

# On the Confidential Auditing of Distributed Computing Systems

Yiping Shen, T. C. Lam, Jyh-Charn Liu<sup>1</sup>, Wei Zhao  
{ypshen, tbl4427, liu, zhao}@cs.tamu.edu  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112

**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 the 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 the 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., the number of transactions, the total volume, the 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

Many auditing services have been developed for the policy compliance, the operational procedures, and the service quality, etc. [1]. The auditor uses random samples to detect fraudulence, errors, or real time intrusion detection [2]-[5]. Auditing requirements for electronic commerce systems are suggested in [6]. In [7], the notion of *secret counting* is proposed to audit system statistics, such as utilization ratio of services and search performance, without unveiling the privacy of the library patrons. In [8], a “confidential auditing control” approach is discussed, where timestamp and multiparty private computations are addressed without discussing details of auditing functions.

Distributed logging and auditing (DLA) has not been widely adopted because unveiling system logs could lead to business loss or litigation. To make the DLA useful for such applications as distributed services provided by multiple independent sources, wide area intrusion detection, etc., the system must guarantee the privacy of the system owners.

DLA requires participants to perform collaborative computation without unveiling local information, whenever possible. To prevent an unsupervised TTP from manipulating the system, we design query processing schemes that require TTP nodes work together, using the *multiparty private computation*, to perform any useful auditing functions. The notion of multiparty private computation that has been extensively studied in the literature:  $n$  different participants  $p_i$  with secret  $x_i$  collectively compute

$y = f(x_1, \dots, x_n)$  such that every participant gets the result  $y$  while keeping  $x_i$  private [9]. For example, the millionaire protocol [10] allows two parties to determine who is richer without revealing their actual wealth. A bitwise *AND* and *NOT* protocol using oblivious transfer is proposed in [11]. An *XOR* and *AND* protocol based on the secure blobs is proposed in [12]. A protocol for linear operators (+, \*) to simulate arbitrary logical circuits is proposed in [13]. Similar protocols are proposed in [14][16]. As of this writing, the most efficient protocol based on secure broadcast channel model is proposed in [17]. The most efficient protocol, based on non-broadcast secure channel model, for perfect security is proposed in [18]. These protocols are cost prohibitive for practical use.

In this paper, we propose a TTP cluster architecture to provide confidential logging and auditing of global events without unveiling details of individual 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 misusing the log information. An auditor can inquire about aggregated log information, e.g., transactions volumes, etc., without accessing all raw data sets, i.e., *confidential auditing service*.

## 2. SYSTEM MODEL

Figure 1 depicts a typical intrusion detection system (IDS) [2]-[5], where the operational information system ( $U$ ) submits the logging data to a log repository ( $L$ ), and then the auditor ( $P$ ) accesses  $L$  to generate auditing reports.  $P$  analyzes audit events to look for positive or negative proof of “interesting events,” such as intrusion or fraudulence.

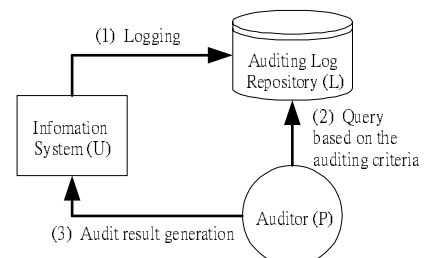


Figure 1. Centralized auditing model.

To expand the centralized auditing model for distributed information systems, 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. To illustrate our system

<sup>1</sup> Correspondence author.

concept, we use logging and auditing of transaction activities across multiple systems, where one can use DLA services to verify common system information such as the order of events, non-repudiation of transactions, the correlation of distributed events, and detection of multiple host intrusion or anomaly.

The architecture of the distributed DLA system is depicted in Figure 2, where the application subsystem consists of  $m$  transaction nodes  $U = \{u_1, \dots, u_m\}$  and  $n$  independent DLA nodes  $P = \{p_1, \dots, p_n\}$ . The proposed system supports three major functions: the secure log distribution, the distributed audit query and the secure distributed computing of the aggregated audit results. The confidential logging-auditing services run on the DLA cluster, which is expected to run only DLA functions.

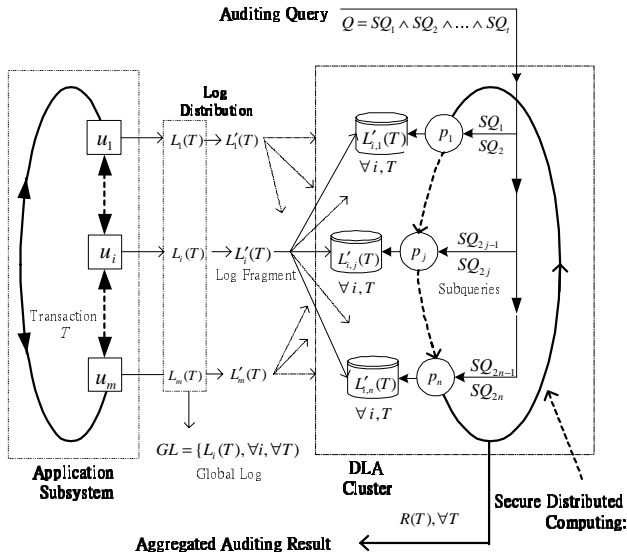


Figure 2. A distributed system architecture for online confidential auditing.

In transaction  $T$ ,  $u_i$  splits the set of its local log records  $L_i(T)$  into  $n'_T$  fragments:

$$L'_i(T) = \{L'_{i,1}(T), \dots, L'_{i,n'_T}(T)\}$$

The log fragments are distributed to  $n'_T$  chosen DLA nodes for storage. Obviously, one will have to design the DLA cluster so that it can obtain complete log from the  $m'_T$  nodes involved in the transaction. Without loss of generality, we assume that  $n'_T = n$  and  $m'_T = m$  in the subsequent discussions.

After the log is distributed to DLA nodes, the DLA cluster has the global log  $GL = \{L_i(T) : \forall i, T\}$ , but a DLA node can only support auditing inquiries that are related to its local attributes. Typically, auditing activities are designed to verify conformance of system activities to certain rules. An auditing query  $Q$  is sent to the DLA cluster based on the transaction specification  $R(T)$ . In general,  $Q$  can be divided into predicates that are connected by logic operators  $\wedge$  (AND),  $\vee$  (OR) and  $\neg$  (NOT).  $Q$  can be normalized to

$t$  atomic subqueries:  $Q = SQ_1 \wedge \dots \wedge SQ_t$ , as depicted in Figure 3. Each subquery is then processed by DLA nodes that has relevant local auditing attributes.

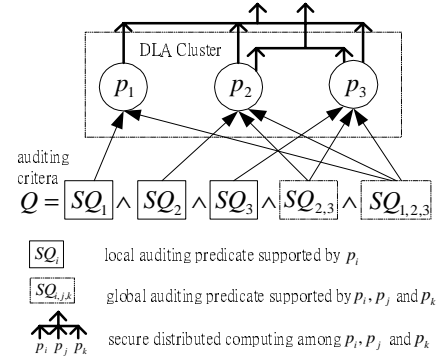


Figure 3. Distributed confidential auditing query processing.

Multiple DLA nodes that collaborate to generate the aggregated auditing result should not leak local information. To achieve this goal, we propose using a relaxed form of multiparty private computing for distributed query processing. To illustrate the key concepts of DLA, we take a typical database transaction example to construct its confidential auditing services. 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 comparison*. Other operators can be constructed in a similar fashion.

### 3. SECURE LOG DISTRIBUTION

Event logs, while in many different formats, usually carry such information as transaction time, event type, involved parties, and other relevant information to characterize the event. Without loss of generality, let  $T = \{R(T), L(T)\}$  denote a sequence of transaction events executed by nodes in  $U$ , where  $R(T)$  and  $L(T)$  respectively denote the transaction specification, and log records of transaction events with respect to  $T$ .  $R(T)$  consists of a number of Boolean equations that governs the rules for log distribution and (distributed) confidential audit query, etc.  $L(T) = \{L_i(T) : 1 \leq i \leq m\}$  is comprised of the local log records of nodes in  $U$ .  $L_i(T) = \{log_{i,k}(T) : 1 \leq k \leq w_i\}$  includes  $w_i$  pieces of log records, where  $w_i$  is the number of atomic events logged by  $u_i$ . Each log record  $log_{i,k}(T)$  contains values of attributes  $A = \{a_1, a_2, \dots, a_h\}$  that specify event details, where  $h$  is the total number of attributes defined in the specification,  $a_1$  the global log sequence number (*glsn*),  $a_2$  the transaction sequence number (*tsn*),  $a_{j>3}$  either the *well known* attributes such as the transaction time, user ID, salary, etc., or the *undefined* attributes, denoted by  $C_1, C_2, \dots$ , etc. We assume that *tsn* is unique to each log record within the

same transaction. And  $glsn$  is uniquely assigned by the DLA cluster for each log record within the global log

$$GL = \{L(T) : \forall T\} = \{L_i(T) : \forall T, i\}$$

which is a virtual view of the complete log information of the whole system, as depicted in TABLE 1. In this example,  $GL$  could be the system log being monitored by an intrusion detection system (IDS).

Let us assume that  $R(T)$  specifies that  $p_j$  supports logging and auditing of attributes  $B_j$ , such that

$$A \subseteq \bigcup_{j=1}^n B_j, \text{ and } \bigcap_{j=1}^n B_j = \{glsn, tsn\} \quad (1)$$

In transaction  $T$ ,  $u_i$  fragments its log records  $L_i(T)$  into  $n$  pieces  $L'_i(T) = \{L'_{i,1}(T), \dots, L'_{i,n}(T)\}$  such that each log record  $log'_{i,k}(T) \in L'_{i,j}(T)$  that is denoted as  $log'_{i,k,j}(T)$  contains values of attributes  $A_j$ , where

$$(A \cap B_j) \subseteq A_j \subseteq B_j \quad (2)$$

An example that depicts the relationships between  $A$ ,  $A_j$ ,  $B_j$  and different log fragments is shown in TABLE 1 – TABLE 4. Note that the log values of  $A_j \setminus (A \cap B) = A_j \setminus A$  do not contribute the  $GL$ , as denoted by italic face in the tables. They are null data generated by  $u_i$  to bewilder  $p_j$  from interpreting semantics of  $GL$ . Details on how to define the confidential audit process will be explained in next section shortly.

Secure log distribution refers to submission and logging of  $log'_{i,k,j}(T)$  from  $u_i$  to  $p_j$ ,  $\forall i, j, k$ , without compromising the secrecy and integrity of log records. (1) and (2) ensure that log fragments in  $p_j$  cannot infer any information about events in  $U$  without communicating with other nodes in  $P$ . With a suitable authentication scheme for the log access between  $u_i$  and  $p_j$ ,  $u_i$  only knows certain rows in  $GL$  (TABLE 1), while  $p_i$  only knows certain columns, and no single node has the full knowledge of  $GL$ . The aggregated audit outcomes about  $GL$  computed by the multiparty private computation protocol are shared by DLA nodes. We assume that the communication protocol is sufficiently transparent so that the probability of undetected collusion between nodes in  $P$  is negligible.

TABLE 1  
AN EXAMPLE OF THE GLOBAL EVENT

$A = \{$		$glsn$	$tsn$	$time$	$uid$	$action$	$vol_1$	$vol_2$	$vol_3$	$eval_1$	$eval_2$	$C_3$	$C_4$	$eval_3$	$eval_4$	$\}$
$L_1(T)$ from $u_1$		78	265	201835	u1	send	10	18	11	gd	bad	gd	nil	nml	gd	$\} log_{1,1}(T)$
		79	267	202036	u1	recv	15	5	24	nml	gd	nml	nml	gd	nml	
$L_2(T)$ from $u_2$		80	290	202316	u2	send	29	12	3	gd	nil	gd	nml	bad	gd	$\} log_{2,1}(T)$
		81	297	202506	u2	recv	23	7	9	nml	nml	gd	nml	nml	nml	

TABLE 2  
EVENT LOG FRAGMENT IN  $P_1$

$A_1 =$		$glsn$	$uid$	$vol_1$	$eval_1$	$eval_2$	$size$	$\}$
$L'_{1,1}(T)$ from $u_1$		78	u1	10	gd	bad		$\} log'_{1,1,1}(T)$
		79	u1	15	nml	gd		
$L'_{2,1}(T)$ from $u_2$		80	u2	29	gd	nil		$\} log'_{2,1,1}(T)$
		81	u2	23	nml	nml		

TABLE 3  
EVENT LOG FRAGMENT IN  $P_2$

$A_2 =$		$glsn$	$tsn$	$time$	$vol_2$	$eval_4$	$eval_5$	$eval_6$	$\}$	
$L'_{1,2}(T)$ from $u_1$		78		265	201835	18	<i>nml</i>	nml	gd	$\} log'_{1,1,2}(T)$
		79		267	202036	5	<i>bad</i>	gd	nml	
$L'_{2,2}(T)$ from $u_2$		80		290	202316	12	<i>gd</i>	bad	gd	$\} log'_{2,1,2}(T)$
		81		297	202506	7	<i>gd</i>	nml	nml	

TABLE 4  
EVENT LOG FRAGMENT IN  $P_3$

$A_3 =$		$glsn$	$action$	$ptcl$	$vol_3$	$C_2$	$C_3$	$C_4$	$\}$
$L'_{1,3}(T)$ from $u_1$		78	send	<i>tcp</i>	11	<i>gd</i>	gd	nil	$\} log'_{1,1,3}(T)$
		79	recv	<i>udp</i>	24	<i>gd</i>	nml	nml	
$L'_{2,3}(T)$ from $u_2$		80	send	<i>tcp</i>	3	<i>nml</i>	gd	nml	$\} log'_{2,1,3}(T)$
		81	recv	<i>icmp</i>	9	<i>gd</i>	gd	nml	

#### 4. DISTRIBUTED AUDIT QUERY

Confidential auditing allows any node to initiate an auditing query on  $GL$  with the auditing criterion  $Q$ .  $Q$  can be composed of nested or non-nested auditing predicates using logical operators  $\wedge$  (AND),  $\vee$  (OR), and  $\neg$  (NOT). The first step of the distributed query processing is to normalize  $Q$  to a conjunctive normal form (CNF) as follows:

$$Q = (SQ_1) \wedge \dots \wedge (SQ_t).$$

Each  $SQ_i$  represents an atomic audit predicate which can be expressed in the format of  $a_1 <_{a_3} \theta(a_2 | c) <_{a_4} >$ , where  $a_1$  and  $a_2$  are the audit trail attributes derived from the partial logs stored at different locations,  $c$  a constant.  $\theta$  is one of arithmetic comparison operators  $<$ ,  $>$ ,  $=$ ,  $\neq$ ,  $\leq$ ,  $\geq$  or set operators  $\in$ ,  $\notin$ ,  $\subset$ ,  $\supset$ ,  $\subseteq$ ,  $\supseteq$ , etc.  $<_{a_3} >$  and  $<_{a_4} >$  are optional selection statements on  $a_1$  and  $a_2$ , respectively, similar to the “where”, “group by”, “order by” statements in SQL. DLA nodes that support  $<_{a_3} >$  and  $<_{a_4} >$  compute the related sets of  $glsn$ , and then return them to DLA nodes that support  $a_1, a_2$  for evaluations of  $a_1 \theta(a_2 | c)$ . When all  $SQ_i$  are processed, the final query result of  $Q$  is computed by a series of conjunctions of  $SQ_i$  and return to nodes that are authorized to receive the results.

To prevent any DLA node from getting the whole raw log data,  $a_{1 < a_3 >} \theta(a_2 | c)_{< a_4 >}$  needs to be evaluated by multiple DLA nodes, *i.e.*, *global auditing predicate* using secret computing operations. These secret computing operations discussed in this paper include set intersection ( $\cap_s$ ), set union ( $\cup_s$ ), equality comparisons ( $=_s$ ), *etc.* For statistics primitives, we only consider secret computation of Sum ( $\Sigma_s$ ), Max( $\cdot$ ), and Min( $\cdot$ ). To reduce the computing overhead, we modify the original form of secret computing into a relaxed form of multiparty private computing for processing of global auditing predicates.

Now we discuss how to design the confidential auditing criterion  $Q$  for the example given in TABLE 1 to TABLE 4. Recall that  $GL$  is the global log of a network based IDS, which is our application system  $U$ . The objective of  $Q$  is to examine (i) balance of the inbound and outbound traffic volumes, AND (ii) validity of sender-receiver pairs, AND (iii) the “adequacy” of security levels at different application components.

In the auditing process, we do not want to unveil the traffic volume and the security levels of individual nodes. Symbols “gd, nml, bad, nil” are used to denote that the security level is “good, normal, poor, unrated”. The system under consideration is “good enough” if: (i) at least one “gd” for each pair  $\{eval_1, eval_2\}$ ,  $\{C_3, C_4\}$  and  $\{eval_5, eval_6\}$ , OR (ii) all entries are not worse than “nml” and at least one “gd” in all above attributes.

The CNF of  $Q$  can be expressed as  $SQ_1 \wedge SQ_2 \wedge SQ_3$ ,

$$SQ_1 : \{uid\}_{<action=send>} = \{uid\}_{<action=recv>}$$

$$SQ_2 : \sum_{i=1}^3 (\Sigma vol_i)_{<action=send>} = \sum_{i=1}^3 (\Sigma vol_i)_{<action=recv>}$$

$$SQ_3 : (\{eval_1, eval_2\} \cap_s \{C_3, C_4\} \cap_s \{eval_5, eval_6\} \supseteq \{ "gd" \}) \vee (\{eval_1, eval_2\} \cup_s \{C_3, C_4\} \cup_s \{eval_5, eval_6\} = \{ "gd", "nml" \})$$

When a DLA node  $p_3$  receives  $SQ_1$  and  $SQ_2$ , which contain the selection statement  $< action = send >$ , it looks up TABLE 4 and returns the set of  $glsn\{78, 80\}$  to  $p_1$  and  $p_2$ . Similarly,  $p_3$  returns  $\{79, 81\}$  for the selection statement  $< action = recv >$ . Based on  $\{78, 80\}$  and  $\{79, 81\}$ ,  $p_1$  evaluates  $SQ_1 = 1$ , because both sides of the equality are  $\{u1, u2\}$ .

Consider the left hand side of  $SQ_2$ ,  $p_1$  computes locally  $\Sigma vol_1 = (10+29) = 39$  using TABLE 2 and similarly,  $\Sigma vol_2 = (18+12) = 30$  by  $p_2$  using TABLE 3,  $\Sigma vol_3 = (11 + 3) = 14$  by  $p_3$  using TABLE 4. However, the computation of  $\sum_{i=1}^3 (\Sigma vol_i) = 39 + 30 + 14 = 83$  requires the collaborations among  $p_1$ ,  $p_2$  and  $p_3$ , using  $SQ_2$  and  $SQ_3$ . One can prevent any node from

getting the complete log data using secure distributed computing to be explained in the next section,

## 5. SECURE DISTRIBUTED COMPUTING

Confidential auditing relies on secret computing of log information without unveiling the intermediate results to DLA nodes. 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 encrypted message in a group for secret information processing in any order, *e.g.*, secure computation the size of a set intersection [20]. This way, a DLA node can process a sub-query only if it has the proper key, and no intermediate results can be unveiled without proper key(s).

In a more formal expression, an encryption algorithm  $E$  is commutative in the message domain  $X$  if given  $\ell$  encryption keys  $k_1, \dots, k_\ell$ , the following equations hold:

$$E_{k_{i_1}} (E_{k_{i_2}} (\dots E_{k_{i_\ell}} (x) \dots)) = E_{k_{j_1}} (E_{k_{j_2}} (\dots E_{k_{j_\ell}} (x) \dots))$$

$\Pr(E_{k_{i_1}} (E_{k_{i_2}} (\dots E_{k_{i_\ell}} (x_1) \dots))) = E_{k_{j_1}} (E_{k_{j_2}} (\dots E_{k_{j_\ell}} (x_2) \dots)) < \varepsilon$  for any permutations  $(i_1, \dots, i_\ell)$  and  $(j_1, \dots, j_\ell)$ ,  $\forall x, x_1, x_2 \in X$ ,  $x_1 \neq x_2$ ,  $\varepsilon < 1/2^N$  for any integer  $N$ .

The Pohlig-Hellman scheme [21] is a commutative encryption system that has a similar construct as the RSA. Here, let  $q$  be a large prime number for which  $(q-1)$  has a large prime factor. Choose an encryption key  $e_a$ , such that  $e_a$  is a relative prime to  $(q-1)$ . Calculate  $d_a$  such that  $(d_a)(e_a) = 1 \pmod{(q-1)}$ . The cipher text of a message  $x$  is  $y = x^{e_a} \pmod{q}$ , which can be decrypted by  $x = y^{d_a} \pmod{q}$ . Let  $(e_b, d_b)$  denote another key pair. Consecutive encryptions of  $x$  by  $e_a$  and  $e_b$  result in  $(x^{e_a})^{e_b} = x^{(e_a)(e_b)} = (x^{e_b})^{e_a} \pmod{q}$ . Given a commutative cryptographic paradigm, such as the one in [21], our next step is to develop secure and secret query processing operators in order to evaluate the auditing predicates.

Existing solutions on multiparty private computation [9]-[18] are too costly for practical implementation even though they support zero information disclosure, which means that no information related to  $x_j$  would be unveiled, and every participant gets only the final result  $y$  [22].

Without compromising confidentiality of raw log data, we propose a relaxed form of multiparty computation to reduce the computing cost. A set of  $n$  DLA nodes is said to perform a relaxed multiparty private

computation  $y = f(x_1, \dots, x_n)$  when selected observers receive  $y$  while  $x_j$  can only be observed by  $p_j$ . The cost of multiparty private computation can be further reduced if a TTP coordinates the (secret) computation, where the TTP can only derive the final results without knowing the plaintext of any  $x_j$ . Traffic patterns of  $x_j$  could be disclosed in the computing process.

### 5.1 Secure Set Intersection ( $\cap_s$ )

For secure set intersection, each DLA node  $p_j$  has a key pair  $(e_j, d_j)$  and a local set  $X_j$ , which represents a local subset of a transaction log records. We securely compute  $Y = X_1 \cap_s \dots \cap_s X_n$  as follows. Each DLA node encrypts its set elements with its own key and passes it to another DLA node. When it receives a set of encrypted set elements from others, it encrypts and relays them and to the next node. This encryption and relay process terminates when every private set is encrypted by all parties and received by every DLA node.

Let  $P_Y$  denote the set of nodes that are authorized to receive  $Y$ , the result of a secure set intersection. After the local sets  $X_1 \dots X_n$  are exchanged between DLA nodes, they can be sent to  $P_Y$  to make the final determination on the set intersection. Using commutative encryption, elements in the  $n$  encrypted sets have the identical value if and only if their plaintext values are also the same.  $P_Y$  can determine the plaintext of the intersection set only if it has access to the raw log data.

Figure 4 illustrates an example of the secure set intersection on three nodes  $p_1$ ,  $p_2$  and  $p_3$  on the first rows of TABLE 2 – TABLE 4, with their private sets  $X_1 = \{gd, bad\}$ ,  $X_2 = \{nml, gd\}$  and  $X_3 = \{gd, nil\}$  respectively. According to the first part of  $sQ_3$  in section 4, our goal is to compute  $X_1 \cap_s X_2 \cap_s X_3 \subseteq \{gd\}$  without unveiling other private information from one another. Let  $E_{i,j}(x)$  denote the nested encryption steps  $E_{k_i}(E_{k_j}(x))$  using keys  $k_i$  and  $k_j$ . Each directional link represents an 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 element for the three sets is the “gd”, encrypted in different orders, satisfying the first part of  $sQ_3$ . The three nodes can decode the plaintext “gd” by using their matched decoding keys. It is a matter of choice to decide which node(s) would receive the three fully encrypted sets, and how to decode them to retrieve the plaintext value of “gd”.

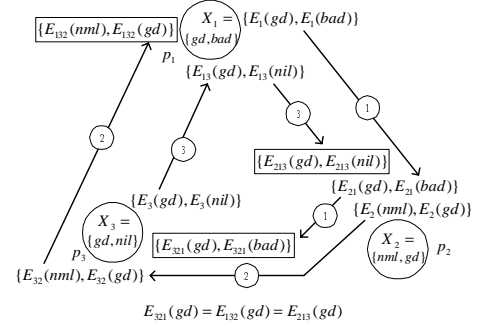


Figure 4. Secure set intersection.

### 5.2 Secure Set Union ( $\cup_s$ )

For secure set union [20],  $n$  nodes securely compute  $Y = X_1 \cup_s \dots \cup_s X_n$  without revealing the owner(s) of each item at the final output. The basic computing procedure is similar to that of the secure set intersection. Each node encrypts its own local set and relays them from one another. The process stops when every local set is encrypted by all nodes. Depending on the service requirements, one or more nodes in  $P_Y$  have the complete collection of encrypted elements of local sets. By keeping only one copy of any redundant entries in these elements, one can recover the plaintext from the union set by sending each encrypted element to every node for decoding. Using the last rows of TABLE 1 – TABLE 4 as an example,  $X_1 \cup_s \dots \cup_s X_n = \{gd, nml\}$ , which satisfies the second part of  $sQ_3$  in section 4.

### 5.3 Secure Sum ( $\sum_s$ )

For *secure sum* [7],  $y$  is collectively computed from  $n$  secret values  $x_1, \dots, x_n$ , without leaking any individual secret  $x_j$ , where  $y = \sum_{j=1}^n x_j$ . Let

$q \gg x_j > 0$  be a large prime and  $\alpha_1, \dots, \alpha_n \in Z_q^*$  are publicly predetermined by  $p_1, \dots, p_n$ . Each  $p_j$  constructs a  $(k, n)$  secret sharing polynomial  $f_j(z) = \sum_{\ell=0}^{k-1} f_{j,\ell} z^\ell \pmod{q}$ , where  $f_j(0) = f_{j,0} = x_j$  and  $f_{j,\ell} \in_R Z_q$ . Then  $p_j$  submits  $s_{i,j} = f_j(\alpha_i)$  to every  $p_i$ .

Given another  $(k, n)$  secret sharing polynomial  $F(z) = \sum_{j=1}^n f_j(z) \pmod{q}$ ,  $p_i$  is able to compute a share of  $F(z)$ , i.e.,  $F(\alpha_i) = \sum_{j=1}^n s_{i,j} \pmod{q}$ . Any  $k$  shares from  $P$  uniquely determine the coefficients of  $F(z)$ , and hence the secure sum  $F(0) = \sum_{j=1}^n f_{j,0} = \sum_{j=1}^n x_j$ .

For example, we try to compute the left hand side of  $SQ_2$  in section 4, *i.e.*,  $\sum_{j=1}^3 (\Sigma vol_j) = 39 + 30 + 14 = 83$ . Let  $q = 103$ ,  $\alpha_1 = 3$ ,  $\alpha_2 = 8$ ,  $\alpha_3 = 13$ ,  $k = 2$ ,  $f_1(z) = 39 + 5z$ ,  $f_2(z) = 30 + 7z$ ,  $f_3(z) = 14 + 9z$ .  $p_1$  receives  $s_{1,1} = 54$ ,  $s_{1,2} = 51$ ,  $s_{1,3} = 41$  and computes the first share  $\sum_{j=1}^3 s_{j,0} + 3 \sum_{j=1}^3 s_{j,1} = 54 + 51 + 41 = 146 = 43 \pmod{103}$ .  $p_2$  receives  $s_{2,1} = 79$ ,  $s_{2,2} = 86$ ,  $s_{2,3} = 86$  and computes the second share  $\sum_{j=1}^3 s_{j,0} + 8 \sum_{j=1}^3 s_{j,1} = 79 + 86 + 86 = 45 \pmod{103}$ . By solving these two equations, we have  $\sum_{j=1}^3 s_{j,0} = \sum_{j=1}^3 (\Sigma vol_j) = 83$ .

Secure weighted sum  $\sum_{j=1}^n \lambda_j x_j$  can be computed in a similar way, given  $n$  publicly known weight values  $\lambda_1, \dots, \lambda_n \in \mathbb{Z}_q$ . Different from the secure sum operation, the secure weighed sum operation defines  $F(z') = \sum_{j=1}^n \lambda_j f_j(z') \pmod{q}$ . Then we can compute  $F(0) = \sum_{j=1}^n \lambda_j f_{j,0} = \sum_{j=1}^n \lambda_j x_j$  from any  $k$  shares  $F(\alpha_j)$ .

#### 5.4 Secure Equality Checking ( $=_s$ )

For secure equality checking, we try to determine if  $X_1 =_s \dots =_s X_n$  when  $X_j$  are held by  $p_j$  privately. Several *secure equality comparison approaches* are discussed in [23]. Secure equality checking can be implemented by comparing if the secure intersection set equals to the secure union set. Randomized mapping is an alternative to the techniques mentioned above. First, two nodes securely agree upon a random mapping table, which transforms  $(X_1, X_2, \dots)$  to a different number space  $(Y_1, Y_2, \dots)$ . Two parties agree on two random numbers  $a \pmod{q}$ , and  $b \pmod{q}$ , where  $a \neq 0 \pmod{q}$  and  $q$  is larger than any number in the number space  $(Y_1, Y_2, \dots)$ . The two nodes send the transformed numbers  $(W_1, W_2, \dots)$  to a TTP, where  $W_j = (aY_j + b) \pmod{q}$ . Then, the TTP can compare the equality of  $(W_1, W_2, \dots)$  without knowing the real information  $(X_1, X_2, \dots)$ .

#### 5.5 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 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 results, or colluding with nodes that submit the inquiry.

## 6. DEFENSE AGAINST MALICIOUS NODES

Security incidents in distributed systems are usually involved with multiple distributed events, each of which alone may appear to be harmless [29]. The DLA system can examine log records in multiple nodes to determine compliance of transaction execution rules for correlation, fairness, atomacy, consistency checking, irregular pattern detection, *etc.*, while preserving the privacy of local nodes.

However, the confidentiality of the individual data and the fidelity of the overall auditing result can be ensured only when the correct log records are submitted from the correct transaction nodes and stored in the correct DLA nodes. Authentication between the participating parties and integrity check of the log data are necessary to detect unauthorized alteration of log data, secret data extraction by excessive queries, *etc.* When offense occurs, the culprit should be identified from undeniable evidence generated from the system.

### 6.1 User Access Control and Distributed Integrity Checking

In our DLA system,  $u_i$  has the full access to its own log fragments, while  $p_j$  has access only to its local log fragments. Authentication between  $u_i$  and  $p_j$  for the log access such as read/query, write/log, delete, *etc.*, can be done by an access control table (TABLE 5) in  $p_j$ , and an authentication ticket, such as the Keberos [28] or other digital signature schemes. Each  $u_i$  has a single ticket for the remote management of all its audit logs on DLA clusters, or multiple types of tickets for difference operation primitives.

Once a  $glsn$  is assigned and authorized by a ticket from the DLA cluster, the  $glsn$  will be added to the access control table under the entry of that ticket ID. Further access to a log record between  $u_i$  and  $p_j$  can be granted/denied based on entries in the access control table.

TABLE 5  
ACCESS CONTROL TABLE

ticket ID	ticket type	glsn
T534	R/W	78,80
T544	R/W	79,81
T290	R/W	82

When  $p_j$  is compromised, its access control tables and log records could be modified [25]. We design a distributed integrity cross checking algorithm to effectively discover any unauthorized alteration of log records, based on a commutative one-way hash function, called the one-way accumulator [26][27].

$$H(x, y) = y^x \bmod v$$

where  $v$  is a composite of two large primes,  $x$  is the message to hash and  $y$  is a seed to the hash function.

One can consecutively hash multiple messages in any order and get the same hash value. The accumulation of  $x_1$ ,  $x_2$  and  $x_3$  corresponding to the seed  $y_0$  is  $((y_0^{x_1} \bmod v)^{x_2} \bmod v)^{x_3} \bmod v$ , independent of the order of  $x_1$ ,  $x_2$  and  $x_3$ .

For simplicity, we use  $H(x_1, x_2, \dots, x_n, y_0)$  to denote  $H(x_n, H(x_{n-1}, H(\dots, H(x_1, y_0))))$ . The initial value  $y_0$  must be agreed upon in advance by  $P$  and  $U$ . When  $u_i$  sends the log fragment  $\log'_{i,k,j}(T)$  to  $p_j$ , it also sends the following accumulator value to  $p_j$ :

$$H(\log'_{i,k,1}(T), \log'_{i,k,2}(T), \dots, \log'_{i,k,n}(T), y_0)$$

To check the integrity of the records while keeping them private,  $p_j$  can periodically check the integrity of the log records in its possession by circulating its intermediate value  $H(\log'_{i,k,j}(T), y_0)$  to the next DLA node with the label  $glsn$ . Each DLA node locates the corresponding log record keyed by  $glsn$  and computes 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, and verify that whether or not it is equal to the original one sent by  $u_i$ .

## 6.2 Anonymous DLA Node Authentication

An important consideration of the DLA design is the notion of anonymous, yet authenticated collaboration. It allows independent users to share log information for global system management, without compromising the privacy of their missions. While certificate offers authentication, it does not provide protection in the operational phase, nor the anonymity of the communicating parties. A more suitable approach would be extending the notion of undeniable evidence chain [30] from a series of participating nodes, while keeping them incognito.

An evidence chain  $\tilde{e}_i$  is derived from the generalized e-token systems, using dynamic attribute binding. As illustrated in Figure 5, every potential auditing node receives certain auditing token  $\tau_i$  from a credential authority before the DLA cluster is formed. The auditing token represents an unforgeable sanction

for  $p_{i+1}$  to engage in the logging and auditing services when it receives the invitation from the lastly joined DLA node  $p_i$ . When  $p_i$  and  $p_{i+1}$  concur to let  $p_{i+1}$  join the DLA cluster, a piece of unforgeable evidence will be created between them to support the claim.  $p_i$  can delegate its authority to  $p_{i+1}$  for inviting other nodes to join the DLA cluster. But once this is done,  $p_i$  can no longer invite other nodes to join the cluster. Doing so will subject to exposure of the true identity of  $p_i$  and its misconduct.

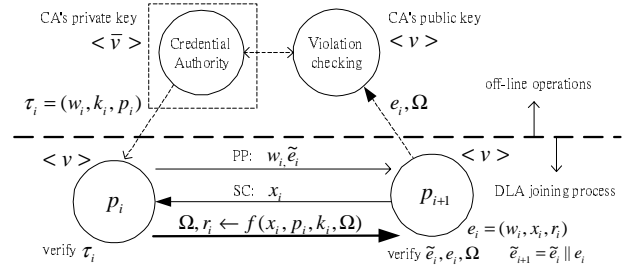


Figure 5. r-binding of the membership  $\Omega$  from newly joining DLA node.

The privileges and responsibilities of  $p_{i+1}$  to serve in the DLA cluster would be negotiated with  $p_i$ . Once they are settled, these logging and auditing attributes become a new part of the DLA service policies. The membership and service terms  $\Omega$  can be bound into the new piece of evidence  $e_i$  between  $p_i$  and  $p_{i+1}$  using the  $r$ -binding techniques [30].

During their negotiation process, the true identities of  $p_i$  and  $p_{i+1}$  can stay anonymous yet verifiable. As depicted in Figure 5, it takes a three-way handshaking message exchange to create a new piece of evidence. In the first phase,  $p_i$  sends  $p_{i+1}$  a *policy proposal* (PP) as an invitation to join the DLA cluster. In the second phase,  $p_{i+1}$  acknowledges  $p_i$  by sending back a *service commitment* (SC), which is the list of services that  $p_{i+1}$  is willing to provide. In the third phase, when  $p_i$  passes  $p_{i+1}$  the new piece of evidence for proving the legitimate DLA membership of  $p_{i+1}$  and the one-time delegation from  $p_i$  to  $p_{i+1}$  for the new member invitation. It is shown in [30] that the evidence generated through these binding operations are unforgeable without the secret knowledge  $k_i$  granted by the credential authority.

## 7. DEGREE OF AUDITING CONFIDENTIALTY

The degree of auditing confidentiality of the whole DLA system is a composite result from the system settings of these three processes. Let  $h = |A|$  be the number of attributes in  $GL$ ,  $c = \min_c (B_1 \cup \dots \cup B_c \supseteq A)$  the minimum number of DLA nodes that can jointly support and cover all

attributes in  $A$ ,  $u$  the number of undefined attributes in  $A$ . An *undefined attribute* is an abstract attribute that is only meaningful to the application subsystem by private agreements. Thus increasing  $u$  makes it more difficult for the individual DLA nodes to infer any hints from the audit trail fragments. It comes up with a similar effect by increasing  $c$  for more DLA nodes are required to recover a complete audit trail.

The logging confidentiality of  $GL$  is defined as

$$\wp_{\log}(GL) = \frac{cu}{h}$$

The auditing confidentiality of the normalized auditing criteria  $Q = SQ_1 \wedge \dots \wedge SQ_t$  is defined as

$$\wp_{\text{audit}}(Q) = \frac{g+t-1}{s+t-1},$$

where  $s$  is the number of atomic auditing predicates,  $g$  the number of global auditing predicates,  $(t-1)$  the number of conjunctive predicates in  $Q$ .

The query confidentiality on  $GL$  based on the auditing criteria  $Q$  is defined as

$$\wp_{\text{query}}(Q, GL) = \wp_{\text{audit}}(Q) \times \wp_{\log}(GL)$$

The DLA confidentiality of the whole system is defined as the average of the query confidentiality for all queries.

$$\wp_{\text{DLA}}(GL) = \frac{\wp_{\text{query}}(Q, GL)}{n}$$

## 8. CONCLUSION

Confidential logging and auditing is suitable for distributed logging and auditing of group activities. Our scheme restrict 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.

## REFERENCES

- [1] Office of the Auditor-General Computer auditing: Background. 2001. <http://www.agsa.co.za/About%20%20us/Corporate%20Information/AGSA%20Information%20Systems%20Auditing/memo/Transversal%20Systems.doc>
- [2] T. F. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey." *Proceedings of the 11th National Computer Security Conference*, 1988.
- [3] W. Lee, S. Stolfo and K. Mok. *Mining Audit Data to Build Intrusion Detection Models*. 1998. <http://www1.cs.columbia.edu/~sal/JAM/PROJECT/HTML-papers/KDD98-audit/Paper/kdd98.html>
- [4] M. Roger and J. Goubault-Larrecq, "Log Auditing through Model Checking." *In Press, I. C. S., editor, Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pp. 220-236, 2001.
- [5] S. Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy*. Chalmers University of Technology: Goteborg, Sweden, 2000.
- [6] L. Strous, "Audit of Information Systems: The Need for Cooperation." *SOFSEM '98: Theory and Practice of Informatics*, 25th Conference on Current Trends in Theory and Practice of Informatics, LNCS 1521, pp. 264-272, 1998.
- [7] L. Jean Camp and J.D. Tygar, "Providing Auditing While Protecting Privacy." *The Information Society*, 10(1), pp. 59-72, 1998.
- [8] N. Szabo, "Confidential Auditing." <http://szabo.best.vwh.net/confidential.html>
- [9] M. Franklin and M. Yung, "Varieties of Secure Distributed Computing." *In Proceedings of Sequences II, Methods in Communications, Security and Computer Science*, pp. 392-417, 1996.
- [10] A. C. Yao, "Protocols for secure computations." *In Proceedings of 2nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 160-164, 1982.
- [11] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority." *In Proceedings of the 19th ACM Symposium on Theory of Computing*, pp. 218 - 229, 1987.
- [12] D. Chaum, C. Crepeau, and I. Damgard, "Multiparty Unconditionally Secure Protocols." *The 20th ACM Symposium on the Theory of Computing*, pp.11--19, 1988.
- [13] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation." *In Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC'88*, pp. 1-10, 1988.
- [14] D. Beaver, J. Feigenbaum, J. Kilian and P. Rogaway, "Security with Low Communication Overhead." *Crypto'90*, pp. 62-76, 1990.
- [15] T. Rabin and M. Ben-Or, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority." *In Proceedings of the 21st Symp. on the Theory of Computing*, pp 73—85, 1989.
- [16] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. *In Advances in Cryptology: CRYPTO '89*, pages 589-590, Berlin, Aug. 1990. Springer.
- [17] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin, "Efficient Multiparty Computations Secure Against an Adaptive Adversary." *Advances in Cryptology - EUROCRYPT '99*, LNCS 1592, pp. 311-326, 1999.
- [18] M. Hirt, U. Maurer, and B. Przydatek, "Efficient Secure Multi-Party Computation." *Advances in Cryptology - ASIACRYPT '00*, LNCS 1976, pp. 143-161, 2000.
- [19] R. Gennaro, M. Rabin, T. Rabin, "Simplified VSS and Fast-Track Multi-party Computations with Applications to Threshold Cryptography." *ACM PODC'98*, 1998.
- [20] C. Clifton, M. Kantarcioglu and J. Vaidya, "Tools for Privacy Preserving Distributed Data Mining." <http://www.acm.org/sigs/sigkdd/explorations/issue4-2/clifton.pdf>
- [21] S. C. Pohlig and M. E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance." *IEEE Trans. Inform. Theory IT-24*, pp. 106-110, 1978.
- [22] W. Du, Z. Zhan, "A Practical Approach to Solve Secure Multi-Party Computation Problems." *In Proceedings of New Security Paradigms Workshop*, 2002
- [23] R. Fagin, M. Naor and P. Winkler, "Comparing Information without Leaking It." *Communication of the ACM*, 39, pp. 77-85, 1996
- [24] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, 1999.
- [25] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics." *ACM Transactions on Information and System Security*, 2(2), pp. 159-176, 1999.
- [26] J. Benaloh and M. de Mare, "One Way Accumulators: A Decentralized Alternative to Digital Signatures." *Advances in Cryptology - Proceedings of Eurocrypt '93*, 1993.
- [27] M. T. Goodrich, and R. Tamassia and J. Hasic, "An Efficient Dynamic and Distributed Cryptographic Accumulator." *In Proceedings of Information Security Conference (ISC 2002)*, 2002
- [28] B. Neumann and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine*, 32(9), pp 33-38, 1994.
- [29] C. Kruegel, T. Toth and C. Kerer, "Decentralized Event Correlation for Intrusion Detection." *In International Conference on Information Security and Cryptology (ICISC)*, LNCS, 2001.
- [30] T.C. Lam and Jyh-Charn Liu "On the Evidence based Peer-to-Peer Resource Management in Distributed Computing System" Technical Report 2003-7-2, Department of Computer Science, Texas A & M University, 2003.