www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 3 Issue 6 June, 2014 Page No. 6347-6352

Tamper Detection in Database using Forensic Analysis

Suraj Dhanawe[1,*], D.B Hanchate[2]
1Research Scholar, Department of Computer Engineering, Vidya Pratishthan College of Engineering, Baramati, 413 133, Dist-Pune, Maharashtra, India.
2Assistant Professor, Department of Computer Engineering, Vidya Pratishthan College of Engineering, Baramati, 413 133, Dist-Pune, Maharashtra, India.

Abstract— *Tampering of database can be determined through cryptographically hash functions. Hash functions determines only the tampering of data but forensic analysis functions determines only the tampering of data but forensic analysis can help to detect when data is tampered, what data is tampered and ultimately who and why. This paper presents a forensic analysis algorithm which is more efficient than forensic algorithms. It provides a solution set (all possible location of detected tampering.) and provides complete features of the solution set and its cardinality.*

***Keywords*— Security, Integrity, temporal databases, forensic analysis and candidate set.**

I. Introduction

Organizations are becoming more concerned about data security, especially as the value of our data continues to increase. However, database security often gets overlooked. So secure data storage is an important requirement for organizations. If data is tampered by insider or outsider it harms the organizations. There are so many reasons why someone modifies data. Number of reasons why someone has tampered with data. For example, a student who receives 30 marks in his Theory of computation subject, in which he needed 50 marks, could be highly tempted to change his marks to 50 in the school's database. This would be an example of how someone hacks the system from the outside, unless of course the student somehow had access to the database containing the grade. Data outsourcing is a new paradigm that allows users & organization to give their (sensitive) data to the external servers that then responsible for the storage, management and distribution. By outsourcing organization can give more importance to their main business activity rather than focusing on data. Data outsourcing provides many benefits especially for parties with limited resources for managing increasing amount of data; it also requires new privacy and security concerns.

Because of data outsourcing, new challenges in data security arose. While outsourcing the sensitive data in business organization, the integrity of the data needs to be ensured. This can be achieved by creating the different views over the data for different users. But this may lead to the problem of enforcing possible data theft or data tampering by the inside employees. In the data outsourcing scenario the data operators are monitored by trusted party. Data owners can use digital signatures to identify the persons for whom they allow to access data which leads to a notarization system. Here the Panel provided to the data owner where he/she can create a Digital signature and by using this digital signature he/ she can always notarize (authenticate the client during the performing each and every database transactions) the transactions. This Operation will be performing by a System called Validator to check the integrity of the data for each transaction. Legitimate users will be checked by the respective digital signature provided by corresponding data owner by the notarized service. In this way, the confidentiality & integrity of information does not rely on an implicit assumption of trust on the server for on the legal protection offered by specific service contracts, but instead relies on the technical guarantees provided by Notarizer and validator. Moreover in this project we are implementing one way MD5 hash key function to ensure data integrity. Where an avalanche effect between master database and application data produces a tamper detection Scenario dynamically. Then a secured and perfectly weaved Forensic analysis System helps to find who, when and where of the tampered data.

II. Literature survey

A Records management is the professional practice or discipline of controlling and governing what are considered to be the most important records of an organization throughout the records life-cycle. There has been a great deal of work on records management. Sarbanes Oxley (SOX) is a United States federal law that set enhanced standards for all U.S. public company boards and management firms. Because of SOX, top management now individually certifies the accuracy of financial information. In addition, penalties for fraudulent financial activity are much wicked. Also, SOX increased the independence of the outside attendees who review the truth of corporate financial statements, and increased the oversight role of boards of directors. In this system, a "record" is a version of a document. These systems use magnetic disks (also tape and optical drives) to provide WORM storage of tractable records. Computer forensics is now dynamic field, with more than 50 books published in the last 10 years. However, these books are 1 generally about preparing permissible evidence for a court case, through breakthrough, germination and preservation of digital evidence. There are few computer books; Mena's book [1] for these tasks, which describes an example of computer tools that can be used for forensic analysis. Forensic analysis can efficiently extract a good deal of information concerning a corruption event. Goodrich et al. introduce new techniques for data forensics by using main memory indexing structures [2]. They encode authentication information according to the way in which a data structure is organized. From this alterations can be detected easily. Their techniques are based on a novel randomness construction for non adaptive combinatorial group testing. Here, with the help of cryptographically strong, one-way hash functions, message authentication codes are built.K.E. Pavlou et [3] introducing additional hash chains to improve forensic analysis. They are addresses the problem of determining what, when, and who, by providing a systematic means of performing forensic analysis has been uncovered after such tampering. Also introduces a schematic representation "Corruption diagram" that used in intrusion investigation. These diagrams are used to fully analyze a linked sequence of hash values. They introduce more sophisticated forensic analysis algorithms namely the monochromatic, RGB, and polychromatic algorithms, and characterize the "forensic strength" of these algorithms. From the studies it is found that, there are no other competing forensic analysis algorithms for high performance databases. The time complexity and cost of these algorithms need to improve. R.T. Snodgrass et. [4] conduct an approach of using cryptographic hash functions to detect database tampering. Audit logs are considered nice practice for business systems but it is critical that they are correct and inalterable. This system proposes mechanisms within a database management system that prevent an intruder, including an auditor or an employee or even an unknown bug within the DBMS itself, from silently corrupting the audit log. With the help of implementation they showed that the validator can correctly determine if the audit log has been compromised or not. But the problem in this system is that we need to scan all pages of audited tables thus increases the time complexity. Strachey et. [5] has considered table lookup to increase the efficiency of bitwise operations. For reversing the bits in a word, they provide a logarithmic time and logarithmic space algorithm. The second algorithm requires only constant time, but the table must be of exponential space. Sheng et. [6] and coworkers et.[7] have developed efficient preimage computation algorithms Sheng present a novel learning algorithm which significantly accelerates an ATPG engine for enumerating all preimages. This algorithm effectively prunes redundant search space due to overlapped solutions. It also constructs a free BDD. with the help of this approach, we are able to compute preimages for large sequential circuits, where OBDD-based method fails. But there is need to for iterating this preimage computation procedure to multiple cycles coworkers [8] present a novel all-solutions preimage SAT solver. They propose a learning algorithm employing smaller cut sets; and a marked CNF database non-trivially combining success/conflict-driven learning. This method saves memory and avoiding inclusion of PI variables. It stores all solutions into a canonical OBDD format.

Symbol	Name	Definition
CE	Corruption Event	This is an event which may be Unauthorized unintended misuse by database administrators by unauthorized users. (E.g. Inappropriate Sensitive inappropriate the database, and security.)
I_v	Validation Interval	The time between consecutive validations.
S_d	Spatial detection Resolution	Finest granularities of CE's spatial bounds uncertainty.
T_{FVF}	Time to first validation failure	The time instance at which the CE is first detected.

Table 1 Summary of Notation Used

These algorithms are similar to the ones we introduced here. We enumerate all possible solutions. The formal verification algorithms are computing preimages of a state transition network but we present it in form of bitwise AND functions.

III. METHODS AND PROCEDURES

3.1 Problem Definition:

The task is to compute from a single target all the possible corruption events, which we term the candidate set.

3.2 The parties involved are:

As In this section, we introduce the parties involved and the underlying threat model.

1) The Database.

2) A digital notarization service: This is a company which can digitally certify the documents and then validates the document.

3) The Validator. This is a DBMS program which periodically communicates with the digital notarization service.

4) The Forensic Analyzer. This is a DBMS application which actually does forensic analysis.

5) Notarization Event (NE): the notarization of a document by the Notarization service.

6) Time of NE: time instant at which a *NE* occurred.

3.3 System Architecture:

The user performs the various transactions operation on the database which is shown in the User Application module. The operations in terms of insert update and delete on the particular table. When we perform any operation on database at that time the audit log is generated. This audit log is combination f original transaction and hashed transaction. Basically hashing is applied on transaction and generates a new digest value. This value is called as Hash value. This log is maintained every time. This log is also used for various operations. Here we use this audit log for validation. Once the hash value is generated from transaction, we also provide it to the Digital Notarization Service (DNS). DNS generate a unique notary ID for each hash value and use it for further validation.

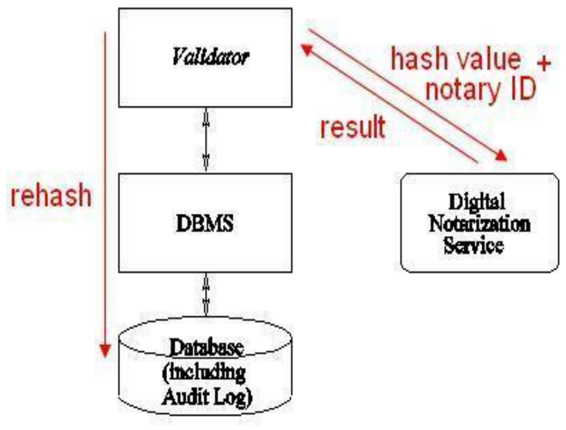

Figure 1 Normal Processing

Validator is a program that runs periodically after every sixteen hours. The generated hash value of each transaction is unique. This is the advantage of cryptography and digital signature. So when any modification on the original database it will generate different hash value. For calculation of Hash value we will also consider the start time as well as end time of the transaction. So no two hash values are same. When data is tampered, we will easily identify, is it tampered or not by checking the hash value, Notary Id and rehashed value. But identifying where it is tampered is not a simple task. So for detecting that we use Forensic Analysis Algorithm (FAA).

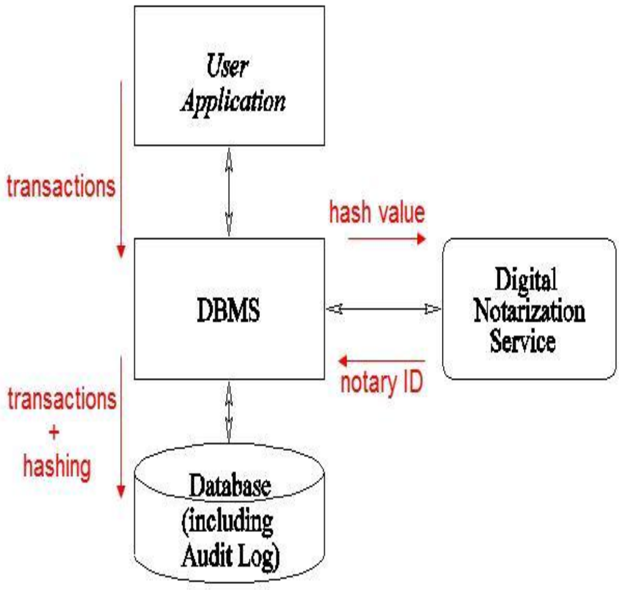

Figure 2 Validation

A. Algorithm Used:

1) Extended Tiled Bitmap Forensic Analysis Algorithm This algorithm constructs a target binary number which is input to candidate function.

/* Input: TFVF is the time of first validation failure IV is the validation interval

k is used for the creation of Ct, k S_d is the spatial detection resolution

Output: Cset, an array of binary numbers */

Function Tiled Bitmap (TFVF, IV, k, S_d)

```
1:  t ← 0 // the *target*
2:  Cset ← Ctemp ← ∅
3:  T← 1
4:  while T< TFV F do
        5:  if ¬ Val_check (c0 (T)) then
        6:  n ← lg (IV /$S_d$)
        7:  for i ← n to 1
        8:  t ← t + $2^{n-I}$ * Val_check (ci (T))
        9:  Ctemp ← candidateSet (t, n, k)
        10: for each r ε Ctemp
                11: g ← renumber(r, T, $S_d$)
                12: Cset ← Cset  U {g}
                13: T← T+ IV
    14: return Cset
```

2) Candidate Set Function

In candidate set function array is created by inspecting bits of target number from left to right & marking at each position in array how many zeros have been encountered so far? The value in array is used throughout generation of candidate solution.

/***Input:** t is a binary target number
l is the length of t
k is a function index for ANDk
Output: Ct, k is an array of binary numbers */

```
1: **Function** candidateSet (unsigned int t, int l, int k)
2: Ct, k ← new array ()
3: z ← 0
4: Z ← new array ()
5: **for** i ← l − 1 **to** 0
6: **if** t& (1 << i) = 0 **then** z ← z + 1
7: Z [l − i − 1] ← z
8: **if** k < 1 || k > 2l **then report** NOT **Defined**
9: **else if** k = 1 **then** Ct, k ← {t}
10: **else if** (z = 0 ^ k > 1) ∨ (l ≥ z > 0 ∧ k > 2z)
11: **then** Ct, k ← ∅
12: **else if** (l ≥ z > 0) ^ (2 ≤ k ≤ 2z) **then**
13: rightmost← createRightmost (t, l)
14: Ct, k ← generate (t, rightmost[l], l, Ct, k)
15: Ct, k ← funkySort (z, Ct, k)
16: **return** Ct, k
```

3) Create Rightmost

This function is called by Candidate Set function in order to construct Rightmost array

```
/*Input: t is the target bit number
l is the length of the bit representation of t
Output: the populated rightmost array */
1: Function createRightmost (unsigned int t, int l)
2: int i, j, flag
3: j ← −1
4: flag ← FALSE
5: rightmost ← new array ()
6: for i ← l − 1 to −1
7: if flag then j ← l − i − 2
8: if t& (1 << i) = 0 then flag ← TRUE
9: else flag ← FALSE
10: rightmost [l − i − 1] ← j
11: return rightmost
```

4) Generate Function

Using rightmost array & the target number Candidate set Function calls the recursive Function generate this will give the candidate set but not in Sorted order.

```
/*Input: t is the modified bit pattern at each stage of the recursion
p is the position of one of the zeros in t
l is the length of the bit representation of t
Ct, k array in which candidate granules are accumulated
Output: Ct, k contains candidate granules (unsorted) */
1: Function generate (unsigned int t, int p, int l, Array Ct, k)
2: if p = −1 then Ct, k.append (t)
3: else
4: Ct, k ← generate (t, rightmost[p], l, Ct, k)
5: Ct, k ← generate (t + (1 << (l − p − 1)),
Rightmost[p], l, Ct, k)
6: return Ct, k
```

5) Funky Sort

For sorting the candidate set it will call the function funky sort.

```
/* Input: z is the number of zeros in t
Ct, k is the result of generate ( )
Output: Ct, k sorted in ascending order */
1: function funky Sort (int z, array Ct, k)
2: sorted ← new array ()
3: indices ← new array ()
4: indices [0] ← 0
5: int i, offset, power
6: offset ← 0
7: power ← 1 << z
8: for i ← 1 to (1 << z) − 1
9: if (i& (i − 1)) = 0 then
10: power ← power >> 1
11: offset ← 0
12: indices[i] ← indices [offset] + power
13: offset ← offset + 1
14: for i ← 0 to (1 << z) − 1
15: sorted[i] ← Ct, k.get (indices[i])
16: return sorted
```

IV. Results

The focal point of the Tiled Bitmap Algorithm [9] is that it
sets out a consistent example (a "tile") of such chains over
touching portions of the database. Likewise, it inherits all the focal points of the Polychromatic Algorithm. The chains in the tile structure a bitmap which could be utilized for simple recognizable proof of the debasement area, and a logarithmic number of anchors might be utilized to diminish Rs. The other focal point of the Tiled Bitmap Algorithm is that it can locate numerous different occasions (up to D of them, i.e., all granules were defiled) something that the Monochromatic, RGB, and Polychromatic Algorithms can't. Then again it experiences false positives while the past three calculations don't.

We have used library book dataset for detecting the tamper in the database. We will detect the tamper by manually doing some changes in our dataset. For detecting such tamper we have used Extended Tiled Bitmap Algorithm (ETBA). So we expect, the result of proposed system will be much better than the previous approach. For improving such result we have used strong hashing technique and hybrid forensic analysis technique. Table1 shows the result of these approaches.

Sr.No	Person Name	Date & Time	Data field
1	Satyajit	15/05/2015,1:00PM	Book Name
2	Sachin	15/05/2015,1:10PM	Publication
3	Mrunmai	15/05/2015,1:16PM	Fine
4	Kishor	15/5/2015,1:23PM	Author Name
5	Sandip	15/05/2015,1:24PM	Price

Table1: RESULTS

V.CONCLUSION

Forensic analysis is used to detect crime which is related to tampering data. When any party compromises the data i.e. when tampering is occurred then we analyze that tampering and finding what data were altered is important. This paper suggests an Extended Tiled Bitmap Algorithm (ETBA), which is better and more useful as well as efficient than previous algorithms. This algorithm takes a logarithmic number of secure hash chains. This algorithm easily detects tampered data of more than one place. This is not achieved by previous algorithm. We have creates a database centrally, because of that all parties can easily get it and perform their operations. We will easily detect what data and where to tamper by using ETBA. So the Doctor's, government agencies protect their data from illegal threats by using this.

REFERENCES

[1] Mena, Investigative Data Mining for Security and Criminal Detection. Butterworth Heinemann, 2003.

[2] M.T. Goodrich, M.J. Atallahand, and R. Tamassia, "Indexing Information for Data Forensics," Proc. Conf. Applied Cryptography and Network Security, pp. 206-221, 2005.

[3] K.E. Pavlou and R.T. Snodgrass, "Forensic Analysis of Database Tampering," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 109-120, June 2006.

[4] R.T. Snodgrass, S.S. Yao, and C. Collberg, "Tamper Detection in Audit Logs," Proc. Int'l Conf. Very Large Databases, pp. 504-515, Sept. 2004.

[5] C. Strachey, "Bitwise Operations," Comm. ACM, vol. 4, no. 3, p. 146, Mar.1961.

[6] J. Mena, Investigative Data Mining for Security and Criminal Detection. Butterworth Heinemann, 2003.

[7] K.E. Pavlou and R.T. Snodgrass, "Forensic Analysis of Database Tampering," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 109-120, June 2006.

[8] B. Li, M.S. Hsiao, and S. Sheng, "A Novel SAT All-Solutions Solver for Efficient Preimage Computation," Proc. IEEE International Conf. Design, Automation and Test in Europe, vol. 1, Feb. 2004.

[9] The Tiled Bitmap Forensic Analysis Algorithm

[10] Kyriacos E. Pavlou and Richard T. Snodgrass, Senior Member, IEEE 2010