate arbitrary logical circuits was proposed in [13]. Similar protocols were proposed in [14][16]. While these protocols can implement any computing functions, their communication and computation costs are very high. As of this writing, the most efficient protocol for the secure channel model with broadcast was proposed in [17]. The most efficient protocol for the non-broadcast secure channel model, with perfect security, was proposed in [18]. Despite their much lower overheads, these approaches are still too costly to be useful for practical systems.

In this paper, we propose a TTP-cluster based architecture for logging and auditing services of distributed information systems. In our scheme, no single node on the TTP cluster owns the full set of log records, and thus it prevents a single node from misuse of log information. The notion of confidential auditing service allows the auditor to audit aggregated log information, e.g., number of transactions, total of volumes, etc., without accessing all raw data sets. Simple auditing query statements together with a relaxed type of multiparty private computations and distributed data mining demonstrate the effectiveness of proposed scheme.

2. SYSTEM MODEL

Figure 1 depicts the architecture of a typical intrusion detection (ID) system [2]-[5], where the operational information systems submit the logging data to a log repository subsystem, and then the auditor uses the log repository to generate the auditing reports to the users. In analyzing the trail of audit events, the auditor typically looks for a positive or negative proof of compliance of business rules. They would examine traces of system events, evidences of illicit access, and intrusion and fraud detection, etc.

![Figure 1. Centralized auditing model.](image-url)
misuse of the log information, not to mention the performance bottleneck and the dire consequence of the logging system being compromised. The other major issue is related to global auditing of independent systems within strictly defined scopes and rules.

Our goal is to create a distributed, confidential logging and auditing network that is difficult for a single auditor to manipulate, forge or leak the log information, without being detected. To illustrate our system concept, we use logging and auditing of transaction/messaging activities across distributed information systems, so that one can use the DLA services to verify common system information such as order of events, non-repudiation of transactions, correlation of distributed events, multiple host intrusion/anomaly detection, etc.

The architecture of the proposed DLA system is depicted in Figure 2. Given a distributed information system under consideration, the application subsystem consists of $m$ nodes $U = \{u_0, u_1, \ldots, u_m\}$ and $n$ independent DLA nodes $P = \{P_0, P_1, \ldots, P_m\}$, where the confidential logging-auditing services run on the DLA cluster. When a node $u_i$ serves any customer, it submits the corresponding log records to the DLA subsystem, so that selected nodes can store the event logs. For instance, $u_i$ may choose to evenly spread their event logs across the DLA nodes serving the application. Obviously, measures must be taken so that the DLA cluster as a whole has the complete log for every node in the application subsystem.

![Figure 2. A distributed system architecture for online confidential auditing.](image)

We assume that the DLA services run on dedicated computing appliance nodes, which perform only the DLA functions. Event logs, while in many different formats, usually carry such information as time, type of event, involved parties, and any other relevant information to characterize the event. Without loss of generality, let $T$ denote a sequence of transaction events executed by nodes in $U$,

$$T = \{R_T, E_T, L_T, tsn, ttn\},$$

where $R_T$ and $E_T$ respectively denote the specification/rules and event set of $T$, $L_T$ the log records set for each event in $E_T$, $tsn$ the unique transaction sequence number, and $ttn$ the unique transaction type number. The specification or property set of $T$ consists of $m$ boolean equations

$$R_T = \{r_j(T) : j = 0, 1, 2 \ldots m-1\}$$

The event set $E_T$ consists of the $w$ atomic events $e_j^{(T)}(i)$ in $T$ executed by $u_i$.

$$E_T = \{e_j^{(T)}(i)(T) : 0 \leq j < w\}$$

The log record set $L_T$ for each of the events in $T$ is

$$L_T = \{log_j^{(T)}(i)(T) : 0 \leq j < w\}$$

where $log_j^{(T)}(i)(T)$ is logged by $u_i$. A log record $log_j^{(T)}(i)$ consists of

$$log_j^{(T)}(i) = \{glsn, \{l_i : 0 \leq k < h_i\}\}$$

where $glsn$ is a monotonically increasing integer that uniquely defines a log record, $l_i (0 \leq k < h_i)$ are the attributes/properties of the event details.

The content of a transaction reflects its business needs, such as e-commerce, business-to-business transactions, etc. It is too complicated to generalize all the different transaction models and their auditing conditions. Instead, we take some typical examples and demonstrate how to construct their confidential auditing services. Specifically, in next section we will show that one can perform confidential auditing by using commutative encryption, secure set intersection, secure set union, secure sum, max, min and secure comparing. Other more sophisticated operations can be constructed in a similar manner.

The objectives of typical auditing activities are to verify the conformance of system states with transaction specifications/properties $R_T$. DLA nodes use secure multiparty computations, threshold signature and distributed majority agreement to provide trusted and reliable auditing. When nodes compute and then pass intermediate query results between DLA nodes, only the final results corresponding to the queries would be made available to nodes that are authorized to receive the results.

Given the system architecture shown in Figure 2, our objective is to give any node the ability to initiate an auditing query $a\sigma(\)GL$ on the global log trail $GL:I = \{l_0, l_1, l_2, \ldots, l_m\}$ to a DLA cluster, with the auditing criteria $Q$. The DLA cluster should return log pieces that meet the auditing criteria ($Q$). In the query processing steps, no single DLA node and $u_i$ would have access to full knowledge of global log trail space $GL$, and the final query outcomes would be computed in a distributed manner.

Let us define auditing criteria $Q$, which is composed by several auditing predicates using logical connectors $\lor$, $\land$, and $\neg$. The auditing predicate
whose terms are of the form $A \theta (B|c)$, where $A$, $B$ are audit trail attributes that may be derived from partial logs stored at different locations; and $c$ is a constant, and $\theta$ is one of the arithmetic comparison operators $<$, $>$, $\neq$, $\leq$, and $\geq$. Furthermore, the auditing predicate does not contain any quantifiers.

$A \theta (B|c)$ can be evaluated in one single DLA node when both $A$ and $B$ are available in the same node (local auditing predicate), or between two DLA nodes (global auditing predicate). To protect data confidentiality, we require that all cross auditing predicates be computed in a confidential manner for such operations as “set intersection,” “equality” comparisons, “less than,” etc. For statistics primitives we only consider secure computing of the “sum,” “max,” and “min.”

First, we normalize an auditing criterion $(Q)$ to a conjunctive form. The conjunctive form is a conjunction ($\land$ predicate) of a subquery $SQ_i$ as follows: $(SQ_{i1}) \land \ldots \land (SQ_{ik}) \land \ldots \land (SQ_{im})$. Each $SQ_i$ is one of several atomic auditing predicates connected by the logical connectors $\lor$, $\land$, and $\neg$. We require that each $(SQ_i)$ is local auditing predicate or a global auditing predicate, so that each of which can be independently processed by a DLA node. Every DLA node supports all auditing attributes, so that any local auditing predicate can be processed by the proper DLA node. For global auditing predicates, DLA nodes would need to collaborate through a relaxed form of secure distributed computation to process each of $SQ_i$. When all $SQ_i$ are processed, the conjunction of $SQ_i$ is processed by a secure set intersection with $\mathbb{glsn}$ as the set element. The final query result of auditing criteria $(Q)$ will be keyed by $\mathbb{glsn}$ and be returned to the user who initiated the auditing criteria $(Q)$.

![Figure 3. Distributed confidential auditing query processing.](image)

Details of the relaxed secure distributed computing methods to serve the auditing functions are discussed next.

3. RELAXED SECURE DISTRIBUTED COMPUTING

Existing solutions of multiparty private computation [9]-[18] focus on the zero information disclosure, which means that it does not permit any information related to $x_i$ to be revealed, and every participant will get the final result $w$. These algorithms have excessive computing and communication overheads to be used for practical applications [22]. The computing cost can be drastically reduced if certain secondary information about private data $x_i$ (packet count, size, etc) need not be protected. Of course, the data itself will need to be protected in all conditions. Furthermore, it is possible that the cost of multiparty private computation will be greatly reduced if a TTP can coordinate the computation. In many situations, only some participants are interested or authorized in receiving the auditing results, and other participants are required to help producing a part of the final result but they need to keep their own $x_i$ private.

**Definition 1** relaxed multiparty private computations - A set of $n$ DLA nodes is said to perform a relaxed multiparty private computation $f(x_1, \ldots, x_n)=w$ when selected observers receive $w$ while $x_i$ can only be observed by $P_i$. The whole computation could also be done under the coordination of a TTP, but the TTP can only derive the final results without knowing the plaintext of any $x_i$. In the computing process, some secondary form of $x_i$ could be disclosed.

An important foundation of the multiparty private computing is commutative encryption, by which a message encrypted by $n$ parties, can be decrypted using the $n$ matched keys, regardless of their operational sequence. The “XOR” Boolean logic with individual keys is a commutative cipher because “XOR” is a commutative operation. We will discuss other commutative operations shortly. A commutative cryptography system gives us the freedom to route a secret (encrypted) message in a group for secret information processing in any order, e.g., secure computation the size of set intersection [20].

In a more formal expression, an encryption algorithm is commutative if given encryption keys $K_i, \ldots, K_n \in K$, for any $m$ in domain $M$, and for any permutation $i=(i_1, \ldots, i_n)$, $j=(j_1, \ldots, j_n)$ the following two equations hold:

$$E_{K_i}(\ldots E_{K_i}(M) \ldots) = E_{K_j}(\ldots E_{K_j}(M) \ldots) \quad (6)$$

$$\forall M_1, M_2 \in M \text{ such that } M_1 \neq M_2 \text{ and for any } k, \ v < 1/2^k \quad \Pr(E_{K_1}(\ldots E_{K_i}(M_1) \ldots) = E_{K_j}(\ldots E_{K_j}(M_2) \ldots)) < \varepsilon \quad (7)$$

The Pohlig-Hellman scheme [21] is a commutative encryption system whose construct is very similar to that of RSA. Let $p$ be a large prime number for which $p-1$ has a large prime factor. Choose an encryption key $e_a$, such that $e_a$ is relatively prime to $p-1$, and calculate
3.1 Secure set intersection ($\cap$)

As the first step to implement confidential auditing, we now consider the notion of secure set intersection. That is, for each DLA node $P_i$, where $P_i$ has its key pair $(E_a, D_a)$, has a local set $S_i$, which represents a local subset of a transaction log, we wish to securely compute $S_1 \cap \ldots \cap S_n$ as follows. Each node encrypts its set elements with its own key and passes it along to other nodes, and when it receives a set of encrypted set elements from others it encrypts and relays them to the next node. Assuming that message routing is handled by the lower network layer, this encryption and relay process is repeated until every private set is encrypted by every party. Every party eventually receives the original set that it sent out, and all items in the original set have been encrypted by all parties.

Let $P_w$ denote the set of nodes that are authorized to receive the secure set intersection outcome $w$. After the local sets $x_i$’s are exchanged between nodes $P_i$, they can be sent to $P_w$ to make the final determination on the set intersection. Under the commutative encryption scheme, elements in the $n$ encrypted sets have the identical value if and only if their plaintext values are also the same. $P_w$ gets to know which items are in the intersection set, if nodes in $P_w$ have access to the raw log data. Otherwise, they will need to receive the log data from the DLA nodes.

Figure 4 illustrates an example of the secure set intersection on three nodes $P_1$, $P_2$ and $P_3$ that have their private sets $S_1=\{c, d, e\}$, $S_2=\{d, e, f\}$, and $S_3=\{e, f, g\}$. Our goal is to compute the $S_1 \cap S_2 \cap S_3 = \{e\}$ without unveiling other information to each other. For encryption keys $K_i$ and $K_n$, we denote $E_{K_i}(M)$ by $E_a(M)$. Each directional link represents encryption of the message after it is received from the sender to the receiver. The circled number over each link represents the original source of the message. As one can see that after two hops of message exchange and local encryption, the system has three copies of the encrypted messages, where the only common elements for the three sets is the element “e”, encoded in different sequences. The three nodes can decode the plaintext “e” by the use of their matched decoding keys as needed. It is a matter of choice to decide which node(s) would receive the three fully encrypted sets, and how to finally decode them to plaintext to retrieve the plaintext value of “e”.

3.2 Secure Equality Checking ($\approx$)

The problem is how to determine if $X_R$ is equal to $X_M$ that are held by two parties privately. Several secure equality comparison approaches are discussed in [23]. When the set size of $S_i = 1$, the secure set intersection in Figure 5 could be used for secure equality comparison. Randomized mapping is an alternative to the techniques mentioned above. Here, two nodes securely agree upon a random mapping table, which transforms $(X_R, X_M, \ldots)$ to a number space $(Y_R, Y_M, \ldots)$. Two parties agree on two random numbers $a \mod p$ and $b \mod p$, where $a \neq 0 \mod p$ and $p$ is larger than any number in the number space $(Y_R, Y_M, \ldots)$. The two nodes send transformed numbers $W_R$, $W_M$ to a TTP, where $W_R=(aY_R+b) \mod p$, and $W_M=(aY_M+b) \mod p$. Then, the TTP can compare the equality of $W_R$, $W_M$ without knowing the real information $(X_R, X_M)$ and send the result back to the two nodes.

3.3 Secure Distributed Sorting (Max$_s$, Min$_s$, Rank$_s$)

We consider $n$ nodes, each of which has a secret number $x_i$. Without disclosing individual secret $x_i$, the nodes have the common interest in knowing who has the maximum number (Max$_s$), and the minimum number (Min$_s$). One or more parties are interested in knowing the ranking of their numbers (Rank$_s$). These problems can be solved at a high computing cost based on the classical definition of secure distributed computing [10]. However, if all $n$ parties negotiate for a transformation, and let a blind TTP to process these transformed numbers, the cost of the three operations will be significantly reduced. Obviously, provision must be made to prevent the TTP from leaking the results, or to collude with the nodes submitting the inquiry.

3.4 Secure Set Union ($\cup$)

Now, we consider the notion of secure set union [20], where the $n$ nodes wish to securely compute $S_1 \cup S_2 \ldots \cup S_n$ without revealing the owner(s) of each of the items at the final output. The basic computing procedure is similar to that of the secure set intersection. That is, each node encrypts its own local set, and it encrypts and relays the (encrypted) set from other nodes. The process stops when every local set is encrypted by every node. Depending on the service requirements, one or more nodes in $P_w$ can have the
complete collection of encrypted elements of the local sets. By keeping only one copy of any redundant entries in these elements, one can recover the plaintext of the set union by sending each of the kept (encrypted) elements to every node for decoding.

3.5 Secure Sum (Σ) 
Now, we consider the secure sum operation [7]. That is, n nodes that have local values a₀, . . . , aₙ₋₁ collectively compute a₀ + . . . + aₙ₋₁ without giving out their own secret values. Let p be a prime, and p >> aᵢ. Integers modulo p form a finite field. Let x₀, x₁, . . . , xₙ₋₁ be non-zero integers that are smaller than p, and xᵢ's are predetermined by P₀, . . . , Pₙ₋₁. Each node Pᵢ constructs a (k, n) secret sharing polynomial over a finite field (k, n) where the 0th order coefficient is its own secret and all other coefficients are randomly selected from E. Pᵢ chooses a random polynomial f(z) ∈ Zₚ[z] of degree no greater than k-1 such that f(0) = aᵢ.

\[ f(z) = f_0 + f_1z + f_2z^2 + . . . + f_{k-1}z^{k-1} \]

where f₀ = aᵢ, and sᵢ = f(xᵢ). Let node Pᵢ sends sᵢ to Pⱼ (j = 0, . . . , n-1), we get

\[ F(z) = \sum_{i=0}^{k-1} f_i(z) = \sum_{i=0}^{k-1} (\sum_{s_i} f_i)z^i \]

We note that F(z) by itself is a (k, n) secret sharing polynomial, and its 0th order coefficient is (f₀ + f₁₀ + f₂₀ + . . . + fₙ₋₁₀), which is just a₀ + . . . + aₙ₋₁. Pᵢ can compute a share of F(z): (xᵢ, F(xᵢ)), F(xᵢ) = \sum_{i=0}^{n-1} sᵢ mod p, any k shares of F(z) could finally decide the F(z), thus get the 0th order coefficient of F(z): a₀ + . . . + aₙ₋₁.

Let λ₀ , λ₁, . . . , λ₋₁ denote publicly known constants, Pᵢ securely compute λ₀a₀ + . . . + λ₋₁aₙ₋₁ without losing their privacy. Let F(z) = \sum_{i=0}^{n-1} λᵢfᵢ(z), Pᵢ can compute a share of F(z): (xᵢ, F(xᵢ)), F(xᵢ) = \sum_{i=0}^{n-1} λᵢsᵢ mod p. Any k shares of F(z) can determine the value of F(z), and thus get the 0th order coefficient of F(z):

\[ \lambda₀a₀ + . . . + \lambda₋₁aₙ₋₁ \]

4. DISTRIBUTED CONFIDENTIAL AUDITING PROTOCOLS
Given the secure distributed computing primitives (\bigcup_{i=0}^{n} = \bigwedge_{i=0}^{n}, \bigvee_{i=0}^{n}, \text{Max}_{i=0}^{n}, \text{Min}_{i=0}^{n}, \text{Rank}_{i=0}^{n}, \sum_{i=0}^{n}), we now consider how to use them to build a DLA cluster. Obviously, other operators can be modified to provide similar functions.

We assume that uᵢ has full access to its own log trail fragments stored in the DLA cluster, through some ticket authentication. Pᵢ has full access to its own stored log fragments and it can share aggregate statistics on all log trails fragments in U.

The log fragments stored in each DLA node Pᵢ cannot infer any information about event happened in U without the collaboration from other nodes in P.

Nodes on the DLA cluster collaboratively maintain the audit trials. Let I = \{i₀, i₁, i₂, . . . , iₘ\} denote a set of all possible audit log attributes, and each DLA node Pᵢ supports partial audit attributes in I. Attributes in I can be well known, such as time, id, pid, salary, price, etc., or undefined (denoted as C₁, C₂, . . . , Cₙ). Let Aᵢ = \{aᵢ₀, . . . , aᵢₘ\} be supported by Pᵢ, nAi = I, Aᵢ ∩ Aⱼ = Φ ( ∀ i,j). Each audit record structure in an application subsystem has a global log sequence number and a few log attributes, Log = \{glsn, L=(l₀,l₁,l₂, . . . , lₙ)\}, (lᵢ ∈ I, i = 0, . . . , n). (SEE TABLE 1).

The glsn is uniquely assigned by DLA cluster. A global log can be splitted into n fragments Logᵢ = \{glsn,Lᵢ=(lᵢ₀,lᵢ₁,lᵢ₂, . . . , lᵢₙ)\} (TABLE 2 – TABLE 5), where Lᵢ ⊆ Aᵢ (0 ≤ i < n), and \sum_{i=0}^{n} Li = L. The Logᵢ (0 ≤ i < n) is sent to Pᵢ by uᵢ ∈ U.

Before a user uᵢ ∈ U can log (write) a message Logᵢ in a DLA clusters, it must obtain a ticket to authenticate the user and control user’s access operations (read/query, write/log, delete). A ticket is generally a digital signature or Keberos like ticket [28]. Each user could have the same ticket for remote management of all its audit logs on DLA clusters, or have several types of tickets for different operation primitive purposes.

In DLA clusters, each audit node maintains the same access control table for every global log sequence number (glsn). Each assigned glsn is authorized by some ticket. Once some glsn is assigned by DLA for user uᵢ ∈ U with the ticket T, this glsn will be added to the access table under the entry of that ticket T’s ID.

4.1 Distributed Integrity Checking
When a DLA node is compromised, its access control tables and log records could be modified [25]. We design a distributed integrity cross checking algorithm to efficiently discover any illegal modification of these data, based on a one-way accumulator method [26][27], which is similar to a one-way hash function, except that it is communicative.

\[ A(xᵢ, y) = xᵢ y \mod n \]  

(8)
One can hash several messages in any order and get the same hash value. The number \( n \) \((n \text{ is the product of two primes}) \) and \( x_0 \) must be agreed upon in advance. Then, the accumulation of \( y_1, y_2, \) and \( y_3 \) would be

\[
((x_0^y \mod n)^y \mod n)^y \mod n, \text{ which is independent of the order of } y_1, y_2, \text{ and } y_3.
\]

\[
A(A(x_0, y_1), y_2, y_3) = A(A(x_0, y_2), y_3, y_1)
\]  

(9)

For simplicity, we use \( A(x_0, y_1, y_2, \ldots y_n) \) to denote \( A(A(A(... A(x_0, y_1), y_2)..., y_n)). \) The initial value \( x_0 \) must be agreed upon in advance by \( P \) and \( U. \) When \( u_i \in U \) sends fragments \( L, \log \{0 \leq j < n\} \) to \( P_i \), it also computes the one-way accumulator of all fragments \( \log \{0 \leq j < n\} \) and sends the value \( A(x_0, \log \{0 \leq j < n\}) \) to each node in the DLA cluster. DLA node can periodically check the integrity of log records it stores by circulating the intermediate value \( A(x_0, \log \{\}) \) to the next DLA node with the label of \( glsn. \)

Each DLA node locates the corresponding log record keyed by \( glsn, \) and calculates the one-way accumulator value based on the received value and forwards it to the next node. The DLA node that initiates the inquiry will receive the final one-way accumulator value, which is \( A(x_0, \log \{0 \leq j < n\} \), and it can verify whether or not this value is equal to the original stored value sent by \( u_i \in U. \) This scheme allows DLA nodes to check the integrity of the records while keeping them private.

For the agreed access control table of global auditing among DLA clusters, since each \( glsn \) is authorized by some ticket, one could use secure set intersection primitive \((\cap_i)\) to check the consistency of each ticket’s authorization set.

### 4.2 Confidential Auditing of Distributed Events

Distributed security bleaching is usually an aggregated effects of distributed events, each of which alone may appear to be harmless [29]. Thus, the auditing system needs to examine records in multiple nodes to see whether or not \( T \) is executed according to the specifications defined in \( RT \). Transaction control is described by the audit trails, which satisfies transaction semantics defined in \( RT = \{r(T) : 0 \leq j < m\} \) (correlation, fairness, non-repudiation, atomic, consistency checking, irregular pattern detection).

An important consideration of the DLA design is the notion of anonymous, yet authenticated collaboration. The objective is to allow independent users to share critical log information without compromising their individual missions. While authentication can easily be done via a certificate authority, it provides little protection in the operational phase, nor the anonymity of connecting parties. A more suitable solution approach would be extending the notion of e-coin to create undeniable evidences even when nodes remain anonymous.

The notion of dynamic attribute binding of e-coin schemes for evidence chains [30] can meet our goals. As illustrated in Figure 6, after a node \( P_y \) is granted a logging/auditing token, \( t \), from the credential authority, it is given unforgeable authority to engage in the logging and auditing services, when it receives the invitation from the last node of a DLA cluster \( P_y \). When \( P_y \) and \( P_x \) agree to let \( P_y \) become a new member of the DLA cluster, a piece of unforgeable evidence will be created between them to support the claim. Furthermore, \( P_y \) will pass to \( P_x \) its authority to invite other qualified nodes to become a part of the DLA cluster. Once this is done, \( P_y \) can no longer invite other

---

**TABLE 1**  
An Example of the Global Event Log

<table>
<thead>
<tr>
<th>glsn</th>
<th>Time</th>
<th>id</th>
<th>protocol</th>
<th>Tid</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>139aef78</td>
<td>20:18:35/05/12/20</td>
<td>U1</td>
<td>UDP</td>
<td>T1100265</td>
<td>20</td>
<td>23.45</td>
<td>&quot;signature&quot;</td>
</tr>
<tr>
<td>139aef79</td>
<td>20:20:35/05/12/20</td>
<td>U2</td>
<td>UDP</td>
<td>T1100265</td>
<td>34</td>
<td>345.11</td>
<td>&quot;evidence...</td>
</tr>
<tr>
<td>139aef80</td>
<td>20:23:35/05/12/20</td>
<td>U3</td>
<td>TCP</td>
<td>T1100265</td>
<td>18</td>
<td>45.02</td>
<td>&quot;salary&quot;</td>
</tr>
<tr>
<td>139aef81</td>
<td>20:23:38/05/12/20</td>
<td>U3</td>
<td>TCP</td>
<td>T1100265</td>
<td>33</td>
<td>687.5</td>
<td>&quot;account&quot;</td>
</tr>
</tbody>
</table>

**TABLE 2**  
Event Log Fragments Stored in \( P_x \)

<table>
<thead>
<tr>
<th>glsn</th>
<th>Time</th>
<th>id</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>139aef78</td>
<td>20:18:35/05/12/20</td>
<td>U1</td>
<td>23.45</td>
</tr>
<tr>
<td>139aef79</td>
<td>20:20:35/05/12/20</td>
<td>U2</td>
<td>354.11</td>
</tr>
<tr>
<td>139aef80</td>
<td>20:23:35/05/12/20</td>
<td>U1</td>
<td>235.00</td>
</tr>
<tr>
<td>139aef81</td>
<td>20:23:38/05/12/20</td>
<td>U2</td>
<td>45.02</td>
</tr>
<tr>
<td>139aef82</td>
<td>20:25:35/05/12/20</td>
<td>U3</td>
<td>678.75</td>
</tr>
</tbody>
</table>

**TABLE 3**  
Event Log Fragments Stored in DLA Node \( P_y \)

<table>
<thead>
<tr>
<th>glsn</th>
<th>Tid</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>139aef78</td>
<td>T1100265</td>
<td>&quot;signature&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139aef79</td>
<td>T1100265</td>
<td>&quot;evidence&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139aef80</td>
<td>T1100267</td>
<td>&quot;bank&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139aef81</td>
<td>T1100267</td>
<td>&quot;salary&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>139aef82</td>
<td>T1100267</td>
<td>&quot;account&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 4**  
Event Log Fragments Stored in DLA Node \( P_z \)

<table>
<thead>
<tr>
<th>glsn</th>
<th>protocol</th>
<th>ip</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>139aef78</td>
<td>UDP</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>139aef79</td>
<td>UDP</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>139aef80</td>
<td>UDP</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>139aef81</td>
<td>TCP</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>139aef82</td>
<td>TCP</td>
<td>53</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 5**  
Access Control Table

<table>
<thead>
<tr>
<th>Ticket ID</th>
<th>Type</th>
<th>glsn</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>W/R</td>
<td>P0</td>
</tr>
<tr>
<td>T2</td>
<td>W/R</td>
<td>P0</td>
</tr>
<tr>
<td>T3</td>
<td>W/R</td>
<td>P0</td>
</tr>
</tbody>
</table>

---

1. TABLE 1 represents the global event log. Each row contains information such as glsn, time, id, protocol, Tid, C1, C2, and C3.
2. TABLE 2 shows the event log fragments stored in \( P_x \). Each row includes glsn, time, and C1.
3. TABLE 3 lists the event log fragments stored in DLA node \( P_y \). Each row has glsn, Tid, C1, C2, and C3.
4. TABLE 4 provides the event log fragments stored in DLA node \( P_z \). Each row contains glsn, protocol, ip, and C1.
5. TABLE 5 illustrates the access control table. Each row specifies Ticket ID, Type, and glsn.
new nodes to join the DLA cluster. Doing so will subject Py to exposure of its true identity and its misconduct.

In the proposed system, we want that each DLA node learns as little information on the audit trail as possible in store procedure, condition checking (auditing) and query processing. For any piece of audit trail \( \text{Log}=\{\text{gln}, L=(l_0, l_1, l_2, \ldots, l_{w-1})\} \), \( l_i \in I, i=0, w-1 \), let \( w \) denote the total number of audit attributes used in \( \text{Log} \), and \( v \) the total number of undefined attributes used in \( \text{Log} \). Let \( u \) denote the minimum number of DLA nodes, whose supported attributes set could cover all attributes appearing in \( \text{Log} \). \( n \) is the total number of DLA nodes \( P = \{P_i; \; 0 \leq i < n\} \) and \( |I| \) is the total number of all possible audit log attributes \( I=\{i_0, i_1, i_2, \ldots, i_{ndf}\} \).

An undefined attribute is an abstract attribute that is only meaningful to the application subsystem by private agreements. Thus increasing the number of undefined attributes in \( \text{Log} \) makes it more difficult for individual DLA nodes to infer any hints from audit trail fragments. Furthermore, it is also more difficult for individual DLA nodes to know about log information if more DLA nodes are needed to recover a complete audit trail.

We define the store confidentiality of audit trail \( \text{Log} \) as
\[
C_{\text{store}}(\text{Log}) = \frac{vu}{w} \quad 0 \leq v \leq w \leq |I|, \; 0 \leq u \leq n
\]  
(10)

For any auditing criterion \( Q \), after normalization \( Q_N = (S_Q) \land \ldots \land (S_Q) \land \ldots \land (S_{Q_{q-1}}) \), we define the auditing confidentiality of criterion \( Q \) (a Boolean condition) as
\[
C_{\text{auditing}}(Q) = \frac{t+q}{s+q}
\]  
(11)

\( s \) is the total number of atomic auditing predicates in \( Q_N \), \( t \) the total number of cross auditing predicates, and \( q \) the total number of conjunctive predicates in \( Q_N \).

For executing any query \( Q \), which will include the auditing criterion \( Q \) and project the satisfied audit trail sets on particular local audit space \( \text{Log} = \{\text{gln}, L=(l_0, l_1, l_2, \ldots, l_{w-1})\} \), we define the query confidentiality of criterion \( Q \) on \( \text{Log} \) space as
\[
C_{\text{query}}(Q, \text{Log}) = C_{\text{auditing}}(Q) \times C_{\text{store}}(\text{Log})
\]  
(12)

The average of query confidentiality of criterion \( Q \) on \( \text{Log} \) space is defined as DLA confidentiality
\[
C_{\text{DLA}}(I, P) = C_{\text{query}}(Q, \text{Log})
\]  
(13)

\( I=\{i_0, i_1, i_2, \ldots, i_{ndf}\} \) denotes a set of all possible audit log attributes, and each auditing server \( P_i \) supports partial audit attributes in \( I \).

6. CONCLUSION

In this paper, we propose a confidential logging and auditing scheme that is suitable for distributed logging and auditing of group activities in a P2P networking setting. Our scheme restricts DLA nodes from gaining complete information of the log trails, so that no single node can abuse their responsibility in the logging and

Figure 6. Undeniable evidence chain for (anonymous) DLA membership, logging and auditing.

The privileges and responsibilities of \( P_x \) to serve in the DLA cluster would be negotiated with \( P_y \). Once being agreed upon by the two nodes, these logging and auditing attributes become a new part of the DLA service policies. The service terms can be bound into auditing attributes become a new part of the DLA being agreed upon by the two nodes, these logging and auditing of group activities in a P2P networking setting. Our scheme restricts DLA nodes from gaining complete information of the log trails, so that no single node can abuse their responsibility in the logging and
auditing process. By proper mapping of logging functions to the e-coin cryptographical system, we can make the DLA’s service roles unforgeable and undeniable. This way, even when the DLA nodes remain anonymous, the application nodes can remain confident about the integrity of the DLA operations.

REFERENCES


