A Blockchain Approach for Data Transparency in a Relational Database System

by

Mohammadamin Beirami

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

in

Computer Science

University of Ontario Institute of Technology

Supervisors: Dr. Ken Q. Pu and Dr. Ying Zhu

August 2018

Abstract

This thesis deals with the development of a framework based on Blockchain technology that is implemented on top of a Relational Database Management System and makes it extremely difficult for the privileged or unprivileged users of the system to conceal their activities on a relational database. We present a mechanism to audit the activities of the users, verify the validness of submitted transactions on the database and create a Blockchain out of submitted transactions. We also present a practical solution to handle large query workloads on the temporal audit tables. By implementing this framework, not only proof of work is available, but also malicious activities such as intentional or unintentional fake data manipulation could be discovered and reported to the moderator of the system.

Acknowledgements

I would like to thank everybody

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Motivation

Today, data is seen as the lifeblood of organizations that is helping them to make strategic decisions more efficient and perform the operations faster. Organizations store their highly confidential data, such as financial documents, customer information, medical records and more in the form of data records in a database and later, use them for their sensitive operations. However, in our current connected era, cyber-security is one of the biggest challenges of organizations. Databases, could be accessible on the public networks where adversaries utilize hacking techniques to manipulate the data that are stored in them. Offline databases also are not safe as the attacks might be carried out within the organization by privileged users of the system. The fake data that are the result of such malicious activities, if not identified on the database, may result in irreversible consequences. As a result, organizations and businesses spend a considerable amount of money to utilize cyber-security techniques in order to make their stored data safe and reliable.

Traditionally, malicious activities on a system including fake data manipulation to the database is prevented by restricting the activities of the users on the system. Also cryptographic techniques such as data encryption or electronically signing the data has proven effective in identifying fake data manipulation. This is primarily done by assigning a pair of cryptographic keys to the users, by which they can securely encrypt or decrypt the data and submit it to the database. Fabricating cryptographic keys is computationally infeasible, hence it is extremely difficult for an outsiders to manipulate data inside the databases without notice. The downside of these techniques is that it requires to fully trust the activities of authenticated users which in turn, brings up a lot of security concerns. Also with ever-increasing complexity of cyber-criminal techniques, each day a new approach to penetrate the database systems is identified. Hence, it is naive to assume that access restriction or cryptographic techniques alone could solve the issue.

Therefore a system is needed to confirm the reliability of data based on verifiable evidences and not by relying on trust. This requires that the transactions on the database system to be transparent, meaning that for a record in a database throughout its life-cycle, it should be evident that its data has always been generated and modified by official sources. To achieve this, the system not only should be able to identify and report the data generated from unrecognized sources but also it should be able to show the proof of work done by privileged users. By providing proof of work, all users who interfered with data manipulation in a database system are identified and their activity information is reported.

In this thesis we have developed a system which ensures transparency of activities in a relational database system. Our developed system identifies and reports any malicious data manipulation by outsiders and provides proof of work by referring to the history of transactions for any records stored in the database. History of records are temper-proof and is protected by cryptographic techniques. We also developed a mechanism to handle large query workloads on the historical records. There were four main challenges that needed to be addressed while developing the system: Auditing the transactions on the database system, handling large query workloads on the audit tables, verifying the validness of query results and making the audit tables to be temper-proof.

1.2 Related Work

1.3 Problem Definition

Given a relational database D, let r to be the relational table in D. Denote attributes of r as $attr(r)$. Assume $attr(r) = [id, m, u, sig(m|u)]$ where id is the id of records in the database, $m = [col_1, ..., col_n]$ is n number of data columns in r, u is the information of the user who submitted the record and $sig(m|u)$ is the digital signature of the record submitted by u. Also let D^T be the temporal database in which the history of the records in D is stored. we denote r^T as the tables in D^T. Assume that the attributes of r^T are $attr(r^T) = [id, m, u, sig(m|u), t, d]$ where t and d are the timestamps in which the record was created/updated or deleted respectively.

A submitted transaction is said to violate transparency, hence untrustworthy in any of the following scenarios:

Scenario 1. Let q be the result of the query $q = \sigma_{(id)}(r)$, which is the record submitted by the user u. The result of query is untrustworthy if $\{q[sig(m|u)] : sig(m|u) \in r\} \neq sig(\{q[m] : m \in r\}|u)$. That is, by digitally signing the record m with the u's cryptographic keys, we get a different result than the submitted signature to the table. The reason that this scenario may happen is that:

- The record was altered accidentally or maliciously.
- A user maliciously claims to be one of the privileged users of the system with fake credentials.

Scenario 2. Let q be the result of query $q = \sigma_{(id)}(r)$. The result of query is untrustworthy if $q[u \vee sig(m|u)] = \emptyset$. In other words, the resulted record was submitted by an anonymous user to the database.

Scenario 3. Given a particular timestamp t_0, the result of query on the temporal database $q^T = \sigma_{(id,t=t_0)}(r^T)$ is untrustworthy if $\{q^T[sig(m|u)] : sig(m|u) \in r^T\} \neq \{sig(q^T[m] : m \in r^T\}|u)$ and if $q^T[u \vee sig(m|u)] = \emptyset$. This means that a former transaction on the record that occurred in t_0, was submitted illegally.

Scenario 4. Given the current timestamp t_{max}, let $q^T = \sigma_{(id,max(m|t_{max}):m \in r^T)}(r^T)$ to be the latest version of a record in D^T and $q = \sigma_{id}(r)$ to be the same record in D. A record is said to be untrustworthy if $q^T \neq q$ meaning that the latest version of a record in the temporal table does not match the record in a normal table. This include the following cases:

- $q^T[m] \neq q[m]$
- $q^T[sig(m|u)] \neq q[sig(m|u)]$
- $q^T[d|id] \neq \emptyset$ but $q[id] \in r$
- $q^T[d|id] = \emptyset$ but $q[id] \notin r$

All in all, we argue that the data in a database is said to be transparent if:

1. The content of the records match the submitters' digital signature.
2. No anonymous transaction was submitted to the system.

3. History of applied transactions is provided for all records.

4. Items 1 and 2 are valid for all former transactions on the records.

5. The latest version of the records in the temporal audit table match the records in the normal table.

1.4 Contributions

1.5 Outline of the Thesis

Chapter 2

Background

This chapter introduces the tools and concepts that the proposed system is based on. In the first section we define the concepts and tools that were used from the Relational Database Management System (RDBMS). Second section covers the basics of cryptography and hashing techniques and finally in the third section we introduce the basics of Blockchain technology.

2.1 Relational Database

Definition 1. (Temporal Database): Let $r_i = r_1, r_2, ..., r_n$, be n number of tables in the relational database D. Denote the attributes of each table as $attr(r_i)$ where $r_i \in D$. A temporal table denoted r_i^T is a table with attributes $atr(r_i^T) = \{(updated, deleted) \in \mathcal{T}\} \cup attr(r_i)$ where $\mathcal{T} = t_0, t_1, ..., t_n$ is the time domain in which transactions on r_i happened. A temporal database denoted D^T is the result of augmenting D by r_i^T:

$$D^T = D \cup \{r_i^T : r_i \in D\}$$

Table 2.1: Normal Relational Table r_1

id	item	value
22	Pencil	7.50$
23	Notebook	12.0$

Table 2.2: Temporal Table r_1^T

id	item	value	updated	deleted
21	Ruler	3.25$	2018-02-10	-
21	Ruler	3.25$	-	2018-02-20
22	Pencil	8.0$	2018-03-21	-
22	Pencil	7.50$	2018-03-30	-
23	Notebook	12.0$	2018-04-01	-

The temporal database D^T contains the entire history of the records ever existed in D.

Example 1. Given the normal relational table r_1 (Table 2.1) and temporal table r_1^T (Table 2.2), the $attr(r_1) = (id, item, value)$ and $attr(r_1^T) = (id, item, value, updated, deleted)$. The result of query $q_1 = \sigma_{id=22}(r_1)$ is $\{(22, Pencil, 7.50\$)\}$ and the result of same query on the temporal table $q_2 = \sigma_{id=22}(r_1^T)$ is $\{(22, pencil, 8.0\$, 2018{-}03{-}21, NULL), (22, pencil, 7.50\$, 20$ $03-30, NULL)\}$. Also the query $q_3 = \sigma_{id=21}(r_1)$ results in $NULL$ however, the same query on the temporal table $q_4 = \sigma_{id=21}(r_1^T)$ has the history of record with id = 21: $\{(21, ruler, 3.25, 2018-02-10, NULL), (21, ruler, 3.25, NULL, 2018-02-20)\}$.

Definition2. (Time domain): The time domain $\mathcal{T}$ consists of discrete timestamps $t_0, t_1, ..., t_n$ in which transactions on tables $r_i \in D$ happened. the range of time domain is: $\mathcal{T} = [t_0, \infty)$ where t_0 is the timestamp in which the first record added to the table r_i. The time domain of a temporal table $r_i^T \in D^T$ is calculated by:

$$\mathcal{T} = r_i^T[updated] \cup r_i^T[deleted]$$

For example in the temporal table r_1^T (Table 2.2), the time domain is: $\mathcal{T} = [2018 - 02 - 10, 2018 - 04 - 01]$.

Definition 3. (Timestamps): A timestamp $t_i \in \mathcal{T}$ is a particular position in the time domain, in which particular transaction(s) happened. For example in the temporal table r_1^T (Table 2.2), "2018-03-30" is a timestamp in which the record with "id = 22" updated.

Definition 4. (Timeline): Let $u_i(t_j)$ be the total transactions on the tables $r_i \in D$ at timestamps $t_j \in \mathcal{T}$, where $i, j = \{0, 1, ..., n\}$. The $u_i(t_j)$ could be represented as an ordered set points on a vector. This vector is called a timeline for the transactions happened on $r_i \in D$. Figure x illustrates the concept of timeline.

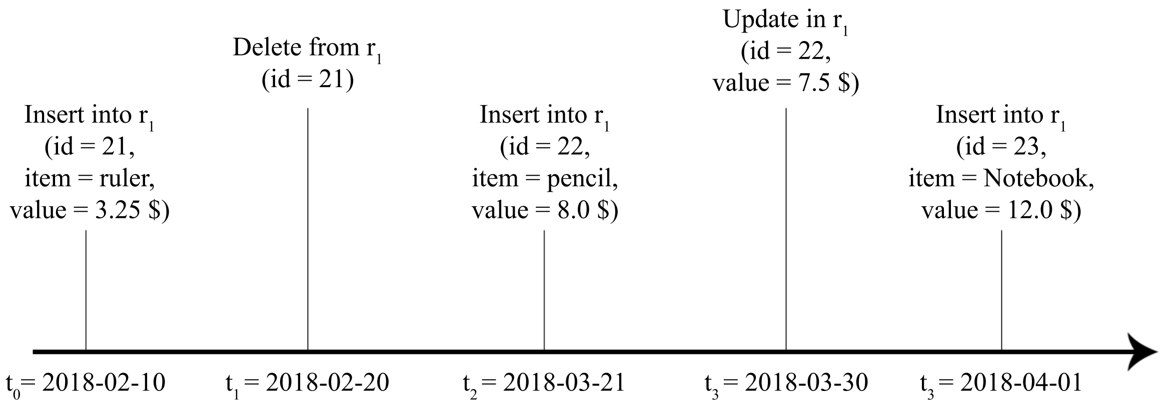
Insert into r_1 (id = 21, item = ruler, value = 3.25 $)
Delete from r_1 (id = 21)
Insert into r_1 (id = 22, item = pencil, value = 8.0 $)
Update in r_1 (id = 22, value = 7.5 $)
Insert into r_1 (id = 23, item = Notebook, value = 12.0 $)
t_0= 2018-02-10
t_1= 2018-02-20
t_2= 2018-03-21
t_3= 2018-03-30
t_3= 2018-04-01

Figure 2.1: Timeline.

Defionition 5. (Snapshot and snapshot-queries): Given a temporal table $r^T \in D^T$ and a timestamp $t \in \mathcal{T}$, we denote $s(t)$ to be the table instance that obtained by calculating the $\{max(r^T[m])|t : m \in r)\} - r^T[deleted]$ for $\mathcal{T} \leq t$. Note that $s(t) \in D^T$ but not necessarily $s(t) \in D$. A snapshot is a materialized version of

$D(t) = \{s_1(t), s_2(t), ..., s_n(t)\}$. A snapshot-query is an arbitrary relational query on $D(t)$.

We can construct the snapshots using simple windowing functions (as in supported by PostgreSQL [?]).

```
snapshot(r, t) =
 WITH T AS (
  SELECT id, {last_value(x) as x : x ∈ attr(r)} OVER W
  FROM r^T
  WHERE updates ≤ t
  WINDOW W AS PARTITION BY id ORDER BY updates
 )
 SELECT id, {x : x ∈ attr(r)} FROM T
 WHERE NOT T.deleted
```

The query snapshot(r, t) computes the snapshot of r at timestamp t by applying the latest update of each tuple up to timestamp t, while removing tuples that have been deleted.

Proposition 1. Linear Time: Assume that the tables are updated at a constant rate over time, then the complexity of snapshot(r, t) is

$$\mathcal{O}(|\{x : x \in r^T \text{ and } x.\text{updates} \leq t\}|) \simeq \mathcal{O}(t)$$

Note that this complexity is also valid for regular query answering (id, t) where the latest version of a particular record at time t is requested.

2.2 Cryptography

Cryptography is a way of secure communication between parties in a network while adversaries might also be present. Using Cryptography, messages sent or received are encrypted so that the adversaries cannot read the normal form of the message. This communication is established through various steps such as cryptographic key assignment, encryption and decryption of messages or digitally signing a message and verifying the digital signatures.

Definition.(Cryptographic Keys): Let u be the authenticated user who is using database D. By the creation of the profile of u in the system, a set of strings $< K_{priv}, K_{pub} > \in \mathbf{N}^+$ is generated and assigned to the user where K_{pub} is the public key that is accessible to everyone on the system, and K_{priv} is the private key that is known only to u. These keys are used to encrypt/decrypt messages which is transmitted between the users.

Definition.(Assymetric Encryption): Let E be the encryption algorithm, D be the decryption algorithm, m be the message which needs to be encrypted and c be the encrypted message. Given the cryptographic keys $< K_{pub}, K_{priv} >$, An encryption technique is said to be asymmetric if:

$$c = E(K_{pub}, m) \text{ and } m = D(K_{priv}, c)$$

or

$$c = E(K_{priv}, m) \text{ and } m = D(K_{pub}, c)$$

Therefore:

$$D(E(m, K_{pub}), K_{priv}) = m$$

and

$$D(E(m, K_{priv}), K_{pub}) = m$$

Note that, if K_{pub} is known, and $E(K_{pub}, m)$ is also known, in asymmetric encryption method, it is impossible to get m without k_{priv}.

Figure x shows the basic steps to send and receive messages between two parties in a secure way by utilizing asymmetric encryption technique. [Figure and description to be added]

Definition. (Hash function): Assume $m \in D$ to be the message with an arbitrary size chosen from domain D. $hash(m) \rightarrow sketch$ is a hash function that maps the m of any size to a fixed size string (normally 256 bits).

Definition. (Digital Signature): Digital signature is used to ensure that the digitally transfered data has not altered while transferring. Also it verifies whether or not the transfered data was submitted by a recognized source. Figure x shows the basic bluh bluh (add more details)

2.3 Blockchain

hash pointers

Components of a Block

Chapter 3

Approach

3.1 Overview

3.2 Creating Temporal database

3.2.1 Auditing

3.2.2 Large query handling

3.3 Applying cryptography

3.3.1 Signing

3.3.2 Signature validation

3.4 Blockchain

3.4.1 Creating Blocks

3.4.2 Block Validation

3.4.3 report fake data

Chapter 4

Experiments

Experiments here

Chapter 5

Related Work

Related works here

Chapter 6

Conclusions

Conclusion here

Bibliography

[1] Random, R. How random is everywhere. In *Proc. of the 2nd Work. of Randomness* (Apr. 2012), pp. 34 –41.