LEBANESE AMERICAN UNIVERSITY

A Hash-Based Assessment and Recovery Algorithm for Distributed Healthcare Systems Using Blockchain Technology

By
Mohammad Walid Jaber

A thesis
Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

School of Arts and Sciences

December 2020

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS APPROVAL FORM

Student Name: Mohammad Jaber I.D. #: 201605721

Thesis Title: A Hash-Based Assessment and Recovery Algorithm for Distributed Healthcare Systems

Program: Masters of Science

Department: Computer Science and Mathematics

School: Arts and Sciences

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

Masters of Science in the major of Computer Science

Thesis Advisor's Name: Ramzi A. Haraty

Signature: [redacted] Date: 28 / 12 / 2020
Day Month Year

Committee Member's Name: Azzam Mourad

Signature: [redacted] Date: 28 / 12 / 2020
Day Month Year

Committee Member's Name: Senthil Athithan

Signature: [redacted] Date: 28 / 12 / 2020
Day Month Year

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS COPYRIGHT RELEASE FORM

Name: Mohammad Jaber

Signature: [redacted]

Date: 6 / 1 / 2021
Day Month Year

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU's Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Mohammad Jaber

Signature: [redacted]

Date: 6 / 1 / 2021
Day Month Year

Dedication Page

To my loving parents

ACKNOWLEDGMENT

This project would not have been possible without the support of many people.
Many thanks to my advisor, Dr Ramzi Haraty, who read my numerous revisions and helped make some sense of the confusion. Also, thanks to
my committee members, Dr Azzam Mourad and Dr Senthil Athithan, who offered guidance and support.

And finally, thanks to my parents and numerous friends who endured this long process with me, always offering support and love.

A Hash-Based Assessment and Recovery Algorithm for Distributed Healthcare Systems Using Blockchain Technology

Mohammad Walid Jaber

ABSTRACT

The improvement of information technology in the past few years has been encouraging the healthcare sector to share medical data online without any barriers and between different parties. Many research works have been done to achieve this goal, and blockchain-based approaches have proved to be decent solutions. However, security challenges can put such distributed databases storing sensitive healthcare data under threat. For example, attackers can access highly sensitive data, altering or deleting some records, or violating the integrity of the database. Many preventive measures have been applied to protect the databases from attacks. However, no one can be confident that the system is safe and secure. Here rises the need for an algorithm that can assess the damage that occurred before recovering the database back to its consistent state. Numerous damage assessment and recovery algorithms have been proposed in the literature. In this work, we present a distributed algorithm that uses blockchain technology and hash tables to solve the information warfare problem in healthcare systems. The proposed algorithm is compared with different previous works and experimental results are recorded.

Keywords: Information Warfare, Distributed Databases, Blockchain Technology, Damage Assessment, Database Recovery, Transactional Dependency.

TABLE OF CONTENTS

I. Introduction....................1

1.1 Overview....................1

1.2 Motivation....................2

1.3 Blockchain Technology....................3

1.4 Information Warfare....................4

1.5 Scope of Work....................5

1.6 Thesis Organization....................5

II. Literature Review....................6

2.1 Overview....................6

2.2 Traditional Models....................6

2.3 Graphs in Recovery....................7

2.4 Clusters and Sub-clusters....................8

2.5 Before Images....................10

2.6 Matrices in Recovery....................11

2.7 Column Dependency....................13

2.8 Distributed Recovery....................13

2.9 Fuzzy Dependency....................15

III. The Model....................16

3.1 Overview....................16

3.2 Definitions....................16

3.3 Assumptions....................17

3.4 Model Structure....................17

3.5 Blockchain Methodology....................20

3.6 Hash Table....................20

3.7 Damage Assessment Algorithm....................21

3.8 Example of the damage assessment algorithm....................23

3.9 Recovery Algorithm....................26

3.10 Example of the Recovery Algorithm....................27

3.11 Model Cases....................28

IV. Performance Analysis....................35

4.1 Overview....................35

4.2 Performance Results and Analysis of the Damage Assessment Algorithm....................36

4.3 Performance Results and Analysis of the Recovery Algorithm....................41

4.4 Performance Analysis of the Memory Consumption....................45

4.5 Conclusion 48

V. Conclusion 49

VI. References 50

LIST OF FIGURES

Figure 1 - Distributed Database Structure 18
Figure 2 - Model Structure....................19
Figure 3 - Hash Dependency....................21
Figure 4: Damage Assessment Algorithm 22
Figure 5 - Hash Dependency Table H....................25
Figure 6: Recovery Algorithm 27
Figure 7 - Case 1: Hash Dependency Table....................29
Figure 8 - Case 2: Hash Dependency Table....................31
Figure 9 - Case 3: Hash Dependency Table of the Principle Database 32
Figure 10 - Case 3: Hash Dependency Table of a Local Database....................33
Figure 11 - Damage Assessment Algorithm Execution Period w.r.t. the ID of the Attacking Entity....................37
Figure 12 - Damage Assessment Algorithm Execution Period w.r.t Traditional Models 39
Figure 13 - Damage Assessment Algorithm Execution Period w.r.t Recent Algorithms........40
Figure 14 - Recovery Algorithm Execution Period w.r.t. Recovered Transaction Numbers ..42
Figure 15 - Recovery Algorithm Execution Period w.r.t. Traditional Models 43
Figure 16 - Recovery Algorithm Execution Period w.r.t. Recent Algorithms 44
Figure 17 - Occupied Memory Analysis in Best Case Scenario....................46
Figure 18 - Occupied Memory Analysis in the Worst Case 47

Chapter One

Introduction

1.1 Overview

Nowadays, we live in the era of data. Data are facts and statistics collected for analysis and decision making. It has become increasingly computerized from paper to electronic form due to the huge improvement in technology in the past few years. One of the common types of data that have been stored in electronics forms are medical data. This information summarizes the history of the patient including medical surgeries, laboratory tests, x-ray images, and much more. If data is corrupted or attacked, the entire medical system will inevitably be damaged or at least harmed. There rises the need for securing these data against unauthorized access, and unwanted modifications or changes. The three key parts of information security are: prevention, detection and correction. Where each part consists of several techniques. Techniques used in prevention methods include authentication, authorization and access control while different techniques such as checksums, message digests and intrusion detection systems are used in detection methods. Finally, backup and logging techniques are mainly applied in correction methods. The main role of this layered system is to prevent hackers' attacks or at least minimize their effects,-yet one cannot be confident that the system is safe as hackers can harm the system after subduing the implemented techniques. Therefore, one must first think about securing their system by applying as many preventive techniques as possible to gain a step ahead of the attacker. After an attack happens, detective and recovery techniques must be applied to identify these malicious transactions and remove their impact on the system. Confidentiality, integrity and availability are the main attributes of data that must be attained:

1- Confidentiality is the state of securing the data and keeping it private against unauthorized access.
2- Integrity is protecting the data from being modified or tampered by unauthorized users.

3- Availability means that only authorized users can at any time access requested information.

Due to the advancement of attacking tools nowadays, it’s hard to prevent malicious attacks.. Therefore, instead of just trying to secure systems from these attacks, many research works have been focusing on recovering databases after the actual attacks.

1.2 Motivation

Sharing medical information online (such as laboratory tests, surgery details, etc.) can help in the improvement of medical treatment. The interoperability between patients and medical institutions (hospitals, doctors, etc.) can be effective in many ways, namely:

1- Improve patient’s safety: for example, limiting the number of times the patient can be exposed to radiation.
2- Decrease medical costs: reducing duplicate medical tests. For example, if a patient undergoes a heartbeat test in hospital A, he/she is not forced to undergo the same test if he/she entered hospital B.
3- Minimizing administrative tasks: decrease time and effort spent on administrative tasks.

However, security challenges expose the entire medical system to danger, which seems to be a major barrier in medical sharing (Azaria et al., 2016). Moreover, the increasing competition in information warfare, raises the need to secure our data and information against malicious attacks, and more precisely to recover our system after an attack occurs.

Between 2008 and 2015, the usage of (EHR) increased by around 966% due to the adoption of Electronic Health Providers (EHR) by the US healthcare providers (Gordon and Catalini, 2018).

A major target for patients, is the ability to share and use all their medical data and test between multiple hospitals; thus, saving the cost of being forced to repeat the same tests at different hospitals. Moreover, it can be very beneficial for the patients to access their medical information in any country or at any hospital; for example, during the current coronavirus pandemic, the patients can avoid repeating the COVID-19 test in every country they visit. To find a solution for this problem, many researches have been done, and blockchain-based approaches proved to be decent solutions.

Due to its decentralized nature, blockchain was key in improving the interoperability between patients and health data systems (Gordon and Catalini, 2018; Jiang et al., 2018; Ali et al., 2016). Due to increasing success of blockchain solutions, we believe that blockchain-based approaches will prove to be a success when adopted in the healthcare sector.

1.3 Blockchain Technology

In Satoshi, 2008, blockchain was introduced as a framework for Bitcoin. In other words, blockchain is literally a series of blocks forming a chain. Each block consists of digital pieces of information. Every block consists of three parts:

1- Information about the transaction: for example, date, time, price, etc.

2- Information about the participants in the transaction: for example, if John buys an item from the Apple store, this block will record both John and Apple store website.

3- Information to distinguish blocks: a unique code called “hash” that makes blocks distinguished.

In the blockchain system, a new block is added when the following conditions are satisfied:

1- A Transaction must exist: back to John’s example, a transaction occurs after John proceeds with the checkout process.

2- A Transaction must be verified: after John completes his purchase from the Apple store, the Apple network verifies that the transaction is legal. That is, it confirms the details of the transaction including its price, the date of purchase, etc.

3- A Transaction must be stored in a block: after being verified, John’s transaction will be packaged with other transactions in a block.

4- A hash must be assigned to the block: after verifying and storing the transaction to a block, a unique hash will be assigned to this block.

When a block is added to the blockchain, it becomes publicly available for anyone to view. The use of smart contracts is another important feature of blockchain. Smart contracts can be defined as self-verifying, temper resistant and self-executing objects that eliminate the use of trusted third parties (Zhang et al., 2018; Panda and Haque, 2002; Panda and Ragothaman, 2003). Blockchain has been proving its success in multiple fields and different applications. Using blockchain technology, users are now able to preserve a decentralized reliable database (Dai et al., 2017). Many research projects have been working on deploying blockchain in other systems such as education, cybersecurity, and IoT (Turkanovi et al., 2018; Chen et al., 2018; Huh et al., 2017). Blockchain has multiple characteristics that it an attractive solution for medical sharing problem. In addition to its decentralized nature, blockchain is immutable and trustless (verifiable).

1.4 Information Warfare

The field of Information warfare has been growing rapidly and getting a lot of interest in the past years from defense planners and policymakers. (Molander et al., 1996). The concept of Information warfare has become a hot topic in the computational field. The main objectives of information warfare are:

1- Taking advantage of an exploit or a bug in the system to cause harmful actions to the whole system such as denial of service.
2- Protecting the system against different types of attacks.

Taking use of the system vulnerabilities and backdoors, the attacker could gain access to the target system and perform their intended harmful actions that would cause damage to the entire system. Some of the techniques used are:

1- Denial of Service (DOS): Preventing authorized users to access the system by applying different types of DOS attacks; thus, will affect the availability of the data.
2- Monitor and Control: Gaining unauthorized access to the system; for example, the attacker would gain an administrative access to the system; thus, affecting both the confidentiality and integrity of the data.

3- Physical Disruption: This happens by physically destroying the data stored on a specific medium; thus, affecting the availability of the data.

4- Data Modification: modification, insertion, and deletion of data; thus, affecting the integrity and availability of the data.

1.5 Scope of Work

In this paper, we present a distributed algorithm that uses blockchain technology and hash tables to solve the problem of information warfare in healthcare systems. Mainly we present:

1- An efficient algorithm to solve the problem of information warfare in healthcare systems.

2- A distributed architecture for our model.

3- Using state of art techniques such as hash tables and blockchain technology in our algorithms.

Our model is implemented, and the results are recorded. We aim is to reach the fastest recovery time to achieve the lowest database offline time.

1.6 Thesis Organization

The remainder of the thesis is organized as follows: Chapter II presents a review of the previous work in literature. In chapter III, we describe our proposed algorithm with some examples to illustrate our approach. Chapter IV presents the computational results with comparisons to similar previous works In chapter V, we provide concluding remarks and introduce potential future work.

Chapter Two

Literature Review

2.1 Overview

The topic of database recovery from malicious attacks has been studied extensively in the literature an Some research projects focus on improving existing algorithms in terms of runtime and memory usage, while others focus on proposing new data structures that would produce better results than using the traditional ones. Almost all the work done in the literature relies on two main algorithms: data dependency (Panda and Haque, 2002) and transactional dependency (Panda and Ragothaman, 2003). Transactional dependency tracks dependency between transactions without taking into consideration the exact data item, unlike data dependency which focuses on the data item itself. In each type, many models were proposed using different data structures such as matrices, clusters, and graphs. In this section, we will review prior artwork found in the literature about these models.

2.2 Traditional Models

In traditional models, the main log file is usually scanned starting from the beginning of the attack, where all affected transactions are re-executed after removing all effects caused by the malicious transactions. In the assessment and recovery phases, traditional approaches adopt full rollback operations on the database. Liu and Jajodia presents a traditional recovery model known as branching, which relies on the tree structure of the database versions (Liu and Jajodia, 2002). In case an attack occurs at any branch, an alternative branch will be used until the attacked branch is returned back to its consistent state.

In Kumar and Son, 1998, three traditional models were presented by Kumar and Son. The first model is known as transactions rollback. In this model, after the detection of a malicious attack, the transactions are rolled back in reverse to their previous logical state before the attack happened. The second model is called redo recovery that works by scanning the log file to redo all transactions in their same order. This model will ensure that the database is returned to its initial state preceding the malicious attack. The third model is known as rollforward recovery. In this model, At every time *T*, the method takes a copy of the database as a backup to ensure that the consistent state of the database can be restored after any potential logical or physical errors.

2.3 Graphs in Recovery

Panda and Zuo (2004) introduce graph-based models for the damage assessment phase and postponed implementing the recovery phase for future work. These models rely on two main assumptions: the log file cannot be damaged, and no blind writes are allowed. These models focus on returning the affected transactions to the recovery algorithm to perform the necessary operations for the aim of returning the database to a reliable state. In these models, A multi-sites distributed database that consists is employed, where every site is managed locally and coordinates with the central coordinator or with the remaining sites. In this work, two main models are presented for the damage assessment phase: peer-to-peer and centralized models.

In the peer-to-peer model, the detection process doesn't work as follows: each local site manager scans the log file to detect affected transactions, and if found, it will notify other site managers that have executed any sub-transactions in their log files. After that, the log files at every notified site will be scanned by the local managers to detect new transactions that may have been affected before sending them to other sites.

On the other hand, a central coordinator is required for the centralized model. Choosing the coordinator site is done through a voting process, where the coordinator to be elected must meet the highest number of the following features:

1- The most convenient place with respect to other sites.

2- The site must be equipped with the best performance capabilities to play its coordination role.

3- Fast network links must exist between the site and other sites.

4- Support backups on the site in case the machine fails.

The centralized model consists of three submodels:

1- Forward and receive models: the coordinator receives global affected transactions and forwards them to other sites with dependent executed sub-transactions.
2- Graph model for local dependency: before building the affected transactions, the coordinator must receive the local graph of each site.
3- Central graph repository model: the coordinator saves the local graphs that is sent by every site performs the needed updates when the ` ones.

Another agent-based model that uses graphs was presented by Saba et al. (2018). A single agent is used to receive and forward messages to other controlled agents. The dependent graph is obtained by adding a node for each dependent transaction below its parent; for example, a node for transaction *T_2* is added below *T_1*'s node if transaction *T_2* reads data item written by *T_1*.

Each agent scans their managed graphs to detect any malicious transaction or activity. If found, the agent sends back a pointer to the recovery manager. This pointer corresponds to the malicious or affected transactions. During recovery phase, the scan picks the smallest ID value, and stops when it reaches a malicious or affected node, then the rollback phase is carried out till it reaches a non-malicious node. The advantage of this approach is that it only isolates the affected portion of the graph; thus, it can be expanded easily with affecting the performance of the database.

2.4 Clusters and Sub-clusters

Clustering approach relies on one main concept which is log file segmentation. A clustered log file is further segmented by the sub-clustering approach to reduce its size.

Haraty and Zeitunlian (2007) presented a new data dependency based log clustering algorithm In this algorithm, two main factors influence the sub-clustering of clusters: the number of committed transactions and the cluster size. To make the process faster, only the sub-clusters are scanned. The proposed algorithm assumes the following for proper operation:

1- Presence of intrusion detection system to detect malicious transactions.

2- Production of customized reads and writes operations by the database scheduler

3- Production of rigorous serializable history

4- Existence of only committed transactions in the clustered log.

5- Sequential incrementation of transaction ID's.

For enhancement purposes, Haraty and Zeitunlian (2007) proposes two data structures: transaction sub-cluster list and sub-cluster data list. A sub-cluster list keeps track of the transaction IDs alongside with the sub-cluster that stores the related data for each transaction. The second data structure stores both IDs too and adds to them the corresponding read, write, overlooked, predicate or statement data item. When some malicious transactions are detected by the intrusion detection system, both lists are checked to identify the affected transactions. During the recovery phase, the scan is done only on sub-clusters containing affected data items. Two additional data structures are used in the detection phase: Damaged_DI and Damaged_PB where all data items damaged by the malicious transactions are tracked by the Damaged_DI, and predicate blocks of all affected transactions are stored in the Damaged_PB data structure. Initially, both data structures are null. To detect affected transactions, the transaction sub-cluster is checked throughout the malicious attack. In addition to the previous steps, recovery phase consists of the following:

1- After assessment, every record in Damaged_PB is scanned.

2- The Obtain the sub-cluster of each transaction is obtained from the transaction sub-cluster list.

3- Every block is evaluated through a new evaluation process.

4- Restored data items are returned to the database.

5- The two data structures, Damaged_DI and Damaged_PB, are set back to null and released.

Panda and Ragothaman (2003) proposes a new cluster-based algorithm where some limitations are set on the number of committed transactions, and size and window time of the cluster. This model proposes three ways for log file segmentation:

1- Segment the log file after checking the number of committed transactions.

2- Segment the log file after a time period *T*.

3- Segment the log file after a specified memory size is occupied.

The presented model assumes the following:

- Using a rigorous two-phase scheduler.
- Identifying the attacker's identity through an intrusion detection system.
- Purging of the log file is not permitted.
- No blind writes are allowed.

In the detection phase, the affected transactions are determined from both the affected items and the read items collected before. The log file segmentation proves its efficiency while comparing the experimental results with that of an unsegmented one.

Tripathy and Panda (2000) proposes a new cluster-based algorithm that relies on a new logging protocol. This protocol maintains all useful information that is used during the recovery of the database. This model defines a predicate-based statement block A predicate is a set of preconditions that must evaluate to true to continue execution. The predicate could be either conditional or unconditional.

The proposed model assumes the following for its assessment and recovery process:

- Usage of log sequence number.
- Usage of write-ahead-logging protocol.
- Application of the steal/no force protocol to the database.
- Usage of check pointing mechanism that is consistent and stable.
- Production of rigorous serializable history
- No modification of the log is allowed. nor purged.
- No nested transactions are permitted.
- Writing a data item is only allowed to a stable database.

2.5 Before Images

Zhu et al. (2008) present a new approach to track damage based on "before images" tables. These tables check the read operations executed on affected transactions. In this model, no log file is needed for database recovery with the usage of before images, one can keep track of the history of the transactions so that the database can be returned to its last consistent state before the attack happened. After a defined time period *T*, all the records in the before image table will be deleted; hence, preserving the size of the table and preventing it from increasing exponentially.

The suggested approach is presented by an inter-transaction dependency graph. In this approach, the last transactions that wrote and deleted data item x are tracked by two new data items: *x.ins_tran* and *x.del_tran* respectively. These two data items are added by the model. This model also includes *TranDepTab:* a table to store inter-transactions. This table consists of three columns where it stores the transactions dependent on each other, their commit order and the transactions dependent by them.

The algorithm runs after a malicious activity is detected. Moving to the recovery process, it consists of two main phases:

1- Detect and undo affected transactions

2- Delete effects of such transactions on the database.

The "before images" approach's main highlights is that it allows fast identification of affected transactions due due to the presence of the inter-transaction dependency graph that keeps track of dependency between transactions.

2.6 Matrices in Recovery

Matrices are one of the popular data structures that were used in the damage assessment and recovery phases. Panda and Lala (2001) a matrix-based algorithm to minimize the time spent during the damage assessment phase. In this model, the following assumptions are made:

- Malicious transactions are identified directly after an attack
- The history produced by the scheduler is strictly serializable.
- The occurrence order of transactions is preserved when tracked by the log file.
- The log file cannot be purged, and it's restricted to specific users.

Another matrix-based approach for database recovery is presented by Haraty and Zbib (2014) and Haraty et al. (2015). In this approach, an external intrusion detection system is necessary to detect malicious transactions and forward them to the proposed model. Also, the history produced is assumed to be rigorously serializable. Moreover, the model uses a checkpoint mechanism on the log file version preventing its size from increasing vastly and thus affecting the overall performance of the proposed model.

In this approach, a two-dimensional array represents the matrix used with columns representing data items and rows representing the committed transactions up to time *T*. The matrix is filled by values *00*, *01* or -T_i to represent the interaction between the transaction and the data item. Interactions taking place between a transaction *T* and a data item *x* is one of the following:

1- *T* blindly writes *x*.

2- Committed transaction *T* modifies *x*.

3- *T* reads *x*, or in other words *T* do not modify *x*.

Two more data structures are used in this model: a two-dimensional array to keep track of the committed transactions that affect a data item *x*. While transactions that will be recovered in the recovery phase are tracked by another one-dimensional array. Following a malicious attack i.e. in the detection phase, the algorithm traverses the matrix and checks all the transactions that proceed the attack. If *01* is found, then the algorithm checks if the transaction is malicious or affected. If a negative value is found, the algorithm similarly traverses the second matrix to check for any further malicious or affected transactions.

After obtaining the list of both affected and malicious transactions, the recovery phase starts with the recovery algorithm undoing the malicious transactions and re-executing the affected ones. Similar to any algorithm, the proposed algorithm has strengths and weaknesses. By using a simple matrix, the detection and recovery algorithms of this model are fast, however, the memory is significantly consumed due to the usage of multiple data structures

2.7 Column Dependency

Chakraborty et al., (2010) presents a column dependency-based technique. The aim of this model is to detect the affected transactions and to apply recovery measures to return the database back to its consistent state. The algorithm presented has two versions: a static version and an online version. In the static version, the database is set offline and becomes unavailable to users during the recovery process. The algorithm takes as inputs:

1- List of committed transactions.
2- Execution schedule of these transactions.
3- Set of detected malicious transactions

In the online recovery version, users can access the database thus allowing transactions to continue running. The three phases that comprise the online recovery version are: assessment, recovery and confinement phases.

In the online version, transactions remain executing, thus allowing the possibility of having new malicious transactions contrary to the static recovery where the database is set offline, and transactions are aborted.

2.8 Distributed Recovery

Panda and Zou (2006) presents a distributed approach for database recovery. The authors propose two approaches: peer to peer and centralized.

In the first approach, affected transactions are shared by every site manager with every site that executed related sub-transactions. The main highlights of this approach:

- Avoid single point of failure.
- Process distribution where each site will run the same algorithm with balanced data.
- Load distribution where the load is distributed among all sites offering a faster assessment.

However, this model has two challenges that needs to overcome:

1- Huge network traffic due to information sharing.

2- Synchronization between all sites.

On the other hand, the centralized model applies a voting process to select a coordinator. Each site manager will forward the affected transactions to the coordinator which in turns, will forwards them to The main highlights of this approach:

1- Low network traffic
2- Assessment is only done by the coordinator

However, this model has some major drawbacks:

1- The coordinator will be placed under huge load.
2- The recovery process will be delayed until the coordinator receives the list of all affected transactions.

A new damage assessment and recovery model is presented by Liu and Yu (2011) The work done contributes to the following:

1- Avoiding single point of failure by distributing the whole process.
2- Incoming transactions are continuously handled.
3- Simultaneous execution of damage assessment and recovery.
4- Easy integration within DBMS.
5- The proposed models are completely transparent.

Each site contains two processes: a local damage assessment and repair (DAR) and a DAR executor. that scans the log file to identify affected sub-transactions. The local DAR manager handles the coordination between the processes at each site.

During the recovery process, the DAR manager and executor produce cleaning transactions and sub-transactions respectively to recover all malicious transactions and sub-transactions.

2.9 Fuzzy Dependency

Fuzzy dependency describes a new type of logical dependency that cannot be expressed by functional dependencies. “It reflects a kind of semantic knowledge about the real world” (Chen, 1995). The three main scenarios for using fuzzy dependency:

1- Fuzzy integrity checking where constraints are applied on a list of relations’ instances. This can be beneficial in case these roles were violated, the modifications can be aborted.

2- Playing the role of an intrusion detection system.

3- Producing fuzzy values for affected data items after a denial of service attack.

Yanjun Zou and Panda (2004) introduce a fuzzy dependency approach for detecting affected transactions and repairing the database. The proposed model is faster than the traditional approaches as it does not require a detailed scanning of the log file.

Chapter Three

The Model

3.1 Overview

In this section we present a hash-based technique for damage assessment and recovery that uses blockchain technology. We optimize our model to increase both accuracy and efficiency and to decrease memory utilization during assessment and recovery phases.

The algorithm starts with the assessment phase. The malicious transactions, identified by intrusion detection system, are assessed in order to detect their effects. Then, all transactions are marked as either clean or affected that need to be recovered. After that, the algorithm redoes the affected transaction and undo the malicious ones in order to recover the database to its consistent state.

3.2 Definitions

Definition 1: If a write operation $write_i[x]$ is done based on the read value of operation $read_j[y]$, then $write_i[x]$ is dependent on $read_j[y]$. Where $write_i[x]$ denotes write operation of a transaction T_i, and $read_j[y]$ denotes read operation of another transaction T_j (Panda and Tripathy, 2000).

Definition 2: If a data item x is modified by a write operation $write_i[x]$ of a transaction T_i before reading its latest value, then $write_i[x]$ is considered a blind write.

Definition 3: If a schedule's effects on a consistent database are similar to that of a a serial one, then it this schedule is serializable (Sumathi and Esakkirajan, 2010).

Definition 4: If uncommitted or unaborted transactions alter a data item, this data item becomes un-accessible (Bernstein et al., 1986). This is because a schedule follows strict property i.e. a strict schedule (Breitbart et al., 1991).

Definition 5: A rigorous schedule is a strict schedule that assert the following: A transaction cannot write a data item, if this data item is already being read by another transaction, unless the latter is either committed or aborted.

3.3 Assumptions

The first assumption in our model is the presence of an intrusion detection system that continuously checks if an attack exists. This intrusion detection system sends a list of malicious transactions during the assessment phase. We also assume the rigorous serializability of history produced where no transaction T_2, that proceed and depends on T_1, can exist

The transactions are assigned unique and sequential IDs. The $ID = 1$ is given for the first transaction, $ID = 2$ for the second one and so on. Therefore, no two transactions can have the same *ID*.

Our damage assessment algorithm follows the transaction dependency paradigm (Panda and Ragothaman, 2003) for building the dependencies, which means that a transaction T_2 is dependent on transaction T_1, if a data item read by T_2 is already being written by T_1. if a transaction T_2 reads a data item being written by another transaction T_1, regardless of the exact data item being read, T_2 is dependent on T_1 since the read operation is based on the previously written value by T_1.

Another assumption in our model is the safety of the log file of every database and its inaccessibility by the users. Moreover, we assume the existence of certification before the committed transactions are uploaded to the principle database.

The proposed hash-based technique is suggested by Haraty and Bokhari (2019). This technique uses a hash table to enhance assessment and recovery phases by allowing swift access and search time. In addition to that, we aim to reduce the recovery time. Also, reducing the recovery time is one of our main objectives. Due to the high demands of accessing the medical databases, one must reduce the database's offline time

3.4 Model Structure

In figure 1, our distributed database structure is sketched. In our model, we assume that the Ministry of Health owns the Principle Database. The doctors/hospitals will upload their patients' data to the replicated database, and at some time (checkpoint T) the changes will be uploaded to the principle database.

Based on blockchain technology, the transactions need to be certified by the government before their commitment. Since some transactions are still upfront, and they are not uploaded to the principle database, so we need to recover locally, and others must be recovered on the principle database.

In figure 2 our model is sketched. Our solution is formed of three main components: an Intrusion Detection System (IDS), to detect malicious transactions, damage assessment algorithm to detect and assess the damaged transactions, and a recovery algorithm that removes the effects of malicious transactions, redoes the damaged transactions and returns the database back to its previous consistent state. After detecting malicious transactions, the IDS will send a list of these transactions which will be received in the damage

assessment phase. In this phase, the first malicious transaction to be handled is the one with minimum

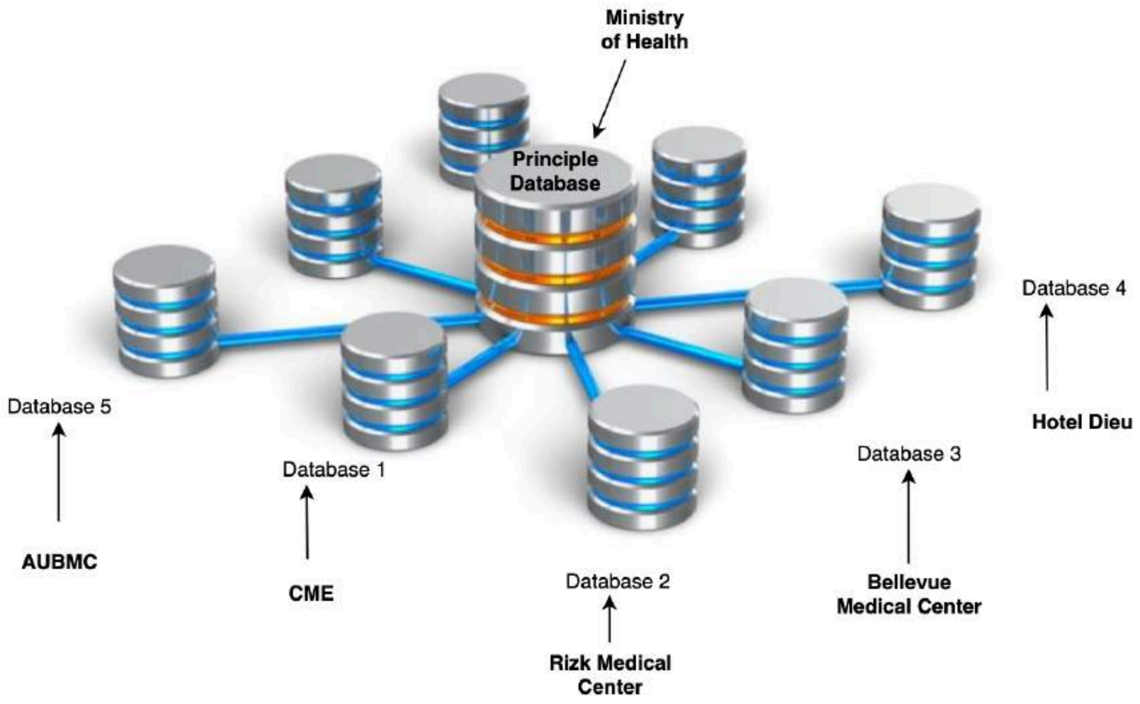


Figure 1 - Distributed Database Structure

ID. Then, based on the list received by the IDS, the algorithm outputs the respective damaged transactions.

Since we are dealing with a distributed database, the IDS is present on each database. When the assessment phase is done, the principle database will check for affected replicated databases and forward the list of affected and malicious transactions to them to recover locally.

Using checkpoint mechanism, the principle database will select the first checkpoint *T* after the attack happens to identify affected replicas. Local recovery is needed since some transactions will be saved locally waiting their confirmation to be uploaded to the principle database.

At the end of the damage assessment phase, the recovery process starts by receiving the list of malicious and damaged transactions. During the recovery process, the recovery algorithm removes all effects of malicious transactions and re-run the affected ones. In this phase, the affected database is taken offline. Hence, proposing a faster algorithm helps in reducing the database offline time. To reach this goal, we use hash tables for both phases.

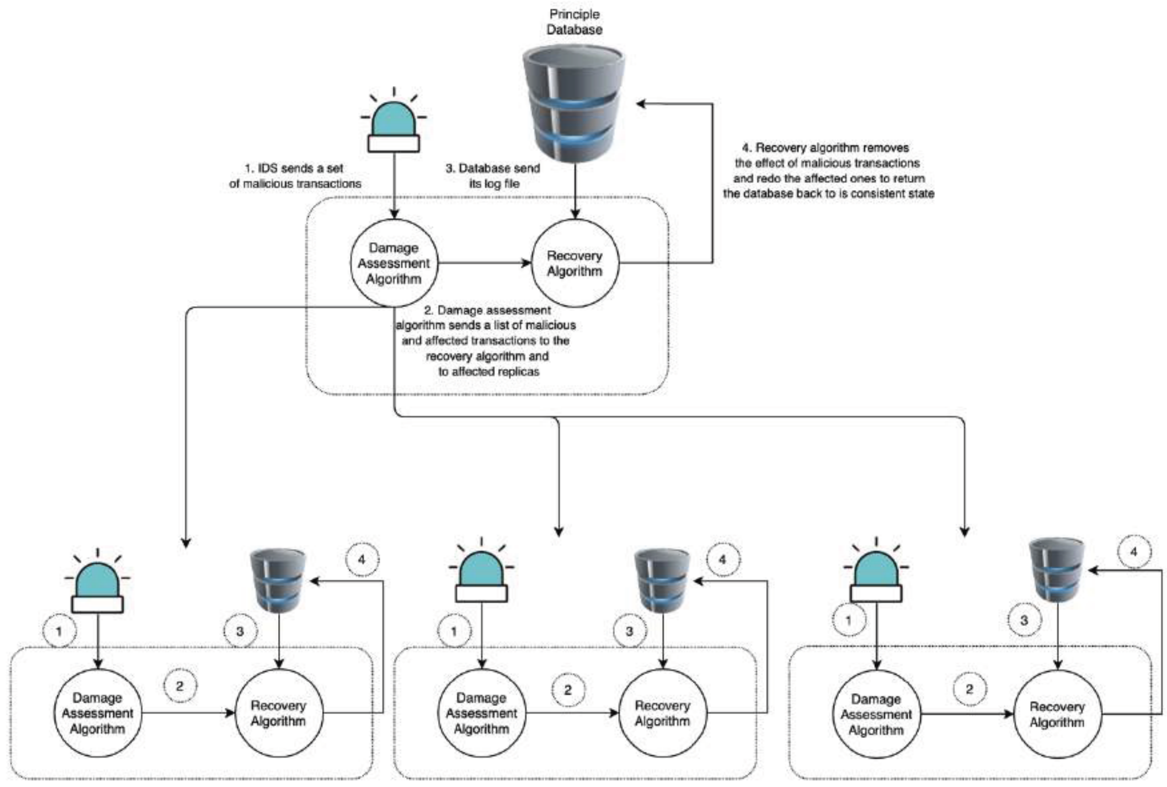

Principle Database
1. IDS sends a set of malicious transactions
3. Database send its log file
4. Recovery algorithm removes the effect of malicious transactions and redo the affected ones to return the database back to is consistent state
Damage Assessment Algorithm
Recovery Algorithm
2. Damage assessment algorithm sends a list of malicious and affected transactions to the recovery algorithm and to affected replicas
1
2
3
4
Damage Assessment Algorithm
Recovery Algorithm
1
2
3
4
Damage Assessment Algorithm
Recovery Algorithm
1
2
3
4
Damage Assessment Algorithm
Recovery Algorithm

Figure 2 - Model Structure

3.5 Blockchain Methodology

As mentioned before, our model presents an assessment and recovery algorithm for distributed healthcare systems using blockchain technology. So, based on blockchain methodology, and due to its decentralized nature, we are motivated to achieve much success in the healthcare sector on all levels.

In order to guarantee authentication and privacy of information, the results of the committed transactions are certified, encrypted and then saved in the secure database at a particular address. Then, instead of saving actual results in the blockchain, only the addresses are encrypted and then saved for the following reasons:

1- Saving more space, since saving the actual results requires much more space than saving the encrypted addresses.
2- Improving security, since an attacker who gains access to this record will have an encrypted address which will be incomprehensible. However, if the actual results are saved and the attacker accesses to our database, it will be easy for him/her to see the actual results and maybe alter or modify them.

Thus, each database in our model will contain encrypted pointers to the medical results in the secure database.

3.6 Hash Table

In our model, a hash table is used for both damage assessment and recovery algorithms. The hash table is built during transactions' execution to store transactions' dependencies. For memory purposes, only dependent transactions are stored in the hash table.

Our algorithm follows the transaction dependency paradigm, so we only care about which transactions are dependent on each other. For example, if some data items, already modified by a previous committed transaction $T_{2,}$ were read by a transaction T_1, the hash table will keep track of this dependency (*T_1* dependent on *T_2*). The details of this dependency are not required since we are focusing on transaction dependency and not on data dependency, which means if *T_2* is malicious.

The hash table stores a number of transactions where every stored transaction can be accessed by referring to its ID's hash value (Haraty and Bokhari, 2019), and has a corresponding list that contains the transactions dependent it. Figure 3 presents an example of the hash dependency.

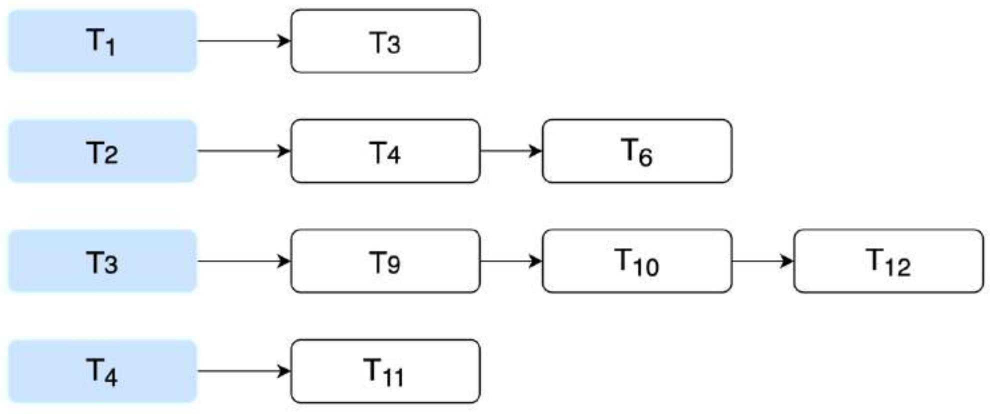


Figure 3 - Hash Dependency

Moreover, the log file is stored in a hash table to improve execution time and to minimize the resources used. Therefore, both damage assessment and recovery phases use only hash tables as data structures. Also, previous logs are deleted using checkpoint mechanism in order to maintain the size of the log file.

3.7 Damage Assessment Algorithm

In assessment phase, the IDS send a list of malicious transactions If multiple transactions were identified as malicious, the algorithm starts with the one with the smallest *ID*. As we mentioned earlier, we assume that the schedule's history is rigorously serializable, so *T_j* cannot depend on *T_i* with *$j < i$*. Also, the transactions committed after the commitment of malicious transaction are considered by the algorithm. So, the assessment phase will spend less time to finish as the affected transactions are considered. For instance, a malicious transaction with ID = 10 requires the damage assessment algorithm to consider the transactions with *ID* greater than 10.

The affected transactions are clean transactions that can be classified into two categories:

1- Directly affected: after reading some data item(s) previously inserted by a malicious transaction.

2- Indirectly affected; after reading some data item(s) previously written by an affected transaction.

To collect both directly and indirectly affected transactions, the hash table is scanned.

To get the index of a malicious transaction from the hash table, the assessment algorithm receives its ID from the IDS. Then, the algorithm creates copies of the malicious transaction's dependency and affected lists. The algorithm loops over all transactions in the affected list and copies their dependency list to cover all affected transactions (directly and indirectly). Figure 4 presents Algorithm 1, that summarizes the damage assessment algorithm. For example, based on Figure 2 if malicious transaction T1 is detected by the IDS, then its index is pinpointed by the assessment algorithm and then T_3 is copied to the affected list. This process is repeated and T_9, T_{10}, and T_{12} are copied to the affected list. Since T_9, T_{10}, and T_{12} has no dependent transactions, the algorithm terminates and outputs T_1 as the malicious transaction and T_3, T_9, T_{10}, and T_{12} as the affected list.

```
Algorithm 1 - Damage Assessment Algorithm

Input: malicious transaction list alongside with the hash table
Output: Malicious and affected lists
1. Locate the malicious transaction index in the hash table
2. Copy the dependency list corresponding to the malicious transaction
   to the affected list
3. for T in affected list:
4.     locate the T's index in the hash table
5.     Copy the T's dependency list to the affected list
6. end for
```

Figure 4: Damage Assessment Algorithm

3.8 Example of the damage assessment algorithm

Suppose we have a database for health sharing management system that stores the following tables:

1- Doctors

- DID: doctor's identification number. This number is distinctive for every doctor.
- DName: doctor's name.
- DMajor: doctor's major.
- DExperience: doctor's experience years

2- Patients

- PID: distinctive identification number of a patient.
- PName: name of a patient
- PDOB: date of birth of a patient
- PAddress: address of a patient

3- Types

- TID: distinctive identification number of a medication type.
- TName: name of the medication type.

4- Medication

- MID: distinctive identification number of a medication.
- MName: name of medication
- MType: type of medication

5- Prescriptions

- PrID: distinctive identification number of each a prescription.
- PID: Id of patients holding this prescription.
- DID: Id of doctor who described this prescription.
- PrDate: prescription's date.

6- PrescriptionDetails

- PDID: distinctive identification number of a prescription's detail.

- PrID: id of the prescription containing this detail
- MID: Id of medication containing this detail
- PDFrequency: number of times this prescription must be taken per day.
- PDDays: treatment duration in days.

The below transactions modify the above stated database:

- T_1 = Doctors (‘11’, ‘Peter’, ‘Heart,’8’);
- T_2 = Types (‘2’, ‘Heart Medication’)
- T_3 = Medication (‘5, ‘Aspirin’, ‘2’)
- T_4 = Types (‘18’, ‘Pain Killers’)
- T_5 = Medication (‘12’, ‘Paracetamol’, ‘18’)
- T_6 = Doctors (‘17’, ‘John’, ‘Dermatology’, ‘10’)
- T_7 = Patients (‘3’, ‘Merry’, ‘1-1-1995’, ’California’)
- T_8 = Patients (‘4’, ‘Frank’, ‘9-9-1985’, ‘Madrid’)
- T_9 = Prescription (‘21’, ‘4’, ‘17’, ‘1-2-2017’)
- T_{10} = Prescription (‘22’, ‘3’, ‘11’, ’23-12-2015’)
- T_{11} = Prescription (‘23’, ‘4’, ‘11’, ’15-2-2017’)
- T_{12} = Prescription_details (‘31’, ‘21’, ‘12’, ‘2’, ‘7’)
- T_{13} = Prescription_details (‘32’, ‘22’, ‘12’, ‘2’, ‘21’)
- T_{14} = Prescription_details (‘33’, ‘23’, ‘5’, ‘3’, ‘14’)

While the above transactions are executed, the hash table *H* is formed. Before inserting T_3 into the database, it reads the type with *ID* = 2, previously inserted by T_2. T_3 is dependent on T_2, so H saves both transactions where T_2.’s dependency list adds T_3. Moreover, category with ID = 18, previously inserted by T_4, is read before T_5 is inserted into the database, so H inserts a new row for T_4 and T_5 is added to T4’s dependency list.

T_6 inserts new records into Doctors while *T_7,* and *T_8* insert records into Patients, but none of these transactions are added to *H* as they depend on no other transactions. It is also clear that *T_9*, *T_{10}* and *T_{11}* depends on *T_6* and *T_8*, *T_1* and *T_7*, *T_1* and *T_8* respectively. Therefore, *T_9* is added to *T_6* and *T_8* dependency lists, *T_{10}* is added to *T_1* and *T_7* dependency lists and *T_{11}* is added to *T_1* and *T_8* dependency lists.

Considering the last three transactions, *T_{12}* reads from *T_5* and *T_9*, so it is added to their dependency lists. *T_{13}* reads from *T_5* and *T_{10}* and is added to their dependency lists. Similarly, *T_{14}* is added to *T_3* and *T_{11}* dependency lists as it reads from them. Figure 5 presents the hash table *H*

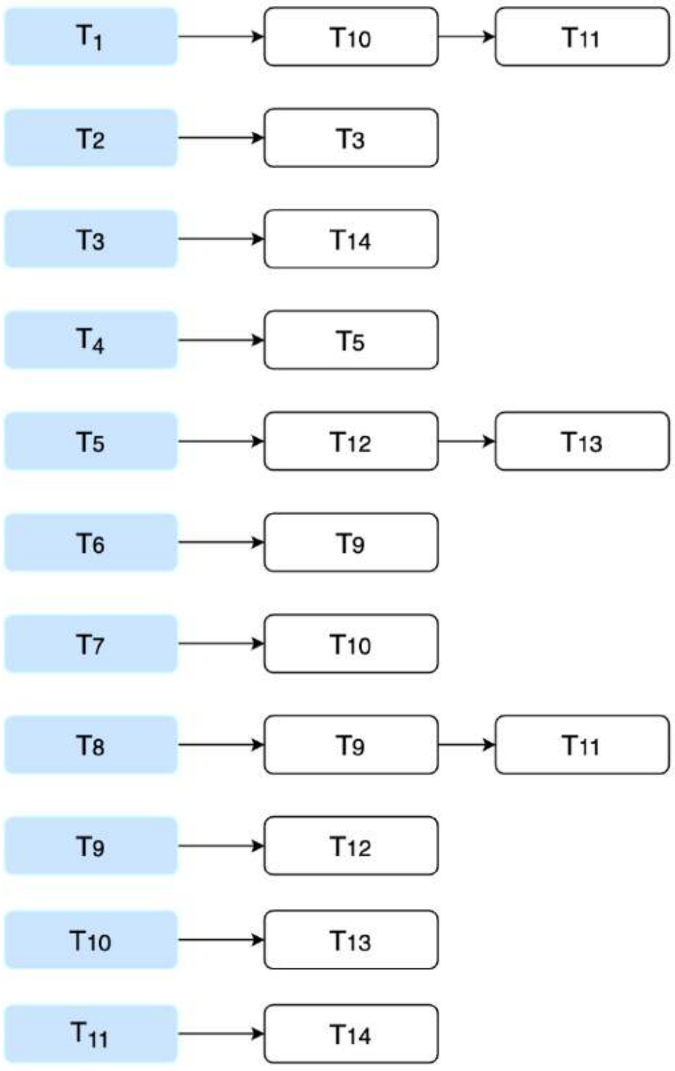


Figure 5 - Hash Dependency Table H.

The list of malicious transactions alongside H are sent to the assessment algorithm. The results obtained from the assessment algorithm are forwarded to the recovery algorithm, to recover the database to its previous consistent state.. One of our assumptions is the rigorous serializability of history so, as an example, the IDS ignores transactions with ID ≤ 6 whenever T_6 is detected by the IDS as a malicious transaction. The algorithm copies the dependency list of T_6 which contains only transaction T_9 to the affected list. The algorithm repeats this process with all affected transactions in order for all indirect damaged transactions to be covered. So, the algorithm copies the dependency list of T_9 and adds it to the affected list that contains now T_{12} and T_9. Since T_{12} has no dependent transactions, the algorithm terminates and outputs a malicious list that contains only T_6 and an affected list that consists of T_9 and T_{12}.

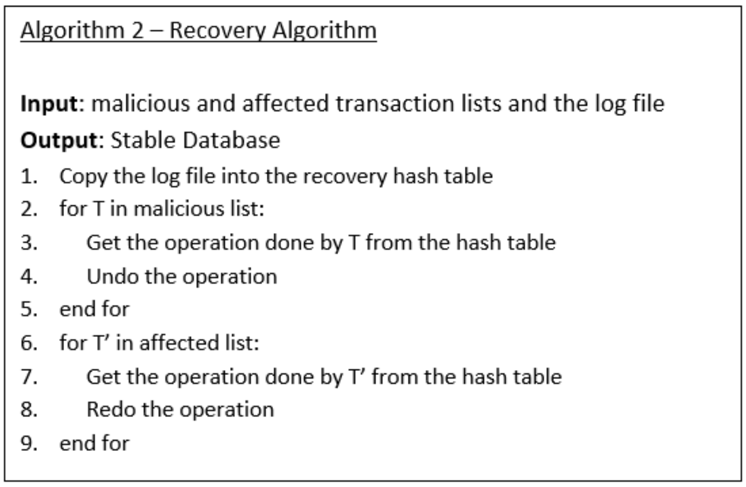
3.9 Recovery Algorithm

The end of the assessment process marks the start of the recovery phase where results of the assessment phase are taken as inputs by the recovery algorithm. At the end of the recovery phase, the database will be recovered back to its consistent state.

As mentioned earlier, the log file is stored in a hash table to enhance the recovery time. In this phase, the malicious transaction with smallest ID is picked by the recovery algorithm. The transactions that occurred before the malicious transaction are ignored since they have smaller IDs, as we assume the rigorous serializability of history.

First, any effects of malicious transactions are removed by the recovery algorithm. Using the recovery hash table, the algorithm selects all operations done by the malicious transaction to remove their effects. As an example, the recovery algorithm deletes any record in the database that is entered by a malicious transaction; thus, undoing this operation.

Next, the recovery algorithm redoes the affected transactions. Similarly, the recovery algorithm selects all operations done by the affected transactions from the hash table to redo them. An example of these operations is updating a record, where the algorithm re-executes this operation that was executed by an affected transaction. Figure 6 presents Algorithm 2 that summarizes the recovery algorithm.



Algorithm 2 – Recovery Algorithm

Input: malicious and affected transaction lists and the log file
Output: Stable Database

```
1.  Copy the log file into the recovery hash table
2.  for T in malicious list:
3.      Get the operation done by T from the hash table
4.      Undo the operation
5.  end for
6.  for T' in affected list:
7.      Get the operation done by T' from the hash table
8.      Redo the operation
9.  end for
```

Figure 6: Recovery Algorithm

The recovery algorithm stops when the last affected transaction is re-executed. At this stage, the database is recovered back to its consistent state.

3.10 Example of the Recovery Algorithm

Based on the example presented in section 3.7, a malicious list that contains only T_6 and an affected list that consists of T_9 and T_{12} were output by the damage assessment algorithm. The first member in the malicious list, transaction T_6, (in our case it is the first and the only transaction in the malicious list) is picked by the recovery algorithm.

The recovery algorithm retrieves all the operations done by T_6 from the hash table. The selected operation is as follows:

Insert a new record into Doctors table with the corresponding values (‘17’, ‘John’, ‘Dermatology’, ‘10’)

So to remove any effects of T_6, the algorithm removes from doctors table the inserted doctor with ID = 17. Then, all the damaged transactions are redone by the algorithm that retrieves all the operations done by each affected transaction from the hash table.

The stored operation for the first affected transaction T_9 is as follows:

Insert a new record into Prescriptions table with the corresponding values (‘21’, ‘4’, ‘17’, ‘1-2-2017’)

In order to re-execute this operation, a new record should be added for patient with *ID* = 4 in Prescription table. However, there is no doctor with *ID* = 17 since this record was deleted in the previous step. Therefore, the prescription with *ID* = 21 is deleted from the Prescriptions table.

The stored operation for the last affected transaction T_{12} is as follows:

Insert a new record into PrescriptionDetails table with the corresponding values (‘31’, ‘21’, ‘12’, ‘2’, ‘7’)

In order to re-execute this operation, a new record should be added for prescription with *ID* = 21 in PrescriptionDetails table. However, the prescription with *ID* = 17 was deleted. Therefore, the prescriptiondetail with *ID* = 31 is also deleted from the PrescriptionDetails table.

By this point, the algorithm re-executed any affected transaction and redid any malicious ones thus returning the database to its previous consistent state. Hence, the database can be set back online.

3.11 Model Cases

In our model, there are three cases for the attack to happen, below we list them and how our algorithm will tackle them. As mentioned previously, we are dealing with a distributed database, so we need to check for possible attacks on each level.

3.10.1 Case 1: Malicious transaction T_1 is present in the log file of the principle database

First, we consider the transactions listed below modifying the principle database. Given that data items *X, N, Y, Z, M* and *K* are modified by these transactions. We assume that the IDS identifies T_1 as the malicious transaction, and that T_1 is only present in the log file of the principle database. In other words, none of the replicated database has T_1 in its log file.

- T_1: $N=K+1$
- T_2: $K=K-5$
- T_3: $X=N+1$

- T_4: $Y=X-4$
- T_5: $M=N+1$
- T_6: $Z=Y+2$

The hash table for the following list of transactions is shown in Figure 7.

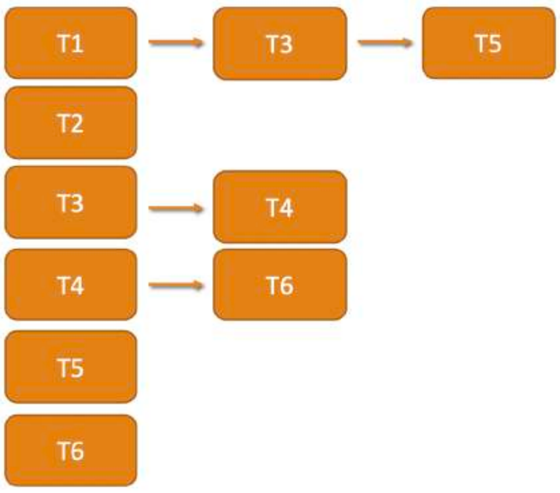


Figure 7 - Case 1: Hash Dependency Table

The principle database will be set offline. The damage assessment algorithm will receive the hash dependency table alongside with list of malicious transactions identified by the IDS. In this case, the algorithm copies the dependency list corresponding to T_1 which contains transactions T_3 and T_5 to the affected list. The algorithm repeats this process with all affected transactions to cover all indirectly affected ones. So, the algorithm copies the dependency list of T_3 and added it to the affected list. Therefore, the affected list becomes $\{T_3, T_4, T_5\}$. Moreover, the algorithm copies the dependency list of T_4 and added it to the affected list to become $\{T_3, T_4, T_5, T_6\}$. Since T_5 and T_6 has no dependent transactions, the algorithm terminates and outputs $\{T_1\}$ as the malicious list and $\{T_3, T_4, T_5, T_6\}$ as the affected one. Moreover, both lists will be forwarded to all replicated databases to check if any of these transactions are present in their log file.

The damage assessment algorithm sends the lists of malicious and affected transactions to the recovery algorithm that starts the recovery process by redoing the malicious transactions to remove their effects Using the recovery hash table, the algorithm selects all operations done by the malicious transaction to remove their effects.

Next, the algorithm redoes the affected transactions by selecting all operations done by the affected transactions from the hash table to redo them. In this case, the algorithm will undo the operation done by transaction T_1 since it's considered the malicious transaction, and will re-run the affected transactions (T_3, T_4, T_5, T_6) to make sure that the obtained results don't interfere with the malicious transaction. Finally, the principle database will be set back online.

3.10.2. Case 2: Malicious transaction T_1 is present in the log file of the local database, but not uploaded to the principle database

First, let us consider the transactions listed below modifying a local database. Given that data items *X, N, Y, Z, M* and *K* are modified by these transactions. We assume that the IDS identifies T_1 as the malicious transaction, and that T_1 is only present in the log file of a local database, and not uploaded yet to the principle database.

- T_1: $N=K+1$
- T_2: $K=K-5$
- T_3: $X=N+1$
- T_4: $Y=X-4$
- T_5: $M=N+1$
- T_6: $Z=Y+2$

The hash table for the following list of transactions is shown in Figure 8.

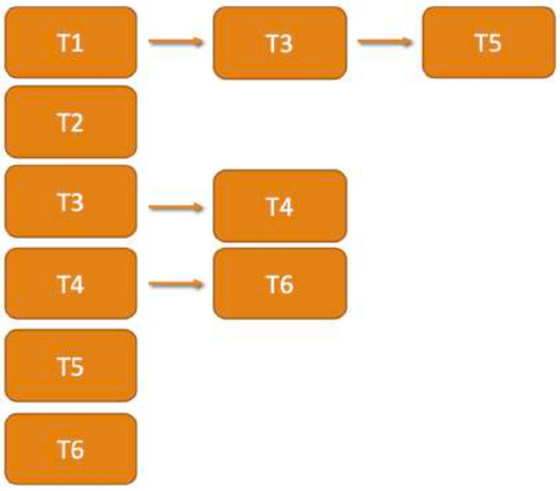


Figure 8 - Case 2: Hash Dependency Table

The local database will be set offline. Similarly, the damage assessment algorithm will receive the hash dependency table corresponding to the local database alongside with list of malicious transactions identified by the IDS. In this case, the algorithm copies the dependency list corresponding to T_1 which contains transactions T_3 and T_5 to the affected list. The algorithm repeats this process with all affected transactions to cover all indirectly affected ones. So, the algorithm copies the dependency list of T_3 and added it to the affected list to become $\{T_3, T_4, T_5\}$. Moreover, the algorithm copies the dependency list of T_4 and added it to the affected list to become $\{T_3, T_4, T_5, T_6\}$. Since T_5 and T_6 has no dependent transactions, the algorithm terminates and outputs $\{T_1\}$ as the malicious list and $\{T_3, T_4, T_5, T_6\}$ as the affected one.

The damage assessment algorithm sends the lists of malicious and affected transactions to the recovery algorithm that starts the recovery process by redoing the malicious transactions to remove their effects. Using the recovery hash table, the algorithm selects all operations done by the malicious transaction to remove their effects.

Next, the algorithm redoes the affected transactions by selecting all operations done by the affected transactions from the hash table to redo them. In this case, the algorithm will undo the operation done by transaction T_1 since it's considered the malicious transaction, and will re-run the affected transactions

(T_3, T_4, T_5, T_6) to make sure that the obtained results don't interfere with the malicious transaction. Finally, the local database will be set back online and will resume its medical operations.

3.10.3 Case3: Malicious transaction T_1 is present in the log file of the principle database, and some local databases read this malicious transaction and perform actions based on its value.

First, let us consider the transactions listed below modifying the principle database. Given that data items *X, N, Y, Z, M* and *K* are modified by these transactions.

- T_1: $N=K+1$
- T_2: $K=K-5$
- T_3: $X=N+1$
- T_4: $Y=X-4$
- T_5: $M=N+1$
- T_6: $Z= Y+2$

The hash table for the following list of transactions is shown in Figure 9.

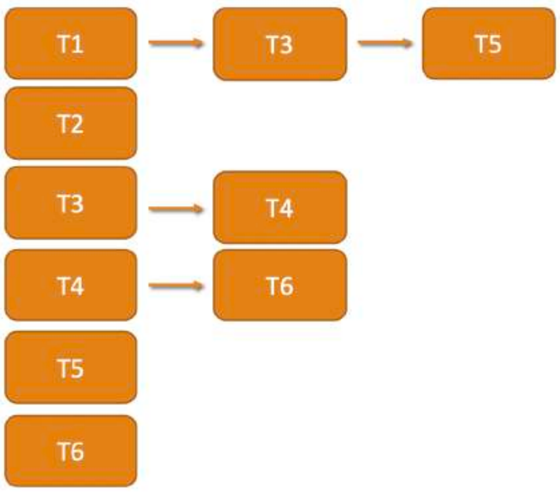


Figure 9 - Case 3: Hash Dependency Table of the Principle Database

Also consider the below transactions modifying a local replicated database. Also, we assume that data items *X, J, Y, L, Y, N, Z* and *K* are modified by these transactions.

- T_1: *N=K +* ***1***
- T_2: *K=K - 5*
- T_3: *X=N +* ***1***
- T_7: *Y=X - 4*
- T_8: *J=N + 8*
- T_9: *Z= K + 2*
- T_{10}: *L= Z + 2*

The hash table for the following list of transactions is shown in Figure 10.

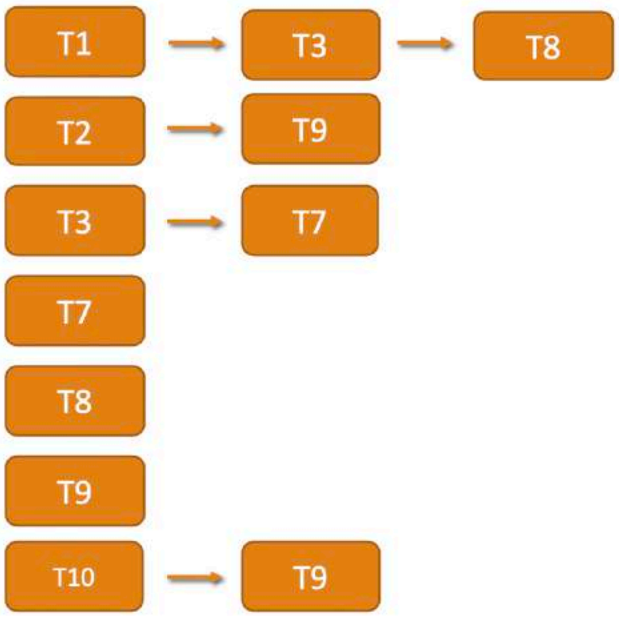


Figure 10 - Case 3: Hash Dependency Table of a Local Database

The local database alongside with the principle database will be set offline. First, the damage assessment algorithm will receive the hash dependency table corresponding to the principle database alongside with list of malicious transactions identified by the IDS. In this case, the algorithm copies the dependency list corresponding to T_1 which contains transactions T_3 and T_5 to the affected list. The algorithm repeats this process with all affected transactions to cover all indirectly affected ones. So, the algorithm copies the dependency list of T_3 and added it to the affected list to become $\{T_3, T_4, T_5\}$. Moreover, the algorithm copies the dependency list of T_4 and added it to the affected list to become $\{T_3, T_4, T_5, T_6\}$. Since T_5 and T_6 has no dependent transactions, the algorithm terminates and outputs $\{T_1\}$ as the malicious list and $\{T_3, T_4, T_5, T_6\}$ as the affected one.

The damage assessment algorithm sends the lists of malicious and affected transactions to the recovery algorithm that starts the recovery process by redoing the malicious transactions to remove their effects Using the recovery hash table, the algorithm selects all operations done by the malicious transaction to remove their effects.

Next, the algorithm redoes the affected transactions by selecting all operations done by the affected transactions from the hash table to redo them. In this case, the algorithm will undo the operation done by the malicious transaction T_1, and will re-run the affected transactions (T_3, T_4, T_5, T_6).

Moving to the local replicated database, the damage assessment algorithm will receive both the hash dependency table corresponding to the local database and the list of malicious transactions identified by IDS. In this case, the algorithm copies the dependency list corresponding to T_1 which contains transactions T_3 and T_8 to the affected list. The algorithm repeats this process with all affected transactions to cover all indirectly affected ones. So, the algorithm copies the dependency list of T_3 and added it to the affected list. The affected list becomes now $\{T_3, T_7, T_8\}$. Since T_7 and T_8 has no dependent transactions, the algorithm terminates and outputs malicious list that contains only T_1 and an affected list that consists of T_3, T_7, and T_8.

The recovery process starts with the recovery algorithm the recovery process redoing the malicious transactions to remove their effects. Using the recovery hash table, the algorithm selects all operations done by the malicious transaction to remove their effects.

Next, the algorithm redoes the affected transactions by selecting all operations done by the affected transactions from the hash table to redo them. In this case, the algorithm will undo the operation done by the malicious transaction T_1, and will re-run the affected transactions (T_3, T_7, T_8). Finally, the principle database and the replicated one will be set back online and will resume their medical operations.

Chapter Four

Performance Analysis

4.1 Overview

In order to analyze the performance of our algorithm, we run our algorithm a virtual environment identical to the actual environment where the algorithm is implementable.

As previously mentioned, an IDS is assumed to exist on each database, in order to detect and forward any malicious transactions. The model receives the these forwarded transactions and then is directly initiated. In our model, we depend mainly on one type of data structures: hash tables. During assessment phase, a hash table is used to track dependencies between transactions. In this stage, since some replicated databases may have run some operations after the last checkpoint *T*, each replicated database must check for affected transactions based on its hash table. Also, during recovery process, a hash table is used to store the log file.

As mentioned earlier, one assumption is the rigorous serializability of history, thus, no transaction can be dependent on any other transaction that occurs after it. In other words, a transaction T_i must happen before a transaction T_j whenever $i < j$. Based on this assumption, all transactions that occur before the malicious transaction are discarded by our algorithm.

The damage assessment phase initiates upon receiving a list of malicious transactions from the IDS. The algorithm starts with the transaction with the smallest ID in case several transactions were identified as malicious. At the end of this phase, the assessment algorithm generates two lists; one containing the malicious transactions and another one containing the affected ones. These lists initiates the recovery process, where the recovery algorithm redoes malicious transactions and re-execute damaged ones.

In our experiments, we use multiple values for the smallest malicious ID to test the performance of our algorithm. Moreover, the analysis done shows how our algorithm performs in terms of memory consumption that we aim to optimize.

During the experimental phase, we used the ‘NorthWind Database’ with slight modifications to achieve our goal. We build our distributed model to cover the four cases mentioned in section 3.10. NorthWind database is provided as a template in Microsoft Access or Microsoft SQL Server. We manage our database using a SQL Server (Microsoft, 17). We use Java as our programming language. The simulated environment is developed on a system with 2.2 GHz CPU, quad-core, and a 4 GB RAM.

4.2 Performance Results and Analysis of the Damage Assessment Algorithm

As mentioned earlier, the recovery process of the database consists of the damage assessment recovery phases. The main role of the first phase is to detect damaged transactions, while the main roles of the second phase is to redo these transactions and undo malicious ones.

In this section, the analysis of the damage assessment algorithm during simulation and the performance results are presented. The Northwind database used in this experiment is slightly tailored to meet the needs of our algorithm

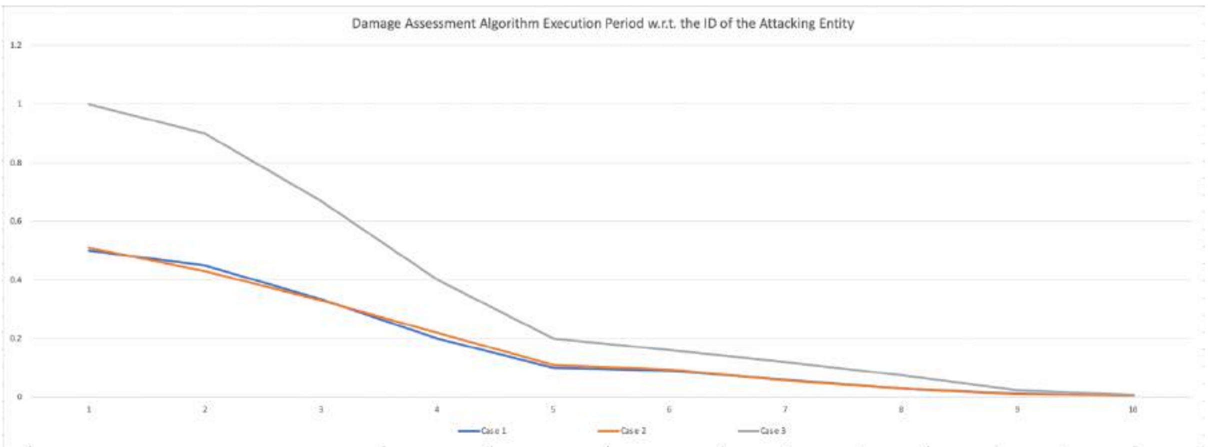
During simulation, we consider the below two assumptions:

1- A transaction can access an utmost number of data items equal to the number of data items in the table that this transaction is operating on.

2- To maintain the unique identity of transactions, a new column is added which to represent a global *ID*. The previous primary key is only unique in the scope of the table that the transaction is operating on. So, using the global *ID* is very important in our case. So a transaction with *ID* = 1 means the existence of single transaction with global *ID* = 1

The number of transactions scanned by the damage assessment algorithm is the main influencer of the algorithm’s performance. So, we vary the ID of malicious transaction while analyzing the running time of the algorithm. It is easy to note that slowest running time required by the algorithm is when the ID of the malicious transaction is the smallest because the algorithm must cover more transactions.

Figure 11 presents the performance of the assessment algorithm respective to the attacker ID in the three cases covered in section 3.10. The attacker ID is the ID of the malicious transaction identified by the IDS. Based on figure 11, while the attacker ID increase, the running time of our algorithm decrease. This proves our assumption above and verifies that our model is perfectly operating.



	Attacker ID	1	20	50	100	150	200	400	600	800	1000
Damage Assessment Time	Case 1	0.5	0.45	0.335	0.2	0.1	0.09	0.06	0.03	0.012	0.005
	Case 2	0.51	0.43	0.33	0.22	0.11	0.095	0.058	0.03	0.011	0.004
	Case 3	1	0.9	0.67	0.4	0.2	0.16	0.12	0.075	0.025	0.009

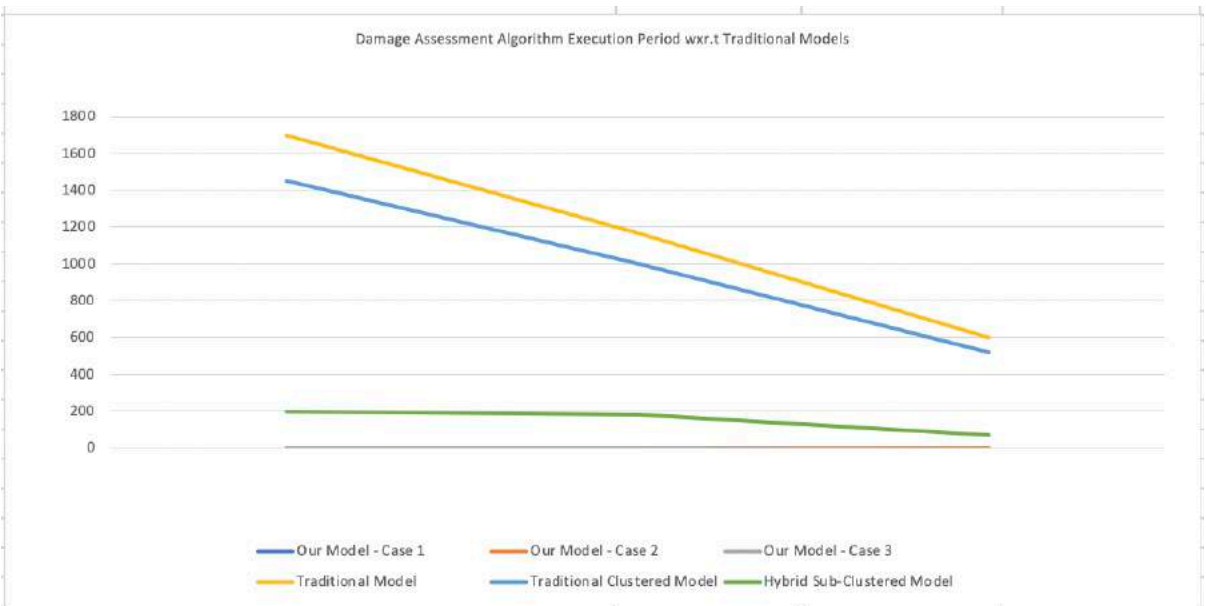
Figure 11 - Damage Assessment Algorithm Execution Period w.r.t. the ID of the Attacking Entity

During this experiment, the database snapshot used consists of 1080 records. The maximum number of data items that can be accessed at a time by any transaction is 18 (the maximum number of columns is 18 and it’s the Employees table).

Since we assume the rigorous serializability of our history, the assessment algorithm ignores any transaction that precedes the malicious one. This is shown in Figure 13. For example, when the ID of the attacker is 100, the algorithm needs to scan 980 transactions (1080 – 100) to check for affected ones. Also, the running time of the algorithm decreases when the ID of the attacker increases. We can conclude that our assessment algorithm proves to be efficient since it scans only transactions that can be affected, without wasting time scanning transactions preceding the malicious one.

To measure the whole performance of our damage assessment algorithm, The results of our assessment algorithm are compared with those of models presented in Haraty and Zeitunlian (2007). The three other models are the traditional, traditional clustered and the hybrid sub-clustered models. The traditional model is presented in Bai and Liu (2009). These experiments are performed on a database with 200 records. So, to properly compare the proposed model with the other models, we use an additional database snapshot. The results are summarized in Figure 14.

From the results presented in figure 14, the superiority of our algorithm 's performance compared to the other models is clear. When our algorithm is compared with the traditional hybrid sub-cluster model, we can see that the case of an attacker's ID = 50, the proposed algorithm spends only 0.00035 ms which indicates that our algorithm is 571428 times faster. Another example is the case of an attacker with ID = 150, where the proposed algorithm spends time that is 368421 faster than the model under comparison. By comparing the run time of our model with that of the traditional model, which is also the slowest model, we can see that the proposed algorithm is 4857142 times faster (case of attacker's ID = 50) and 3157894 (case of attacker ID = 150) times faster.



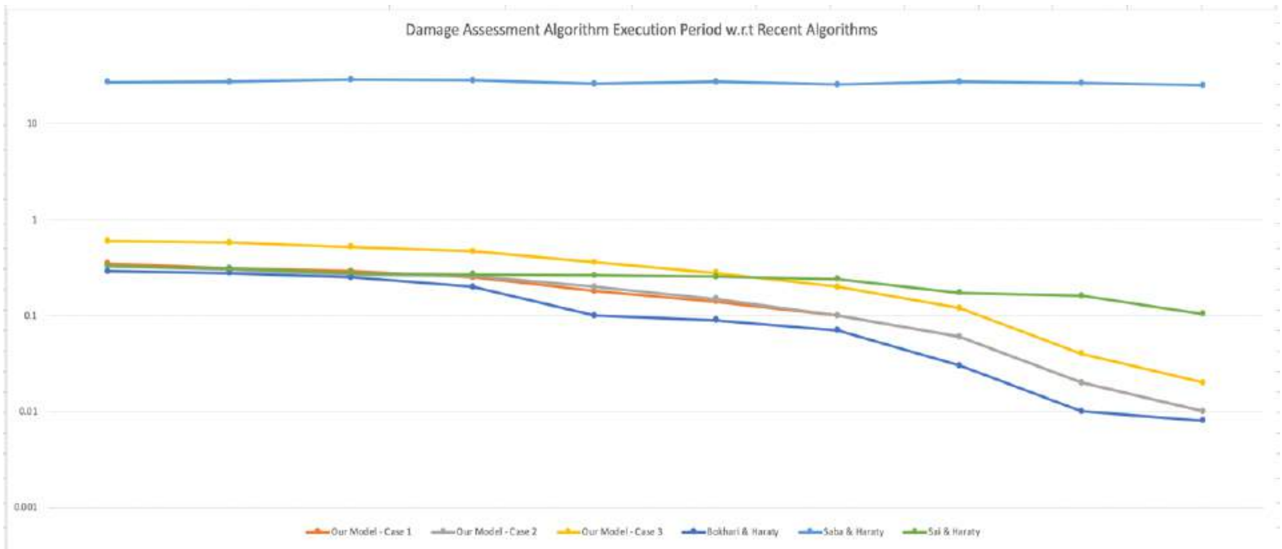
Attacker ID	50	100	150
Our Model - Case 1	0.00035	0.00025	0.00019
Our Model - Case 2	0.00033	0.00022	0.00017
Our Model - Case 3	0.00067	0.0004	0.00025
Traditional Model	1700	1170	600
Traditional Clustered Model	1450	1000	520
Hybrid Sub-Clustered Model	200	180	70

Figure 12 - Damage Assessment Algorithm Execution Period w.r.t Traditional Models

The main reason for the superior results of our algorithms is selecting a hash table to store dependencies between transactions. This data structure allows quick retrieval of the transactions ID's hash value. In addition to that, only the required information needed to find affected transactions are stored by the hash dependency table. The enhanced performance of our assessment algorithm is mainly due to the right selection of the data structure used to store dependencies between transactions. The hash table used provides fast access on the hash value of the transaction ID. Also, the hash dependency table only stores the needed information which is used to find the affected transactions.

Moreover to compare the results our damage assessment algorithms with other algorithms, a new experiment the is conducted to test algorithm's performance In this experiment we use a database snapshot containing 1080 records and compare our results with other models that uses hash tables (Haraty and Bokhari, 2019), linked lists (Haraty and Sai, 2016), and two-dimensional array (Saba, 2018). The results are summarized in figure 13.

Looking at the results in figure 13, we can conclude that our assessment algorithm outperforms the two models based on linked lists and two-dimensional array. However, our model has the same performance as the model presented by Haraty et al. (2019) which also uses the hash table for storing transaction dependencies.



Attacker ID	50	100	200	300	400	500	600	700	800	1000
Our Model - Case 1	0.35	0.31	0.29	0.25	0.18	0.14	0.1	0.06	0.02	0.01
Our Model - Case 2	0.33	0.3	0.27	0.26	0.2	0.15	0.1	0.06	0.02	0.01
Our Model - Case 3	0.6	0.58	0.52	0.47	0.36	0.28	0.2	0.12	0.04	0.02
Bokhari & Haraty	0.29	0.279	0.25	0.2	0.1	0.09	0.07	0.03	0.01	0.008
Saba & Haraty	27.4	27.6	28.9	28.5	26.3	27.6	25.8	27.6	26.7	25.5
Sai & Haraty	0.33	0.31	0.28	0.271	0.263	0.255	0.241	0.174	0.161	0.104

Figure 13 - Damage Assessment Algorithm Execution Period w.r.t Recent Algorithms

Our algorithm's running time scores a lowest running time equal to 0.01 µs and highest running time equal to 0.35µs. Using hash tables for storing transactional dependencies is a plus due to the fast retrieval of the transactions.

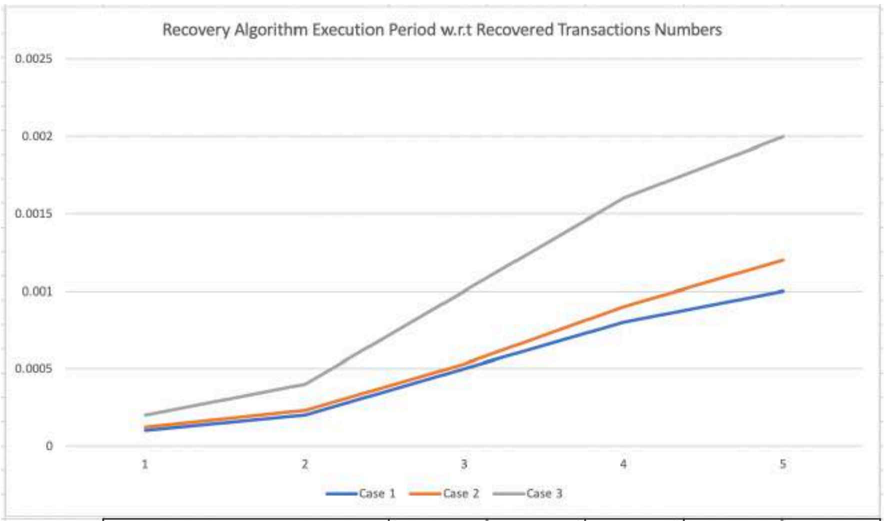
On the other hand, Haraty and Sai (2016) uses a matrix of linked lists to store the transaction dependencies. where all these transactions are checked by the algorithm in case of an attack. To detect affected transactions, the related row of every malicious transaction is checked by the algorithm. malicious transaction; thus, affecting the running time of the assessment algorithm. The running time of the assessment phase shows a slower performance than our algorithm especially when the attacking Id increases. An example is the case of an attacker with ID = 1000, where the proposed algorithm spends time that is 10 times less than that spent by Haraty and Sai's algorithm (Haraty and Sai, 2016).

Moreover, the algorithm presented by Haraty and Saba (2018) has the slowest running time compared to other presented models. In their work, the authors use graphs to store dependencies between transactions. Although using graphs has some advantages in terms of scalability and damage isolation, it still has some major disadvantages in terms of graph construction and coverage. Thus, our algorithm clearly outperforms this model with a ratio factor of 78 in their best-case scenario with the smallest attacker ID.

As a conclusion, we compare the running time of the proposed assessment algorithm with several models including recent and traditional model where Northwind database is adopted. The results (the results obtained during assessment phase for one database) prove that our algorithm outperforms almost all other models.

4.3 Performance Results and Analysis of the Recovery Algorithm

We display in this section the results and analysis of the recovery algorithms' performance during simulation. We can expect that the increase of recovery time is directly related to the increase of the number of transactions that needs recovery. The experiment was done using a snapshot of Northwind database consisting of 200 records where a maximum of 18 data items can be accessed by any transaction at a time. Figure 14 summarizes the results.

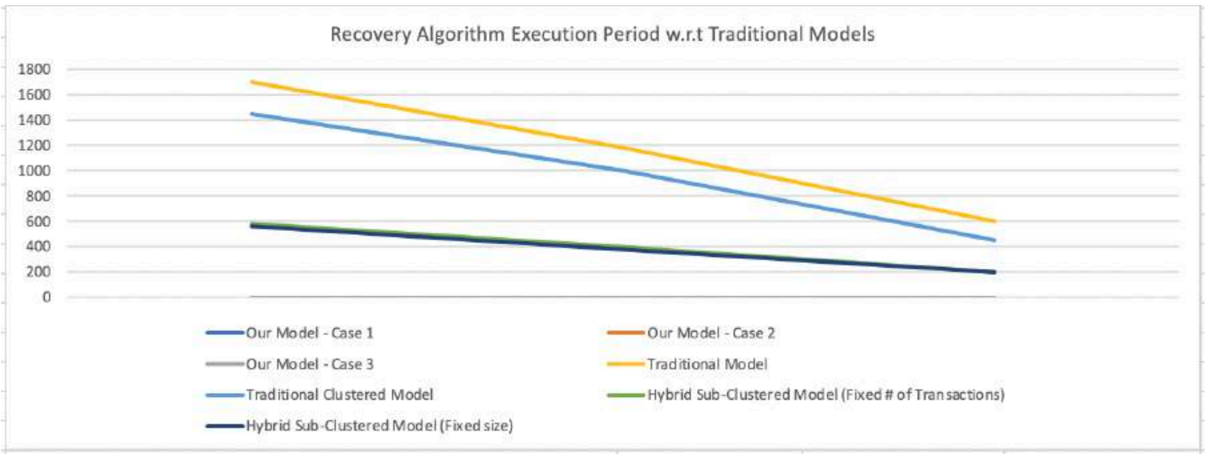


	Transactions to be Recovered	5	10	15	20	25
Recovery Time	Case 1	0.0001	0.0002	0.0005	0.0008	0.001
	Case 2	0.00012	0.00023	0.00053	0.0009	0.0012
	Case 3	0.0002	0.0004	0.001	0.0016	0.002

Figure 14 - Recovery Algorithm Execution Period w.r.t. Recovered Transaction Numbers

Malicious and damaged transactions are both included in the transaction that requires recovery. Based on Figure 14, the increase in the number of transactions the need recovery is directly related to the increase in the time spent by the recovery algorithm. However, this additional time required by our recovery algorithm is well reasonable. For instance, the increase in the number of transactions by five folds leads to an acceptable increase in the time required by the algorithm: it increases from 0.0001 to

0.001 µs in the best-case scenario (case1) and from 0.0002 µs to 0.002 µs in the worst-case scenario (case 3).



Attacker ID	50	100	150
Our Model - Case 1	0.00000012	0.00000033	0.00000051
Our Model - Case 2	0.00000011	0.00000035	0.0000005
Our Model - Case 3	0.00000025	0.0000006	0.000001
Traditional Model	1700	1180	600
Traditional Clustered Model	1450	1000	450
Hybrid Sub-Clustered Model (Fixed # of Transactions)	580	400	200
Hybrid Sub-Clustered Model (Fixed size)	560	380	200

Figure 15 - Recovery Algorithm Execution Period w.r.t. Traditional Models

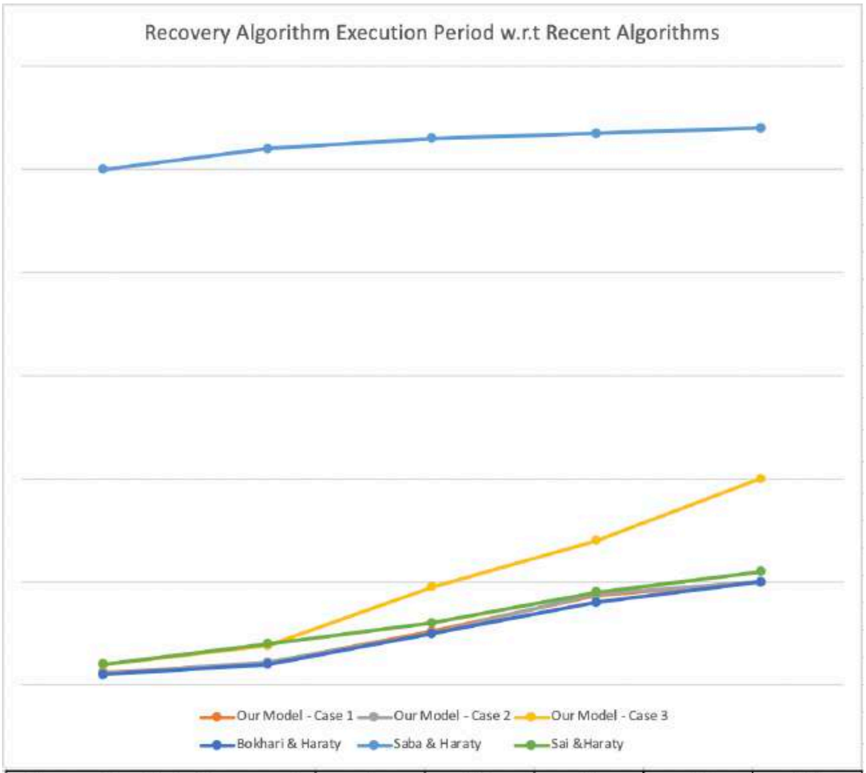
Another experiment is conducted to compare our recovery algorithm with different algorithms based on the running time. The models selected for comparison are those introduced by Haraty and Zeitunlian (2007). Figure 15 summarizes the results obtained during this experiment.

Based on the results shown in figure 14, it is clear that our model outperforms all other traditional models proposed by Haraty and Zeitunlian (2007). For example, in the case of an attacker with ID = 50, the proposed recovery algorithm in our model takes 12 x 10^{-8} ms which is at least 14,166,666,666 times faster than the slowest traditional model. However, when comparison depends on sub-clusters and fixed size, comparing our algorithm with the top traditional model increases this factor to 4,666,666,666.

When only the required part of the log file is stored in a hash table, the recovery time improves. Also, we store only the operations done by affected transaction that happened after the attack occurs.

As we already assume the rigorous serializability of history, then any preceding transaction to the malicious one is not affected.

A new experiment is conducted to compare the proposed recovery algorithm with the recent hash-based model presented by Haraty and Bokhari (2019) in terms of performance. Moreover, we compared our results to the recent matrix-based lists (Haraty and Sai, 2016) and graph-based (Saba, 2018) models. Figure 16 summarizes the results obtained during this experiment.



Transactions to be Recovered	5	10	15	20	25
Our Model - Case 1	0.00012	0.00021	0.00052	0.00087	0.001
Our Model - Case 2	0.00011	0.00022	0.0005	0.00088	0.001
Our Model - Case 3	0.0002	0.00038	0.00095	0.0014	0.002
Bokhari & Haraty	0.0001	0.0002	0.0005	0.0008	0.001
Saba & Haraty	14	15	25	29	30
Sai &Haraty	0.0002	0.0004	0.0006	0.0009	0.0011

Figure 16 - Recovery Algorithm Execution Period w.r.t. Recent Algorithms

Based on the results shown in figure 15, it is clear that our algorithm perform faster than the algorithms under comparison. except for the algorithm by Haraty and Bokhari (2019) that has very close results to our algorithm because both models depend on hash tables. Our recovery algorithm has better recovery time compared to Haraty and Sai (2016) algorithm in at least 1.5 times. Moreover, the recovery time of Haraty and Saba (2018) algorithm has the slowest running time since their work is based on traversing the received subgraph twice until it reaches back the first node which is not malicious.

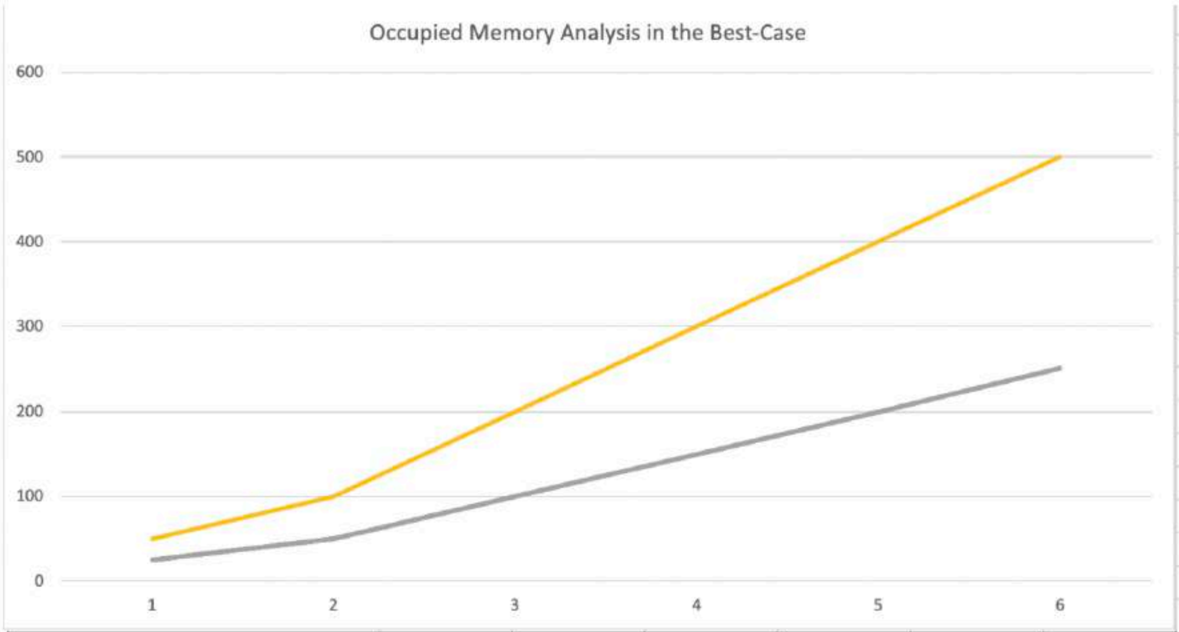
Therefore, by comparing the running time with some recent models, it is clear that our model improves the recovery and damage assessment algorithms; thus, it enhances the availability of our healthcare databases. It can also be regarded as a unique answer to the problem of recovering and assessing a distributed database.

4.4 Performance Analysis of the Memory Consumption

In this section, we prove that the proposed recovery algorithm performs also efficiently with respect to memory consumption. Due to the high scalability of the proposed algorithm, our model can be regarded as an efficient solution for large medical databases due its efficient memory consumption. The performance of our model based on memory consumption, in both best and worst cases scenarios, is compared to the performance of models presented by Haraty and Bokhari (2019), Haraty and Sai (2016), and Haraty and Saba (2018). One assumption is that the number of occupied content entries in all data structures represents the memory consumed by an algorithm. As an example, a transaction hat is dependent on four different transactions leads the algorithm to represent this dependency relation using four memory slots.

In this experiment, we assume the presence of a database operating on 100 data items and consisting of 1000 transactions. Firstly, memory consumption results of the proposed model in the best case scenario, are compared with those of Haraty and Bokhari (2019), Haraty and Saba (2018), and of Haraty and Sai (2016). Figure 17 summarizes the obtained results.

In the best-case scenario, each transaction depends only on one single transaction. Based on the results shown in figure 17, we can prove that our model has the least memory consumption. Similarly, the algorithms presented by Haraty and Bokhari (2019) and Haraty and Sai (2016) has the same memory consumption as our algorithm since all previous models insert to the data structures dependent transactions only. It is also significant to mention that the number of dependent transactions is double the number of occupied slots in memory by two.

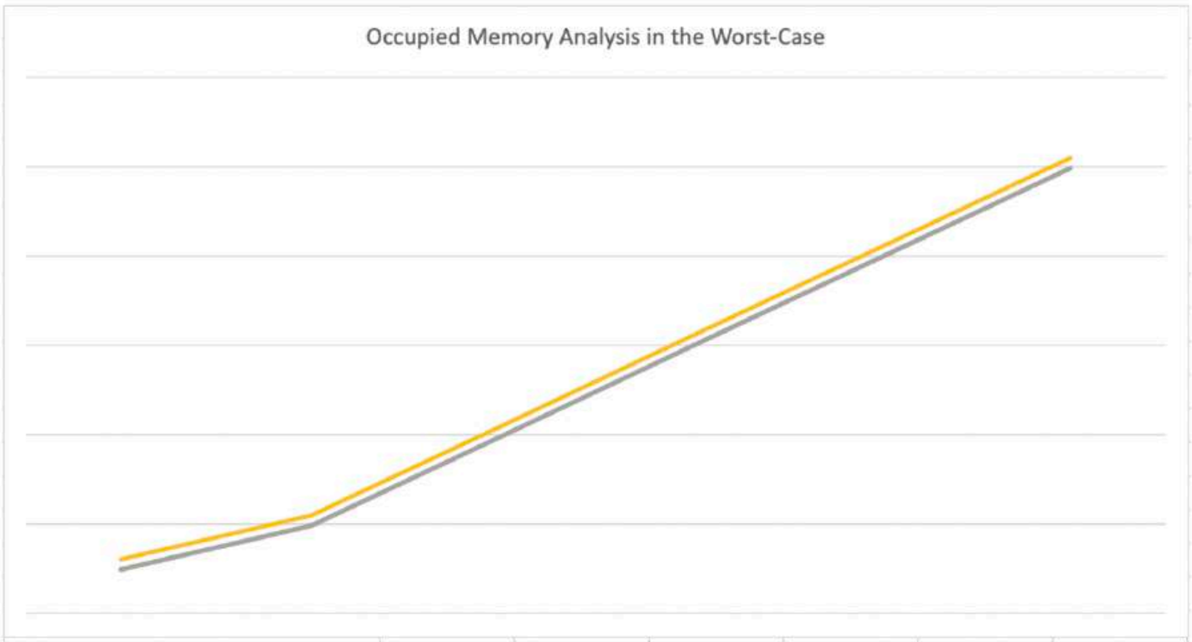


Number of dependent transactions	50	100	200	300	400	500
▬ Our Model	25	50	100	150	200	250
▬ Haraty and Bokhari	25	50	100	150	200	250
▬ Haraty and Sai	25	50	100	150	200	250
▬ Haraty and Saba	50	100	200	300	400	500

Figure 17 - Occupied Memory Analysis in Best Case Scenario

For example, in case of 50 dependent transactions, our hash table, similarly to hash table for Haraty and Bokhari (2019), contains 25 rows that is the same number of memory slots occupied. This is due to the dependence of every transaction on one other transaction only. The same applies for Haraty and Sai (2016) that utilizes a matrix of linked lists.

On the other hand, our algorithm outperforms that proposed by Haraty and Saba (2018) in terms of the number of utilized memory slots. The algorithm presented by Haraty and Saba (2018) uses graphs to store transactional dependencies; thus, each transaction has one memory slot for each connection in the graph. So, the algorithm uses 50 memory slots when the number of dependent transactions is 50. Hence, the number of dependent transactions is equal to the number of slots occupied in memory.



Number of dependent transactions	50	100	200	300	400	500
— Our Model	49	99	199	299	399	499
— Haraty and Bokhari	49	99	199	299	399	499
— Haraty and Sai	49	99	199	299	399	499
— Haraty and Saba	50	100	200	300	400	500

Figure 18 - Occupied Memory Analysis in the Worst Case

A new experiment is conducted to compare the worst-case results of the proposed model with Haraty and Saba (2018), Haraty and Sai (2016), Haraty and Bokhari (2019) in terms of memory consumption. Figure 20 summarizes the obtained results.

Every transaction depends on all other transactions in the worst-case situation; thus, the memory consumption is the highest. For example, if our database has 100 transactions, each transaction will have 99 other transactions dependent on it.

Based on the results shown in figure 18, we can prove that in the worst-case, the proposed model consumes the least memory. Similarly, the algorithms presented by Haraty and Bokhari (2019) and Haraty and Sai (2016) has the same memory consumption as our algorithm since all of these models inserts to the used data structure dependent transactions only. For example, a number of dependent transactions equal to 100, leads our algorithms to occupy 99 memory slots as each transaction depends on 99 others.

In Haraty and Saba (2018), the best-case memory consumption is maintained because a every node is connected to all other nodes. So, the slots occupied in memory and dependent transactions have equal numbers.

The proposed model in this paper is proved to be very efficient when memory consumptions is considered. This fits our needs especially when dealing with distributed databases. Also, this allows our model to improve in terms better in terms of running time as less memory is consumed.

4.5 Conclusion

The distributed hash-based model present in this thesis proves its efficiency in all the stages of the recovery process. In the assessment process, the algorithm omits all transactions the happened before the malicious one; thus, it enhances the running time. Also, we tested our algorithm in all possible cases that could happen in a distributed database. And for proper comparison, we compared the results applied on a single database and compared it to recent algorithms. Similarly, our recovery algorithm outperforms all recent and traditional algorithms.

In the last part, we provide a comparative analysis of our algorithm with other algorithms where the performance in terms of memory consumption is considered. The results prove that our algorithm uses less memory slots compared to other algorithms by using a hash table to store the required portion only of the log file. The significance of using a hash table is allowing fast retrieval and access of operations.

Chapter Five

Conclusion

The capability of sharing medical records between multiple hospitals has been a main concern for patients; thus, saving the cost of being forced to repeat the same test at multiple hospitals. Moreover, during this global panic around coronavirus, it can be much beneficial for the patient to access their medical information in any country or at any hospital in order to avoid repeating the COVID-19 test in each country he/she visits. Many studies have been proposed to find a solution for this problem, and blockchain has been found to be a good solution.

In most of the information system protection, a layered system is a used. This system starts by applying prevention methods to reduce the possibility of an attack. No one can prevent all attacks; hackers always try to find a way to breach the system. The second layer is the detection layer. In this phase, we should detect any attack before damaging or corrupting the data. Also, detection tools sometimes fail to detect the attack. Here emerges the need for of a recovery layer. During recovery, the algorithm is responsible to restore the database to its consistent sate by removing the effects of malicious attacks.

In this thesis, we present a hash-based technique for damage assessment and recovery that uses blockchain technology. Our algorithm is implemented, and the results were compared with the latest damage assessment and recovery algorithms. The results prove the superiority of the proposed algorithm over all different algorithms considered with respect to assessment, recovery and memory consumption.

For future work, we will be focusing on a new efficient scheduler mechanism for the assessment phase. A scheduler with high level of accuracy that will allow the database to continue execution during the assessment phase. Moreover, the model presented in this thesis provides numerous opportunities for further research efforts in an edge computing environment; thus, enhancing the services provided while reducing the bandwidth consumed.

References

Azaria, A. Ekblaw, T. Vieira, and A. Lippman," MedRec: Using Blockchain for Medical Data Access and Permission Man- agement," in International Conference on Open and Big Data, 2016, pp. 25-30.

Abdulwahab Alazeb and Brajendra Panda, "Ensuring Data Integrity in Fog Computing Based Health-Care Systems", Spring Nature Switzerland, 2019.

Bai, K. and Liu, P. (2009). A data damage tracking quarantine and recovery (dtqr) scheme for mission-critical database systems. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, pages 720–731, New York, NY, USA. ACM.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1986). Concur- rency Control and Recovery in Database Systems. Addison- Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bitshares - your share in the decentralized exchange. [Online]. Available: https://bitshares.org/

Bosu, Amiangshu & Iqbal, Anindya & Shahriyar, Rifat & Chakraborty, Partha. (2018). Understanding the Motivations, Challenges and Needs of Blockchain Software Developers: A Survey.

Breitbart, Y., Georgakopoulos, D., Rusinkiewicz, M., and Silberschatz, A. (1991). On rigorous transaction scheduling. IEEE Transactions on Software Engineering, 17(9):954–960.

Buterin V, et al. Ethereum white paper; 2013

Chakraborty, A., Majumdar, A. K., and Sural, S. (2010). A column dependency-based approach for static and dynamic recovery of databases from malicious transactions. International Journal of Information Security, 9(1):51–67.

Chen G. (1995) Fuzzy Functional Dependency and a Series of Design Issues of Fuzzy Relational Databases. In: Bosc P., Kacprzyk J. (eds) Fuzziness in Database Management Systems. Studies in Fuzziness, vol 5. Physica, Heidelberg.

Chen, Jieying Ma, Xiaofeng Du, Mingxiao Wang, Zhuping. (2018). A Blockchain Application for Medical Information Sharing. 1-7. 10.1109/TEMS-ISIE.2018.8478645.

D. Johnston, S. O. Yilmaz, J. Kandah, N. Bentenitis, F. Hashemi, R. Gross, S. Wilkinson, and S. Mason, The general theory of decentralized applications, dapps, GitHub, June, vol. 9, 2014.

F. Dai, Y. Shi, N. Meng, L. Wei and Z. Ye," From Bitcoin to cybersecurity: A comparative study of blockchain application and security issues," 2017 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, 2017, pp. 975-979. doi: 10.1109/ICSAI.2017.8248427.

Francesco Galati, "Blockchain as a Process: Ideologies and Motivations behind the Technology", Mar 2018.

Haraty R, Zbib M (2014) A matrix-based damage assessment and recovery algorithm. In: Innovations for community services (I4CS), pp 22–27.

Haraty R, Zbib M, Masud M (2015) Data damage assessment and recovery algorithm from malicious attacks in healthcare data sharing systems. J Peer-to-Peer Netw Appl. doi:10.1007/s12083-015-0361-z.

Haraty, R. and Zeitunlian, A. (2007). Damage assessment and recov- ery from malicious transactions using data dependency for defensive information warfare. ISESCO Science and Technology Vision, 3(4):43–50.

Hyperledger project, 2015. [Online]. Available: https://www. hyper-ledger.org/

Kim, T., Wang, X., Zeldovich, N., Kaashoek, M. F., et al. (2010). Intrusion recovery using selective re-execution. In OSDI, pages 89– 104.

Kumar, V. and Son, S. H. (1998). Database recovery. Springer.

Lala, C. and Panda, B. (2001). Evaluating damage from cyber attacks: a model and analysis. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 31(4):300–310.

Liu, P. and Jajodia, S. (2002). Trusted Recovery Models, pages 27–38. Springer US, Boston, MA.

Liu, P. and Yu, M. (2011). Damage assessment and repair in attack re-silient distributed database systems. Computer Standards Interfaces, 33(1):96 – 107. Special Issue: Secure Semantic Web.

M. Ali, J. Nelson, R. Shea, M. Freedman, Blockstack: Design and Implementation of a Global Naming System with Blockchains, Last visited on, 2016, 25(2).

M. Turkanovi, M. Hlbl, K. Koi, M. Heriko and A. Kamiali," EduCTX: A Blockchain-Based Higher Education Credit Platform," inIEEE Access, vol. 6, pp. 5112-5127, 2018.

N. Satoshi, Bitcoin: A peer-to-peer electronic cash system, Consulted, 2008, pp:1-9.

Panda B, Haque K (2002) Extended data dependency approach: a robust way of rebuilding database. In: Proceedings of the 2002 ACM symposium on applied computing, pp 445–452.

Panda B, Ragothaman P (2003) Analyzing transaction logs for effective damage assessment. In: Gudes E, Shenoi S (eds) Research directions in data and applications security, vol 128. Springer, Cambridge, pp 121-134.

Panda B, Zuo Y (2004) Fuzzy dependency and its applications in damage assessment and recovery. In: Proceedings of the 2004 IEEE Workshop on Information Assurance, pp 350–357.

PandaB, TripathyS (2000) Data dependency-based logging for defensive information warfare. In: Proceedings of the 2000 ACM symposium on applied computing, pp 361–365.

R.A.Haraty and Bahia Boukhari, "Hashing Based Assessment and Recovery Algoeithm for Information Warfare", thesis, Lebanese American University, 2019.

R.A.Haraty and M.El Sai, "Information Warfare: a lightweight matrix-based approach for database recovery", Springer-Verlag London 2016.

R.A.Haraty, M.Jaber, M.Dahini, A.Fakhereldine, "A Novel Privacy-preserving Healthcare Information Sharing Platform Using Blockchain", Lebanese American University, 2019.

R.A.Haraty, S.Kaddoura, A.S.Zekri, "Recovery of business intelligence systems: Towards guaranteed continuity of patient centric health systems through a matrix-based recovery approach", Telematics and Informatics, pp. 801-804, 2018.

Ragothaman, P. and Panda, B. (2003). Analyzing Transaction Logs for Effective Damage Assessment, pages 89–101. Springer US, Boston, MA.

Roger C. Molander, Andrew Riddile, Peter A. Wilson: Strategic Information Warfare, A New Face of War, 1996.

S. Huh, S. Cho and S. Kim," Managing IoT devices using blockchain platform,"2017 19th International Conference on Advanced Communication Technology (ICACT), Bongpyeong, 2017, pp. 464-467.

S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma and J. He," BlocHIE: A BLOCkchain-Based Platform for Healthcare Information Exchange,"2018 IEEE International Conference on Smart Computing (SMARTCOMP), Taormina, 2018, pp. 49-56.

S. King and S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, Self-Published Paper, August, vol. 19, 2012.

Saba, R. (2018). Information reconciliation through agent controlled graph model. (c2018). PhD thesis, Lebanese American University.

Sumathi, S. and Esakkirajan, S. (2010). Fundamentals of Relational Database Management Systems. Springer Publishing Company, Incorporated, 1st edition.

Weikum, G. and Vossen, G. (2001). Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Con- trol and Recovery. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

William J. Gordon, Christian Catalini, Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability, Com-putational and Structural Biotechnology Journal, Volume 16, 2018.

Xie, M., Zhu, H., Feng, Y., and Hu, G. (2008). Tracking and repairing damaged databases using before image table. In 2008 Japan-China Joint Workshop on Frontier of Computer Science and Technology, pages 36–41.

Yanjun Zuo and Panda, B. (2004). Fuzzy dependency and its applica- tions in damage assessment and recovery. In Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004., pages 350–357.

Zhang, P., White, J., Schmidt, D. C., Lenz, G., Rosenbloom, S. T. (2018). FHIRChain: Applying Blockchain to Securely and Scalably Share Clinical Data. Computational and Structural Biotechnology Journal, 16, 267-278. doi: 10.1016/j.csbj.2018.07.004.

Zheng, J., Qin, X., and Sun, J. (2007). Data dependency based recov- ery approaches in survival database systems. In Shi, Y., van Albada, G. D., Dongarra, J., and Sloot, P. M. A., editors, Computational Science – ICCS 2007, pages 1131– 1138, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zuo, Y. and Panda, B. (2006). Distributed database damage assessment paradigm. Inf. Manag. Comput. Security, 14:116–139.