Forensic analysis of tampering database using RGBY algorithm

Rajni kori[1], Vinod Kumar Sripuram[2], Saurabh Suman[3] and Dr.B.B.Meshram

V.J.T.I/Computer, Mumbai, India
[1]Email: kori07rajni@yahoo.co.in
[2] Email: vinodkumargate@gmail.com
[3]Email: saurabh.mvjti@gmail.com

***Abstract* - We are using cryptographically-strong hash functions to detect tampering of a database to identify the people who have made any changes to the database silently we are applying forensic analysis algorithm and to determining who, when, and what, by providing a systematic means of performing forensic analysis after such tampering has been uncovered. We will use more sophisticated forensic analysis algorithms: RGBY to perform forensic analysis on tampered data .**

Index Terms - Forensic analysis algorithm , RGBY algorithm

1. INTRODUCTION

Due to recent federal laws e.g., Health Insurance Portability and Accountability Act: HIPAA, Canada's PIPEDA, Sarbanes-Oxley Act and standards e.g., Orange Book for security, and in part due to widespread news coverage of collusion between auditors and the companies they audit (e.g., Enron, WorldCom), which helped accelerate passage of the aforementioned laws, there has been interest within the file systems and database communities about built-in mechanisms to detect or even prevent tampering.

Audit log security is one component of more general *record management systems* that track documents and their versions, and ensure that a previous version of a document cannot be altered. As an example, *digital notarization services* when provided with a digital document, generate a *notary ID* through secure one-way hashing, thereby locking the contents and time of the notarized documents . Later, when presented with a document and the notary ID, the notarization service can ascertain whether that specific document was notarized, and if so, when *Compliant records* are those required by myriad laws and regulations (10,000 in the US) to follow certain "processes by which they are created, stored, accessed, maintained, and retained" . It is common to use Write-Once-Read-Many (WORM) storage devices to preserve such records .The original record is stored on a write-once optical disk. As the record is modified, all subsequent versions are also captured and stored, with metadata recording the timestamp, optical disk, filename, and other information on the record and its versions.

Such approaches cannot be applied directly to high-performance databases. A copy of the database cannot be versioned and notarized after each transaction. Instead, audit log capabilities must be moved into the DBMS.

An cryptographically strong one-way hash functions prevent an intruder, including an auditor or an employee or even an unknown bug within the DBMS itself, from silently corrupting the audit log . This is accomplished by hashing data manipulated by transactions and periodically *validating* the audit log database to detect when it has been altered. The question then arises, what do you do when an intrusion has been detected? At that point, all you know is that at some time in the past, data somewhere in the database has been altered. *Forensic analysis* is needed to ascertain *when* the intrusion occurred, *what* data was altered, and ultimately, *who* is the intruder.

2. TAMPER DETECTION

In this section we study the *tamper detection* approach.

There are several related ideas that in concert allow tamper detection.

1) The first insight is that the DBMS can maintain the audit log in the background, by rendering a specified relation as a *transaction-time table*. This instructs the DBMS to retain previous tuples during update and deletion, along with their insertion and deletion/update time (the start and stop timestamps), in a manner completely transparent to the user application. An important property of all data stored in the database is that it is *append-only*: modifications only add information; no information is ever deleted. Hence, if old information is changed in any way, then tampering has occurred. Oracle 10g supports transaction-time tables.
2) The second insight is that the data modified (inserted/ updated/deleted) by a transaction can be cryptographically hashed to generate a secure one-way hash of the transaction.

3) The third insight is to digitally notarize this hash value with an external notarization service. So even if the intruder has full access to the database itself, the DBMS, and even the operating system and hardware, the intruder cannot change the hash value. This makes it exceedingly difficult to make a series of changes to the audit log that generate the same hash value.

4) Finally validation service rehash the tuple and match with the previously computed hash .if matching is there then no problem , but if not match then we need to apply forensic analysis RGBY algorithm to find out where and when that tampering has occurred

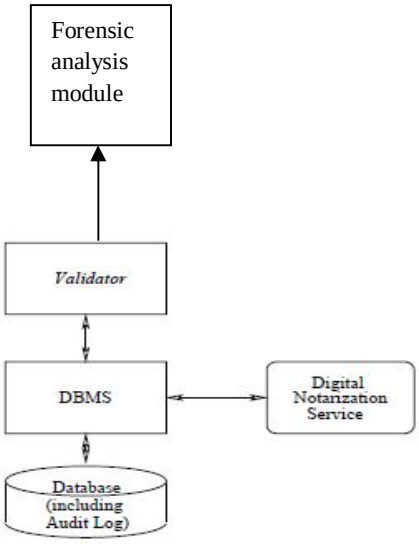

Fig (a) forensic analysis of tampered data

3. NOTATION USED IN ALGORITHM

Sysmbol	Name	Definition
CE	Corruption event	An event that compro-mises the database
VE	Validation event	The validation of the Audit log by the notarize-tion service
NE	Notarizati on event	The notarization of a document by the notarization service
lc	Corruption locus data	The corrupted data
$\mathbf{t_n}$	Notarizati on time	The time instant of a NE
$\mathbf{t_c}$	Corruption time	The time instant of a CE
$\mathbf{t_{RVS}}$	Time of most recent validation success	The time instant of the last NE whose revalidation yielded a true result
$\mathbf{t_{FVF}}$	Time of first validation failure	Time instant at which the CE is first detected
USB	Upper spatial bound	Upper bound of the spatial uncertainty of the corruption region
LSB	Lower spatial bound	Lower bound of the spatial uncertainty of the corruption region
UTB	Upper temporal bound	Upper bound of the temporal uncertainty of the corruption region
LTB	Lower temporal bound	Lower bound of the temporal uncertainty of the corruption region
V	Validation factor	The ratio Iv/I_N
N	Notarizati on factor	The ratio I_N/Rs
$\mathbf{t_v}$	Validation time	The time instant of VE
$\mathbf{t_l}$	Locus time	The time instant that lc was stored
Iv	Validation interval	The time between two successive VE's
I_N	Notarizati on interval	The time between two successive NE's
Rt	Temporal detection resolution	Finest granularity chosen to express temporal bound uncertaintiy of CE
Rs	Spatial detection resolution	Finest granularity chosen to express temporal bound uncertaintiy of CE

4. FORENSIC ANALYSIS

Once the corruption has been detected, a *forensic analyzer* springs into action. The task of this analyzer is to ascertain, as accurately as possible, the *corruption region*: the bounds on "where" and "when" of the corruption.

From this validation event, we have exactly one bit of information: validation failure. For us to learn anything more, we have to go to other sources of

information. One such source is a backup copy of the database.

We could compare, tuple-by-tuple, the backup with the current database to determine quite precisely where (the locus) of the CE. That would also delimit the corruption time, to after the locus time (one cannot corrupt data that has not yet been stored!). Then, from knowing where and very roughly when, the CIO and CSO and their staffs can examine the actual data (before and after values) to determine who might have made that change.

However, it turns out that the forensic analyzer can use just the database itself to determine bounds on the corruption time and the locus time.

- RGBY has a more regular structure and avoids some of RGB's ambiguities.
- The RGBY chains are of the same types as in the original RGB Algorithm.
- The black cumulative chains are used in conjunction with new partial hash chains, Another difference is that these partial chains are evaluated and notarized during a validation scan of the entire database.
- The introduction of the partial hash chains will help us deal with more complex scenarios, e.g., multiple data-only CEs or CEs involving timestamp corruption. The partial hash chains in RGB are computed as follows.
- We assume throughout that the validation factor V = 2 and I_N is a power of two.

  for odd i the Red chain covers NE $_{2 . i-3}$ through NE $_{2 . i-1}$

  for even i the Green chain covers NE $_{2 . i-3}$ through NE $_{2 . i-1}$

  for even i the Blue chain covers NE $_{2 . i-2}$ through NE $_{2 . I}$
- In this algorithm (RGBY) we simply introduce a new Yellow chain, computed as follows:

— for odd i the Yellow chain covers NE $_{2 . i-2}$ through NE $_{2 . i}$.

- It is indexed as Chain[color, number], where number refers to the subscript of the chain while color is an integer between 0 and 3 with the following meaning.

—if color = 0 then Chain refers to a Blue chain

—if color = 1 then Chain refers to a Green chain

—if color = 2 then Chain refers to a Red chain

—if color = 3 then Chain refers to a Yellow chain

RGBY forensic analysis algorithm

// input : t_{RVs} is the time of first validation failure

// I_N is the notarization interval

//Output : C_{set} is the set of corrupted granules

// UTB , LTB are the temporal bounds on t_c

Procedure RGBY(t_{FVF} , I_N)

1.Iv ← 2.I_N // v=2

2.C_{set} ← Φ

3.t_{RVS} ← find_ t_{RVS}(t_{FVF} , I_N)

4.USB ←t_{RVS}+I_N

5.LSB ←t_{RVS}

6.UTB ←t_{FVF}

7.LTB ← max(t_{FVF}-I_V , t_{RVS})

8.C_{set} ←C_{set} U{t_{RVS}+1}

9. v←(t_{FVF}/I_V)

10.lastchain ← chain[1+v mod 2 ,v]

11.n ←(LSB/I_N)

12.S ← [(n/2.0)]+1

13. currChain ← Chain[(n+3)mod 4 ,s]

14. While currChain ≤ lastchain do

15.If (currChain.color =Green) U (currChain.color=Yellow) then

16.succChain.number ← currChain.number+1

17. Else succChain.number ←currChain.number

18.succChain.color ← (currchain.color+1)mod 4

19.If ⌐ val_check(currchain) then

20. If ⌐ val_check(succChain)then

21.If currchain.color=Blue U currchain.color=Red then

22.C_{set} ← C_{set} U {2.(currChain.number-1) .I_N +1}

23.Else C_{set} ← C_{set} U {2.(currChain.number. I_N - I_N +1}

24. currChain ← succchain

25 . Return C_{set} , LTB< t_c≤ UTB

// input : t_{FVF} is the time of first validation failure

// I_N is the notarization interval

// output : schema Corruption if it exists

// t_{RVS} is the time of most recent validation success

Procedure find_t_{RVS}(t_{FVF} ,I_N)

1. Left← 1
2. Right←t_{FVF}
3. t_{RVS}← (left +right)/2

// since t_{RVS} may not coincide with a NE

4. If (t_{RVS} mod I_N)≠0 then t_{RVS} ←t_{RVS} - (t_{RVS} mod I_N)

5.While (⌐ Blackchain[max(1+(t_{RVs} / I_N),0)] U BlackChain[t_{RVS}/I_N])

∩(right≥left) do

6.If ⌐ BlackChain[t_{RVS}/I_N]then

7. If tRVS=0 then

8. Report "schema corruption : cannot proceed…"

9. Exit

10.If t_{RVS}-I_N< 0 then right←0 else right← t_{RVS}-I_N

11.Else

12. If $t_{RVS+}I_N > t_{FVF}$ then left ← t_{FVF} else left ← $t_{RVS+}I_N$

13.t_{RVS} ← (left+right)/2

14.If (t_{RVS} mod I_N) 0 then t_{RVS}←t_{RVS} – (t_{RVS} mod I_N)

15. Return t_{RVS}

- The RGBY Algorithm was designed so that it attempts to find more than one CE.
- However, the main disadvantage of the algorithm is that it cannot distinguish between three contiguous corruptions and two corruptions with an intervening I_N between them.

5. NOTARIZATION AND VALIDATION INTERVALS

We assumed a notarization interval of $I_N = 2$ and validation interval of $I_V = 6$. In this case, notarization occurs more frequently than validation and the two processes are in phase, with I_V a multiple of I_N . In such a scenario, we saw that the spatial uncertainty is determined by the notarization interval and the temporal uncertainty by the validation interval.

The validation interval should be equal to or longer than the notarization interval, should be a multiple of the notarization interval, and should be aligned, that is, periodically be simultaneous with notarization. When the two do align, validation should occur immediately after notarization. Thus the *validation factor V*such that $I_V = V \cdot I_N$.

6. CONCLUSION

In this paper we have discussed how to find out the tampering in database . we are using the cryptographic application apply the hash function to generate the hash value. Database management system and validation system both are using same one-way hash function to calculate the hash value. Notarization service which is a third party that generate the notarized id to the document that is being sent by database management system .after finding tampering we will apply the RGBY algorithm for tampering analysis. RGBY algorithm is more powerful then other because it found out multiple corruption region.

6. REFERENCES

[1] I. Ahn and R. T. Snodgrass, "Partitioned Storage Structures for Temporal Databases," *Information Systems*, Vol. 13, No. 4, December 1988, pp. 369–391.

[2] J. Bair, M. B¨ohlen, C. S. Jensen, and R. T. Snodgrass, "Notions of Upward Compatibility of Temporal Query Languages," *Business Informatics (Wirtschafts Informatik)* 39(1):25–34, February, 1997.

[3] K. Fu, M. F. Kaashoek and D. Mazi`eres, "Fast and secure distributed read-only file system," in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pp. 181–196, October 2000.

[4] P. A. Gerr, B. Babineau, and P. C. Gordon, "Compliance: the effect on information management and the storage industry," Enterprise Storage Group Technical Report, May 2003.

[5] S. Haber and W. S. Stornetta, "How To Time-Stamp a Digital Document," *Journal of Cryptology* 3:99–111, 1999.

[6] W. W. Hsu and S. Ong, "Fossilization: A process for establishing truly trustworthy records," IBM Research report RJ 10331, 2004.

[7] C. S. Jensen and C. E. Dyreson (eds), "A Consensus Glossary of Temporal Database Concepts—February 1998 Version," in **Temporal Databases: Research and Practice**, O. Etzion, S. Jajodia, and S. Sripada (eds.), Springer-Verlag, pp. 367–405, 1998.

[8] C. S. Jensen and R. T. Snodgrass, "Temporal Specialization and Generalization," *IEEETransactions on Knowledge and Data Engineering*, Vol. 6, No. 6, December 1994, pp. 954–974.

[9] LabCompliance,www.labcompliance.com/-e-signatures/overview.htm, viewed November 14,2005.

[10] D. Lomet, R. Barga, M. F. Mokbel, G. Shegalov, R. Wang, and Y. Zhu, "Immortal DB: transaction time support for SQL server," in *Proceedings of the International ACM Conference on Management of Data (SIGMOD)*, pp. 939–941, June 2005.

[11] D. Mazi`eres, M. Kaminsky, M. F. Kaashoek and E. Witchel, "Separating key management from file system security," in *Proceedings of the ACM Symposium on Operating Systems Principles*, pp. 124–139, December 1999.

[12]Oracle Corporation, "Oracle Database 10g Workspace Manager Overview," Oracle White Paper, May 2005.

[13] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics," ACM *Transactions on Information and System Security* 2(2):159–196, May 1999.

[14] R. T. Snodgrass, S. S. Yao, and C. Collberg, "Tamper Detection in Audit Logs," in *Proceedings of the International Conference on Very Large Databases*, pp. 504–515, Toronto, Canada, September 2004.

[15] Q. Zhu and W. W. Hsu, "Fossilized Index: The Linchpin of Trustworthy Non-Alterable Electronic Records," in *Proceedings of the ACM International Conference on Management of Data*, pp. 395–406, Baltimore, Maryland, June 2005.