International Journal of Information Management Data Insights 5 (2025) 100320

Contents lists available at ScienceDirect

International Journal of Information Management Data Insights

journal homepage: www.elsevier.com/locate/jjimei

A robust scheme for securing relational data incremental watermarking ☆

Maikel Lázaro Pérez Gort *, Agostino Cortesi

Ca' Foscari University of Venice, Via Torino 155, Venice, 30170, Italy

ARTICLE INFO

Keywords:
Incremental watermarking
Ownership protection
Public system
Relational data
Security

ABSTRACT

Watermarking techniques aim to protect relational databases by embedding on them a copyright signal known as the watermark without imposing additional restrictions. However, unlike other digital assets, such as multimedia data, relational data are often subject to frequent updates that may dramatically compromise the quality of the embedded watermark. Hence, it is relevant to implement incremental watermarking for this type of data. Although incremental watermarking is defined in theory as the requirement of generating and inserting a mark whenever data is inserted or updated in a watermarked database (if the new value requires marking), its practical deployment is often ignored in the validation of proposed techniques, possibly due to how its deployment affects other requirements, such as the public system and security. In this work, we present different architectural approaches that, rather than conflicting with security and the public system, are built upon and contribute to them. The experimental results validate their applicability in terms of deployment, portability, scalability, and performance. As an architectural proposal, our work can be applied to different watermarking techniques, regardless of their particularities and the protected databases, making the preservation and enhancement of the watermark possible. Thus, we face the silent threats to security posed by opportunistic malicious operations in the absence of incremental watermarking.

1. Introduction

The use of digital technologies increases daily to the point where it is unthinkable for an organization to adequately function without computer networks. Particularly, as the Internet of Things (IoT) and the Fourth Industrial Revolution (4IR) gain more relevance among organizations and clients, online data access and management face significant challenges. Data that are not protected can be tampered with or stolen (Pavlou & Snodgrass, 2013). In the case of data tampering, organizations are left vulnerable to using compromised information for decision-making. On the other hand, in the event of data theft, the leakage of sensitive information compromises a corporation's reputation. Both outcomes can easily disrupt a business's finances.

In this context, watermarking techniques have emerged as promising tools for data protection, mainly (but not only) for data tampering detection, copyright protection, and tracking the origin of unauthorized data copies. One of the key features contributing to the success of watermarking is the ability to apply it without requiring additional data restrictions, such as access constraints or deployment limitations.

1.1. From multimedia to relational data

Watermarking schemes, along with steganography, are classified as information-hiding techniques (Katzenbeisser & Petitcolas, 2000). They act by embedding a signal to protect the data (a.k.a the watermark), which is detected and extracted when suspicions of data copyright violations or tampering arise. Some watermarking approaches do not require modifying the data to embed the watermark, making them distortion-free. On the other hand, approaches that distort the data during watermark insertion are defined as distortion-based. In distortion-based approaches, the watermark must be embedded without compromising the digital asset's usability while ensuring the watermark's imperceptibility. Therefore, when the authenticity or ownership of the data is under scrutiny, the watermark signal must remain within the digital asset with sufficient quality to facilitate data tampering detection or copyright identification.

Watermarking techniques were initially employed to protect multimedia data and later extended to relational databases. However, they have been used on many other types of digital assets. Unlike

☆ This study was partially funded by the European Union - NextGenerationEU, in the framework of the iNEST- Interconnected NordEst Innovation Ecosystem, Italy (iNEST ECS_00000043 – CUP H43C22000540006), SERICS, Italy (PE00000014 - CUP H73C2200089001), and PADS4Health, Italy (PRIN PNRR 2022 P2022MSMAW - CUP N. H53D23010880001). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.

* Corresponding author.
E-mail addresses: maikel.perezgort@unive.it (M.L Pérez Gort), cortesi@unive.it (A. Cortesi).

https://doi.org/10.1016/j.jjimei.2025.100320

Available online 22 January 2025

Table 1
Differences between relational and multimedia data.

	Relational data	Multimedia data
Data redundancy	Low	High
Update frequency	High	Low
HVS & HAS	Not exploitable	Exploitable
Data ordering	Changeable	Not changeable
Data type	Multiple	Single

multimedia data, relational data are often subject to frequent updates. As a result, the hidden watermark faces greater preservation challenges. In this context, the performance of *benign updates* (i.e., the daily operations performed on relational databases, such as tuple insertion, tuple deletion, and update of attributes) leads to a "natural degradation" of the watermark that increases as the degree of modified data grows over time.

Compared to multimedia data, the structure used to store relational data presents additional challenges in the design and implementation of watermarking techniques. These challenges include the lack of a fixed order for storing values, low redundancy of the stored content, and the inability to directly leverage the human visual system (HVS) and the human auditory system (HAS) for processing information (Pérez Gort, Olliaro, & Cortesi, 2022). On the other hand, the diversity of data types often present in a database relation offers new opportunities for digital watermarking with respect to multimedia content. Table 1 presents the main differences between relational and multimedia data regarding redundancy, update frequency, if their representation directly relies on HVS and HAS, data ordering, and the number of contained data types.

1.2. Challenges for watermark preservation

Watermarking techniques encounter the challenge of watermark degradation over time, particularly in highly dynamic environments such as those found in relational databases once deployed on public servers. Some examples reflecting this situation are (i) a database containing the profiles of researchers along with their published work and number of citations and (ii) a catalog of products traded online featuring information such as the number of units sold and buyers. These databases are deployed to allow updates and the insertion of new content, provided that business rules are respected.

The literature defines incremental watermarking as a method to prevent watermark degradation resulting from *benign updates* (Agrawal & Kiernan, 2002; Halder, Pal, & Cortesi, 2010). It constitutes a key requirement for relational data due to their high frequency of updates. Incremental watermarking dictates that marks must be computed and embedded whenever new data are added to the database. This must be done using the same algorithm and parameters employed to synchronize the watermark with the original unwatermarked database the first time the watermark was embedded (Pérez Gort, Feregrino Uribe, & Nummenmaa, 2017).[1] Therefore, it focuses on insertion and update operations.

Despite its purpose, state-of-the-art proposals often fail to address the implementation and deployment of incremental watermarking. They clearly define it in theory but exclude it from their experimental validation. We suspect this may be due to a misunderstanding of the way it interacts with the public system and the security requirements, or a misconception stemming from the belief that concealing the watermark embedding implementation enhances the technique's robustness and security. For a better understanding, we introduce in Section 2.3 the main requirements linked to incremental watermarking (including incremental watermarking itself).

Instead of implementing incremental watermarking, a common strategy to prevent watermark degradation is to scatter the marks throughout the data, hoping that when a column or tuple is updated, the number of affected marks does not compromise watermark recognition. This approach is practical against attacks based on tuple deletion or attribute updates. However, in the absence of incremental watermarking, no marks are generated when new data are added to the database, resulting in the addition of noise to the watermark. Consequently, watermark recognition is inevitably degraded over time, underscoring the importance of implementing incremental watermarking, especially during the insertion and update of data. Nevertheless, the literature provides limited and inadequate references to the deployment of incremental watermarking.

1.3. Research contribution

The primary objective of this paper is to bridge the gap between the definition of incremental watermarking and its implementation. Here, we discuss the main alternatives for addressing the "natural degradation" of the watermark in the context of relational data.

We design a watermarking architecture that contributes to restoring data owners' trust by implementing incremental watermarking across different nodes while ensuring the preservation and enhancement of the watermark despite operations performed on the database. Our proposal can be applied to different watermarking techniques, regardless of their specific characteristics and the protected databases. Our architecture goes beyond the traditional deployment of watermarking services. It allows watermark synchronization in batches once a given number of tuples is inserted or updated but also makes mark embedding at a single-tuple level possible whenever required. As far as we know, this is the first time an architecture of this type has been proposed in relational data watermarking. It not only guarantees the robustness of the watermark thanks to the deployment of incremental watermarking but also contributes to a more secure implementation of watermarking techniques.

The key idea behind our approach is to manage iterations in a scenario that sees *in-house* data watermarking and *on-server* data deployment, with the use of a synchronization matrix that allows us to optimize the watermarking processes.

Besides analyzing watermark robustness and security, we focus on non-functional requirements such as database deployment, data portability, scalability, and incremental watermarking performance to guarantee our proposal's applicability. We present scenarios where the watermark quality is analyzed, comparing the effects of considering incremental watermarking vs. ignoring it. We carry out this by performing watermark synchronization with two techniques, as commonly presented (i.e., without deploying incremental watermarking services), vs. combined with our approach (i.e., deploying incremental watermarking services as proposed in this work). Our approach does not work alone but instead requires applying it to an existing technique, adding new functionalities to incremental watermarking processes while offering various alternatives for their implementation. We also use the same techniques combined with our approach to record the metrics we proposed for measuring the effects of incremental watermarking while evaluating its performance. Thus, we assess the reliability and performance for scenarios where incremental watermarking is carried out, taking into account different volumes of data.

To summarize, our work addresses the following question: *Does implementing incremental watermarking conflict with the public system and security requirements, and if so, to what extent?*

There is a lack of trust in deploying incremental watermarking due to the gap between its definition and implementation in the related literature. This lack of trust is evidenced through robust testing, which reveals that the watermark signal degrades in proportion to the number of tuples or attributes affected by *benign updates* or *malicious operations*.

[1] Watermark synchronization refers to aligning two signals in time or space (Cox, Miller, Bloom, Fridrich, & Kalker, 2007), corresponding to the embedding and extracted watermarks.

1.4. Paper structure

The rest of this paper is organized as follows. Section 2 presents the preliminaries related to our research, focusing on the watermarking architecture and the requirements most relevant to incremental watermarking. Section 3 provides details on how proposals introduced in the literature address the problem of watermark degradation in practical environments, along with potential conflicts arising from implementing incremental watermarking. Section 4 describes our proposal from both architectural and implementation perspectives, introducing various alternatives considering researchers' and data owners' priorities and the reliability levels of the database deployment scenarios. Section 5 presents the experimental results validating our proposals, with a primary focus on false positive detection, watermark recognition precision, and watermark synchronization performance. Finally, Section 6 concludes this work.

2. Preliminaries

Preserving the quality of the watermark is directly linked to how the technique manages capacity and robustness. Nevertheless, despite guaranteed robustness, *benign updates* can still compromise watermark recognition in the absence of *malicious operations* or attacks. Therefore, ensuring 100% detection accuracy of embedded marks is insufficient for optimal performance of watermarking techniques in protecting relational data. This section presents the main theoretical aspects to consider when addressing the effects of *benign updates* on watermark quality.

2.1. Structure of a database relation

According to the relational model, each relation represents an entity of the business for which the database is designed. In this model, a relation is depicted as a table, with columns representing the entity's attributes and rows representing the instances of the entity, known as tuples. Each tuple is uniquely identified by a value obtained from one or multiple attributes, known as the relation's primary key (denoted by PK). Consequently, a relational database comprises one or several relations linked through their respective primary keys (Date, 2003).

For the sake of simplicity, our work models embedding a watermark W in a single relation R. We identify each attribute of the relation as $a_i \in A$, where A is the set of attributes, with $i \in [0, \nu - 1]$, and ν being the cardinality of A, defined as $\nu = |A|$. The tuples of the relation are identified as $t_j \in T$, where T is the set of tuples, with $j \in [0, \eta - 1]$, and η being the cardinality of T, according to $\eta = |T|$. The primary key of the tuple is denoted as $t_j.PK$, and the ith attribute of the jth tuple is represented as $t_j.a_i$.

It is important to consider different attributes for each tuple to scatter the marks within R. Among the diversity of watermarking techniques proposed thus far, some approaches embed only one mark per tuple (e.g., Jawad & Khan, 2013), while others attempt to embed a fixed number of marks for each tuple (e.g., Mehta & Rao, 2011). Finally, certain approaches strive to vary the number of marks embedded in each tuple as much as possible to increase randomness and make the technique less predictable for attackers (e.g., Pérez Gort et al., 2017).

2.2. Watermarking architecture

The architecture of distortion-based watermarking techniques consists of the watermark embedding and extraction processes, regardless of the protected digital asset type. However, ensuring proper watermark synchronization does depend on the features of the structure storing the protected content.

Some techniques generate the watermark from external sources or fragments of the protected data, categorizing them as meaningful watermarking. In contrast, techniques that employ a meaningless stream of bits as a watermark are categorized as meaningless watermarking. Meaningful watermarking offers the advantage of applying enhancement algorithms to improve the quality of the extracted watermark, allowing the mitigation of damage caused by attacks. Additionally, depending on the source type, various metrics can be used to assess the quality of the extracted watermark (Pérez Gort et al., 2022).

A watermark, denoted as W, comprises a set of marks, each storing a bit according to $m_l \in W$, where m_l represents the lth mark and $l \in [0, |W| - 1]$. For meaningful techniques, an external source, denoted as S, consists of meta-marks used to generate the marks. Formally, $\overline{m}_k \in S$ represents the kth meta-mark, where $k \in [0, |S| - 1]$. To increase the probability of successful synchronization, it is crucial to fulfill the condition $|S| \ll |W|$; otherwise, only a portion of the watermark source is considered during embedding, rendering the technique vulnerable to *benign updates* and attacks.

For cases that use a binary image as a watermark source, the meta-marks are the image's pixels. Fig. 1 depicts the architecture of meaningful watermarking techniques. The embedding process takes as input the relation to be protected R, the watermark source S, and the set of parameters P to perform the embedding, which also contains the data owner's secret key SK. The secret key is considered to fulfill the *key-based system* (Agrawal & Kiernan, 2002; Halder et al., 2010) (see Section 2.3).

The embedding process comprises three subprocesses: (i) meta-mark selection, (ii) mark generation, and (iii) mark insertion. In the figure, s denotes the iteration of the embedding process, considering that mark generation occurs after selecting the corresponding meta-mark. The output of the embedding process is a version of the relation containing the watermark (denoted by R').

After the watermark is embedded, the database is deployed on an online server. In this scenario, users can access the database and manage its content, leading to *benign updates*. Furthermore, considering the possibility of accessing the data from the network, malicious operations are also likely to occur. The extraction process takes as input a version of the watermarked relation (denoted by R"), which differs from R' due to the execution of *benign updates* and the potential occurrence of attacks. Additionally, the set of parameters $\overline{P}$ is used for watermark extraction, which is expected to be the same as the one used for watermark embedding (i.e., $\overline{P} = P$). Otherwise, watermark synchronization will likely fail unless the technique is defined as asymmetric (Nadhir, Ali, & Abdelkrim, 2021). Situations where $\overline{P} \neq P$ are also likely due to *brute force attacks* (Pérez Gort, Olliaro, & Cortesi, 2023).

The watermark extraction comprises three subprocesses: (i) detection of marks, (ii) their extraction, and (iii) watermark reconstruction. Watermark reconstruction involves transforming the marks into their respective meta-marks. Since different marks may be linked to the same meta-mark, majority voting is performed to handle marks with contradictory values resulting from *benign updates* and *malicious operations*. After extracting all meta-marks, the watermark source is assembled, and the process concludes with the generation of S' (a.k.a the version of S constructed from the signal detected during extraction). Finally, both sources are compared, and the final assertion is made. Ownership claims are valid for robust techniques if $S \equiv S'$ (where the symbol $\equiv$ represents the equivalency of the sources, considering that they do not have to be equal as long as both signals can be matched). At the same time, data tampering is confirmed for fragile techniques if $S \not\equiv S'$.

2.3. Watermarking requirements

For the applicability of a technique, it is vital to consider both watermarking and non-functional requirements, such as performance and precision. In the case of relational database servers, performance is optimized by deploying the architecture, while considering the nodes involved in an information management system (see Section 2.4). On the other hand, precision relates to how watermarking parameters are applied during the embedding and extraction processes. In the

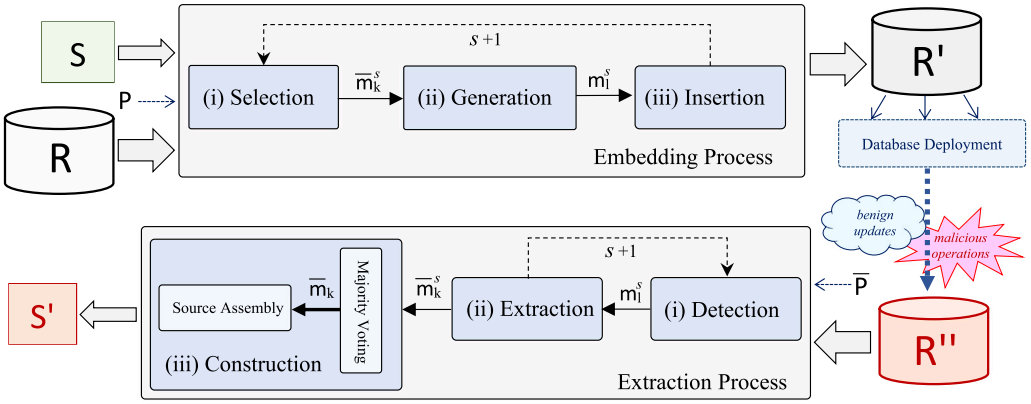


Fig. 1. Architecture of meaningful watermarking techniques depicting the roles of meta-marks and marks (Pérez Gort et al., 2022).

context of relational data, compliance with incremental watermarking is crucial for mitigating the effects of daily updates on the embedded watermark. This requirement cannot be adequately addressed if others are neglected.

Among watermarking requirements are the *blind system*, *false positives* and *false negatives detection*, *accuracy*, *non-interference*, *usability*, *imperceptibility*, and many more (Agrawal & Kiernan, 2002; Halder et al., 2010). Below, we describe the requirements directly affected by the deployment of incremental watermarking:

- **Incremental watermarking:** mandates that for a watermarked database, whenever data is inserted or updated, the corresponding mark must be generated and embedded if the new value requires marking.
- **Public system:** according to Kerckhoffs' principle, the algorithm underlying the watermarking processes must be public. The resilience of the architecture should depend on secret parameters, such as the secret key SK (Mrdovic & Perunicic, 2008).
- **Security:** it should rely solely on private parameters that are kept secret from unauthorized third parties. This includes the secret key SK, which should be known only to the data owner.
- **Key-Based System:** defines the requirement of using cryptographic keys to trigger the watermark embedding and extraction processes.
- **Robustness:** the embedded watermark signal should remain strong enough to guarantee watermark detection despite *benign updates* and *malicious operations* as long as attacks performed to remove the watermark do not compromise the database usability.
- **Capacity:** defines the number of marks that can be embedded without compromising data quality. The higher the number of marks, the stronger the embedded signal and the technique's robustness. However, the watermark capacity cannot be increased to a point that compromises the watermark's imperceptibility.

Our approach focuses on these requirements to enable the implementation and deployment of incremental watermarking while maintaining data owners' trust in its execution.

2.4. Information systems architecture

The two-tier software architecture, generally referred to as the client/server architecture, represents the simplest case of an N-tier architecture (Laudon & Laudon, 2004). There are systems built upon more complex models that involve additional tiers with specialized functionalities. One example is the deployment of different types of servers in the architecture nodes (see Fig. 2).

In information management systems, data management occurs within a node that houses the database server where the database is deployed. To access or retrieve data, datasets are encapsulated in packages and transmitted between the tiers. Given that the type of architecture influences non-functional requirements, it is crucial to consider this during the implementation and execution of incremental watermarking.

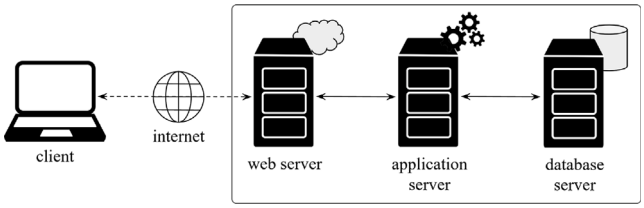


Fig. 2. Example of a 4-tier software architecture.

Overloading the database servers with new functionalities might compromise the system's performance. Conversely, data transmission among the architecture nodes (or tiers) is also costly in terms of computational resources. Security risks increase every time data is extracted from the context of the database management system, as it becomes vulnerable to acquisition by unauthorized third parties. Moreover, data can be susceptible to consistency issues if communication among nodes fails or if the system crashes. Therefore, it is crucial to interact with the least possible data when implementing incremental watermarking.

To ensure the optimal performance of a watermarking scheme while safeguarding a deployed database, it is important to consider certain properties, such as *efficiency* and *reliability* (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 2008). *Efficiency* pertains to utilizing resources available for executing the software and encompasses response times. It is influenced by the speed of programming algorithms and the design and deployment of the architecture (Buschmann et al., 2008). Given its significance in our work, it is crucial to note that implementing incremental watermarking may impact the system's response times during data source queries. On the other hand, *reliability* concerns the overall ability of the software system to maintain functionality, addressing system errors and unexpected scenarios. To guarantee the system's proper operation, it is essential to consider how data changes and the system functionalities vary with the deployment of incremental watermarking.

3. Related work

Agrawal and Kiernan (2002) proposed the first watermarking technique for relational data. They introduced the notations of the elements composing R and an algorithm for watermark synchronization known as AHK. Their approach utilizes a meaningless watermark and does not address the deployment of incremental watermarking.

There are numerous watermarking techniques based on the AHK algorithm, with several examples published in Pérez Gort et al. (2017, 2023). Some of these techniques address the issue of false ownership claims (Manjula & Settipalli, 2010), while others concentrate on mitigating the distortion caused during watermark embedding (Rani & Halder, 2022; Waheeb Yaqub & Aung, 2018). Conversely, some distortion-based techniques aim to minimize rather than prevent distortion (Franco-Contreras & Coatrieux, 2015), while others prioritize preserving query results when accessing watermarked data (Gross-Amblard, 2003, 2011). Additionally, in terms of distortion, certain techniques focus on preserving the semantic content of the database (Hu, Zhao, & Zheng, 2018; Iftikhar, Kamran, & Anwar, 2014; Kamran, Suhail, & Farooq, 2013; Li et al., 2022).

Some approaches, such as those described in Kamran et al. (2013) and Rani, Koshley, and Halder (2017), define tuple partitions to synchronize the watermark. Other works, such as Hu et al. (2018) and Wang and Li (2023), focus on watermark synchronization for numerical attributes. In Li et al. (2022), a semantic-based approach is introduced for both numerical and non-numerical attributes. The proposed method employs a reversible scheme to embed the watermark on numerical attributes, aiming to preserve their distribution. For non-numerical attributes, natural language processing is utilized to segment and embed words in textual content.

While the related literature defines the need for incremental watermarking, few techniques address the challenges of its application and trust conflicts that arise between designers and data owners. Regarding approaches aimed at protecting digital assets in cloud computing, Guang, Xiaoping, Sha, Yingjie, and Huifang (2015) present a proposal to safeguard services through software watermarking. Although this research addresses critical aspects of piracy in that environment, incremental watermarking is not considered necessary.

On the other hand, in Naz et al. (2020), the authors target databases containing medical records and aim to protect them through watermarking services. Their approach involves a zero watermarking scheme that ensures no distortion to the protected data and places significant emphasis on the buyer–seller watermarking (BSW) protocol (Khan et al., 2016). Additionally, they focus on ensuring the anonymity of patients' records before selling the data. The authors utilize a trusted third party, although its convenience may vary depending on the deployment conditions of the data and the involvement of additional roles in the watermarking scheme.

Table 2 presents characteristics of a representative group of watermarking techniques and approaches that have recently been developed. We outline general features to convey the diversity of these approaches, naming in the first column of the table the reference of the proposal, in the second column its cover type, and in the third one the granularity level of the technique (i.e., at that level of value the mark is embedded in the attribute). The next column shows whether the approach is an IBW, and column 5 shows whether the technique is robust. If it is not robust, the technique can be fragile or semi-fragile (fragile techniques are created to detect data tampering, so if the watermark is compromised, data tampering can be assumed). Finally, the last column in the table indicates whether the authors have implemented the incremental watermarking requirement. In the table, negative values are displayed as a red cross and positive values as a green check mark.

There are a couple of approaches from Table 2 that we address in more detail. In Hou and Xian (2021), the authors achieved high watermark preservation despite tuples insertion, but never addressed incremental watermarking. In Iftikhar et al. (2014), the authors do not implement incremental watermarking. The watermark is embedded in a one-time process and the method does not support continuous or progressive updates to the watermark as the data changes. In Lin et al. (2021) and Tzouramanis (2011), the authors claim to achieve incremental watermarking but do not provide details regarding the design or implementation. Furthermore, the experimental results contradict their assessment. Finally, in Zhang et al. (2022), the authors can guarantee the total persistence of the watermark after high-degree attacks, but this was not achieved through incremental watermarking, which is never addressed.

Our proposal is designed to work in combination with a main watermarking approach. Therefore, it adjusts to the main technique's cover type and granularity level. We define the cover type as *universal* and the granularity level as *scalable*. Although we present our proposal using an IBW approach, this is not mandatory, as it is another requirement derived from the main technique.

None of the previous approaches directly deal with the deployment of incremental watermarking or address the conflicts that might arise with other requirements. So far, no research has addressed this issue, nor has it examined how trust can be guaranteed when implementing it in environments where stakeholders might doubt its application.

4. Proposed approach

It is reasonable to think that implementing incremental watermarking might provide additional clues about how the algorithm synchronizes the watermark. Given the database's deployment on a public server, the watermark embedding process must be triggered somewhere between the database and the client's access point. Therefore, data owners and security technicians may be concerned about the possibility of an unauthorized party accessing the code used to embed the marks. This concern is heightened when the implementation is done in plain SQL, where all details of the watermarking scheme are exposed and could be intercepted and exploited to support false ownership claims. In summary, security concerns are the main reason for not implementing incremental watermarking.

In related work, some authors address the basics of incremental watermarking but avoid its deployment and execution while performing robustness experiments. Paradoxically, the requirement for a public system (see Section 2.3) sets the basis for implementing incremental watermarking, even if the implementation reveals algorithm details. As shown in Section 2.3, security must rely solely on the secrecy of private parameters rather than on hiding the details of the watermarking processes. To our knowledge, the literature has not yet addressed the costs of deploying and executing incremental watermarking.

4.1. Targeting the protected data

Based on the *public system*, the *security*, and the *key-based system* requirements, incremental watermarking can be implemented, deployed, and executed, even if it reveals details of the embedding process' implementation, as long as the parameters used to synchronize the watermark are sufficiently complex to thwart attackers' attempts. Direct contributions to the performance of the watermarking scheme are achieved when these requirements are carefully considered.

In relational data watermarking (see Fig. 1), the database relation R transforms into R' after undergoing watermarking. Subsequently, *benign updates* and potential attacks applied to R' produce a new version of the watermarked relation, denoted as R''. Then, when required, the watermark extraction process is triggered using R'' as input. Within this architectural framework, the challenges of incremental watermarking primarily revolve around the location of the node where the database is deployed.

We present a case study where the watermarking embedding process (denoted by E) occurs in a controlled environment called *in-house*. The watermarked database is deployed on the database server within an environment designated as *on-server*. Notice that both *in-house* and *on-server* environments correspond to logical nodes of an architecture, each functioning as a server. Therefore, *in-house* does not identify a client; rather, it denotes a more secure server that implements incremental watermarking and offers it as a service. In contrast, *on-server* is where the database operates and accepts connections from clients outside the watermarking scheme.

Table 2
Consideration of incremental watermarking in diverse techniques.

Proposal	Cover type	Granularity	IBW	Robust	Inc.
Cao, Sun, and Hu (2010)	Numeric	Bit	✓	✓	✗
Farfoura, Horng, and Wang (2013)	Numeric	Bit	✓	✓	✗
Franco-Contreras, Coatrieux, Cuppens-Boulahia, Cuppens, and Roux (2014)	Numeric	Bit	✗	✓	✗
Hamadou et al. (2024)	Numeric	Bit	✗	✗	✗
Han, Peng, Xian, and Yang (2024)	Numeric	Bit	✗	✓	✗
Hou and Xian (2021)	Numeric	Bit	✗	✓	✗
Hu et al. (2023)	Numeric	Value pair	✗	✓	✗
Iftikhar et al. (2014)	Numeric	Entire value	✗	✓	✗
Li et al. (2022)	Numeric/textual	Digit/word	✓	✓	✗
Lin, Nguyen, and Chang (2021)	Categorical	Entire value	✗	✓	✗
Rani et al. (2017)	Numeric	Multi-bit	✗	✓	✗
Shen, Zhang, Wang, and Sun (2020)	Numeric	Value pair	✗	✓	✗
Siledar and Tamane (2020)	Numeric	Entire value	✗	✓	✗
Tzouramanis (2011)	Numeric	Bit	✗	✓	✓
Zhang, Wang, Wang, and Liu (2022)	Numeric/textual	Bit/symbol	✗	✓	✗
Our Proposal*	Universal	Scalable	✓	✓	✓

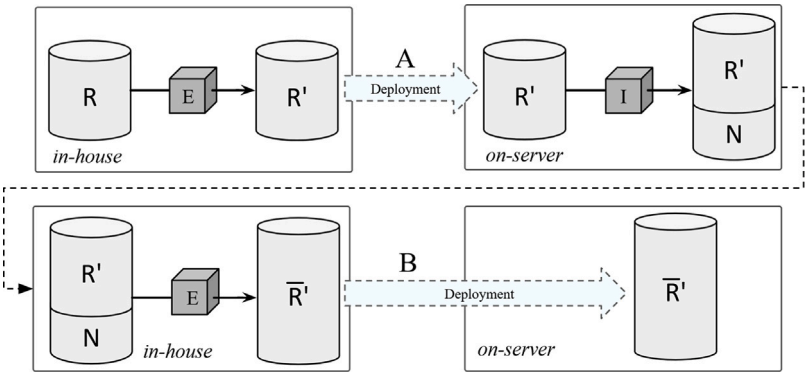


Fig. 3. Case 𝕀: Incremental watermarking targeting the entire database.

After deploying the watermarked database, *benign updates* introduce modifications to the data in R'. We specifically examine the scenario of tuple insertion (denoted as I), which involves inserting new data, denoted by N, into the watermarked relation. These database modifications are represented as R'' ⇔ R' ∪ N, where ∪ denotes the *union* operator and ⇔ is the equivalence relationship (See step A of Figs. 3 and 4). However, since the new data is not watermarked, the question arises: how and where should incremental watermarking be deployed and executed to maintain the trust of data owners in the scheme? We outline two scenarios considering the volume of data deployed each time watermarking is performed. To ensure the safety of developers' and data owners' interests, we perform watermarking within the secure *in-house* environment. However, this decision entails significant costs in terms of computing resources and performance.

Fig. 3 illustrates Case 𝕀, where all data, identified by R' ∪ N, are deployed in the *in-house* environment for watermarking. This version of the architecture is recommended when there is low trust in accurately identifying modified data from the deployed database or in the scheme's ability to ensure the persistence of marks despite data modifications. This scenario prioritizes optimizing accuracy, even at the expense of performance. In Fig. 3, R̄' is generated as a result of watermarking R' ∪ N. This process involves embedding watermarks on a database fragment that is already watermarked (see step B of Fig. 3). After obtaining R̄', the database is redeployed in the *on-server* environment. The analysis of new unwatermarked data added to R̄' is managed by iterating the transitions between steps A and B.

The second case (defined as Case 𝕀𝕀) exclusively targets the new content inserted into the watermarked database (see Fig. 4). It involves deploying N into the *in-house* environment to initiate the watermark embedding process, resulting in N'. Subsequently, the watermarked content is combined with R' and deployed in the *on-server* environment (see step C of Fig. 4). This case also establishes new data protection through the iteration of transitions between steps A and C. Generally, Fig. 4 represents the lowest-cost scenario, given the smaller number of analyzed tuples. However, Fig. 3 illustrates a more reliable alternative from an accuracy perspective and presents a case that requires a lower frequency of connections between environments.

To evaluate the cost of incorporating incremental watermarking into the architecture, we define C_I as the cost of running incremental watermarking processes according to (1), where $C_S \propto \eta * \nu$ denotes the expenses of moving the data. This is formally described by the *on-server* ⇄ *in-house*, where ⇄ represents the transmission of data from *on-server* to *in-house*, and then back to *on-server*. In the equation, C_W denotes the cost of the actual watermarking process, which is directly proportional to the number of tuples in the relation, expressed as $C_W \propto \eta/\gamma$ based on the AHK algorithm.

$$C_I = C_S + C_W \tag{1}$$

In this case, γ represents the tuple fraction used to determine the number of marked tuples ω, according to $\omega \approx \eta/\gamma$. For techniques that do not utilize the tuple fraction, the watermarking cost remains proportional to the number of protected tuples. For example, in Case 𝕀𝕀, if the embedding process E is executed in the *on-server* environment after the insertion of tuples I, then $C_S = 0$. However, by doing so, the reasons for distrust in the watermarking technique persist.

4.2. Frequency of watermarking iterations

Incremental watermarking is designed to safeguard new content resulting from the insertion or update of data in a watermarked database. It involves executing the watermark embedding process and selecting the content to be watermarked using the same parameters previously employed to synchronize the watermark in the initial dataset before

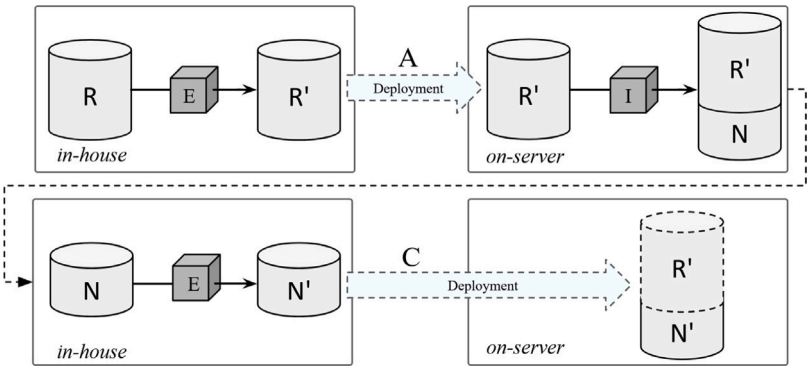


Fig. 4. Case III: Incremental watermarking targeting only N.

its deployment in the *on-server* environment. Thus, the frequency of incremental watermarking execution and the amount of data involved in each iteration are crucial factors that significantly impact the technique's performance and the trust of data owners, given their direct influence on non-functional requirements. Therefore, optimizing the watermarking scheme is essential.

In this case, we differentiate between content that has already been watermarked and content undergoing incremental watermarking. Similar to all versions of R (i.e., R', R̄', and R''), N consists of tuples and attributes, allowing us to utilize the notation introduced in Section 2.1 to reference its elements. We formalize the distinctions between updating attributes and inserting tuples as follows: for tuple insertion, N∩R' = ∅ (as depicted in Figs. 3 and 4), while for update operations, N ⊂ R'. To elaborate further, T_N denotes the set of tuples from N, where $t_{N_j} \in T_N : j \in [0, \eta_N - 1]$ identifies the jth tuple in T_N, and η_N represents the number of tuples in the set (or its cardinality) according to $\eta_N = |T_N|$. Regarding the attributes, we maintain consistency with previous notations, as benign operations do not involve attribute insertion (which would entail modifying the database schema rather than just its data).

We propose two approaches to determine when to trigger incremental watermarking: (i) checking every time a group of at least G tuples is updated/inserted (a.k.a block-based iterations), or (ii) checking each time a new tuple is updated/inserted (a.k.a tuple-based iterations).

4.2.1. Block-based iterations

Performing incremental watermarking for a data volume composed of G tuples entails redefining previous cases (Cases I and III of Figs. 3 and 4, respectively). Fig. 5 depicts details of this approach, where S represents the *on-server* ⇄ *in-house* transmissions. Each set of tuples obtained from N can be identified by T_{N_S}, where $S \in [0, (|T_N|/G) - 1]$. Therefore, for Cases I and III, we refer to N_S instead of N, where $N_{S_0} \cup N_{S_1} \cdots \cup N_{S_{(|T_N|/G)-1}} = N$. Similarly, we identify the new watermarked content as N'_S instead of N', where $N'_{S_0} \cup N'_{S_1} \cdots \cup N'_{S_{(|T_N|/G)-1}} = N'$.

Consequently, N remains within the *on-server* environment, awaiting fragmentation before transmission to the *in-house* environment. Once all tuples from N are watermarked, the scheme waits until G new tuples are inserted to trigger incremental watermarking again. When considered convenient, the database administrator may restart the metric values to manage model performance by reporting the occurrence of N' and R', which resets the number of tuples awaiting incremental watermarking. This will also restart the computation of the cost of executing incremental watermarking.

This approach reduces network traffic but heightens vulnerability risks as the number of unwatermarked tuples increases until reaching G. Additionally, higher values of G will require more network resources when triggering watermarking, thereby increasing the risk of leaving unprotected data on the server.

4.2.2. Tuple-level iterations

For the second approach, where the process is triggered for each tuple, it can be established that $G = 1$. Depicted in Fig. 6, this approach increases the network traffic but ensures the watermark on the database remains up to date. Therefore, it is crucial to consider the trade-off between G and S, bearing in mind that $G \propto S^{-1}$ (when T_N remains constant). Furthermore, lower values of G reduce C_W but increase C_S, as the number of transitions S also rises.

In (2), we formalize the relationship between the number of transitions and the number of tuples used to trigger incremental watermarking. When using tuple-level iterations, the number of transitions equals the number of tuples in N. On the other hand, when $G > 1$, the watermarking complexity increases for each group according to $c_{uw} \times G$, where c_{uw} represents the unitary watermarking cost per tuple. If the cost of watermarking each group is consistent, then $C_W = S \times c_{uw} \times G$, resulting in $C_W = c_{uw} \times |T_N|$. Additionally, to identify differences resulting from watermarking each group, it is advisable to set $C_W = \sum_{i=0}^{|T_N|/G-1} c_{w_i}$. In this scenario, the actual cost of watermarking each group is denoted as c_{w_i}.

$$|T_N|/G = S \tag{2}$$

Moreover, the cost of transferring data between environments is given by c_{us}, representing the unitary cost of moving a tuple between environments for marking and deploying the data. For block-based iterations, $C_S = \sum_{j=0}^{|T_N|/G-1} c_{s_j}$, whereas for tuple-level iterations, $C_S = c_{us} \times |T_N|$. In theory, for each group of tuples, $c_{s_j} = c_{us} \times G$. However, it is important to note that network conditions may fluctuate when a watermarking process is initiated for each group of tuples.

4.3. In-house and on-server interactions

Although the public system and security requirements justify the implementation of incremental watermarking, it cannot be properly addressed without considering the protection of trust in the watermarking scheme. Deploying functions of the watermarking technique on different nodes increases the risk of interfering with data, which is an inherent challenge of client/server architecture. Regardless of how incremental watermarking is implemented, it is essential to address the risks and potential solutions for its execution alongside data access, management, and possibly watermark synchronization.

We design the watermarking processes considering a new element called the synchronization matrix M. Our focus is to ensure concurrency and consistency between the processes deployed across different nodes involved in incremental watermarking, enabling real-time updates and watermark synchronization. The structure of M is defined by η rows and 4 columns (i.e., $M[\eta, 4]$). Each row of the matrix corresponds to a tuple of the protected data. Therefore, whenever a new tuple is inserted, a row is added to M, incrementing η. Meanwhile, the columns (labeled as $\mathcal{A}$, $\mathcal{B}$, C, and D) denote the status of each tuple during incremental

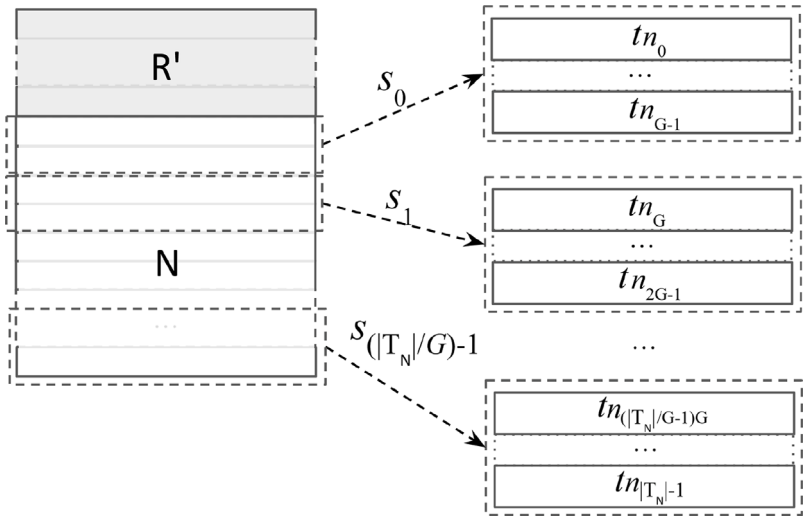


Fig. 5. Incremental watermarking triggered by groups of G tuples, featured by $G > 1$, which results in $S = |T_N|/G$.

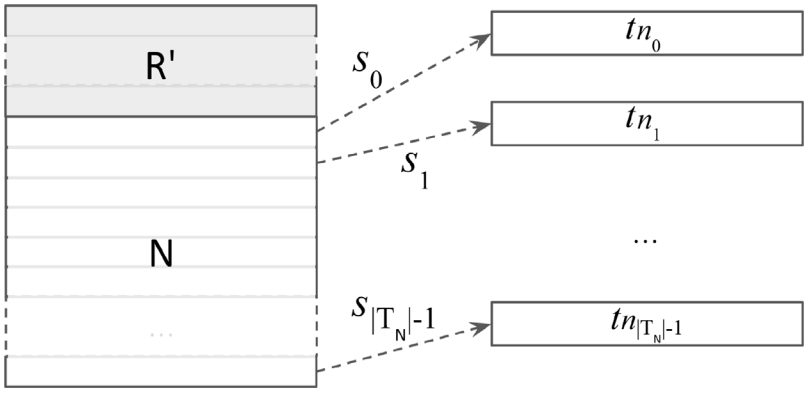


Fig. 6. Incremental watermarking triggered at a single tuple level, characterized by $G = 1$, which results in $S = |T_N|$.

Table 3
Meaning of M columns to classify the status of a tuple.

$\mathcal{A}$	→	Tuple protected on-server
$\mathcal{B}$	→	Tuple marked
$\mathcal{C}$	→	Tuple fragmented (grouped)
$\mathcal{D}$	→	Tuple processed or packaged

Table 4
Map of possible stages of the synchronization matrix M.

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	Meaning
0	0	0	0	New tuple to be added to N.
0	0	0	1	Tuple added to N (see Algorithm 3).
0	0	1	1	Tuple added to N_S (see Algorithm 2).
0	1	1	1	Tuple marked (see Algorithm 1).
1	1	1	1	Tuple deployed *on-server* (see Algorithm 4).

watermarking. Table 3 presents each column of M with its respective meaning in the tuple status classification.

A combination of statuses defines the stage of the tuple. Some stages cannot be assigned unless the tuple is already linked to another stage, which is considered a prerequisite. Therefore, there is a natural precedence among stages while describing the process of incremental watermarking. For instance, a tuple must initially be added to N when updated/inserted in the protected database before being marked by the incremental watermarking process deployed in the *in-house* environment. Additionally, certain combinations of statuses are invalid due to logical constraints. Therefore, rows such as $M[\mathcal{B} = 0, \mathcal{D} = 0]$, $M[\mathcal{B} = 0, \mathcal{D} = 1]$, and $M[\mathcal{B} = 1, \mathcal{D} = 1]$ are considered valid for any tuple, whereas $M[\mathcal{B} = 1, \mathcal{D} = 0]$ is not. Table 4 contains the valid status combinations represented in M for each tuple, presented in stages according to their logical order while performing incremental watermarking. In the table, the columns of the matrix are presented with the value they most take for, in combination, define the stage described under the caption **Meaning** of the table.

Tuples that are already stored on the server and have been watermarked initially (i.e., marked but not through the incremental watermarking processes) are associated with entries $[1, 1, 1, 1]$ in M. This indicates that these tuples have undergone the complete watermarking process.

Utilizing M enhances the security of the watermarking technique, as it can be strengthened with additional measures such as Hamming codes, binary transformations, and various validation procedures for the matrix entries. Moreover, it facilitates the incorporation of additional roles, such as trusted third parties, and enables elements of multi-party computing to be considered. Finally, the convention for the values of statuses and valid stages can be modified according to the technique's requirements, increasing the flexibility of the approach.

4.4. In-house data watermarking

Watermark embedding is performed in the *in-house* environment based on the AHK algorithm, illustrated in Algorithm 1. This algorithm

takes as input the watermark source S, the owner's secret key SK, the tuple fraction γ, and the attribute fraction δ (which is used to vary the number of attributes selected for watermarking, similarly to how γ varies the number of tuples considered by the technique). Additionally, the input includes the number of most significant bits β and less significant bits ξ, along with the sets of attributes available for marking A and the set of tuples N_S sent from the *on-server* environment. It is worth noting that the group of tuples G is not explicitly required, as it can be inferred from N_S.

Algorithm 1: In-house Embedding.

```
1  Input: S, SK, γ, δ, β, ξ, A, N_S
2  Output: N'_S
3  foreach tuple t ∈ T_{N_S} do
4      K_T = F_H(t.PK, SK)
5      if K_T mod γ = 0 then
6          foreach attribute a ∈ A do
7              K_A = F_V(t, t.a, K_T, β)
8              if K_A mod δ = 0 then
9                  b_β = B_β(K_A, t.a, β)
10                 p_ξ = B_ξ(K_A, t.a, ξ)
11                 m̄ = H(S, t, K_T, K_A)
12                 m = m̄ ⊕ b_β
13                 t.a = M(t.a, m, p_ξ)
14     M[K_T] ← [0, 1, 1, 1]
```

The distortion caused to the data can be controlled by parameters such as γ, δ, and ξ. However, the restrictions imposed by the database server to maintain data quality and adhere to business rules must be respected during watermark embedding. The process involves computing the virtual primary key $\mathrm{K_T}$ for each tuple from N_S (stored in $\mathrm{T_{N_S}}$) using the one-way hash-based function $F_H(\cdot)$, which takes the current tuple's primary key and the secret key as input (see lines 3 and 4). This leverages the exclusivity of the primary key for identifying the tuple and the secrecy of SK, enhancing detection accuracy. Subsequently, if $\mathrm{K_T}\ mod\ \gamma = 0$, the attributes of the current tuple are considered for the process (see lines 5 and 6). Similarly to how $\mathrm{K_T}$ is created to identify the tuple, $\mathrm{K_A}$ is generated to identify each attribute. In this case, the content of the tuple, along with the attribute being analyzed, is used, as well as $\mathrm{K_T}$, and the number of most significant bits β (see line 7). The value of $\mathrm{K_A}$ is obtained using the function $F_V(\cdot)$ that implements the generation of keys and may consider the most significant bits of other attributes in the tuple as well.[2] If $\mathrm{K_A}\ mod\ \delta = 0$, the current attribute is considered for mark embedding (see line 8).

After selecting an attribute for marking, one of its most significant bits is *pseudo-randomly* chosen and stored in b_β using the function $B_\beta(\cdot)$ (see line 9). Similarly, a position from the given range of possible less significant bits to be marked is obtained using the function $B_\xi(\cdot)$ and stored in p_ξ (see line 10). These functions take as input the virtual key identifying the attribute $\mathrm{K_A}$, the attribute value $t.a$, and the allowed ranges of bits specified by β and ξ, respectively.

The meta-mark is extracted from the source S using a second one-way hash function, $H(\cdot)$, which takes as input the keys identifying the tuple, the attribute, and the entire tuple content (see line 11). Subsequently, the meta-mark is combined with the most significant bit value b_β selected from the attribute using an *xor* operation, resulting in the mark m (see line 12). Finally, the mark is embedded into the less significant bit position p_ξ of the attribute using the function $M(\cdot)$ (see line 13). Additionally, the corresponding entry for that tuple in M is flagged as marked (see line 14). Following the attribute marking, the process proceeds to analyze the remaining attributes and tuples in N_S until all are considered. After obtaining the watermarked version of the data group, $\mathsf{N'}_S$, the fragment is sent back to the *on-server* environment for its deployment.

Note that the corresponding entry for the stage of the tuple t in M is obtained using the virtual primary key $\mathrm{K_T}$. Thus, the uniqueness and secrecy of $\mathrm{K_T}$ are crucial for ensuring watermark synchronization. For generating high-quality virtual primary key sets, it is important to use a scheme resilient against duplicate and deletion problems (Li et al., 2003). In Li et al. (2003) and Pérez Gort et al. (2019), the authors address the main topics related to designing and implementing virtual primary key schemes for relational data watermarking.

4.5. On-server data deployment

The synchronization matrix M identifies the tuples already marked from those that are yet to be marked, reducing the management time required by the watermarking processes. As previously described, the tuples watermarked at the beginning (without using the incremental watermarking engine) are linked to $[1, 1, 1, 1]$ entries in M. On the other hand, tuples inserted after the watermarked database has been deployed on the server and yet to be watermarked are linked to $[0, 0, 0, 0]$ entries.

For each iteration, the first task performed in the *on-server* environment is to generate N to build N_S fragments. Algorithm 2 outlines the first process triggered, which takes as input the secret key SK, the version of the watermarked data resulting from the practice of *benign updates* R", the number of tuple groups used by the incremental watermarking engine G, and the synchronization matrix M.

First, the counter of tuples added to the current group g and the new package of tuples to be processed N_S are initialized (see lines 2 and 3). Subsequently, the method `getUnwatermarkedData(·)`, which takes SK, R" and M as inputs to create the dataset N, is invoked (see line 4). Then, for each tuple in N, stored in $\mathrm{T_N}$, the virtual primary key $\mathrm{K_T}$ is created (see lines 5 and 6).

Algorithm 2: On-server Packaging.

```
1  Input: SK, R", G, M
2  g ← 0
3  N_S[] ← 0
4  N ←getUnwatermarkedData(R",M)
5  foreach tuple t ∈ T_N do
6      K_T = F_H(t.PK, SK)
7      if M[K_T] = [0, 0, 0, 1] then
8          if g ≠ G then
9              N_S.add(t)
10             M[K_T] ← [0, 0, 1, 1]
11             g++
12         else
13             send(N_S)
14             g ← 0
15             N_S[] ← 0
```

Tuples with an entry equal to $[0, 0, 0, 1]$ in M are added to the N_S fragment being generated, as long as the number of tuples under consideration is fewer than G (see lines 7 to 9). For these tuples, the entry in M is updated to $[0, 0, 1, 1]$, indicating that they are already contained in an N_S fragment and will not be added to another fragment in the same iteration (see line 10). Additionally, each time a tuple is added to the current fragment, g is incremented (see line 11).

[2] The reader can find more details about the implementation of $F_V(\cdot)$ under the category of virtual key schemes in Li, Swarup, and Jajodia (2003) and Pérez Gort, Feregrino-Uribe, Cortesi, and Fernández-Peña (2019).

When the number of tuples considered for the current fragment N_S reaches G, the fragment is sent to the *in-house* environment for mark embedding, the counter g is reset, and the structure for the next fragment N_S is cleared (see lines 12 to 15). This process continues until all tuples in the dataset N, obtained for the current iteration, have been considered.

Algorithm 3 details the method `getUnwatermarkedData(·)` used to create N. Initially, an empty structure N is created (see line 3). Then, for each tuple in R", stored in the set T", the virtual primary key K_T is computed (see lines 4 and 5). If the M entry of the tuple being analyzed equals $[0,0,0,0]$, the tuple is added to N, and its M entry is updated to $[0,0,0,1]$ (see lines 6 to 8). This process continues until all tuples from R" have been analyzed.

Algorithm 3: Method `getUnwatermarkedData(·)`

```
1 Input: SK, R", M
2 Output: N
3 N[] ← 0
4 foreach tuple t ∈ T" do
5     K_T = F_H(t.PK, SK)
6     if M[K_T] = [0,0,0,0] then
7         N.add(t)
8         M[K_T] ← [0,0,0,1]
```

Finally, whenever a watermarked package is returned from the *in-house* environment to the *on-server* environment, the process outlined in Algorithm 4 is triggered.

Algorithm 4: On-server Deploying.

```
1 Input: SK, N'_S, R", M
2 Output: R"
3 foreach tuple t ∈ T_{N'_S} do
4     K_T = F_H(t.PK, SK)
5     if M[K_T] = [0,1,1,1] then
6         R".ovr(t)
7         M[K_T] ← [1,1,1,1]
```

Algorithm 4 takes as input the secret key SK, the watermarked fragment N'_S, the modified version of the relation R", and the synchronization matrix M. For each tuple of the watermarked package, its status is checked in M to confirm that mark embedding was performed in the *in-house* environment (see lines 3 and 4). If mark embedding is confirmed (see line 5), the tuple is overwritten in R" (see line 6) and flagged as redeployed in the *on-server* environment by updating its entry in M, as shown in line 7.

5. Experimental results

We conducted a series of experiments to assess the alternatives proposed in this paper. Our analysis focused on evaluating the quality of the extracted watermark. We used the source built from the extracted watermark as a reference and compared it with the source used to generate the embedded watermark. Additionally, we employed metrics to evaluate the accuracy of mark extraction and meta-mark reconstruction, examining their contributions to the reconstructed watermark signal. Furthermore, we reported the time required under different values of G to offer a clear understanding of the trade-off between accuracy and performance.

5.1. Experimental setup

We employed two image-based watermarking (IBW) techniques of different cover-type to evaluate our proposal. They were selected to generalize the outcome of our approach application. The first technique (identified as IBW-NUM) is the numerical cover-type approach proposed in Pérez Gort et al. (2017). The second one (identified as IBW-TEXT) is the multi-word textual cover-type approach proposed in Pérez Gort, Olliaro, Cortesi, and Uribe (2021). These techniques utilize a binary image as the watermark source, allowing the generation of only one mark per pixel, thus associating each pixel with a meta-mark. They were implemented based on a client–server architecture. The client tier was developed in Java 1.8 using Eclipse Integrated Development Environment (IDE) 4.20. We utilized a binary image of the World Wildlife Fund (WWF) logo (40×45 pixels) as the watermark source. Despite being a binary image, we used red as a pseudo-color to emphasize pixels overlooked during watermark synchronization[3] a low-quality watermark signal to avoid introducing higher distortion to the data.

To evaluate our approach, when applied to IBW-NUM, we utilized the numerical relational dataset *Forest Cover Type*.[4] We used the first 30,000 tuples (out of 581,012) and the initial 10 attributes (out of 54) to facilitate a fair comparison and correlation of our research findings with those previously published in Pérez Gort et al. (2017, 2022). On the other hand, we used the textual dataset *Amazon Fine Food Reviews*[5] to evaluate our approach when applied to IBW-TEXT. We selected the attribute '*Text*' from this data set to embed the marks, given the significant content it stores. The Oracle Database 19C was selected as the database engine, and Oracle SQL Developer 20.4 served as the IDE for database management. The experiments were conducted on a PC with an Intel i7-7700K processor clocked at 4.20 GHz, 32.0 GB of RAM, and Windows 10 Pro OS.

We utilize the Correction Factor (CF) and the Structural Similarity Index (SSIM) metrics, defined in (3) and (4) respectively, to assess the quality of the image reconstructed from the extracted watermark. Eq. (3) defines the correction factor as a percentage (i.e., CF $\in [0, 100]$) computed by using the XOR operator (denoted by $\oplus$) to compare pixels in the image generated from the embedded watermark (denoted by I_{emb}) with the opposite value of the corresponding pixels in the extracted watermark image (denoted by Img_{ext}). The resulting values are summed up. Both images must have identical dimensions, determined by the height h and width w. A CF value of 100 indicates perfect correlation between the images, while a CF value of 0 signifies no correlation.

$$\mathrm{CF} = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w}(Img_{emb}(i,j) \oplus \overline{Img_{ext}(i,j)})}{h \times w} \times 100 \tag{3}$$

The Structural Similarity Index (SSIM), as defined in (4), is a metric that provides a value more closely aligned with the human visual perception of image similarity. It is computed using multiple windows of a common size $N \times N$, with window positions on the images defined by coordinates x and y. In this work, SSIM values range between 0 and 1 (i.e., SSIM $\in [0, 1]$), where 1 indicates perfect structural similarity between the images, and 0 indicates a complete lack of similarity. Other terms in (4) include μ_x and μ_y, representing the averages of x and

[3] In the absence of *benign updates* or *attacks*, the pixels ignored during watermark synchronization result from the data owner's intentional embedding of

[4] The *Forest Cover Type* dataset is available online in the University of California Irvine (UCI) machine learning repository at http://kdd.ics.uci.edu/databases/covertype/covertype.html (Colorado-State-University, 1999).

[5] The *Amazon Fine Food Reviews* dataset is available online in the kaggle repository at https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews (McAuley & Leskovec, 2013).

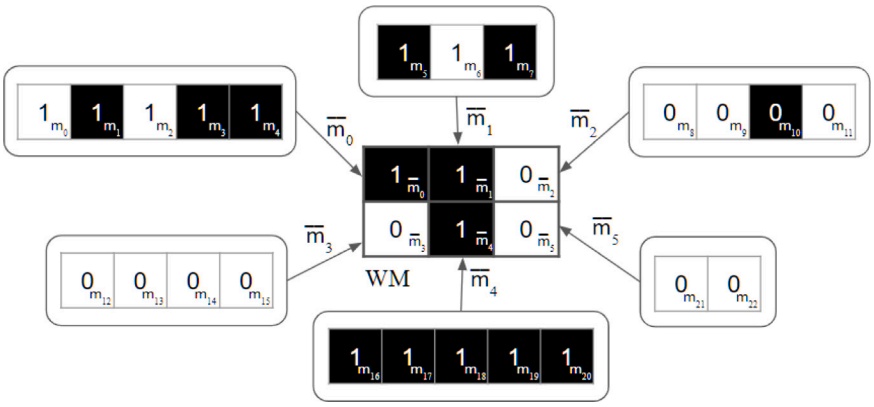


Fig. 7. Graphical view of the link between meta-mark and mark structures, illustrating how majority voting corrects meta-mark values.

y, respectively; σ_x^2 and σ_y^2 denote their respective variances; and σ_{xy} defines the covariance. Symbols C_1 and C_2 are stabilization constants.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (4)$$

Each mark can potentially enhance or undermine the value of its corresponding meta-mark. Understanding the nature of the relationship between marks and meta-marks requires consideration of both recurrent embedding in watermark insertion and the majority voting during watermark extraction. For modeling these processes, we employed two structures: (i) a matrix of meta-marks representing the watermark and (ii) a linear array representing the extracted mark stream (see Fig. 7).

Not all marks necessarily match their respective meta-mark values. Some techniques, such as IBW-NUM, generate the embedded mark, as shown in line 12 of Algorithm 1. Thus, a single meta-mark may compute different mark values. On the other hand, other techniques may be more faithful to the value of the selected meta-mark. For example, IBW-TEXT performs mark embedding by replacing words with synonyms selected according to the sense given by the context (Pérez Gort et al., 2021). Depending on the meta-mark value, words in different positions from the selected set of synonyms are utilized to replace the original word. Nevertheless, since sense disambiguation is not 100% accurate, some values may not match the meta-mark during the extraction because the set of synonyms selected may differ from the one selected during the embedding. This makes the use of majority voting more critical. Nevertheless, in some cases, majority voting may not be enough to avoid adding noise to the extracted watermark. Despite this, IBW-TEXT increases watermark capacity and preserves data usability, but IBW-NUM features a higher detectability.

To assess the impact of watermark synchronization processes during benign updates and the relevance of implementing incremental watermarking, we calculate the percentages of true and false detected meta-marks (designated as $\overline{\mathsf{m}}_T$ and $\overline{\mathsf{m}}_F$, respectively). Additionally, to gauge the contribution to the watermark at the mark level, we record in $\mathsf{m}_{I=I}$ the number of detected marks matching the values used during watermark embedding. The number of mismatched marks is also noted in $\mathsf{m}_{I\neq I}$. Finally, we determine in $\mathsf{m}_{R(\overline{\mathsf{m}})}$ the number of marks that generate correct meta-mark values and in $\mathsf{m}_{W(\overline{\mathsf{m}})}$ those generating incorrect ones.

5.2. Effects of tuple insertion

After watermarking both datasets, we inserted different numbers of tuples, followed by watermark extraction in every case. We utilized two main scenarios for comparison, examining the quality of the recovered signal when incremental watermarking was applied and when it was not. For this, the watermark signal is first synchronized using the techniques described in Section 5.1 without our approach's benefits, and then combined with it. Additionally, we considered various parameters for watermark synchronization to analyze the effects of embedding different quantities of marks.

Table 5 presents the results obtained when marking one out of every 40 tuples (i.e., $\gamma = 40$). It contains two main groups of columns, showing the watermark synchronized using the numerical cover-type technique and the multi-word textual cover-type one (see columns IBW-NUM and IBW-TEXT, respectively). The percentage of tuples inserted, relative to the initial number of tuples in R', is denoted by ϕ. The different percentages go from 0 to 400, increasing by 100% each time. The data describing the "no incremental" and "incremental" watermarking scenarios are displayed in the group of rows labeled *No Inc.* and *Inc.*, respectively. The column $\phi = 0$ contains data corresponding to the watermark detected before any tuples were inserted into the relation. For each case, the table includes the generated watermark image, and the corresponding SSIM and CF values. Note that the SSIM and CF values in the table are identifiable by their respective domains (see Section 5.1).

Using a high tuple fraction results in fewer embedded marks, leading to the omission of many meta-marks. These omissions manifest as red pixels in the reconstructed watermark figures in the table. However, increasing the number of tuples inserted into R' enhances watermark quality, particularly when incremental watermarking is employed. Conversely, in cases where incremental watermarking is not applied, the extracted watermark exhibits white noise resulting from a high incidence of false positive meta-mark values, which also escalates with the number of tuples. Nevertheless, noise is always higher in the watermark detected when using IBW-TEXT, given the imperfect accuracy of the word sense disambiguation, as previously mentioned. In instances labeled *No Inc.*, a discrepancy between SSIM and CF values is noted, highlighting the unreliability of CF as false positives accumulate. On the other hand, when incremental watermarking is applied, SSIM and CF demonstrate similar trends (refer to Fig. 8).

Table 6 presents the results of a similar experiment as the previous one, using the same template of Table 5. In this case, a higher number of tuples is considered for embedding the watermark, with a tuple fraction of $\gamma = 5$. As a result, each detected watermark exhibits improved quality. Notably, there is a reduced presence of red pixels compared to Table 5. Additionally, due to the high redundancy of marks embedding resulting from the tuple fraction value, the damage to the watermark when incremental watermarking is not applied is lower than when $\gamma = 40$ (see Fig. 9).

The impact of incremental watermarking at the meta-mark level is illustrated in Figs. 10 and 11 for the scenarios outlined in Tables 5 and 6, where incremental watermarking is not applied. In both cases, the red bars indicate the percentage of false meta-marks detected (denoted as $\overline{\mathsf{m}}_F$), while the green bars represent the percentage of true meta-marks detected (denoted as $\overline{\mathsf{m}}_T$). Notably, $\overline{\mathsf{m}}_F$ increases without

Table 5
Quality of the detected watermark for $\gamma = 40$ (incremental vs. no incremental watermarking).

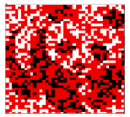
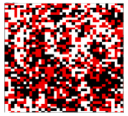
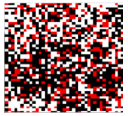
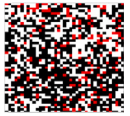
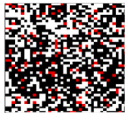
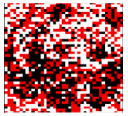
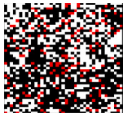
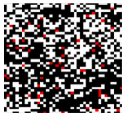
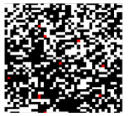
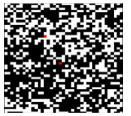
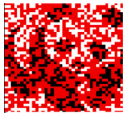
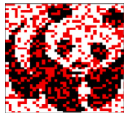
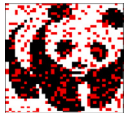
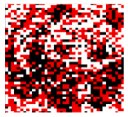
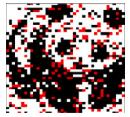
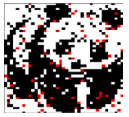
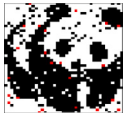
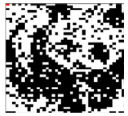
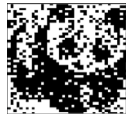
	IBW-NUM					IBW-TEXT				
ϕ	0	100	200	300	400	0	100	200	300	400
										
No Inc.	0.43 *42.44*	0.31 *52.05*	0.25 *55.94*	0.18 *56.05*	0.22 *60.05*	0.38 *54.77*	0.24 *61.16*	0.19 *61.22*	0.17 *61.61*	0.16 *60.44*
										
Inc.	0.43 *42.44*	0.64 *67.83*	0.75 *80.27*	0.85 *88.94*	0.88 *93.72*	0.38 *54.77*	0.55 *78.88*	0.63 *86.66*	0.73 *91.55*	0.79 *94.22*

Table 6
Quality of the detected watermark for $\gamma = 5$ (incremental vs. no incremental watermarking).

	IBW-NUM					IBW-TEXT				
ϕ	0	100	200	300	400	0	100	200	300	400
										
No Inc.	0.94 *98.83*	0.78 *94.16*	0.63 *87.55*	0.55 *84.66*	0.47 *80.83*	0.94 *98.27*	0.72 *91.77*	0.57 *85.11*	0.49 *81.11*	0.49 *79.38*
										
Inc.	0.94 *98.83*	0.99 *99.88*	1.00 *100*	1.00 *100*	1.00 *100*	0.94 *98.27*	0.99 *99.88*	1.00 *100.00*	1.00 *100.00*	1.00 *100.00*

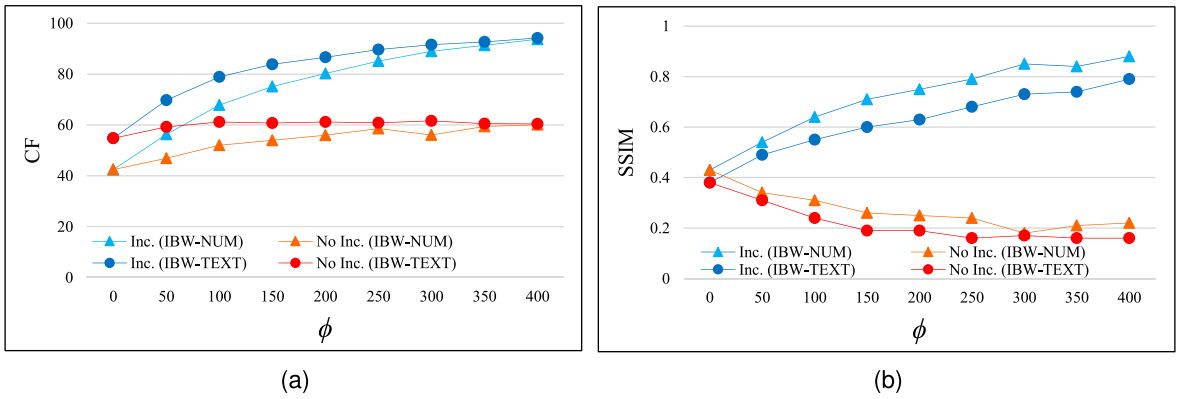


Fig. 8. Quality of the synchronized watermark for $\gamma = 40$ (Incremental Watermarking vs. No Incremental Watermarking). (a) Quality of the watermark according to CF. (b) Quality of the watermark according to SSIM.

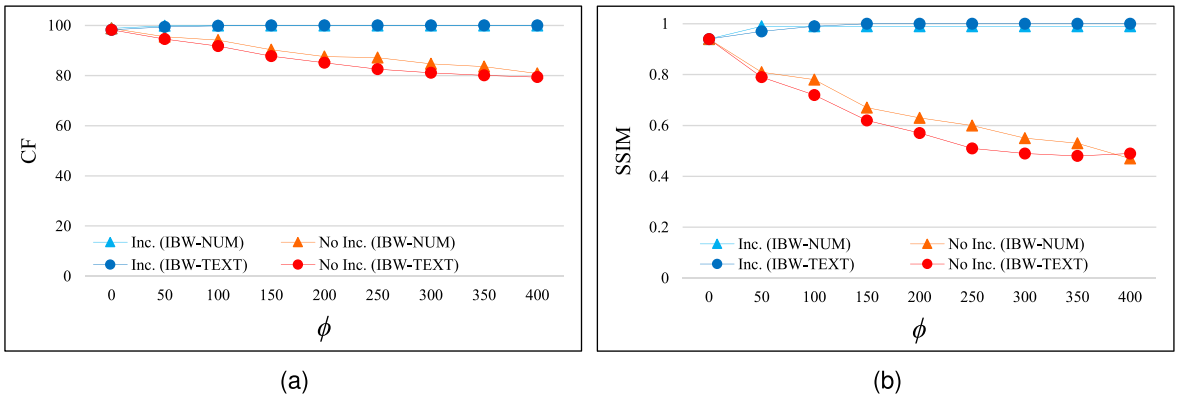


Fig. 9. Watermark synchronized using $\gamma = 5$ (Incremental Watermarking vs. No Incremental Watermarking). (a) Quality of the watermark according to CF. (b) Quality of the watermark according to SSIM.

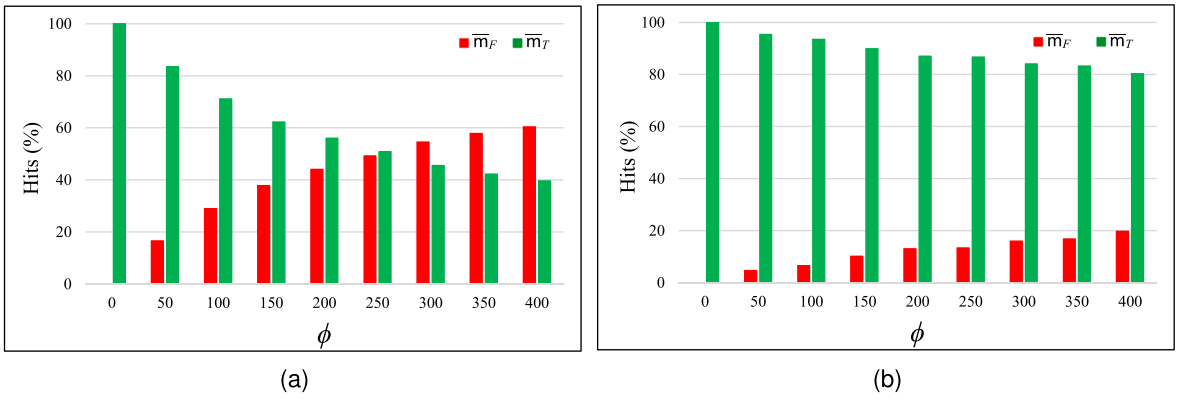


Fig. 10. False positives and true positives of detected meta-marks (given by $\overline{m}_F$ and $\overline{m}_T$ respectively) for IBW-NUM after adding different percentages of tuples (given by ϕ). In both cases, incremental watermarking is ignored. (a) Watermark synchronized using $\gamma = 40$. (b) Watermark synchronized using $\gamma = 5$.

restraint due to the lack of incremental watermarking. Particularly, when $\gamma = 5$, the high number of embedded marks sufficiently encompasses almost all meta-marks without needing tuple insertion, resulting in significantly fewer false positives compared to true positives (refer to Fig. 10(b)). Consequently, compromising the watermark becomes more challenging, necessitating the insertion of many tuples before impacting watermark detection.

Conversely, when incremental watermarking is implemented and executed applied, true positives enhance the watermark signal, resulting in $\overline{m}_T = 100\%$ and $\overline{m}_F = 0\%$ for each value of ϕ. Consequently, the watermark signal strengthens with increasing redundant insertion, bolstering the scheme's resilience against future attacks. Contrary to the relatively low numbers of true and false detected meta-marks, the number of detected marks is significantly higher. This is primarily attributed to the recurrent selection of meta-marks for generating marks (see Fig. 7). Additionally, it is worth noting that the size of the watermark source constrains the number of meta-marks.

We obtain the marks matching their values during watermark extraction to evaluate the performance of the watermarking scheme. This process may not always directly affect the final watermark quality, owing to the application of a majority voting mechanism. Therefore, considering only the values of $\overline{m}_F$ and $\overline{m}_T$ is insufficient for describing synchronization accuracy, as these metrics only represent the final values of the computed meta-marks, rather than the values of the

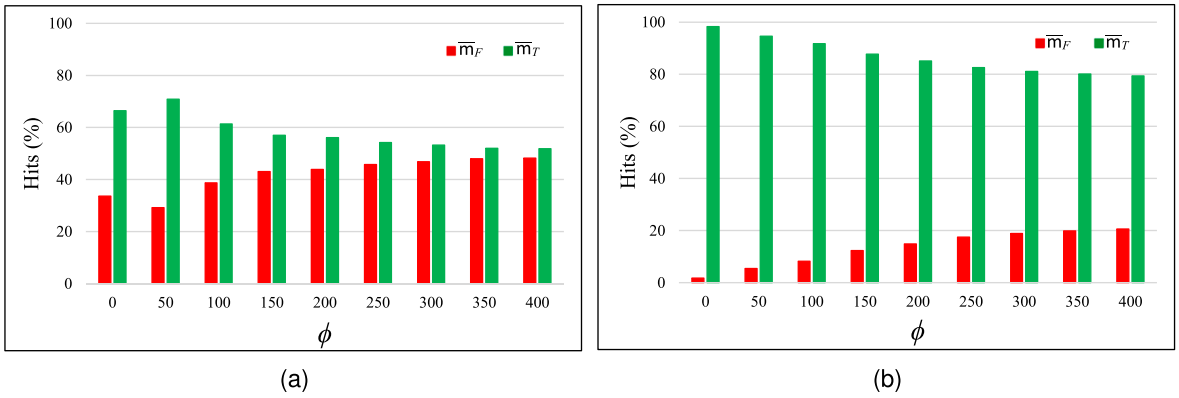


Fig. 11. False positives and true positives of detected meta-marks (given by $\overline{m}_F$ and $\overline{m}_T$ respectively) for IBW-TEXT after adding different percentages of tuples (given by ϕ). In both cases, incremental watermarking is ignored. (a) Watermark synchronized using $\gamma = 40$. (b) Watermark synchronized using $\gamma = 5$.

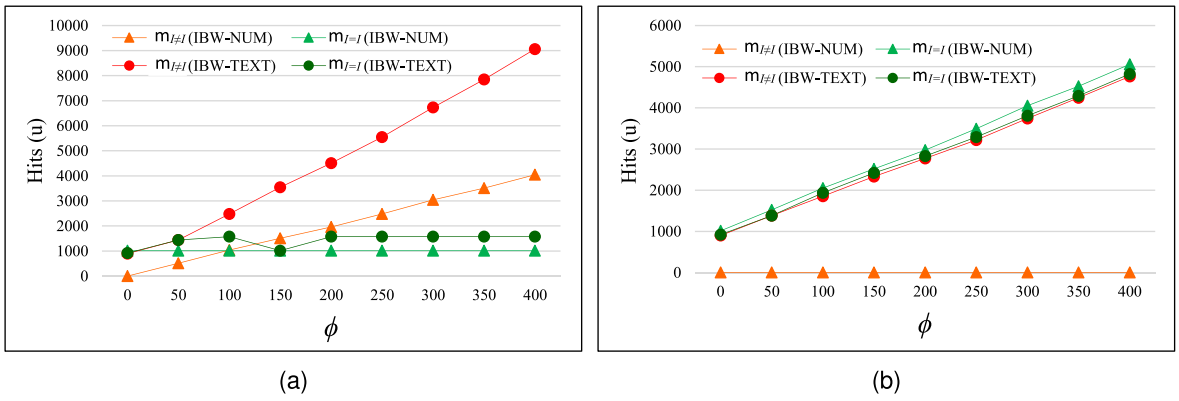


Fig. 12. Quality of the streams of marks detected (given by $m_{I=I}$ and $m_{I\neq I}$) after adding different percentages of tuples (given by ϕ). The watermark synchronization was performed using $\gamma = 40$. (a) No incremental watermarking (b) Incremental watermarking.

individual marks.

Figs. 12 and 13 display the values of $m_{I=I}$ and $m_{I\neq I}$, which involve comparing the array of inserted marks with the array of detected marks, counting the matching values in $m_{I=I}$ and the non-matching ones in $m_{I\neq I}$. The number of marks in both figures varies, reflecting the higher watermark capacity when $\gamma = 5$. However, the scheme's behavior remains consistent. In the scenarios depicted in Figs. 12(a) and 13(a), where incremental watermarking is not applied, while the number of matching values remains almost constant, incorrect values increase significantly. This demonstrates that regardless of the scheme's resilience and the number of inserted marks (as shown in Figs. 10(b) and 11(b) when $\gamma = 5$), failure to implement incremental watermarking inevitably leads to eventual watermark recognition failure.

On the other hand, in cases involving the execution of incremental watermarking (see Figs. 12(b) and 13(b)), while the number of mismatched marks remains at zero for IBW-NUM due to its high detectability, and increases for IBW-TEXT, give that the addition of noise is proportional to the number of tuples, the watermark signal continues to strengthen due to the increasing number of detected values that match their original value.

Finally, we obtain the values of $m_{R(\overline{m})}$ and $m_{W(\overline{m})}$ to analyze how each mark contributes to the generation of its corresponding meta-mark in cases where incremental watermarking is not implemented (see Figs. 14 and 15). Similar to Figs. 10 and 11, the number of meta-marks generated with incorrect values is higher than those with the correct values when $\gamma = 40$ for IBW-NUM, and very close to the correct ones for IBW-TEXT, reflecting a more passive embedding (see Figs. 14(a) and 15(a), respectively).

However, even with watermark synchronization using $\gamma = 5$, the higher number of correct meta-mark values compared to incorrect ones may not offer complete assurance. Notice in Figs. 14(b) and 15(b) how the increase in ϕ leads to a reduction in the difference between $m_{W(\overline{m})}$ and $m_{R(\overline{m})}$. Fig. 16(a) illustrates the behavior of $m_{A(\overline{m})}$, given by the difference between $m_{R(\overline{m})}$ and $m_{W(\overline{m})}$. While changes in $m_{A(\overline{m})}$ may not pose an immediate problem, with a considerable increase in the number of tuples, the significant rise in incorrect meta-marks could eventually jeopardize watermark recognition. This underscores the importance of implementing incremental watermarking regardless of the number of embedded marks or the size of the watermark source.

Moreover, regarding the generated meta-marks, with the implementation of incremental watermarking, $m_{R(\overline{m})}$ increases linearly, whereas $m_{W(\overline{m})}$ remains at zero as long as high detectability is guaranteed. This indicates no incorrect meta-marks are generated, as all recovered marks match their inserted values.

5.3. Impact on robustness, trust and security

We extended the analysis of our architecture to identify how other operations may affect it. This section addresses our proposal's robustness against attribute updates and its behavior when performing tuple deletion, even though tuple deletion is out of the scope of incremental watermarking (see Section 2.3). Furthermore, we discuss how our approach contributes to trust and the way security is achieved.

5.3.1. Dealing with update and deletion of content

It is essential to ensure that our approach does not negatively interfere with the watermark synchronization process of the techniques to which it is applied. If it cannot enhance the quality of the watermark signal, it must remain neutral. For this reason, we include tuple deletion among the operations used to test our approach. This allows us to

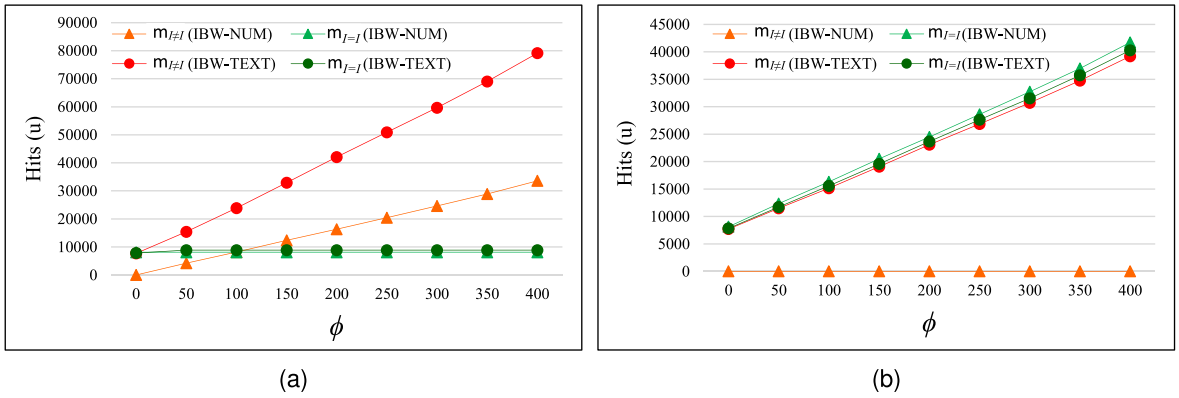


Fig. 13. Quality of the streams of marks detected (given by $m_{I=I}$ and $m_{I\neq I}$) after adding different percentages of tuples (given by ϕ). The watermark synchronization was performed using $\gamma = 5$. (a) No incremental watermarking (b) Incremental watermarking.

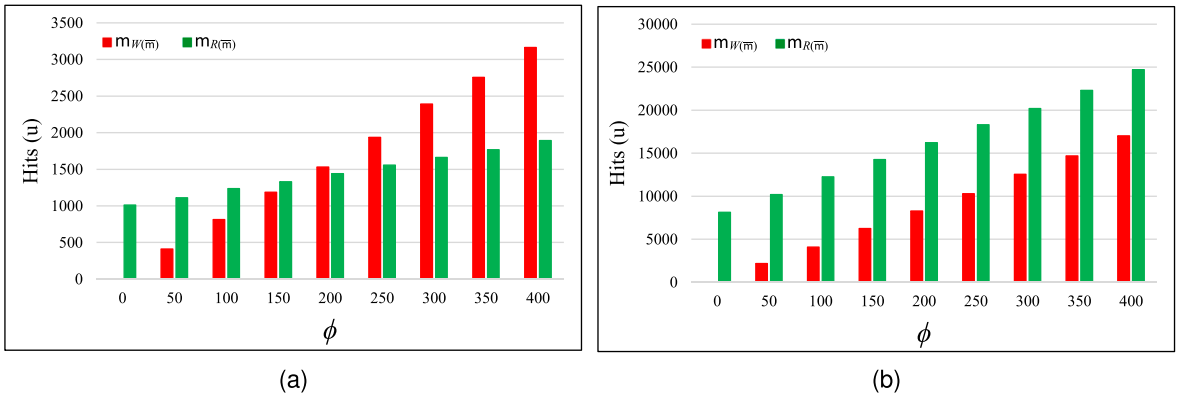


Fig. 14. Number of wrong and right generated meta-marks (given by $m_{W(\overline{m})}$ and $m_{R(\overline{m})}$ respectively) for IBW-NUM after adding different percentages of tuples (given by ϕ). In both cases, incremental watermarking is ignored. (a) Watermark synchronized using $\gamma = 40$. (b) Watermark synchronized using $\gamma = 5$.

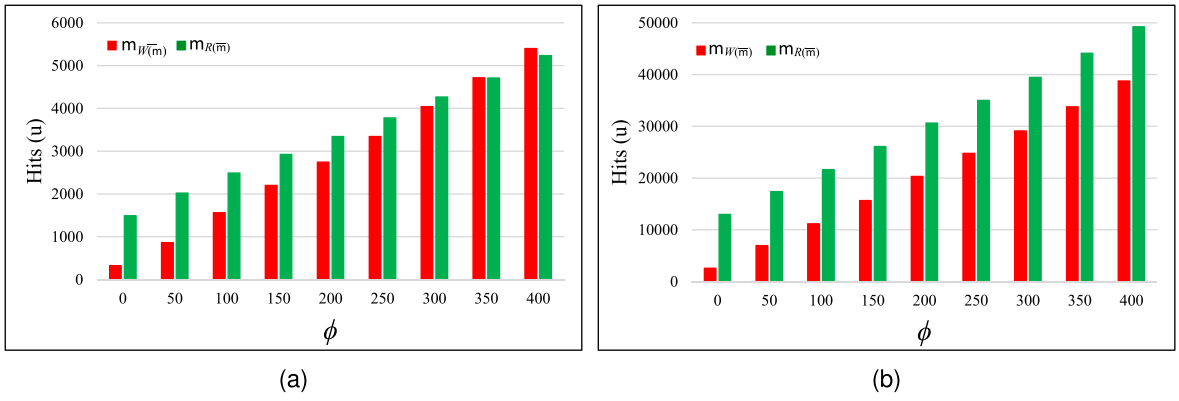


Fig. 15. Number of wrong and right generated meta-marks (given by $m_{W(\overline{m})}$ and $m_{R(\overline{m})}$ respectively) for IBW-TEXT after adding different percentages of tuples (given by ϕ). In both cases, incremental watermarking is ignored. (a) Watermark synchronized using $\gamma = 40$. (b) Watermark synchronized using $\gamma = 5$.

validate its robustness by improving the watermark signal without compromising the core features of the techniques, which are crucial for resisting malicious operations such as subset reverse order and brute force attacks.

The following experiments focus on demonstrating the contribution of our approach by implementing incremental watermarking. In this case, we test our approach on another operation targeted by incremental watermarking: the updating of attributes. For this, we update one attribute per tuple, progressively increasing the proportion of tuples selected for updates by 10%, up to 90%. We compare the quality of the watermark signal by deploying the technique within our architecture, evaluating incremental watermarking versus ignoring its execution. As with previous experiments, this operation was performed using the IBW-NUM and IBW-TEXT techniques. The results are shown in Fig. 17.

Fig. 17(a) illustrates the CF obtained for both $\gamma = 40$ (top) and $\gamma = 5$ (bottom). Despite the differences in the number of tuples involved in the watermarking process due to the varying γ values, the watermark quality exhibits behavior consistent with previous observations. Specifically, the watermark quality is preserved for this operation when incremental watermarking is applied but does not increase as observed during tuple insertion. This is because updates do not increase database content, preventing the insertion of additional marks. Conversely, when incremental watermarking is ignored, watermark quality degrades significantly for IBW-TEXT, while IBW-NUM

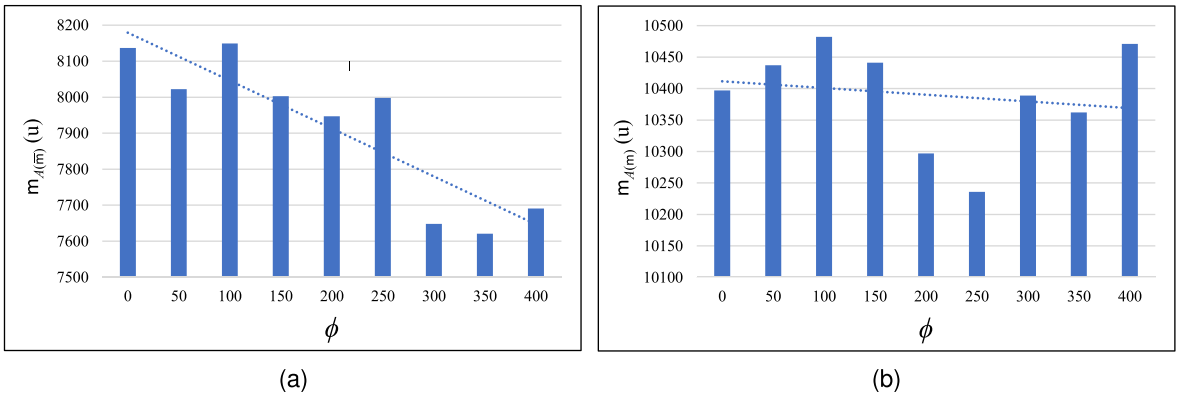


Fig. 16. Recorded changes for $m_{A(\overline{m})}$ due to the increasing number of inserted tuples. (a) IBW-NUM. (b) IBW-TEXT.

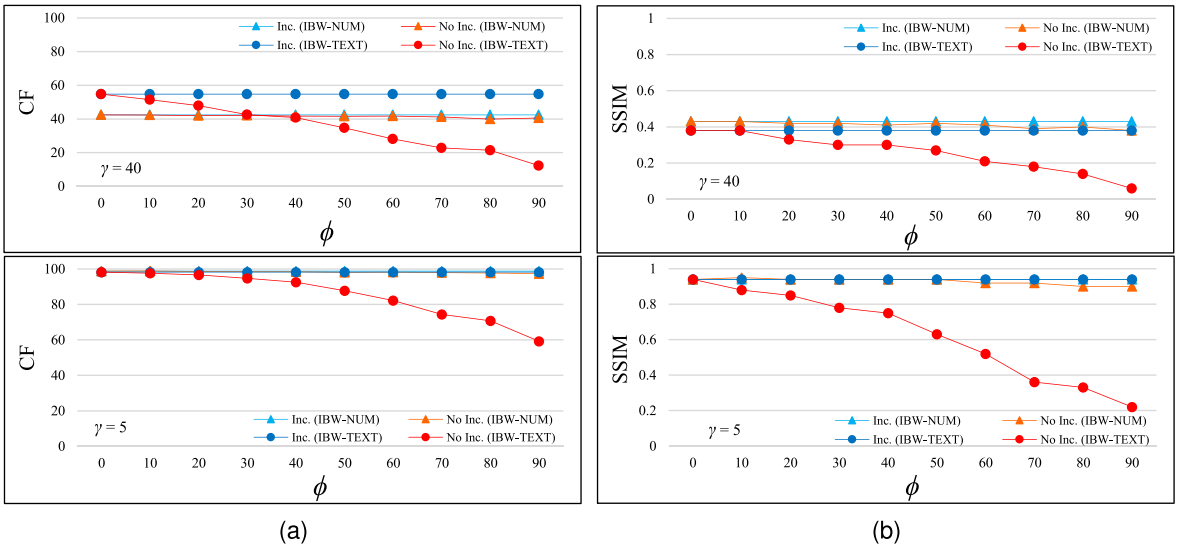


Fig. 17. Quality of the synchronized watermark after update operations (Incremental Watermarking vs. No Incremental Watermarking). (a) Quality of the watermark according to CF. (b) Quality of the watermark according to SSIM.

experiences only minor degradation, which is almost imperceptible in the figure. The superior performance of IBW-NUM can be attributed to its high detectability, combined with majority voting during watermark extraction. In contrast, the lack of perfect accuracy in word sense disambiguation negatively impacts watermark quality for IBW-TEXT, which has the worst effect when incremental watermarking is not used.

The quality of the watermark exhibits a similar trend when measured using SSIM. This indicates that, in the case of tuple updates, both SSIM and CF are reliable metrics and produce consistent results, unlike the discrepancies observed when measuring watermark quality after tuple insertion.

After testing content updates, we conducted experiments to ensure that our approach does not interfere with the technique's performance. Specifically, we analyzed how our proposal impacts the main technique's performance during tuple deletion. As previously noted, tuple deletion falls outside the scope of incremental watermarking since marks contained in deleted content cannot be recovered. For this analysis, we compared IBW-NUM and IBW-TEXT results only, as incremental watermarking does not apply in this scenario (see Fig. 18).

The results confirm the 'natural degradation" of the watermark caused by tuple deletion. Both CF and SSIM illustrate how the quality of the detected watermark signal decreases as the number of deleted tuples increases. This behavior validates that our approach does not interfere with the techniques, as the observed degradation aligns with the expected behavior of the watermark signal under deletion.

It is important to note that these experiments are designed solely to validate the approach. Such aggressive data loss is unlikely in organizational environments, as it would compromise database quality and result in critical content loss.

5.3.2. *Analysis of trust and security*

As previously demonstrated, our approach enhances the watermark signal or avoids negative impact when inserting or updating content in the protected database and does not interfere with the performance of the main technique during content deletion. This highlights the robustness of our approach in handling operations categorized as *benign updates*, which databases encounter daily. Additionally, proving that our approach does not compromise the technique's performance helps build stakeholder trust in its application. However, we further address security and trust when dealing with malicious operations.

The robustness of any watermarking technique is enhanced by the application of incremental watermarking. However, it is essential to assess how the deployment of incremental watermarking influences security and trust. In this regard, the features of the incremental watermarking technique play a crucial role. We address two critical scenarios closely tied to security testing: resilience against subset reverse order attacks and brute force attacks.

First, we address how relational data watermarking techniques can achieve robustness against subset reverse order attacks. This type of

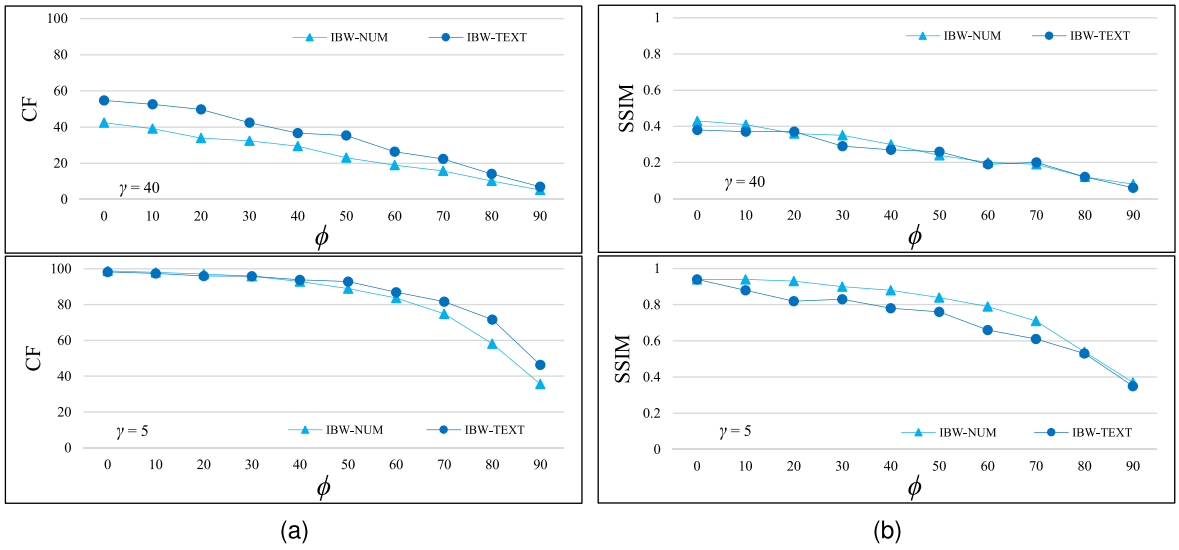


Fig. 18. Quality of the synchronized watermark after tuple deletion. (a) Quality of the watermark according to CF. (b) Quality of the watermark according to SSIM.

malicious operation exploits the lack of fixed order for tuples and attributes in a database relation (see Section 1.1). The order presented in the table structure is primarily for human understanding of the relation design and does not impose a fixed data position when the database is deployed on a server (Agrawal & Kiernan, 2002; Halder et al., 2010). Consequently, if a watermark relies on a specific order of tuples and attributes, attackers can undermine its detection by reordering the data without altering its content quality.

Resilience against subset reverse order attacks can be achieved by avoiding sequential watermark embedding. This can be accomplished by performing a pseudo-random selection of meta-marks and embedding locations within the relation. We consider this approach for deploying incremental watermarking, ensuring that watermark recognition cannot be compromised by updating values or shuffling the order of the elements in the relation. The key to success in this context is selecting by content rather than by position, which is implemented using the virtual keys K_T and K_A, as well as the one-way hash-based functions $F_H(\cdot)$ and $F_V(\cdot)$ (see Sections 4.4 and 4.5).

On the other hand, the key to achieving resilience against brute force attacks lies in the very definition of security, as outlined in Section 2.3. If security depends on the secrecy of the parameters, then the number of parameters and the complexity of their values are crucial. To validate the security of our proposal, we use (5), introduced by Agrawal and Kiernan (2002), to compute the probability of an attacker successfully removing the watermark. Since our scheme increases the number of parameters required for watermark synchronization, its robustness against brute force attacks also improves.

$$P\{success|\omega\} = \left(1 - \frac{1}{2\gamma_A \nu_A \xi_A}\right)^{\omega} \tag{5}$$

In (5), ω denotes the number of watermarked tuples, γ_A represents the fraction of tuples guessed by the attacker, ν_A is the number of attributes used by the attacker for watermark detection, and ξ_A is the range of *lsb* considered for searching the marks. Considering that once our approach is applied, the attacker must account for additional parameters, such as the synchronization matrix M and the number of tuples required to trigger incremental watermarking, G, this equation describes a low probability of success for watermark removal or detection. Finally, with respect to trust, by placing incremental watermarking security in a controlled environment that can be personalized for the data owner's assistance and control, we aim to contribute to the data owner's sense of security regarding where incremental watermarking services are carried out.

5.4. Incremental watermarking performance

We evaluate the performance of incremental watermarking through experiments that combine Cases I and III. We use different values of G (ranging from 1 to 20) to analyze the costs as both G and S vary, thereby highlighting the relationship between these two parameters. For these experiments, watermark insertion was conducted by varying the number of embedded marks, initially using $\gamma = 40$ and then $\gamma = 5$. The number of attributes considered for mark embedding varies using $\delta = 7$, where δ represents the fraction of attributes marked per tuple, as introduced in Pérez Gort et al. (2017). We increase the number of inserted tuples by 10% at each step until the total reaches 100% of the tuples initially stored in the relation (a.k.a $eta = 30.000$).

Fig. 19 illustrates the same pattern regarding the time required to run incremental watermarking *in-house* for cases where $\gamma = 40$ and $\gamma = 5$, as introduced in Section 4. As shown in the figure, the lowest time is required when $G = 1$, and the time increases as both G and ϕ increase. Overall, note the linear behavior of the time, which remains proportional to the number of tuples inserted and the size of the tuple groups. This reliability in the process's implementation and execution stems from its known complexity, allowing for estimating required resources based on the given parameters.

In cases where incremental watermarking is deployed and executed on a node outside the *in-house* environment, such as external servers, additional costs should be factored into the times depicted in Fig. 19. These costs may vary depending on factors such as network bandwidth and concurrent access. However, a performance behavior similar to that shown in the figure is expected, primarily influenced by the values of G and ϕ.

6. Conclusions

In this work, we highlighted the relevance of implementing incremental watermarking and showed the consequences of ignoring it, often due to the lack of trust between developers and data owners. We focus on boosting trust by addressing the nodes where incremental watermarking is triggered after deploying the watermarked database. This paper proposes a set of alternatives for keeping the watermark updated on the protected database if the lack of trust among the watermark scheme stakeholders remains. Some of the proposals in this work require additional computational resources. Through experiments, we validated how they contribute to maintaining requirements such as security and robustness.

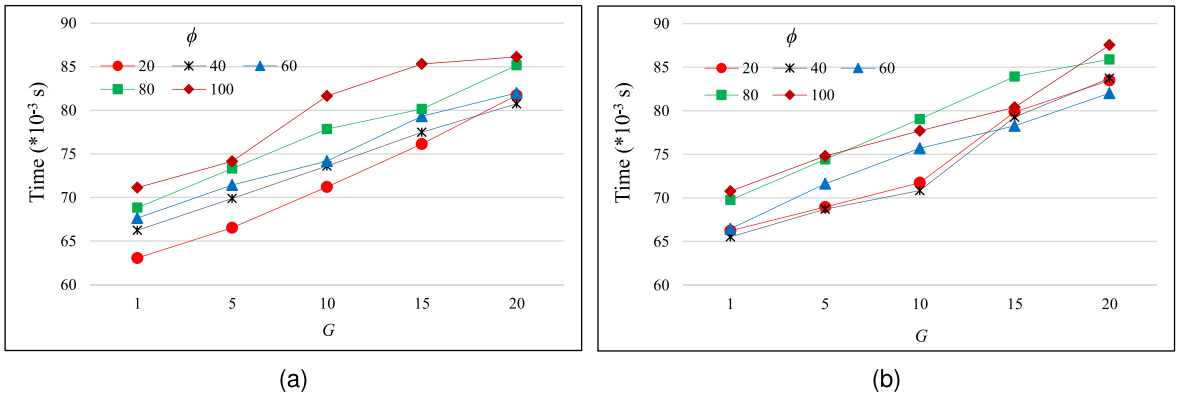


Fig. 19. Time required (in milliseconds) to execute incremental watermarking *in-house* according to Case III of Fig. 4: (a) Watermark synchronized using $\gamma = 40$. (b) Watermark synchronized using $\gamma = 5$.

Thanks to the proposals in our work, the roles involved in the watermarking scheme can consider different alternatives, giving higher relevance to the precision of protection or the performance of the services offered by the database engine. Furthermore, considering the scenarios presented in this paper, watermarking can be deployed as a service, where the database administrator can define the frequency to trigger the processes involved in incremental watermarking.

As future work, we aim to design and implement an engine for deploying incremental watermarking in a distributed environment. We intend to incorporate the benefits of distributed data management into the watermarking scheme, making it possible to boost performance and balance workload. This idea comes with new challenges, such as concurrency management and a different way to deal with security in additional connection points. With the help of distributed database management systems and further considerations on the watermarking scheme, dealing with these challenges should pay off, considering the benefits of the distributed environment. We do not necessarily intend to constrain our solution to a distributed environment; instead, we want to make it robust to deployment in distributed and centralized environments.

CRediT authorship contribution statement

Maikel Lázaro Pérez Gort: Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Agostino Cortesi:** Writing – review & editing, Supervision, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Agrawal, R., & Kiernan, J. (2002). Watermarking relational databases. In *VLDB'02: Proceedings of the 28th international conference on very large databases* (pp. 155–166). Hong Kong, China: Elsevier.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (2008). *Pattern-oriented software architecture: A system of patterns: Vol. 1*. Prague, Czech Republic: John wiley & sons.

Cao, Z., Sun, J., & Hu, Z. (2010). Image algorithm for watermarking relational databases based on chaos. In *Advances in wireless networks and information systems* (pp. 411–418). Springer.

Colorado-State-University (1999). Forest CoverType, the UCI KDD archive.

Cox, I., Miller, M., Bloom, J., Fridrich, J., & Kalker, T. (2007). *Digital watermarking and steganography* (2nd ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Date, C. J. (2003). *An introduction to database systems* (8th ed.). Inc.: Addison-Wesley Longman Publishing Co..

Farfoura, M. E., Horng, S.-J., & Wang, X. (2013). A novel blind reversible method for watermarking relational databases. *Journal of the Chinese Institute of Engineers*, *36*(1), 87–97.

Franco-Contreras, J., & Coatrieux, G. (2015). Robust watermarking of relational databases with ontology-guided distortion control. *IEEE Transactions on Information Forensics and Security*, *10*(9), 1939–1952.

Franco-Contreras, J., Coatrieux, G., Cuppens-Boulahia, N., Cuppens, F., & Roux, C. (2014). Ontology-guided distortion control for robust-lossless database watermarking: Application to inpatient hospital stay records. In *2014 36th annual international conference of the IEEE engineering in medicine and biology society* (pp. 4491–4494). IEEE.

Gross-Amblard, D. (2003). Query-preserving watermarking of relational databases and xml documents. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems* (pp. 191–201). San Diego, CA, USA: ACM.

Gross-Amblard, D. (2011). Query-preserving watermarking of relational databases and xml documents. *ACM Transactions on Database Systems*, *36*(1), 1–24.

Guang, S., Xiaoping, F., Sha, F., Yingjie, S., & Huifang, L. (2015). Software watermarking in the cloud: Analysis and rigorous theoretic treatment. *Journal of Software Engineering*, *9*(2), 410–418.

Halder, R., Pal, S., & Cortesi, A. (2010). Watermarking techniques for relational databases: Survey, classification and comparison. *Journal of Universal Computer Science*, *16*(21), 3164–3190.

Hamadou, A., Hassane, A. A. I., Naroua, H., et al. (2024). Reversible semi-fragile watermarking technique for integrity control of relational database. *Engineering*, *16*(9), 309–323.

Han, J., Peng, X., Xian, H., & Yang, D. (2024). A distortion free watermark scheme for relational databases. In *2024 33rd international conference on computer communications and networks* (pp. 1–6). IEEE.

Hou, R., & Xian, H. (2021). A graded reversible watermarking scheme for relational data. *Mobile Networks and Applications*, *26*, 1552–1563.

Hu, D., Wang, Q., Yan, S., Liu, X., Li, M., & Zheng, S. (2023). Reversible database watermarking based on order-preserving encryption for data sharing. *ACM Transactions on Database Systems*, *48*(2), 1–25.

Hu, D., Zhao, D., & Zheng, S. (2018). A new robust approach for reversible database watermarking with distortion control. *IEEE Transactions on Knowledge and Data Engineering*, *31*(6), 1024–1037.

Iftikhar, S., Kamran, M., & Anwar, Z. (2014). RRW—A robust and reversible watermarking technique for relational data. *IEEE Transactions on Knowledge and Data Engineering*, *27*(4), 1132–1145.

Jawad, K., & Khan, A. (2013). Genetic algorithm and difference expansion based reversible watermarking for relational databases. *Journal of Systems and Software*, *86*(11), 2742–2753.

Kamran, M., Suhail, S., & Farooq, M. (2013). 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.

Katzenbeisser, S., & Petitcolas, F. A. (2000). *Information hiding techniques for steganography and digital watermarking* (1st ed.). USA: Artech House, Inc..

Khan, A., Jabeen, F., Naz, F., Suhail, S., Ahmed, M., & Nawaz, S. (2016). Buyer seller watermarking protocols issues and challenges—A survey. *Journal of Network and Computer Applications*, *75*, 317–334.

Laudon, K. C., & Laudon, J. P. (2004). *Management information systems: Managing the digital firm*. Boston, MA, USA: Pearson Educación.

Li, W., Li, N., Yan, J., Zhang, Z., Yu, P., & Long, G. (2022). Secure and high-quality watermarking algorithms for relational database based on semantic. *IEEE Transactions on Knowledge and Data Engineering*, *35*(7), 7440–7456.

Li, Y., Swarup, V., & Jajodia, S. (2003). Constructing a virtual primary key for fingerprinting relational data. In *Proceedings of the 3rd ACM workshop on digital rights management* (pp. 133–141).

Lin, C.-C., Nguyen, T.-S., & Chang, C.-C. (2021). LRW-CRDB: Lossless robust watermarking scheme for categorical relational databases. *Symmetry*, *13*(11), 2191.

Manjula, R., & Settipalli, N. (2010). A new relational watermarking scheme resilient to additive attacks. *International Journal of Computer Applications*, *10*(5), 1–7.

McAuley, J. J., & Leskovec, J. (2013). From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on world wide web* (pp. 897–908). Rio de Janeiro, Brazil: ACM.

Mehta, B., & Rao, U. P. (2011). A novel approach as multi-place watermarking for security in database. In *SAM'11, 10th international conference on security and management* (pp. 703–707). Las Vegas, NV, USA: arXiv.

Mrdovic, S., & Perunicic, B. (2008). Kerckhoffs' principle for intrusion detection. In *Networks 2008-the 13th international telecommunications network strategy and planning symposium* (pp. 1–8). Budapest, Hungary: IEEE.

Nadhir, N., Ali, S., & Abdelkrim, G. (2021). Medical image watermarking scheme in transform domain based on asymmetric crypto-system and arnold chaotic map. In *2021 44th international conference on telecommunications and signal processing* (pp. 267–272). IEEE.

Naz, F., Khan, A., Ahmed, M., Khan, M. I., Din, S., Ahmad, A., et al. (2020). Watermarking as a service (WaaS) with anonymity. *Multimedia Tools and Applications*, *79*(23), 16051–16075.

Pavlou, K. E., & Snodgrass, R. T. (2013). Generalizing database forensics. *ACM Transactions on Database Systems*, *38*(2), 1–43.

Pérez Gort, M. L., Feregrino-Uribe, C., Cortesi, A., & Fernández-Peña, F. (2019). HQR-scheme: A high quality and resilient virtual primary key generation approach for watermarking relational data. *Expert Systems with Applications*, *138*, Article 112770.

Pérez Gort, M. L., Feregrino Uribe, C., & Nummenmaa, J. (2017). A minimum distortion: High capacity watermarking technique for relational data. In *Proceedings of the 5th ACM workshop on information hiding and multimedia security* (pp. 111–121). Philadelphia, PA, USA: ACM.

Pérez Gort, M. L., Olliaro, M., & Cortesi, A. (2022). Reducing multiple occurrences of meta-mark selection in relational data watermarking. *IEEE Access*, *10*, 62210–62231.

Pérez Gort, M. L., Olliaro, M., & Cortesi, A. (2023). Relational data watermarking resilience to brute force attacks in untrusted environments. *Expert Systems with Applications*, *212*, Article 118713.

Pérez Gort, M. L., Olliaro, M., Cortesi, A., & Uribe, C. F. (2021). Semantic-driven watermarking of relational textual databases. *Expert Systems with Applications*, *167*, Article 114013.

Rani, S., & Halder, R. (2022). An efficient format-independent watermarking framework for large-scale data sets. *Expert Systems with Applications*, *208*, Article 118085.

Rani, S., Koshley, D. K., & Halder, R. (2017). Partitioning-insensitive watermarking approach for distributed relational databases. In A. Hameurlain, J. küng, R. Wagner, T. Dang, & N. Thoai (Eds.), *Lecture notes in computer science*: *Vol. 10720*, *Transactions on large-scale data- and knowledge-centered systems XXXVI* (pp. 172–192). Springer, Berlin, Heidelberg: Springer.

Shen, X., Zhang, Y., Wang, T., & Sun, Y. (2020). Relational database watermarking for data tracing. In *2020 international conference on cyber-enabled distributed computing and knowledge discovery* (pp. 224–231). IEEE.

Siledar, S., & Tamane, S. (2020). A distortion-free watermarking approach for verifying integrity of relational databases. In *2020 international conference on smart innovations in design, environment, management, planning and computing* (pp. 192–195). IEEE.

Tzouramanis, T. (2011). A robust watermarking scheme for relational databases. In *2011 international conference for internet technology and secured transactions* (pp. 783–790). IEEE.

Waheeb Yaqub, I. K., & Aung, Z. (2018). Distortion-free watermarking scheme for compressed data in columnar database. In *Proceedings of the 15th ICETE* (pp. 343–353). Porto, Portugal: SciTePress.

Wang, C., & Li, Y. (2023). A copyright authentication method balancing watermark robustness and data distortion. In *2023 26th international conference on computer supported cooperative work in design* (pp. 1178–1183). Rio de Janeiro, Brazil: IEEE.

Zhang, Y., Wang, Z., Wang, Z., & Liu, C. (2022). A robust and adaptive watermarking technique for relational database. *Cyber Security*, 3.