A Watermarking Framework for Outsourced and Distributed Relational Databases

Sapana Rani[1], Dileep Kumar Koshley[1], and Raju Halder[1,2]

[1] Indian Institute of Technology Patna, India,
{sapana.pcs13, dileep.pcs15, halder}@iitp.ac.in
[2] HASLab, INESC TEC, Braga, Portugal,
raju.halder@inesctec.pt

Abstract. Unlike centralized databases, watermarking of distributed databases faces serious challenges for various reasons, *e.g.* (*i*) Distribution of data (*ii*) Existence of replication (*iii*) Preservation of watermarks while partitioning and distributing databases, etc. In this paper, we propose a novel watermarking technique for distributed relational databases considering a generic scenario that supports database outsourcing and hybrid partitioning. Our approach addresses the above challenges in an effective way by maintaining meta-data and by making the detection phase partition independent. To the best of our knowledge, this is the first proposal on watermarking of distributed relational databases that supports database outsourcing and its partitioning and distribution in a distributed setting.

Key words: Watermarking, Distributed Databases, Security

1 Introduction

Enormous amount of data is being generated day by day due to the rapid development of internet-based technologies. This huge data need to be stored and managed effectively. Distributed database system is one of the best solutions aiming at improving data-sharing, local autonomy, availability, reliability, performance, etc [18]. To achieve all these, distributed system divides databases into various partitions (fragments) and stores them physically across various locations along with the associated database-applications. These locations are interconnected by means of communication networks. In recent time, there is a trend to outsource databases, as a cost effective solution, to third party who has required resources to support such distributed settings. This can be understood as three level hierarchy depicted in figure 1.

Database contents are always prone to various threats, *e.g.* illegal reselling, ownership claim, tamperation, copyright infringement, etc [1]. As an effective solution, database watermarking has emerged as a promising technique to detect or prevent such kind of threats. This embeds some kind of information (known as watermark) into data of the database using a secret key which is extracted

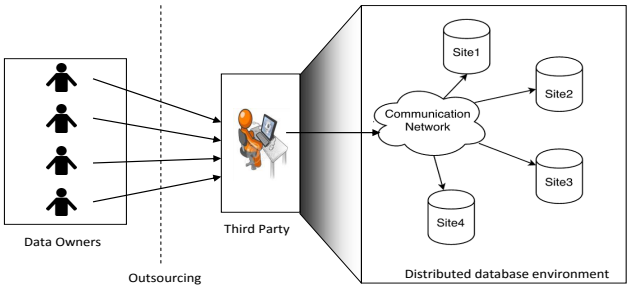

Fig. 1. Three-level hierarchy of distributed database system

later to reason about a suspicious database. Figure 2 depicts watermark embedding and detection process where a watermark W is embedded into the original database using a private key K (known only to the owner) and later the detection process is performed on any suspicious database using the same private key K by extracting and comparing the embedded watermark (if present) with the original watermark information [4].

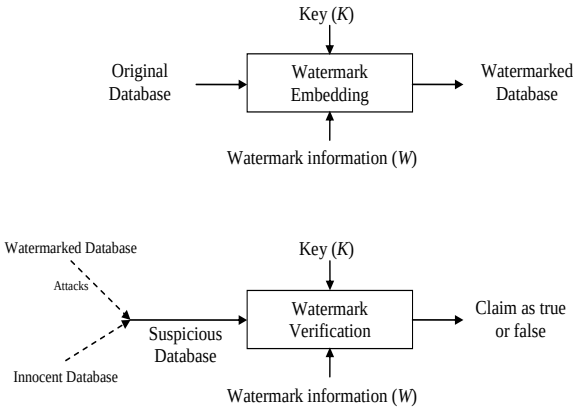

Fig. 2. Basic Watermarking Technique [4]

Like centralized databases, distributed databases also suffer from all the above-mentioned threats. However, the existing watermarking frameworks for centralized databases [1, 9, 13, 14] are not directly applicable to address those threats in case of distributed databases for the following reasons: (*i*) Distribution of data (*ii*) Existence of replication (*iii*) Preservation of watermarks while

performing partitioning and distribution by third party, etc. To the best of our knowledge, till now there is no significant contribution in case of watermarking of distributed relational database systems. Two related works in this direction are found in [21, 6]. Authors in [21] proposed a real-time watermarking technique for any kind of digital contents which are distributed among a group of parties in hierarchical manner. However, their proposal has not considered any kind of relational databases and their partitioning over distributed environment. The major drawback is that the data owner has to extract all the watermarks from top to bottom in the hierarchy during verification. Authors in [6], although title refers, have not addressed any challenge in distributed database scenario. In fact, the main technical contributions have not considered any distributed scenario at all.

All the above facts motivate us to propose a novel watermarking technique for distributed database system. The major contributions in this paper are:

- We consider a more generic scenario of distributed relational database systems that supports Database Outsourcing and Hybrid Database Partitioning (*i.e.* both vertical and horizontal partitioning).
- We propose a watermarking framework for distributed databases which allows us to apply any existing suitable centralized database watermarking algorithm separately for each database-partition. However, as the choice of suitable algorithm depends on many factors of each partition (*e.g.* capacity, cover type, etc.), we keep this out of the scope of this work.
- We consider key management scheme in such a way to make the watermarked database more robust *w.r.t.* various threats.
- Most importantly, our proposal aims at making the embedded watermark safe *w.r.t.* database partitioning and distribution by third party. Moreover, the detection phase is completely partition-independent.

This is worthwhile to mention that our approach is suitable for static partitioning and infrequent dynamic partitioning [22], where in the later case a re-watermarking is necessary to make the detection partition-independent.

The structure of the paper is as follows: Section 2 describes briefly the existing watermarking techniques in the literature. Section 3 discusses the proposed watermark embedding and detection technique for distributed databases. The experimental results are discussed in section 4. Finally we conclude our work in section 5.

2 Related Works

A series of works on watermarking of centralized databases has been proposed for last 15 years [1, 8, 20, 14, 15, 5, 19, 7, 13, 12]. A comprehensive survey can be found in [9]. Among these, a large number of proposals [1, 19, 13, 25] refer to distortion-based watermarking, whereas many others [8, 5, 14, 12, 3] refer to distortion-free watermarking of relational databases. All these techniques consider various attribute-types ranging from numerical, categorical, string, etc. as cover

to embed watermarks. Among the recently proposed works, an extensive survey on reversible watermarking techniques for relational database is reported in [12]. These techniques ensure original data recovery from watermarked data. A fragile zero-distortion watermarking technique for textual relational database is proposed in [3]. This scheme is based on local characteristics of the relation itself such as frequencies of characters and text length to generate the watermark aiming at preserving data integrity and data quality. Authors in [5] proposed a new approach based on fragile zero watermarking for the authentication of numeric relational data. Here the database relation is partitioned into independent square matrix groups and the watermark is generated using the determinant and minor of the generated square matrix. To protect integrity of database relations, Khan and Hussain [14] proposed a fragile scheme based on zero watermarking technique extracting the local characteristics of the database content, *e.g.* frequency distribution of digits, lengths, ranges of data values, etc. The proposed technique in [13] embeds each bit of a multibit watermark (generated from datetime) in every selected tuple for having maximum robustness even if an attacker is somehow able to successfully corrupt the watermark in some selected part of the data set.

As already mentioned in the previous section, authors in [21, 6] proposed watermarking technique in the context of distributed relational databases. However, they have not considered the core properties of distributed scenario during watermark embedding and detection. To be more precise, [21] considers digital contents which are distributed among a group of parties in hierarchical manner. Similarly, the main technical contributions in [6] have not considered any distributed scenario at all.

3 Proposed Watermarking Technique

In this section, we propose a novel watermarking technique for distributed databases. The proposal is based on the scenario where database owner outsources data to a third party as depicted in figure 1, assuming that third party has required resources to manage it. Let us describe in detail each of the phases of our proposed watermarking technique.

3.1 Watermark Embedding

The watermark embedding phase consists of the following three phases:

Phase 1: Initial exchange of partition information

Data owner will initiate this process to exchange some basic information with the third party in order to obtain some initial information about the partitioning and distribution of the database.

Let DB_schema be a relational database schema. Let INF be a set of specifications and requirements about the database and its associated applications,

which must be preserved after partitioning and distribution by the third party. For example, *INF* may include confidentiality and visibility constraints [24], user access information [18], query behaviours [2], etc.

To start this process, the data owner provides DB_schema and *INF* to the third party. As a result, the third party will send back to the owner a partition overview ψ of the database. This partition overview may be a set of partitions where each partition is either a subset of attributes (vertical partitioning) or a subset of tuples with common properties (horizontal partitioning) or both (hybrid partitioning).

Let us formalize the partition overview ψ. Let t_schema be a schema of a database relation that belongs to DB_schema. The horizontal partitioning of t_schema is formally represented by $\langle t\_schema, f_h \rangle$ where A is the set of all attributes in t_schema and f_h is a partial function defined over A. For instance, f_h can be a mapping of A to a set of properties represented by first order predicate formulas [10] or any other algebraic functions like hash [2]. The horizontal partitioning of tuples in an instance of t_schema is performed by using f_h where each partition contains tuples with similar properties. Similarly, the vertical partitioning can be formalized by $\langle t\_schema, f_v \rangle$ where $\wp(A)$ is the power set of A and $f_v(A) \subseteq \wp(A)$. Observe that the definitions of f_h and f_v depend on *INF* in order to satisfy the specifications and requirements. Therefore, in general, the hybrid partitioning is formally defined as $\langle t\_schema, f_h, f_v \rangle$. The partition overview ψ of DB_schema satisfying *INF* is formally defined as

$$\psi \triangleq \{ \langle t\_schema, f_h, f_v \rangle \mid t\_schema \in DB\_schema \}$$

Phase 2: Watermarking by data owner

Given a partition overview ψ (provided by the third party) and a secret key K, the data owner embeds watermark into the original database DB. To this aim, the data owner performs the following two:

- *Key Management*: Obtain a set of n different sub-keys $\{K_i \mid i = 1, 2, \ldots, n\}$ from K where n represents the number of fragments obtained from the partition overview ψ (denoted $|\psi|$), and
- *Watermark Embedding*: Embed the watermark W into DB using n sub-keys.

Let us describe each in detail:

Key Management. As our aim is to make the watermark detection partition-independent, the prime challenge here is to select private key K properly and to watermark the database by using K in such a way that partitioning of the database DB by third party must not affect this watermarking.

Watermarking by the data owner considering the future partitioning (by third party) leads to following four possibilities:

- Same Watermark, Same Key: Embedding same watermark into different partitions using same key.

- Different Watermark, Same Key: Embedding different watermarks into different partitions using same key.
- Same Watermark, Different Key: Embedding same watermark into different partitions using different keys.
- Different Watermark, Different Key: Embedding different watermarks into different partitions using different keys.

In our approach, we consider "Same Watermark, Different Key" scenario in which if somehow the watermark is extracted at one site, it will not expose the watermarks embedded into other database-partitions at other sites. Moreover, this serves the purpose of making watermark detection partition-independent as well.

To achieve our objective, we consider k out of n secret sharing schemes [23, 17] which states that the secret key K can be recovered from any set of k shares (where k is a threshold) out of n shares of K. Observe that this reduces the challenges in managing and distributing large number of independent keys for all database-partitions in distributed settings. In our approach, we use Mignotte's scheme as this leads to small and compact shares [11]. Algorithm 1 provides detail steps of the Mignotte's scheme to obtain n shares of secret key. Here $n = |\psi|$ that indicates the number of partitions. We have a secret key K which is partitioned into different shares, $\{K_i \mid i = 1, 2, \ldots, n\}$ that is used in watermarking of various partitions.

Algorithm 1 `KEY-COMPUTATION`

Input : Partition overview ψ, Secret key K
Output : Shares $\{K_i \mid i = 1, 2, \ldots, n\}$ of the secret key K

1: Let $n = |\psi|$ and k be a threshold, where $|\psi|$ represents the number of partitions.
2: Choose n pairwise co-prime integers $m_1, m_2, ..., m_n | (m_1 \times ... \times m_k) > (m_{n-k+2} \times ... \times m_n)$.
3: Select secret key K such that $\beta < K < \alpha$ where $\alpha = (m_1 \times ... \times m_k)$ *and* $\beta = (m_{n-k+2} \times ... \times m_n)$.
4: Compute shares of secret key as $K_i = K \bmod m_i$ // $\forall\ i \in 1$ to n.
5: Return $\{K_i \mid i = 1, 2, \ldots, n\}$.

Watermark Embedding. Data owner watermarks the database DB using shares $\{K_i \mid i = 1, 2, ..., |\psi|\}$, obtained from the secret key K by using Algorithm 1. Suppose DB^i represents i^{th} database-partition in the partition overview ψ. The distributed watermarking is formalized as:

$$\begin{aligned}
&\texttt{DistWM_Embed}(DB, \psi, W, K) \\
&\quad = \bigcup_{i \in 1 \ldots |\psi|} \texttt{WM_Embed}(DB^i, W, K_i) \\
&\quad = \bigcup_{i \in 1 \ldots |\psi|} DB_w^i \\
&\quad = DB_w
\end{aligned}$$

Observe that DB^i is watermarked using the share K_i. Owner may use any existing suitable centralized database watermarking algorithm `WM_Embed` [3] to watermark each partition DB^i, $i \in 1 \ldots |\psi|$. Once watermarked, data owner then outsources the watermarked database DB_w to the third party.

The overall watermarking process performed by the data owner is summarized in Algorithm 2.

Algorithm 2 `Dist_WM_Embed`

Input : Database DB, Watermark W, Specifications INF, Secret key K.
Output : Watermarked database DB_w.

1: Send $\{DB\_schema, INF\}$ to the third party.
2: Receive a partition overview ψ computed from $\{DB\_schema, INF\}$ by the third party.
3: Generate n shares $\{K_i \mid i = 1, 2, ..., n\}$ of the secret key K using algorithm `KEY-COMPUTATION`(ψ, K), where $n = |\psi|$.
4: Watermark the database: $DB_w = \bigcup_{i \in 1 \ldots |\psi|}$ `WM_Embed`(DB^i, W, K_i), where $DB_i \in \psi$.
5: Send the watermarked database DB_w to the third party.

Phase 3: Partitioning and distribution by third party

Once the third party receives the watermarked database DB_w from the data owner (using Algorithm 2), the third party partitions and distributes DB_w as per ψ. In addition, the third party maintains a metadata table which contains information about the data distribution over the servers. The metadata information consists of partition ID P_i, property description of the data in the partition in the form of first-order formula, the server ID S_j where partition P_i is located, etc. Table 1 depicts a hypothetical example of metadata, where $A_1, \ldots, A_5$ represent attributes.

Table 1. Example of Metadata

Partition ID	Partition Description		Server ID
	Schema	Properties	
P_1	$\{A_1, A_2, A_3\}$	$A_3 \leq \texttt{avg}(A_3)$	S_3
P_2	$\{A_1, A_4, A_5\}$	$A_4 \leq \texttt{avg}(A_4)$	S_1
P_3	$\{A_1, A_2, A_3\}$	$A_3 > \texttt{avg}(A_3)$	S_2
P_4	$\{A_1, A_4, A_5\}$	$A_4 > \texttt{avg}(A_4)$	S_4

3.2 Watermark Detection

If the data owner finds any suspicious database partition or a part of it (denoted DB_s), he/she will initiate the detection process. The main issue that arises in this phase is how to know the actual key which was used at the time of watermark insertion for DB_s. For this purpose, the data owner communicates with

[3] The selection of suitable watermarking algorithms is an orthogonal research topic.

the third party and obtains a partition ID P_i based on the matching of the suspicious database data with the property description of P_i in the metadata table. Once the partition ID P_i is obtained, the owner uses the Mignotte's scheme [17] to obtain i^{th} share K_i from K. This way using K_i the data owner extracts a watermark $W^{'}$ by applying `WM_Detect`(DB_s, K_i) and compares it with the original watermark W. Algorithm 3 formalizes the watermark detection phase. Observe that detection algorithm `WM_Detect` corresponds to the watermark embedding algorithm `WM_Embed`.

Algorithm 3 `Dist_WM_Detect`

Input : Suspicious database DB_s.
Output : Watermark detection as *true* or *false*.

1: Data owner asks third party for partition ID of the suspicious database DB_s.
2: Third party refers metadata and returns P_i based on the matching of data of DB_s with P_i's property description in metadata.
3: Data owner computes i^{th} share K_i by applying Mignotte's scheme and extracts watermark $W^{'} =$ `WM_Detect`(DB_s, K_i).
4: If $W^{'} \approx W$, claim := *true* else claim := *false*.

4 Experimental Analysis

We have performed experiment on a real data set Forest Cover Type [16] that contains 581012 tuples. We have added an extra attribute *id* to the dataset that serves as primary key. The experiment is performed on a server configured with Intel Xeon Processor, 3.07 GHz clock speed, 64 GB RAM, and Linux operating system.

For partition-level watermarking algorithm `WM_Embed`, we have used AHK algorithm [1]. The notations used in our experiment are defined below:

Notations	Description
$Count$	no. of tuples used for particular experiment
ν	no. of attributes used for marking and detection in the relation
γ	fraction of tuples used in the experiment
χ	no. of least significant bit available for marking in an attribute.
TC	total count that is marked during embedding.
α	significance level of the test for detecting a watermarking
τ	threshold parameter for detecting a watermark.
match-count	count of the marks matched successfully during detection.

Tables 2 depicts the watermark embedding results (watermark embedding time in millisecond) for various number of partitions in partition-overview ψ.

Tables 3, 4, 5 and 6 depict watermark detection results (detection time in millisecond, successfully detected or not, etc.) for various number of partitions after random value modification attack took place on 0%, 50% and 90% of the tuples in the partitions.

The rate of watermark detection $(=(\textit{match-count}/TC) \times 100)$ after performing random value modification attack in each partition is graphically shown in Figure 3. The observations are summarized below:

Table 2. Results of Watermark Embedding

Count	$\|\psi\|$	ν	χ	γ	TC	time (msec)
581012	2	10	15	50	11851	14213425
581012	4	5	15	50	23702	27380395
581012	6	3	15	50	35553	44380295
581012	8	2	15	50	47404	57248920

Table 3. Watermark detection results after random value modification attack in case of 2 partitions of the data-set

Partition		**Random Modification Attack**		**Detection**			
No.of fragments	Count	percent updated	χ-bit updated	match count	τ	time (msec)	Detect?
2	337195	0	NA	6791	3395	4610860	✓
		50	8	6341	3395	4104676	✓
		90	8	5977	3395	4247292	✓
		90	10	4741	3395	4395566	✓
	243817	0	NA	5060	2530	2668006	✓
		50	8	4682	2530	2239336	✓
		90	8	4449	2530	2495227	✓
		90	10	3525	2530	2303253	✓

Table 4. Watermark detection results after random value modification attack in case of 4 partitions of the data-set

Partition		**Random Modification Attack**		**Detection**			
No.of fragments	Count	percent updated	χ-bit updated	match count	τ	time (msec)	Detect?
4	337195	0	NA	6791	3395	3086946	✓
		50	8	6317	3395	3148158	✓
		90	8	5988	3395	3248881	✓
		90	10	4759	3395	3052083	✓
	243817	0	NA	5060	2530	1609398	✓
		50	8	4618	2530	1668649	✓
		90	8	4446	2530	1677063	✓
		90	10	3535	2530	1677395	✓
	280962	0	NA	5753	2876	2192104	✓
		50	8	5509	2876	2099414	✓
		90	8	5320	2876	2208180	✓
		90	10	3990	2876	2198221	✓
	300050	0	NA	6098	3049	2477041	✓
		50	8	5824	3049	2630520	✓
		90	8	5629	3049	2501038	✓
		90	10	4307	3049	2473426	✓

- For 0% value modification, we have 100% detection rate for all the partitions.
- For 50% value modification, the rate of detection is 93%, 93.9%, 95.2% and 98.1% for 2, 4, 6 and 8 partitions respectively.
- For 90% value modification, the rate of detection is 88%, 90.2%, 93.1% and 97% for 2, 4, 6 and 8 partitions respectively.
- Therefore, detection rate in presence of random value modification attack increases as we increase the number of partitions.

We have also computed average detection time (in presence of random value modification attack) for all the partitions from tables 3, 4, 5 and 6 which is depicted in figure 4. Observe that the average detection time decreases as we increase the number of partitions.

Table 5. Watermark detection results after random value modification attack in case of 6 partitions of the data-set

Partition		Random Modification Attack		Detection			
No.of fragments	Count	percent updated	χ-bit updated	match count	τ	time (msec)	Detect?
6	334876	0	NA	5646	2832	2977686	✓
		50	8	5646	2832	2857632	✓
		90	8	5646	2832	2829288	✓
		90	10	5646	2832	2768492	✓
	246136	0	NA	5609	3093	1568444	✓
		50	8	5490	3093	1562750	✓
		90	8	5351	3093	1583879	✓
		90	10	4872	3093	1522225	✓
	371245	0	NA	6290	3165	3588523	✓
		50	8	6290	3165	3611780	✓
		90	8	6290	3165	3473582	✓
		90	10	6290	3165	3479286	✓
	209767	0	NA	4912	2760	1126138	✓
		50	8	4488	2760	1171351	✓
		90	8	4217	2760	1124066	✓
		90	10	4198	2760	1102890	✓
	280877	0	NA	2787	2834	1994684	×
		50	8	2781	2834	2089529	×
		90	8	2809	2834	2009919	×
		90	10	2809	2834	2013371	×
	300135	0	NA	3276	3091	2341871	✓
		50	8	3240	3091	2317585	✓
		90	8	3223	3091	2307532	✓
		90	10	3223	3091	2221183	✓

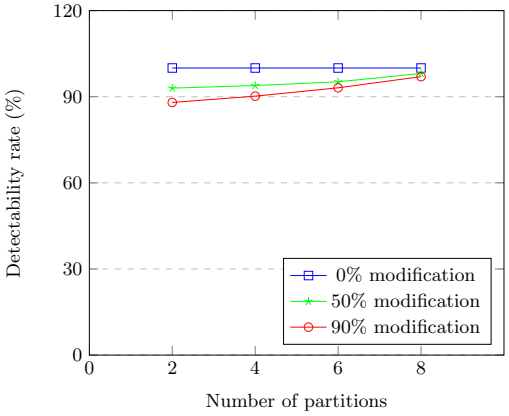

Fig. 3. Watermark detection rate after random value modification attack

5 Conclusions and Future Plans

In this paper, we proposed a novel watermarking technique for distributed database that supports hybrid partitioning. The detection phase in the proposed scheme is partition independent. The key management scheme that we have considered makes the watermark more robust against various attacks, as if anyhow some partitions are attacked, it will not affect the watermarks in other database-partitions. The experimental results show the strength of our approach by an-

Table 6. Watermark detection results after random value modification attack in case of 8 partitions of the data-set

Partition		**Random Modification Attack**		**Detection**			
No.of fragments	Count	percent updated	χ-bit updated	match count	τ	time (msec)	Detect?
8	330969	0	NA	6721	3360	2781232	✓
		50	8	6205	3360	2699055	✓
		90	8	5887	3360	2650933	✓
		90	10	4678	3360	2630398	✓
	250043	0	NA	5130	2565	1600756	✓
		50	8	4756	2565	1568929	✓
		90	8	4545	2565	1528359	✓
		90	10	3612	2565	1523670	✓
	337195	0	NA	6791	3395	2790800	✓
		50	8	6791	3395	2795596	✓
		90	8	6791	3395	2768237	✓
		90	10	4761	3395	2775518	✓
	243817	0	NA	5060	2530	1523984	✓
		50	8	5060	2530	1466059	✓
		90	8	5060	2530	1471725	✓
		90	10	3541	2530	1490088	✓
	335646	0	NA	6833	3416	2836748	✓
		50	8	6833	3416	2791317	✓
		90	8	6833	3416	2809398	✓
		90	10	4804	3416	2765616	✓
	245366	0	NA	5018	2509	1484451	✓
		50	8	5018	2509	1465203	✓
		90	8	5018	2509	1508419	✓
		90	10	3489	2509	1480657	✓
	246035	0	NA	4956	2478	1516136	✓
		50	8	4956	2478	1475345	✓
		90	8	4956	2478	1514514	✓
		90	10	3469	2478	1526597	✓
	334977	0	NA	6895	3447	2676237	✓
		50	8	6895	3447	2705557	✓
		90	8	6895	3447	2817721	✓
		90	10	4830	3447	2720335	✓

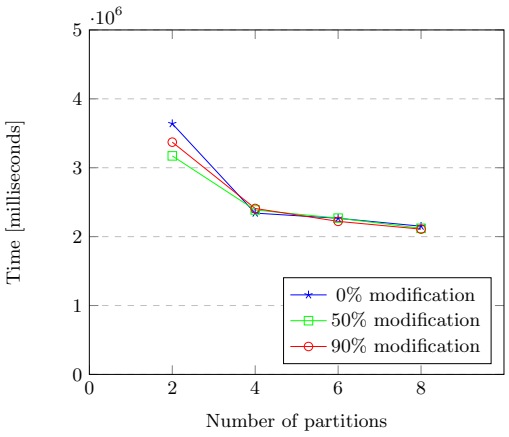

Fig. 4. Average detection time after random value modification attack for $\gamma = 50$

alyzing the detection rate with respect to random modification attack. To the best of our knowledge, this is the first work that supports database outsourcing

and its partitioning and distribution in a distributed setting. The future works aim to design an efficient watermarking technique for each partition leading to possible improvements in this proposed generic framework and to extend it to the case of big data and cloud computing environment.

Acknowledgment

This work is partially supported by the Council of Scientific and Industrial Research (CSIR), India, and by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project "POCI-01-0145-FEDER-006961", and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia as part of project "UID/EEA/50014/2013".

References

1. Agrawal, R., Haas, P.J., Kiernan, J.: Watermarking relational data: framework, algorithms and analysis. The VLDB Journal 12(2), 157–169 (2003)
2. Agrawal, S., Narasayya, V., Yang, B.: Integrating vertical and horizontal partitioning into automated physical database design. In: Proc. of the ACM SIGMOD international conf. on Management of data. pp. 359–370 (2004)
3. Alfagi, A.S., Manaf, A.A., Hamida, B., Olanrewajub, R.: A zero-distortion fragile watermarking scheme to detect and localize malicious modifications in textual database relations. Journal of Theoretical and Applied Information Technology 84(3), 404 (2016)
4. Bhattacharya, S., Cortesi, A.: A generic distortion free watermarking technique for relational databases. In: Proceedings of the 5th International Conference on Information Systems Security. Springer LNCS, Kolkata, India (December 2009)
5. Camara, L., Li, J., Li, R., Xie, W.: Distortion-free watermarking approach for relational database integrity checking. Mathematical Problems in Engineering 2014 (2014)
6. El-Bakry, H.M., Hamada, M.: A developed watermark technique for distributed database security. In: Computational Intelligence in Security for Information Systems 2010, pp. 173–180. Springer (2010)
7. Farfoura, M.E., Horng, S.J., Wang, X.: A novel blind reversible method for watermarking relational databases. Journal of the Chinese Institute of Engineers 36(1), 87–97 (2013)
8. Halder, R., Cortesi, A.: A persistent public watermarking of relational databases. In: Proceedings of the 6th International Conference on Information Systems Security. pp. 216–230. Springer LNCS 6503, India (2010)
9. Halder, R., Pal, S., Cortesi, A.: Watermarking techniques for relational databases: Survey, classification and comparison. Journal of Universal Computer Science 16(21), 3164–3190 (2010)
10. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and reasoning about systems. Cambridge University Press (2004)

11. Iftene, S.: General secret sharing based on the chinese remainder theorem with applications in e-voting. Electronic Notes in Theoretical Computer Science 186, 67–84 (2007)
12. Iftikhar, S., Kamran, M., Anwar, Z.: A survey on reversible watermarking techniques for relational databases. Security and Communication Networks 8(15), 2580–2603 (2015)
13. Kamran, M., Suhail, S., Farooq, M.: A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints. IEEE Transactions on Knowledge and Data Engineering 25(12), 2694–2707 (2013)
14. Khan, A., Husain, S.A.: A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations. The Scientific World Journal 2013 (2013)
15. Khanduja, V., Chakraverty, S., Verma, O.P., Singh, N.: A scheme for robust biometric watermarking in web databases for ownership proof with identification. In: Active Media Technology, pp. 212–225. Springer (2014)
16. Forest Cover Type: `kdd.ics.uci.edu/databases/covertype/covertype.html`
17. Mignotte, M.: How to share a secret. In: Cryptography, pp. 371–375. Springer (1983)
18. Özsu, M.T., Valduriez, P.: Principles of distributed database systems. Springer Science & Business Media (2011)
19. Rani, S., Kachhap, P., Halder, R.: Data-flow analysis-based approach of database watermarking. In: Advanced Computing and Systems for Security, pp. 153–171. Springer (2016)
20. Rao, B.V., Prasad, M.V.: Subset selection approach for watermarking relational databases. In: Proc. of the Second International Conference on Data Engineering and Management, pp. 181–188. Springer (2012)
21. Razdan, R.: Real-time, distributed, transactional, hybrid watermarking method to provide trace-ability and copyright protection of digital content in peer-to-peer networks (Mar 7 2001), uS Patent App. 09/799,509
22. Rodríguez, L., Li, X.: A dynamic vertical partitioning approach for distributed database system. In: Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on. pp. 1853–1858. IEEE (2011)
23. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
24. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragments and loose associations: Respecting privacy in data publishing. Proceedings of the VLDB Endowment 3(1-2), 1370–1381 (2010)
25. Zhou, X., Huang, M., Peng, Z.: An additive-attack-proof watermarking mechanism for databases' copyrights protection using image. In: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing. pp. 254–258. Seoul, Korea (2007)