Implementing Chain of Custody Requirements in Database Audit Records for Forensic Purposes

Denys A. Flores*† and Arshad Jhumka*
*University of Warwick. Department of Computer Science. Coventry, United Kingdom
{d.flores-armas, h.a.jhumka}@warwick.ac.uk
†Escuela Politécnica Nacional. Departmento de Informática y Ciencias de la Computación (DICC). Quito, Ecuador
denys.flores@epn.edu.ec

***Abstract*—During forensic database investigations, audit records become a crucial evidential element; particularly, when certain events can be attributed to insider activity. However, traditional reactive forensic methods may not be suitable, urging the adoption of proactive approaches that can be used to ensure accountability through audit records whilst satisfying Chain of Custody (CoC) requirements for forensic purposes.**

In this paper, role segregation, evidence provenance, event timeliness and causality are considered as CoC requirements in order to implement a forensically ready architecture for the proactive generation, collection and preservation of database audit records that can be used as digital evidence for the investigation of insider activity. Our proposal implements triggers and stored procedures as forensic routines in order to build a vector-clock-based timeline for explaining causality in transactional events recorded in audit tables. We expect to encourage further work in the field of proactive digital forensics and forensic readiness; in particular, for justifying admissibility of audit records under CoC restrictions.

***Index Terms*—database forensics, proactive, chain of custody, admissibility, architecture, audit, trigger, stored procedure, role segregation, provenance, timeline, causality, vector clock.**

I. Introduction

Database Forensics allows investigating malicious activities performed by *trusted employees or insiders* who, motivated by financial gain, could misuse their privileged access [1] in order to disclose or contaminate [2] transactional databases. Since *audit records* may be considered legal evidence [3], *accountability and forensics* become crucial investigation elements for analysing and justifying insider behaviour [4].

However, there is a difficulty in considering audit records as legally relevant or admissible if the *lack of accountability and forensic features*, within the database environment, enables malicious insiders to cover up their activities, and eventually make them appear as authorised [5]. For instance, unauthorised payments were made by malicious employees of a public institution in Ecuador [1], who used privileged system credentials for making them look legitimate. Although evidence could have been retrieved from the database, its audit records were inconsistent as it was reported to be inadequate and vulnerable.

On the other hand, ensuring admissibility also requires forensic practitioners to establish an *unbroken accountability trail* in order to show 'due dilligence' when handling any form of data and records. This requirement is known as *Chain of Custody (CoC)*, which basically describes the 'evidence continuum', delivering proof of adequate handling, and justification of actions performed on any evidential item. Nonetheless, when investigating databases, initiating and maintaining CoC is difficult because, unlike *proactive forensics*, the generally accepted *reactive approach* [6] may not be able to properly analyse and justify insider actions.

First of all, *reactive database forensics* is comprised of bottom-up methods that adapt traditional digital forensics techniques for recovering scattered pieces of evidence in order to reconstruct the database state [7]. Examples of these methods are table-relationship analysis [8] and data file carving [9]. However, these methods either lack formalisation and scientific background [10], or may not be suitable for investigating databases [11]. As a consequence, ad hoc database investigations over rely on the practitioner's knowledge and expertise, leading to conjectures about insider behaviour since the only available evidence to fully explain such actions may be partially recovered or unavailable.

Alternatively, *the proactive approach* is an emerging top-down method which is based on the premise that databases per se were designed with *forensically ready features*, such as triggers [12], for auditing insider activities [10]. Hence, audit records can be generated, collected and preserved in order to draw conclusions based on more generalistic behavioural traces than those which may (or may not) be present within reactively recovered evidence. This research takes on this approach, introducing a proactive architecture for database forensics so that the generation, collection and preservation of audit records can be done under CoC restrictions.

In section II, *role segregation, evidence provenance, event timeliness and causality* are considered as *Chain of Custody (CoC) requirements* for the proactive investigation of databases. In section III, the previous requirements are implemented as *functional features of a distributed architecture* for the *generation, collection and preservation* of audit records that can be used to explain insider activity. *A vector clock mechanism* is implemented in a stored procedure for recording causality and timeliness every time *Data Manipulation Language (DML) events* are triggered. In section IV, experimental results are presented, regarding the construction of *DML event timeliness*, the relationship between *causality and provenance*, and the

[1]As reported in 2012 by local newspapers *Ecuador Inmediato [https://goo.gl/08KHsi]* and *El Comercio [https://goo.gl/aOCyBp]*.

architectural performance. Finally, related and future work along with conclusions are given in sections V and VI, respectively.

II. Chain of Custody Requirements in Proactive Database Forensics

Traditionally, digital forensics has been known as a scientific approach for the identification, collection, validation, preservation, and subsequent analysis of digital evidence [13]. This life cycle has a slight variation during *proactive investigations*, where evidence must be generated, collected and preserved before the analysis phase [14]. However, regardless the approach, Chain of Custody (CoC) must be initiated and maintained according to the generally accepted *4 principles of digital evidence* [2] [15]:

- *Principle 1:* No action taken by [any insider] should change [evidence].
- *Principle 2:* In circumstances where ... [accessing] original data [is required], ... [evidence must be provided] explaining the relevance and implications of [such] actions.
- *Principle 3:* An audit trail [or similar record] of all [events] ... should be [generated, collected] and preserved. An independent third party should be able to examine [those events] and achieve the same [conclusion].
- *Principle 4:* The person in charge of the investigation [must ensure the application of these principles].

When applying the proactive approach, accountability and forensics are important elements for investigating databases since audit records become digital evidence for explaining the occurrence of insertions, deletions and updates which, in the context of this article, are referred as *Data Manipulation Language (DML) events*. Hence, DML event attributability can be explained when tuples in audit tables capture changes in transactional data [16] with their corresponding actor (insider causing the event). However, in order to guarantee the applicability of these principles as established in *Principle 4*, Chain of Custody (CoC) must be initiated and maintained, considering that the generation, collection and preservation of audit records must be performed whilst transactional database operations are also being executed. This brings on the consideration of role segregation, evidence provenance, event timeliness and causality as CoC requirements so that reproducibility and verification of insider activity can be ensured within a forensically ready environment.

A. Separation of Concerns

As established by *Principle 1*, a clear functional separation of concerns [17] is required in order to prevent potential changes in audit records whilst avoiding overlapping functional responsibilities. Although the administrator role (DBA) is normally in charge of managing audit functions [18], a explicit *forensic role* and a corresponding *forensic database* should be created for preventing discretionary violations of administrative functions, such as disabling audit mechanisms on convenience [19].

[2]Where necessary, bracketed text denote paraphrasing for adapting the concept of the principles to the article's context.

Definition II.1. ***Role Segregation***

Let S be the set of database users, A and F the set of administrator and forensic roles, respectively:

$$segregation = \{usr, role | usr \in S \wedge role \notin A \cap F\}$$

The function segregation prevents a database user having administrator and forensic permissions at the same time.

By placing transactional event accountability, and controlling access to audit functionally [3], Def. II.1 follows *Principle 2*, allowing monitoring insider actions in order to justify that audit records were produced without negligent insider intervention.

B. Evidence Provenance

Principle 3 states that audit records should reflect a trail of events in order to ensure third-party verification; specially, after an insider security violation [3]. Thus, provenance becomes a CoC requirement during the generation of audit records, allowing investigators to relate DML events with their actors.

Definition II.2. ***Provenance***

Let provenance be a 6-attribute tuple representing the description level of audit records:

$$provenance = \{p_n : n \in \mathbb{N}^+ \wedge 1 \leq n \leq 6\}$$

In Table I, the required granularity provenance description level [20] on audit records is described in order to explain DML events:

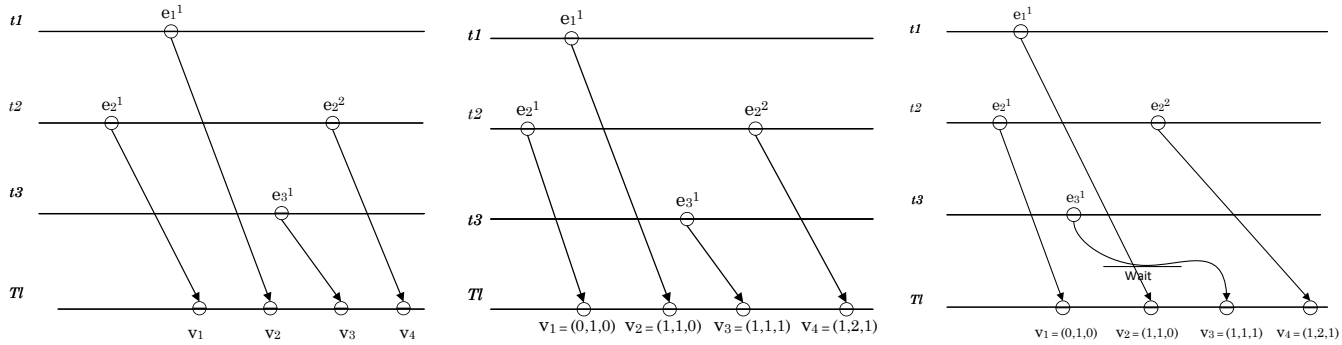
TABLE I: The 6-Attribute Provenance Tuple

Attr.	Value	Description
p_1	**What** audit record has been generated.	*Audit record identifier (Id)*
p_2	**When** the audit record was generated.	*Real (Hardware) Clock Timestamp.*
p_3	**Why** the audit record was generated.	*Type of DML Event: Insert, Update or Delete.*
p_4	**Who** the actor is.	*User identifier.*
p_5	**Which** the DB actor role is.	*Type of DB role: DBadmin, DBforensics, DBuser.*
p_6	**Where** the DML event was triggered.	*Originating IP Address.*

C. Event Timeliness

Principle 3 implies considering timeliness as an important CoC requirement in order to monitor insider behaviour by applying time constraints to audit records. Subsequently, building a *timeline of DML events* can be useful not only to explain their global ordering in the entire computation, but also to keep an audit trail during their generation, collection and preservation:

a. *Generating Audit Records:* Audit records are generated every time a DML event is 'triggered' in a transactional table. At the same time, provenance of DML events is also possible to capture by retrieving DML event-related attributes (Table I) during the generation of such records.



(a) A timeline T_l receives DML event-related audit records e_i^j from each audit table t_i.

(b) Representation of Vector Clock values v_i in T_l as received from each audit table t_i.

(c) A subsequent audit record e_3^1 must 'wait' for a previous record e_1^1 to be registered in T_l.

Fig. 1: Global Ordering Representation, where T_l represents a timeline constructed by an ideal external observer of the computation.

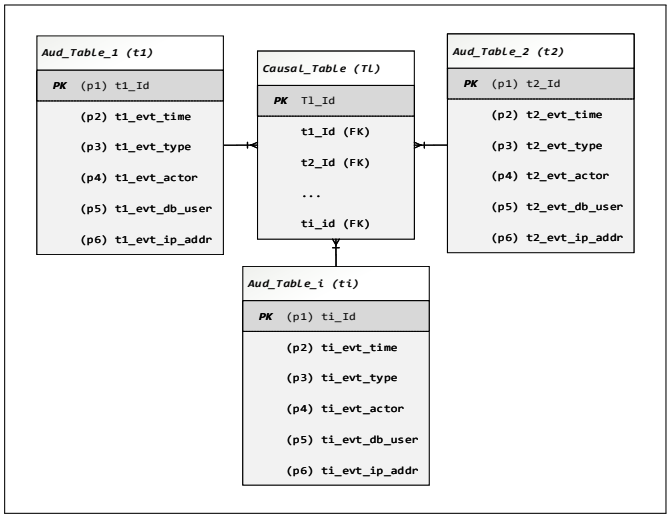


Fig. 2: A de-normalised forensic database collects DML events in tables t_i, and constructs a causal timeline in table T_l.

b. *Collecting Audit Records:* When a DML event is triggered in a transactional table, its corresponding *6-attribute audit table* t_i in the forensic database (Fig. 2) collects its audit record, along with its provenance attributes (Def. II.2).

c. *Preserving Audit Records:* Whilst audit records are generated and collected in audit tables, their temporal occurrence can be preserved and ordered in a timeline using *causal audit records* stored in a causal table T_l within the forensic database (Fig. 2).

By using *causal audit records* preserved in the causal table not only losing sequentiality of DML events can be prevented, but also their temporal occurrence can be explained. For example, if an insider *inserts* data in a transactional table, a DML event is triggered. A corresponding audit record is generated along with its provenance attributes, which is later collected inside an audit table, and finally, the corresponding causal audit record is preserved in the causal table. In Fig. 1a, these interactions are represented from the perspective of an *ideal external observer* [21] who is in charge of receiving, recording and ordering DML event-related audit records e_i^j in a causal table T_l. Then, a timeline is built using causal audit records per each DML event-related audit record generated and collected in a finite number of audit tables t_i. In the following definitions, these event timeliness characteristics are formalised for better understanding:

Definition II.3. ***Evidence Sources***
Let D be the set of i evidence sources, then:

$$D = \{t_i : i \in \mathbb{N}^+\}$$

where t_i is the i^{th} audit table considered as evidence source.

Definition II.4. ***Evidential Events***
Following from Def.II.3, let E be a set of audit records registering DML events in their corresponding audit tables, then:

$$E = \{e_i^j : i, j \in \mathbb{N}^+\} \tag{1}$$

where e_i^j is the j^{th} audit record generated by the i^{th} audit table.

Each audit record e_i^j in t_i is denoted using canonical enumeration, where j denotes the j^{th} audit record generated in the i^{th} audit table. For example, some audit records in the 1^{st} and 4^{th} audit tables can be identified as:

$$e_1^1, e_1^2, e_1^3, \ldots$$
$$e_4^5, e_4^6, \ldots, e_4^{10}, \ldots$$

From the vector clock definition in [21], in Fig. 1b, the *logical order of audit records* generated in an audit table t_i and recorded into a timeline T_l, is a *vector clock mechanism* which is used to track audit record order values, making them causally consistent. I.e., a vector clock is a simple logical order

of DML event occurrence represented by the *Cartesian power* V^n of the corresponding audit record timestamps v_i:

$$V^n = \{(v_1, v_2, v_3, \ldots, v_n) | v_i \in \mathbb{N}, i \in \mathbb{N}^+\} \quad (2)$$

From (1) and (2), timestamps v_i can be expressed in terms of audit records e_i^j in order to explicitly identify the audit table t_i which they belong to:

$$v_i = T_s(e)[i] \quad (3)$$

The index i is sufficient to represent the n-tuple of the vector clock V^n since an audit table t_i must be identified for assigning its corresponding timestamp T_s. Whereas the index j becomes irrelevant as it denotes local ordering of the audit record e in its corresponding audit table t_i.

Definition II.5. ***Event Timestamps***
For building a DML event timeline T_l as shown in Fig. 1b, and following from (3) in Def.II.4, the timestamp T_s of an audit record e registered in the i^{th} audit table can be defined as follows:

$$T_s(e)[i] := \begin{cases} \textit{(a)} \\ T_s[i] + 1, & \textit{if } e = send_{ti}(T_s) \\ \\ \textit{(b)} \\ max\{T_s[i], send_{ti}(T_s)\}, & \textit{if } e = receive_{ti}(T_s) \end{cases}$$

Where:

(a) if an audit record e is being generated or 'sent', then the local vector clock component in its audit table t_i is incremented.
(b) if an audit record e is being 'received' for registration in the timeline T_l, then the value of the reporting component of the vector clock in T_l is updated to the maximum value, obtained by comparing the corresponding previous vector clock value in T_l with the received timestamp T_s from the corresponding audit table t_i.

Definition II.6. ***Causal Audit Record***
Following from Def. II.5, in order to represent the n-tuple vector clock V^n in Def. II.4 (2), let a causal audit record, in a causal table T_l, be an array of timestamps $T_s(e)[i]$, recording the occurrence of a DML event e collected in its corresponding audit table t_i:

$$record : t_i \mapsto T_s(e)[i]$$

Definition II.7. ***Event Timeline***
Following from Def. II.6, let T_l be a sequence of records, representing the global occurrence of DML events in their corresponding audit records e, considering the timestamps T_s, as reported by each audit table t_i:

$$T_l = [(t_i, T_s(e)[i]) \mid t_i \in D, e \in E]$$

Hence, the timestamp values of each component of the vector clock can be registered on the timeline T_l and retrieved when required.

D. Event Causality

Form the forensic point of view, applying *Principle 3* not only enables the construction of a timeline (section II-C), but also allows sequencing DML events in order to identify and explain their interactions [22]. Likewise, for accountability purposes, timeliness allows the generation and collection of audit records with time restrictions, so they can be stored and reviewed in later investigation stages [3]. As shown in Def. II.7, the global history of audit records in a timeline initiates and maintains CoC requirements by introducing an element of causality [23] for explaining the sequential relationship or their corresponding DML events. Since audit records in databases are strictly bound to timestamps, sequencing them requires establishing a 'happen-before' relation (represented by $\rightarrow$) with a strong timestamp condition [21][24].

Definition II.8. ***Event Sequentiality Property***
Being e_a (sending) and e_b (receiving), two sequential DML events recorded in their corresponding audit records; then, the timestamp T_s of e_a must be less than the timestamp value $T_s(e_b)[a]$ of the vector clock corresponding to the receiving DML event e_b:

$$\forall e_a, e_b \in E \bullet (e_a \rightarrow e_b \Rightarrow send_{ta}(T_s) < T_s(e_b)[a])$$

Definition II.9. ***Event Transitive Property***
Similarly, by transitivity, having three evidential DML events registered in their corresponding audit records e_a, e_b, and e_c, if e_a 'happens before' e_b, and e_b 'happens before' e_c, then e_a precedes e_c, and the timestamp T_s of the sending event e_a is less than the timestamp value $T_s(e_c)[a]$ of the vector clock corresponding to the receiving event e_c:

$$\forall e_a, e_b, e_c \in E \bullet$$
$$(e_a \rightarrow e_b \wedge e_b \rightarrow e_c \Rightarrow e_a \rightarrow e_c \wedge send_{ta}(T_s) < T_s(e_c)[a])$$

Definition II.10. ***Event Concurrency Property***
Since concurrency explains the occurrence of DML events that are not affected by a "happen-before" relation [24], they can be defined in a more general perspective as they are not restricted by a timestamp condition.

Given two DML events e and e', if they are not sequential with each other then they are concurrent:

$$\forall e, e' \in E \bullet (e || e' \Rightarrow \neg(e \rightarrow e' \wedge e' \rightarrow e))$$

From the transitive property in Def. II.9 and its implication in concurrent events (Def. II.10), one can infer that an ideal external observer must be "informed" of the existence of an intermediate event e_b [21] as concurrent events are not bound to timestamp restrictions. However, determining whether or not such an event actually 'happened before' a receiving event e_c is a concurrency challenge for building the timeline T_l. This requires the introduction of an asynchronous method for preventing inconsistent observations, and therefore make an intermediate event 'wait' if an ongoing event has not been registered yet (Fig. 1c). The solution to this problem is explained later in section III-C3.

III. Implementing a Distributed Environment for Proactive Database Forensics

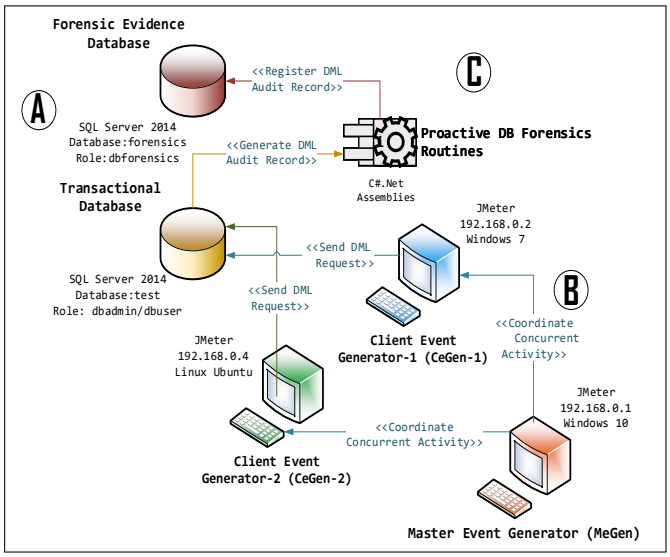


Fig. 3: Experimental Architecture

In a proactive forensics approach audit records must be generated, collected and preserved within a forensically ready environment. However, for making them admissible, the *Chain of Custody (CoC) requirements* explained in section II must be considered as functional specifications. In Fig. 3, an experimental architecture is outlined in order to proactively generate and collect audit records, and at the same time, preserve a timeline of their occurrence (sections II-C and II-D). In the following sections, these architectural components are explained in detail.

A. Separation of Concerns

For achieving *role segregation* (section II-A), separation of concerns is implemented by using a transactional and a forensics database (Fig. 3-A). Whilst the transactional database (and its corresponding roles) is in charge of transactional operation and administration, the forensics database with a explicit forensic role deploys the *forensic routines* for the generation, collection and generation of DML event-related audit records (section II-C, lit. a - c). As a result, *an event timeline* can be created based on the causal relation of sequential events with timestamp restrictions, as explained in section II-D.

B. Concurrent DML Event Generation

For experimental purposes, in order to generate concurrent events, synthetic workload is produced by means of a *Master Event Generation Server (MeGen)* and two *Event Generation Clients (CeGen)* (Fig. 3-B). *MeGen* is a master terminal coordinating concurrent activity using *threaded database connections* to emulate transactional behaviour in a distributed environment. Meanwhile, *CeGen* is comprised of two slave terminals which are in charge of passing concurrent *DML requests* from *MeGen* to the transactional database.

C. Proactive Database Forensics Routines

Triggers and stored procedures are implemented as external forensic routines (Fig. 4) with explicit enable/disable permissions assigned to a specific forensic database role (section II-A). This prevents them to be easily accessed, or conveniently disabled by malicious insiders with administrator privileges. Additionally, *abstraction* can be provided by obscuring their implementation particularities from normal database roles, achieving access control for ensuring Chain of Custody compliance during the generation, collection and preservation of audit records:

1) Evidence Generation - Fig. 4-1: Every time DML requests are sent from *CeGen* to the transactional database, a corresponding *evidence generation trigger* in a receiving table is executed. These triggers not only generate audit records, but also automatically capture specific *provenance descriptive attributes* (section II-B) related to the occurrence of a DML event in the transactional database.

2) Evidence Collection - Fig. 4-2: Data tables within the transactional database have their corresponding audit tables in the forensic database (Fig. 2). Then, *evidence generation triggers* in the transactional database execute *evidence collection stored procedures* in the forensic database for storing audit records and their provenance descriptive attributes in the audit tables (Def. II.2).

3) Evidence Preservation - Fig. 4-3: A *causal table* in the forensic database (Fig. 2) is used to build an event timeline T_l (Def. II.7). Whilst audit records are generated and collected in the audit tables t_i, their corresponding *evidence preservation triggers* execute an *evidence preservation stored procedure* in order to create a *causal audit record*, assigning timestamp values T_s to build a timeline T_l in the *causal table*. The implementation of this stored procedure, following the *vector clock mechanism specification* in Def. II.5, is shown in the following pseudo code (List. 1):

```
#begin causalStoredProcedure
public static void spLogCausalEvent (timestamp Ts,
     identifier ti)}
  #begin serialisedTransaction
    /*1. Get the number of audit tables ti*/
    numTables := audit_tables.getNumber();
    /*2. Get the timeline Tl of events*/
    timeline := event_records.getSequence();
    /*3. If there are no events reported*/
    if timeline.getRecords() = [] then
      tbl_index := 0;
      /*3.1 Timestamp Ts is assigned to the
    sending table ti*/
      while tbl_index < numTables do
        if ti = tbl_index then
          event_record[tbl_index] := Ts;
        else
          event_record[tbl_index] := 0;
        end if
      end while
    /*4. If at least one event has been reported*/
    else
      /*4.1 Retrieve last event record*/
```

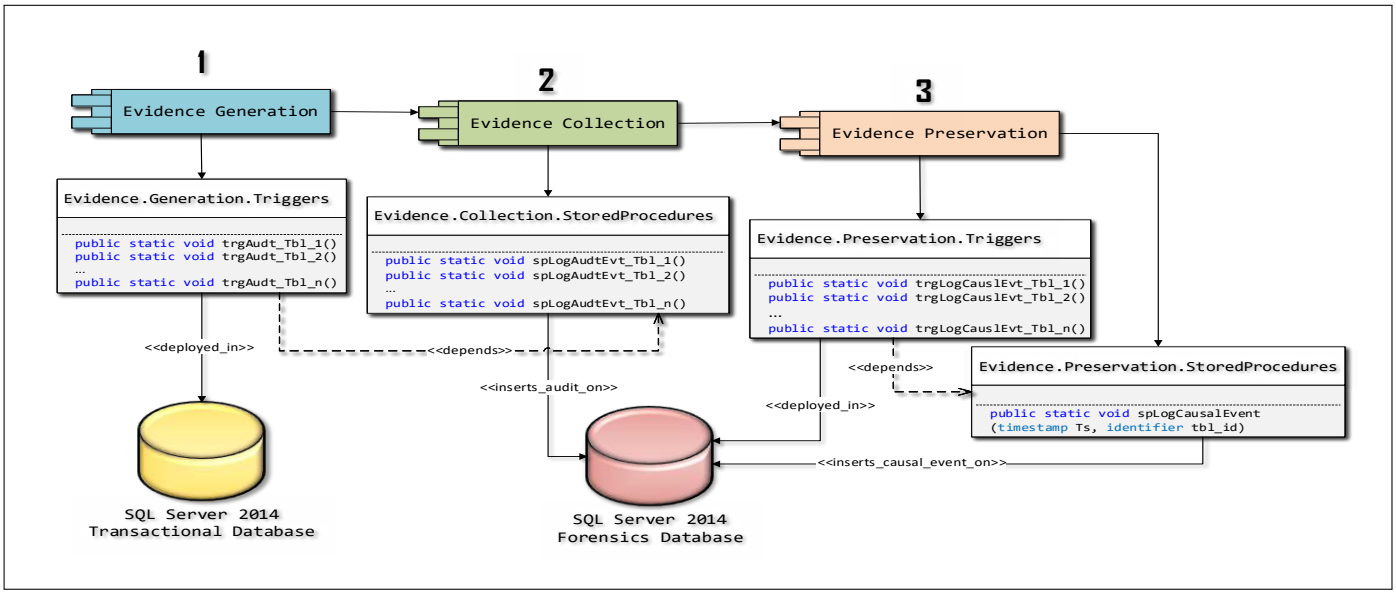


Fig. 4: Architecture of Proactive DB Forensic Components implemented as CLR C# Static Methods

```
22        event_record := timeline.getLastRecord();
23        tbl_index := 0;
24        /*4.2 Assign timestamps to receiving events
      */
25        while tbl_index < numTables do
26        /*4.3 The sending timestamp Ts is compared
      with its previous Ts value*/
27          if ti = tbl_index then
28            /*4.4 The max Ts value is assigned to
      the corresponding sending table ti*/
29            event_record[ti] := max(event_record[
      tbl_index], Ts);
30          else
31            /*4.5 Previous timestamp values of non-
      sending tables are maintained*/
32            event_record[tbl_index] := event_record[
      tbl_index];
33          end if
34        end while
35      end if
36      /*5. Add reported event to the timeline*/
37      timeline.add(event_record);
38      #commit serialisedTransaction
39  }#end causalStoredProcedure
```

Listing 1: Causal Event Registration Pseudo code

Since the construction of a timeline T_l must be accurate with no 'events lost' due concurrent user activity (Def. II.10), in List. 1, causal event registration is executed using *serialised transactions*. This prevents the occurrence of *concurrent intermediate events* (Fig. 1c) which, due to incorrect transitivity (Def. II.9), may be recorded before an ongoing sending event is received.

IV. Experimental Results

The technical specifications of the *forensically ready environment* depicted on Fig. 3 are briefly explained as follows:

a. *A transactional and a forensics database* (Fig. 3-A) were implemented in MSSQL Server 2014 with operative (DBUser), administrative (DBAdmin) and forensic (DBForenics) roles enabled.

b. For concurrent DML event generation, both *MeGen* and *CeGen* implement JMeter [3] in master-slave mode, respectively (Fig. 3-B).

c. *Proactive DB forensic routines* (Fig. 3-C) are implemented using *Common Language Runtime (CLR) C# Assemblies*, and deployed in their respective databases, following the deployment architecture shown in Fig. 4.

720 concurrent DML request samples were modelled and executed in *MeGen* and *CeGen*, and captured using the forensic routines. This allowed analysing timeliness, the relationship amongst causality and provenance, and finally, measuring the architectural performance.

TABLE II: Vector Clock Components in the Causal Table T_l

seq	$T_s[1]$	$T_s[2]$	$T_s[3]$	$R_{Ts}[seq]$
...	...	...	...	...
17	5	5	[7]	2017-03-06 16:51:49.000
18	[6]	5	7	2017-03-06 16:51:49.263
19	6	[6]	7	2017-03-06 16:51:49.283
...	...	...	...	...
24	[7]	6	11	2017-03-06 16:51:49.360
25	7	6	[12]	2017-03-06 16:51:49.367
26	7	[7]	12	2017-03-06 16:51:49.370

Sequences of events recorded in audit table t_1
Sequences of events recorded in audit table t_2
Sequences of events recorded in audit table t_3

[3] *JMeter official site* explains its deployment in master-salve mode [https://goo.gl/qK1tCt].

A. Analysing Timeliness

In Table II, a sample of captured DML event sequences shows the timestamps $T_s[i]$ assigned to them in the causal table T_l. The causal relationship amongst the 17th, 18th and 19th sequence can be proved using Def. II.9:

$$\text{Being } send_{t3}(T_s[2]) := 5 \wedge T_s(e_{19})[2] := 6$$
$$\text{As } 5 < 6 \text{ (true)} \Rightarrow e_{17} \rightarrow e_{19}$$
$$\therefore$$
$$e_3^7 \rightarrow e_1^6 \wedge e_1^6 \rightarrow e_2^6 \text{ by transitivity.}$$

If *real clock timestamps* $R_{Ts}[seq]$ are assigned, then:

$$R_{Ts}[17] < R_{Ts}[18] \wedge R_{Ts}[18] < R_{Ts}[19]$$

This proves that using *hardware and logical clock timestamps* is equivalent since the causal relationship between the 17th, 18th and 19th sequence remains. This is very useful considering that auditors and forensic practitioners usually rely on timestamps associated with hardware clock values for explaining DML event sequencing; i.e., date and time of a particular DML event. However, if these values are tampered with by a malicious insider, audit record integrity can be compromised. Also, if hardware clocks were used, they have to be synchronised which is transactionally expensive if the database is geographically distributed.

B. Relation of Provenance and Causality

Using the causal timestamps $T_s[i]$ as conditions, *provenance descriptive attributes* can be queried on their corresponding audit tables t_i as shown in List. 2:

```
use forensics;
select  t1.t1_Id as 'Ts[t1]', t1.t1_evt_time,
  t1.t1_evt_type, t1.t1_evt_actor,
  t1.t1_evt_db_user, t1.t1_evt_ip_addr
from dbo.t1 where t1.Evt_ri_us = 7;
select  t3.Evt_ri_tr as 'Ts[t3]', t3.t3_evt_time,
  t3.t3_evt_type, t3.t3_evt_actor,
  t3.t3_evt_db_user, t3.t3_evt_ip_addr
from dbo.t3 where t3.Evt_ri_tr = 12;
select  t2.Evt_ri_py as 'Ts[t3]', t2.t2_evt_time,
  t2.t2_evt_type, t2.t2_evt_actor,
  t2.t2_evt_db_user, t2.t2_evt_ip_addr
from dbo.t2 where t2.Evt_ri_py = 7;
```

Listing 2: Provenance Queries on Audit Tables

TABLE III: Querying Provenance Descriptive Attributes

Ts	Time	Type	Actor ID	DB User	IP Addr
[7]	16:51:49.360	Insert	1705997013	DBUser	192.168.0.2
[12]	16:51:49.363	Insert	1705997013	DBAdmin	192.168.0.4
[7]	16:51:49.367	Insert	1705997013	DBUser	192.168.0.2

Description of event e_7 recorded in audit table t_1
Description of event e_{12} recorded in audit table t_2
Description of event e_7 recorded in audit table t_3

Table III shows the resulting provenance queries, providing a fine grained description of the 24th, 25th, and 26th event recorded in the causal table T_l. If a DBUser role has been assigned to *external application users* for interacting with the transactional database, the *provenance attributes* can detect misuse, for example when the DBAdmin has performed an insertion with the same Actor ID as the one used by a DBUser.

C. Measuring Architectural Performance

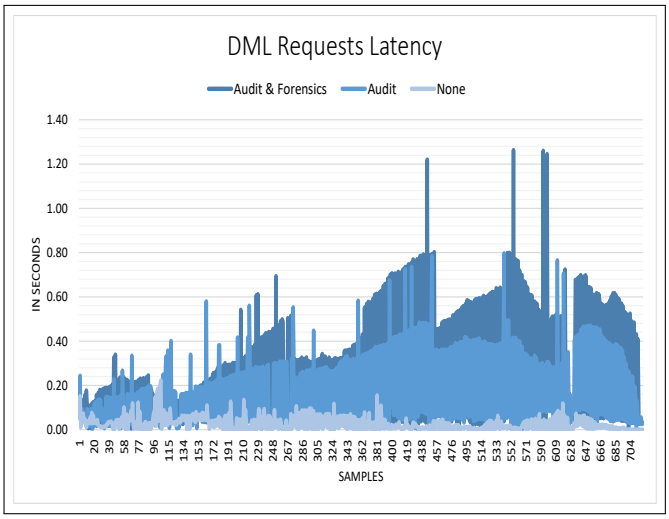


Fig. 5: Latency Graph of DML Requests

Fig. 5 shows *three stress test scenarios* (Table IV) used to measure the *database response latency* per each one of the 720 DML Request samples: Despite having some latency, the values

TABLE IV: DML Requests Latency Results (Seconds)

	Maximum	Minimum	Average
Audit & Forensics	1.263	0.005	0.24404
Only Audit	0.797	0.003	0.17255
None	0.223	0.000	0.02750

in Table IV are acceptable for a concurrent scenario, where using rudimentary audit features may even increase latency beyond the shown thresholds. For avoiding adding throughput on the database, our *featured audit and forensic components* (Fig. 4) perform serialised operations in relays so that a causal record can be produced only after its corresponding audit event has been recorded, adding 3 to 5 ms. of latency, when enabling 'only audit' and 'audit & forensics' features, respectively.

V. Related and Future Work

An early attempt to capture a notion of timeliness was developed in [25]. Later, in [23], research on separation of concerns in NoSQL databases was conducted whilst the usage of de-normalised tables for handling evidence was introduced in [26]. On the contrary, despite not being strictly related to databases, [27] recently introduced timeline construction based on audit trails. Our research considers these contributions in order to formalise *Chain of Custody requirements*, and implement a forensically ready architecture for the generation, collection and preservation of database audit records. Our

findings have established a relationship between provenance and causality for databases, also inspired by the approach used in [28]. Future work is expected to be developed for capturing provenance during the occurrence of SELECT events, which cannot be done using database triggers.

VI. Conclusions

For justifying Chain of Custody (CoC) requirements in proactive database forensics, role segregation, provenance, timeliness and causality must be captured within a forensically ready architecture.

Regarding *role segregation*, although trigger auditing functionality can be used, conventional SQL implementations cannot prevent them to be disabled by malicious or negligent insiders. We use an explicit forensic database role to deploy CLR C# triggers and stored procedures, obscuring implementation details and restricting their access from operational and administrative database roles (Section III-C).

With regard to *provenance*, the implemented forensic database uses de-normalised audit tables to capture provenance descriptive attributes (Section II-B) along with audit records. This enables capturing time and type of DML events along with information about their actors.

Finally, *timeliness and causality* are mutually related CoC requirements because one cannot be explained without the other. Although auditors and forensic investigators may rely in hardware clock timestamps for explaining DML event sequentiality, we have proved that vector clock logic timestamps are equivalent to hardware clock timestamps. Thus, DML events become independent of hardware clock failures and manipulations because their occurrence real time becomes a descriptive provenance attribute rather than an element for explaining their causality.

Acknowledgements

This research was funded by the Secretariat of Higher Education, Science, Technology and Innovation of the Republic of Ecuador. We also thank Warwick Cyber Security GRP for partially funding our research. A special thanks to our colleague Matthew Bradbury for his valuable comments towards enhancing this work. A special recognition to Bolívar Palán who was, and still is, a dear source of inspiration and example.

References

[1] G. B. Saathoff, T. Nold, and C. P. Holstege, "Chapter 3 - we have met the enemy and they are us: Insider threat and its challenge to national security," in *Strategic Intelligence Management*, B. Akhgar and S. Yates, Eds. Butterworth-Heinemann, 2013, pp. 24–35.

[2] D. A. Flores, F. Qazi, and A. Jhumka, "Bring your own disclosure: Analysing byod threats to corporate information," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 1008–1015.

[3] B. Guttman and E. A. Roback, *An Introduction to Computer Security : NIST SP 800-12*. Gaithersburg, US: National Institute of Standards and Technology, 1995.

[4] D. Takahashi and Y. Xiao, "Retrieving knowledge from auditing logfiles for computer and network forensics and accountability," *Security and Communication Networks*, vol. 1, no. 2, pp. 147–160, 2008.

[5] F. Cohen, "Forensic methods for detecting insider turning behaviors," in *2012 IEEE Symposium on Security and Privacy Workshops*, 2012, pp. 150–158.

[6] S. Alharbi, J. Weber-Jahnke, and I. Traore, "The proactive and reactive digital forensics investigation process: A systematic literature review," *International Journal of Security and Its Applications*, vol. 5, no. 4, pp. 59–72, 2011.

[7] O. M. Fasan and M. S. Olivier, "On dimensions of reconstruction in database forensics," in *Seventh International Annual Workshop on Digital Forensics and Incident Analysis (WDFIA)*, Hersonissos, Crete, Greece, 2012, pp. 97–106.

[8] D. Lee, J. Choi, and S. Lee, "Database forensic investigation based on table relationship analysis techniques," in *2nd International Conference on Computer Science and its Applications*, 2009, pp. 1–5.

[9] J. Wagner, A. Rasin, and J. Grier, "Database forensic analysis through internal structure carving," *Digital Investigation*, vol. 14, no. 1, pp. S106–S115, 2015.

[10] W. K. Hauger and M. S. Olivier, "The state of database forensic research," in *Proceedings of the Information Security for South Africa (ISSA) Conference*. IEEE, 2015, pp. 1–8.

[11] K. E. Pavlou and R. T. Snodgrass, "Generalizing database forensics," *ACM Transactions on Database Systems*, vol. 38, no. 2, pp. 12–12:43, 2013.

[12] W. K. Hauger and M. S. Olivier, "The role of triggers in database forensics," in *2014 Information Security for South Africa*, 2014, pp. 1–7.

[13] G. Palmer, "A road map for digital forensic research," *DFRWS Technical Report*, 2001, [Accessed 03 March 2017]. [Online]. Available: http://bit.ly/28YEaXP

[14] A. Al-Dhaqm, S. A. Razak, S. H. Othman, A. Nagdi, and A. Ali, "A generic database forensic investigation process model," *Jurnal Teknologi*, vol. 78, no. 6-11, pp. 45–57, 2016.

[15] Association of Chief Police Officers. (2012) Good Practice Guide for Digital Evidence. [Accessed 03 March 2017]. [Online]. Available: https://goo.gl/UUHFwQ

[16] W. Lu, G. Miklau, and N. Immerman, "Auditing a database under retention policies," *The VLDB Journal*, vol. 22, no. 2, pp. 203–228, Apr. 2013.

[17] L. Liu and Q. Huang, "A framework for database auditing," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, 2009, pp. 982–986.

[18] Y. A. Rathod, M. B. Chaudhari, and G. B. Jethava, "Database intrusion detection by transaction signature," in *Computing Communication Networking Technologies (ICCCNT), 2012 Third International Conference on*, 2012, pp. 1–5.

[19] Q. Huang and L. Liu, "A logging scheme for database audit," in *2009 Second International Workshop on Computer Science and Engineering*, vol. 2, 2009, pp. 390–393.

[20] A. Rani and S. Thalia, "Knowledge driven decision support system for provenance models in relational database," in *2014 International Conference on Data Science Engineering (ICDSE)*, 2014, pp. 68–75.

[21] O. Babaoğlu and K. Marzullo, in *Distributed Systems (2nd Ed.)*, S. Mullender, Ed. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1993, ch. Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms, pp. 55–96.

[22] E. Casey, *Handbook of Digital Forensics and Investigation*. Elsevier, 2010.

[23] P. Bailis, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "Bolt-on causal consistency," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 761–772.

[24] S. Vömel and F. C. Freiling, "Correctness, atomicity, and integrity: Defining criteria for forensically-sound memory acquisition," *Digital Investigation*, vol. 9, no. 2, pp. 125 – 137, 2012.

[25] G. Bhargava and S. K. Gadia, "Relational database systems with zero information loss," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 1, pp. 76–87, 1993.

[26] Y. Zhang and Y. Lin, "Research on the key technology of secure computer forensics," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, 2010, pp. 649–652.

[27] W. Halboob, R. Mahmod, M. Abulaish, H. Abbas, and K. Saleem, "Data warehousing based computer forensics investigation framework," in *2015 12th International Conference on Information Technology - New Generations*, 2015, pp. 163–168.

[28] M. Vieira and H. Madeira, "Detection of malicious transactions in dbms," in *11th Pacific Rim International Symposium on Dependable Computing (PRDC'05)*, 2005, pp. 8–15.