GeoInformatica (2026) 30:6
https://doi.org/10.1007/s10707-026-00566-2

RESEARCH

Earth observation data provenance protection through self-recalibrated watermarking

Maikel Lázaro Pérez Gort[1] · **Agostino Cortesi**[1]

Received: 27 February 2025 / Revised: 13 January 2026 / Accepted: 29 January 2026

Abstract
Earth observation (EO) data are the foundation for any web service relying on remote sensing and geographic information systems. Protecting their provenance is crucial yet challenging, given their susceptibility to rapid and widespread copying and dissemination. Often, when tampering or unauthorized use of the data is detected, the data has already been disseminated. This work presents an effective watermarking technique for tracking EO data copies and validating their provenance and authenticity without compromising remote sensing algorithms' functionality. Our approach allows the relocation of the watermark while ensuring the operability of the database. This feature is particularly useful when the database content's priorities and attributes' tolerance to distortion change over time. Experimental results indicate that our method outperforms existing techniques regarding data provenance and tampering detection when considering multi-type digital assets repositories. It also establishes the foundation for protecting linked content stored using different data types.

Keywords Digital watermarking · Earth observation data · Geographic features · Provenance verification

1 Introduction

The evolution of programming technologies has made it possible to store and manage data in sophisticated ways. The different formats utilized guarantee portability, deployment, and content replication. Furthermore, the information derived from the data can be corrected and enhanced using additional sources targeting the same objects of interest and periods of observation. In these circumstances, protecting and increasing data accuracy is imperative,

✉ Maikel Lázaro Pérez Gort
maikel.perezgort@unive.it

Agostino Cortesi
cortesi@unive.it

[1] Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Via Torino 155, 30172 Venice, Italy

Published online: 11 February 2026

considering that many data providers are widespread, and their products can be easily acquired. The benefits of the provider's diversity and the high speed of data dissemination also favor disinformation and misinformation when sources are not subjected to content accuracy checks.

The case of Earth observation (EO) data is a clear manifestation of previous statements. Data targeting the same observation time and region of interest can be obtained with different devices and sensors. The generated digital content is often stored in different formats and registered using different measurement units (see Fig. 1). A diverse set of EO data providers on the Internet allows downloading and managing the content easily and rapidly [1]. This makes it possible to corroborate and improve content accuracy.

EO data has characteristics that make it possible to link content beyond the format used to store it. Table 1 shows the features that make EO data unique. Like all digital content, EO data products are challenged by undetected or late-detected data tampering, which might lead to inaccurate information that misleads the decision-making process of organizations and users, causing economic loss and putting their competitiveness and future at risk.

This work focuses on EO data, as they enable the representation of the same information in different formats. By matching the data through the various levels of content representation, it is possible to move beyond the traditional static selection of items within the data when protecting them with watermarking techniques. Nevertheless, the idea of dynamic selection of content for watermark generation extends beyond existing approaches that merely introduce variability during watermark embedding and extraction. The principles introduced in our work can be extended to any kind of content as long as they exhibit the features of EO data presented in Table 1.

As the main objective, this work aims to develop a mechanism to safeguard and identify EO data provenance, making copyright and authenticity protection possible. We focus on checking data format without discriminating among their types or performing modifications that can compromise their usability or interfere with remote sensing algorithms' accuracy and performance.

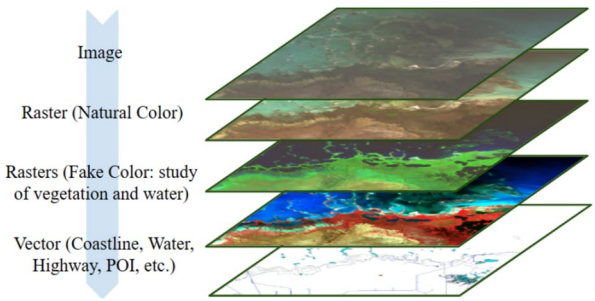


Fig. 1 Example of links between different OE data types focusing on the same study region, storing content in rasters and vectors containing polygons and points of interest (POI), among others [2]

Table 1 EO data features relevant to our work

Feature	Description
Format	Several (e.g., image, documents, table)
Dimensions	More than one (e.g., region of interest and observation period)
Providers	Diverse and large in number

The main contribution of this work is an engine for resynchronizing the watermark during the database's operational phase. To the best of our knowledge, no mechanism for dynamic watermark resynchronization has been previously introduced or discussed in the literature. Our proposal obtains data from different sources that point to the same objects of interest and links them, considering matching values of dimensions such as space and time. This way, a repository containing extensive EO data is built. We developed a complex watermark scheme to protect the provenance of EO data using vertical watermark synchronization (comprised of standard watermark embedding and extraction processes) and horizontal synchronization (a new synchronization approach proposed in this work). The horizontal synchronization consists of relocating the marks to other locations in the repository while guaranteeing the functionality of the database (i.e., without interfering with the outcomes of remote sensing algorithms). This approach makes it possible to relocate the distortion caused by the watermark while increasing the robustness of the scheme against malicious operations such as data tampering and false ownership claims.

Experimental results show that vertical and horizontal synchronization do not interfere and guarantee watermark detection at any time. We also performed a set of experiments to validate the performance of our proposal. Regardless of the volume of data being protected, all processes experience linear complexity with respect to the number of cells in the repository. Finally, the results also validate the robustness of our scheme. As a result of our approach, provenance information cannot be corrupted despite the high rate of malicious operations. We are also able to detect when data integrity cannot be reliably guaranteed. We implemented our proposal within an architecture designed for provenance identification. It combines the principles of robust and fragile watermarking (see Section 2), increasing the dynamism of the mark's carrier selection.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries and related work, focusing on the current status of EO data providers, relevant concepts of digital watermarking, and existing techniques for ownership protection and data tampering detection. Section 3 contains details of our proposal, and Section 4 presents the experimental results. Finally, Section 5 outlines the conclusions and future work.

2 Background and related work

Providers of EO data must implement policies defined by international norms and organizations to access and manage their products. They are responsible for guaranteeing data quality and authenticity data. Furthermore, clients are required to use the data in accordance with the established distribution and usage conditions. This section introduces a generic model of EO data distribution, considering the main aspects that providers must address. It also discusses the capabilities of digital watermarking as a promising solution for EO data protection. Given the different features considered in our approach, this section is not limited to a specific type of watermarking technique. In addition to the fundamentals of digital watermarking, both robust and fragile techniques are formally introduced.

Providers of EO data must implement policies defined by international norms and organizations to govern access to and management of their products, and they are responsible for ensuring data quality and authenticity. In addition, clients are required to use the data in accordance with established distribution and usage conditions.

2.1 The challenges of EO data distribution

A large number of EO data providers allow easy access to their data. This practice is motivated by the generally beneficial use of EO data for human development, which underpins open data distribution policies for non-commercial purposes. In line with international laws and policies, organizations such as the European Union and the G7 (previously G8) have established that digital data serving the public sector should be made accessible with minimal barriers [1].

To clarify how data are offered and exchanged, it is important to define concepts such as open data, free of charge, and full access. *Open data* refers to data that can be freely used, managed, and redistributed without commercial restrictions. *Free data* refers to data free of charge, considering that free availability does not necessarily imply open usage rights. Notice that open data inherently include free-of-charge access. Finally, *full access* refers to the availability of the complete dataset rather than part of it. This level of access may be constrained by security and data protection mechanisms or by technical limitations [1].

An illustrative example of the different data access and distribution strategies is provided by the data-sharing principles of the Global Earth Observation System of Systems (GEOSS)[1] The first principle promotes the full and open exchange of EO data, metadata, and related products, while recognizing applicable international instruments as well as national policies and legislation. The second principle states that all data shall be made available with minimal time delay and at minimal cost. The third principle specifies that shared data should be free of charge (or provided at no more than the cost of reproduction) for research and education purposes. Another example of a data access strategy is provided by the International Disaster Charter (Charter on Cooperation to Achieve the Coordinated Use of Space Facilities in the Event of Natural or Technological Disasters), which enables the provision of full, open, and free of charge EO data to disaster-related agencies submitting formal requests through authorized users.

In line with the principle of easy access, requesting EO data from any provider typically involves three main data packages: (i) Region and Features of Interest, (ii) Contact Information, and (iii) High-level (HL) Access Data. These packages differ in content, with their importance determined by how frequently they are requested and how essential they are for each request. Figure 2 illustrates the data packages involved, although the order in which they are acquired may vary depending on the provider policies. Providers may offer different versions of EO data for a given time and region of interest, with varying levels of completeness depending on user access rights. However, access to higher-quality versions often requires additional confirmation steps and may incur financial costs.

Each data package is described as follows:

- **Region and features of interest:** It contains data defining the geographical region of interest, including the time at which the observation was performed. The features of interest are required to target a specific sensor or satellite instrument, as well as to identify data pre-processed using feature-related models (e.g., Vegetation Drought Response Index (VegDRI)[3, 4])[2]. These data are mandatory and independent of the provider.

[1] Information about GEOSS is available at its official website, https://earthobservations.org/geoss.php.

[2] Defining the features of interest in the requested EO data is important, considering that different sensors may generate different data from the same region and observation time. For example, studying forest conditions

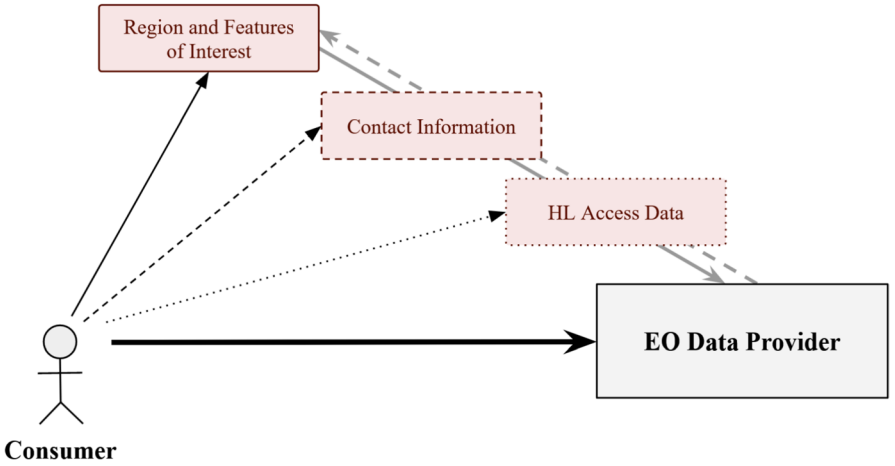


Fig. 2 Packages of data considered when requesting EO data

- **Contact information:** It contains the contact data about the requesting user, as well as access-related information used to verify the research domain or affiliation of the individual downloading the data. It may include information such as email address and access credentials (e.g., username and password). This package is not always required by EO data providers, and the security mechanisms used to deliver this information may vary (e.g., sometimes it might be enough to create an account and confirm the email address provided).
- **HL access data:** This category of data is required when the information to be acquired is highly sensitive or classified as premium. Providers request this information when some regions are restricted from public access, such as government institutions, military facilities, or sensitive research areas. This package may also include requests for real-time data. The HL Access Data package comprises advanced (and highly verified) contact and payment-related information.

Table 2 presents examples of EO data providers operated by major space exploration and remote sensing organizations. These institutions may represent national governments, international collaborations, or private companies. In most cases, providers are implemented as web applications. In addition, some providers offer direct download links and function primarily as data repositories, whereas others manage data requests through web services.

Despite the security policies enforced through the different packages, acquiring most EO data typically requires only the first package. Nevertheless, even when contact information (a.k.a the second package) is required , no *blind*[3] method exists to ensure data quality in the presence of suspected tampering or unauthorized redistribution. On the other hand,

requires different data than analyses focused on changes in water formations.

[3] The term *blind* denotes the requirement of not utilizing a previously generated copy of the data prior to an unauthorized modification. Similarly, when external digital signals are employed for protection, such as watermarks, the original watermark cannot be used. In contrast, *non-blind* methods allow the use of the original data copy together with the source used to generate the watermark.

Table 2 Example of EO data providers

Provider	Details
Earth Data portal	From NASA's Earth Science Division. It is part of NASA's Earth Science Data Systems (ESDS) Program, which focuses on maximizing the scientific return from NASA's missions and experiments. Earth Data portal is available at https://www.earthdata.nasa.gov/
Copernicus	From the European Union's Earth observation programme implemented in partnership with the Member States, the European Space Agency (ESA), and many others. Copernicus is available at https://www.copernicus.eu/en
Group on Earth Observations (GEO)	A partnership of more than 100 national governments and over 100 Participating Organizations. GEO's official website is https://earthobservations.org/geo_community.php

non-blind approaches may also be compromised when multiple versions of EO data are deliberately mixed[4] as part of a malicious operation.

Beyond earlier assumptions, the notion that data can be freely distributed solely for benign purposes of human progress is increasingly difficult to sustain. The current international context, characterized by growing geopolitical tensions, poses significant challenges to the implementation of such open data paradigms makes . Trust among governments has diminished, and cyberspace has emerged as an additional domain of confrontation, placing the authenticity and reliability of data at risk. Cyber-attacks targeting governments and public institutions have become increasingly frequent, while misinformation and disinformation have gained prominence. In this context, protecting data and providing verifiable evidence of provenance without undermining the benefits of open and free data distribution has become both critical and urgent.

2.2 Digital watermarking

Watermarking techniques enable data ownership protection and tampering detection without constraining database deployment and management. They contribute to addressing the security gaps inherent in traditional approaches such as authentication and authorization. Some watermarking schemes are classified as distortion-based (mostly created for ownership protection and traitor tracing) and embed the watermark while ensuring its imperceptibility while preserving data usability. Given the resilience required against operations intended to remove the watermark or compromise its detection, these schemes are also classified as robust. On the contrary, distortion-free approaches do not introduce any perturbation into the data but must ensure authenticity verification at any time. Considering their primary objective (a.k.a enabling watermark destruction once data modifications exceed thresholds defined by the data owner so that tampering can be detected), most distortion-free approaches are classified as fragile.

Formally, the watermark constitutes a signature composed of bits, each referred to as a mark. Distortion-based approaches select data locations that can tolerate distortion without affecting usability in order to embed these marks. By contrast, some distortion-

[4] Mixing EO data can be used to compromise hidden signals intended for data protection, following principles similar to those of *collusion attacks* used to compromise fingerprinting schemes. [5] (see Section 2.2.6)

free approaches derive a watermark directly from the data and use it later to verify data integrity. Most watermarking approaches proposed to date rely on a static selection of data elements associated with the watermark. This implies that once a data element is selected to play a role in embedding or extracting a mark, this role remains fixed according to the parameters used for performing the watermark synchronization[5]. Such static selection introduces vulnerabilities to malicious operations aimed at compromising data quality or watermark detection. When watermark detection is undermined, the evidentiary value of the watermark for ownership verification is lost, enabling false ownership claims. Furthermore, once a scheme is compromised, data tampering and unauthorized copies of the data can be performed without compliance with acquisition contracts.

The watermarking architecture comprises two processes: watermark embedding and extraction. We denote the watermark as W and define each mark composing W as $\mathrm{m}_j \in \{0, 1\}$, where $j \in [0, (|W| - 1)]$ and $|\cdot|$ denotes the cardinality operator applied to the set of marks composing W. The embedding and extraction of W are defined by the functions $\mathcal{I}(\cdot)$ and $\mathcal{E}(\cdot)$, respectively. Throughout the paper, multiple symbols are used; for clarity, the reader may refer to Table 3.

Formally, watermark embedding is defined as $\mathrm{A}' = \mathcal{I}(P, W, \mathrm{A})$, where A denotes the digital asset into which W is embedded and A' represents its watermarked version. The symbol P denotes the set of parameters used to select the fragments of A where the watermark is embedded. In accordance with the requirement of the key-based system [5], at least the secret key (known only to the data owner) used for watermark synchronization must be included in P. We denote this key as K_{S}, such that $K_{\mathrm{S}} \in P$. On the other hand, watermark extraction is defined as $W' = \mathcal{E}(P, \mathrm{A}'')$, where A'' is presumably a version of A' that may have undergone *benign updates*[6] and/or *malicious operations*[7]. The symbol W' represents the watermark extracted from A''.

To prove ownership or the absence of data tampering, the extracted watermark must be similar to the embedded one (formally, $W \approx W'$). Comparing watermarks based on similarity rather than strict equality accounts for the possibility that the extracted watermark may be degraded within a tolerable threshold due to *benign updates* or *malicious operations* that do not remove the watermark and do not compromise data quality.

2.2.1 Robust watermarking

Robust watermarking techniques are designed to be resilient against malicious operations aimed at watermark removal or at compromising its detection. Owing to their robustness, these techniques are commonly used to prove ownership. Their implementation follows the architecture previously described and depicted in Fig. 3.

Some robust techniques are defined as *fingerprinting* and are created to identify different copies of the same data. These techniques are particularly useful for data protection in scenarios involving multiple buyers of the same digital content. Fingerprinting approaches

[5] Synchronization refers to the process of aligning two signals in time or space [6], in this case the embedded and extracted watermarks.

[6] *Benign updates* are defined as regular and authorized operations performed on the protected data during its management.

[7] *Malicious operations* (also referred to as attacks) are intended to remove the watermark while preserving data quality or to tamper with the data.

Table 3 Description of notations and symbols

Symbol	Description	Symbol	Description
W	watermark	$\lvert\cdot\rvert$	cardinality operator
w	size of the watermark, $w = \lvert W \rvert$	$\approx$	similarity operator
m	mark contained in the watermark	$\overline{\mathrm{m}}$	meta-mark extracted from the watermark source
$\mathcal{I}(\cdot)$	watermark embedding function	$\mathcal{E}(\cdot)$	watermark extraction function
A	digital asset to watermark	A'	watermarked digital asset
P	set of embedding/extraction parameters	K_{S}	data owner's secret key
A''	modified version of A'	W'	watermark extracted from A''
$+p$	provenance signal watermark	$+a$	authorship signal watermark
$+o$	ownership signal watermark	$\varkappa$	number of observed objects
α	number of focused objects	O_i	observed object
CD	composed data package	ND	normalized data package
D	set of dimensions	$D[i]$	repository dimensions
F	set of facts	F_j	facts stored in the repository
δ	threshold for detecting fact's versions	C_i	mark carriers
$\mathbb{A}$	stage identifying carriers selected after watermark embedding	$\mathbb{B}$	stage identifying carriers right before watermark extraction
$\equiv$	equivalent relationship between digital asset stages	$\not\equiv$	not equivalent relationship between digital asset stages
κ	number of elements to consider from a dimension of the repository	F_0	actual fact
RND( )	random selection function	SEED	number used as seed
β	number of most significant bits (*msb*)	ξ	number of less significant bits (*lsb*)
$\mathcal{X}$	dimension reduction filter	N	array of number of elements per dimension to contemplate
φ	fact's version fraction	$\Gamma(\cdot)$	function to obtain the dimensions set of the repository
$\Lambda(\cdot)$	function to apply the filter $\mathcal{X}$ to reduce dimension values	$\hat{d}_j$	dimension values contained in D
d	set of current dimension values	$\mathcal{F}$	set of versions of a particular fact
f	fact version obtained from $\mathcal{F}$	$H(\cdot)$	one-way hash function to generate the virtual primary key (VPK)
$H_\beta(\cdot)$	one-way hash function to obtain a *msb* value	b_β	selected *msb* value
$H_\xi(\cdot)$	one-way hash function to obtain a *lsb* position	p_ξ	selected *lsb* position to embed the mark
$H_{\overline{\mathrm{m}}}$	hash function used to select the meta-mark from the watermark source	$\oplus$	*xor* operator
f_V	flag determining the fact's availability for watermark synchronization	f_C	flag identifying the availability of a fact's version as a carrier
c_V	flag storing the status of the carrier in case of being available	f_S	flag determining if the node is available for seed's generation
B_I	input block of a cell	B_O	output block of a cell
f_A	input/output cell's availability flag	b_V	original bit container
Θ	number of links of horizontal synchronization strings	θ	number of times a mark is moved in horizontal synchronization
M	majority voting matrix	R	hash value of a node's excluded items
C	hash value of a node's carriers	S	hash value of a node's synchronizers

Table 3 (continued)

Symbol	Description	Symbol	Description
μ	mean of an unwatermarked numerical distribution	μ'	mean of a watermarked numerical distribution
σ	standard deviation of an unwatermarked numerical distribution	σ'	standard deviation of a watermarked numerical distribution
M	absolute difference between the original and distorted means.	Σ	absolute difference between the original and distorted standard deviations
D_{KL}	Kullback–Leibler divergence	$\mathbb{X}$	probability space on which the original and watermarked numerical distributions are defined
$\mathcal{D}_i$	set of unwatermarked numerical values	$'\mathcal{D}_i$	set of watermarked numerical values
$P_{\mathcal{D}_i}$	discrete probability distribution of $\mathcal{D}_i$	$'P_{\mathcal{D}_i}$	discrete probability distribution of $'\mathcal{D}_i$

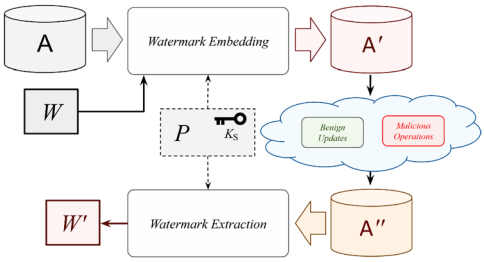


Fig. 3 General architecture of digital watermarking

rely on embedding different watermarks into each data copy, enabling traceability and the identification of traitor buyers (i.e., buyers who violate data acquisition contracts by selling or distributing their data copies without authorization).

Almost all robust techniques are distortion-based, although this is not a strict requirement. On the other hand, some distortion-based techniques restore the data to its original unwatermarked version after watermark extraction. They are classified as reversible and, while preserving data quality, are primarily intended for short-term protection, as the data revert to a vulnerable state once the watermark is removed.

2.2.2 Fragile watermarking

Fragile watermarking is commonly used for data tampering detection. This approach differs from robust techniques, as the watermark is intentionally designed to be compromised when violations of data integrity occur. Consequently, the the absence of a detectable watermark raises suspicion of data tampering.

For fragile techniques, the previously described architecture differs from that of robust approaches. While robust techniques rely on comparing W and W' for final assessment, fragile techniques frame watermark detection as a binary decision (i.e., whether the watermark is detected or not). Failure to detect the watermark is sufficient to conclude that the data are untrustworthy. On the contrary, if the watermark is still detected in A'', the data are considered authentic.

Figure 4 depicts the main architectural difference between robust and fragile watermarking techniques. The figure shows the watermark extraction process for fragile watermarking,

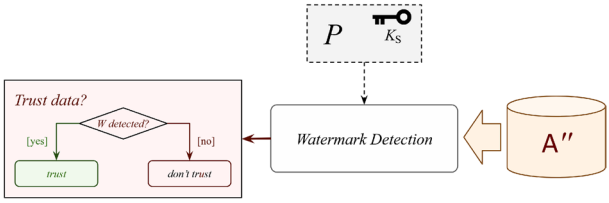


Fig. 4 Particularities of fragile watermarking detection

which is instead formulated as watermark detection. As previously mentioned, the detection output consists of a binary response to the question: *Is the watermark detected in* A''?

2.2.3 Properties and requirements

When developing a watermarking technique, a set of requirements must be considered. First, it is essential to address the trade-off among watermark imperceptibility, capacity, and robustness. Watermark imperceptibility requires that the presence of the watermark in the digital asset be unnoticeable and that data quality be preserved despite watermark embedding.

On the other hand, capacity refers to the amount of watermark information embedded in the data and the manner in which it is embedded. In this context, increasing the number of marks embedded in A makes it more difficult to compromise the detectability of *W*. However, indiscriminately increasing capacity may render the watermark perceptible and compromise data quality.

Finally, robustness refers to the persistence of the watermark in the data despite *benign updates* and *malicious operations*. Robustness can be enhanced by increasing capacity, which may, in turn, compromise imperceptibility [7].

Not all watermarking techniques are required to provide robustness. Fragile approaches used for data tampering detection are intentionally designed to forgo robustness [8].

Other requirements commonly considered in digital watermarking are the following:

- **Usability:** It is related to watermark imperceptibility and capacity. It states that modifications introduced by watermark embedding should not affect data quality [9].
- **Public system:** Watermarking techniques are typically designed under the assumption that synchronization mechanisms are public, and the security of the scheme relies solely on the secrecy and values of private parameters [10, 11].
- **Security:** It relies on the secrecy of private parameters, such as the secret key K_S. In addition to robustness, security is a critical factor to ensure the effectiveness of a watermarking technique [12].
- **Blind system:** A watermarking technique may be blind, semi-blind, or non-blind. In blind techniques, watermark detection is performed without requiring the original unwatermarked data, watermark information, or any other auxiliary content, except for private parameters such as the secret key. Semi-blind techniques require additional reference information, whereas non-blind approaches require full access to the original data and watermark information [10, 13].
- **Incremental watermarking:** It states that any content updated or inserted into already watermarked data must itself be marked if the algorithm and its parameters specify so.

This process cannot involve reprocessing the remaining watermarked data [14, 15].

- **False positiveness & false negativeness:** False positiveness refers to the probability of identifying valid marks in unwatermarked data, whereas false negativeness refers to the likelihood of missing valid marks during watermark detection. The values of these two metrics should not compromise the reliability of the detection outcome [11].
- **Non-interference:** It defines the non-overlapping condition when multiple watermarks are embedded in the same data. Embedding locations selected for embedding marks of different watermarks should not interfere with one another [16].
- **Detectability:** It addresses whether mark detection is deterministic or probabilistic. It also specifies whether detection is blind, semi-blind, or non-blind, as well as whether the scheme is public or private [5].

2.2.4 Data type constraints on watermarking

When implementing watermark synchronization, it is important to consider the structure of the digital assets storing the data and how they are reproduced or interpreted. In the case of EO data, multiple data structures are typically combined, with the most common being images, text, and relational databases. Many watermarking techniques have been created to protect each of these structures. However, due to their differences, they cannot be directly applied across different digital assets without additional considerations. Table 4 summarizes key features of the digital assets mentioned before.

Relational data are characterized by low redundancy, which reduces the cover for watermark embedding compared to images or other multimedia data types. By contrast, text data exhibits higher redundancy than relational data, manifested through repetitions in articles, prepositions, and other grammatical structures. However, this redundancy is largely constrained by the need to preserve semantic coherence. Overall text data exhibit lower redundancy than multimedia data. For instance, the higher redundancy present in images enables the concealment of data without compromising content usability, as evidenced by the correlation among pixel values in neighboring regions.

Furthermore, images are generally modified less frequently than relational data, which often undergo daily updates, thereby increasing the difficulty of maintaining watermark integrity in the protected content. In the case of text data, updates are also possible but constrained by contextual coherence and the limited ways in which phrases can be built. Under these circumstances, the watermark may also undergo natural degradation, similar to that observed in relational data, although to a lesser extent.

When dealing with multimedia and text data, it is possible to leverage the limitations of the human visual and auditory systems (HVS and HAS, respectively), as these data

Table 4 Differences between relational, multimedia, and text data

Criteria	Relational Data	Multimedia Data	Text Data
Data Redundancy	low	high	medium
Update Frequency	high	low	variable
HVS & HAS	not exploitable	exploitable	exploitable
Data Ordering	no fixed	fixed	tied (limited)
Data Type	multiple	single	single

types are intended for direct perception. Consequently, the constraints of these systems allow watermark embedding in regions that remain imperceptible even when distorted. By contrast, relational data are typically processed through a middle layer constructed using programming technologies. This layer is more restrictive, as it excludes alterations resulting from unauthorized data updates. For example, in an SQL query that relies on parameterized conditions, changes to data values caused by watermark embedding may prevent records from matching query parameters, thereby excluding them from the query results and affecting the generated information.

Images and other multimedia data types store their content in fixed positions, facilitating the design of sequential watermark synchronization techniques. By contrast, relational data lack a fixed ordering, rendering sequential methods vulnerable to malicious operations such as data reordering (see Section 2.2.6). In the case of text data, ordering is partially constrained. While some elements within a sentence can be rearranged without altering its meaning, the possibilities for rearrangement are constrained by the surrounding grammatical structures.

Finally, multimedia and text data are represented using unique data types (streams of bits and text, respectively), which simplifies watermark synchronization but also imposes certain limitations. By contrast, relational data can incorporate various data types, thereby expanding watermarking capabilities to encompass multiple cover types and increasing capacity while enabling distortion o be managed through diverse mechanisms.

2.2.5 Virtual primary keys

Relational data refer to the data stored in relational databases defined according to the relational model [17]. A relational database comprises a set of relations, each representing a business entity and modeled as a table, where columns represent the entity's attributes and rows correspond to stored data records (a.k.a tuples). Each tuple is identified by one or more attributes storing unique values, collectively referred to as the relation's primary key. Primary keys are used to define links among relations, thereby enforcing referential integrity [17].

Owing to their unique values, primary keys are useful for watermark synchronization, as they increase the likelihood of selecting different marks and embedding locations each time. Nevertheless, when relations are distributed independently from the rest of the database, primary keys lose relevance and may become a weakness for primary key-dependent watermarking techniques. In such cases, attackers can delete or update the primary key, thereby compromising watermark detection. Consequently, proving data ownership becomes infeasible, even if the watermark remains embedded in the data.

The virtual primary key (VPK) is introduced to avoid the dependency of watermarking scheme on primary keyss. They are generated using the rest of the attributes of the relation; therefore, they often exhibit duplicate values (called *duplicate problem*), and their values may change when some attributes are updated or deleted (called *deletion problem*). Therefore, while the use of VPKs mitigates the limitations associated with primary key dependency, it also introduces additional challenges.

Virtual primary keys are generated using VPK schemes. Although some schemes enable the generation of VPK sets with higher quality than others, all existing approaches remain

vulnerable to the *duplicate* and the *deletion problems*. Several widely used VPK schemes are presented in [18–23].

The VPK concept can be extended beyond relational databases to create virtual keys that allow the generation of seeds for watermark synchronization. In this work, such virtual keys are very valuable, regardless of the type of structure used to store normalized EO data.

2.2.6 Adversary model

The primary roles of the digital watermarking adversary model are the data owner and the attacker. While the data owner tries to protect intellectual property, the attacker attempts to steal or tamper with it. Accordingly, the attacker's main goal is to compromise watermark detection (in the case of robust techniques) or tamper with the data without raising suspicion (in the case of fragile techniques).

The malicious operations performed by an attacker depend on the technique goals and the type of carriers used to store the watermark. A carrier is the reference structure within a digital asset that stores the marks (e.g., in relational data, a carrier may correspond to an attribute of a given relation).

Robust watermarking techniques are subject to a variety of attacks, many of which involve data updates through set operations (e.g., subset deletion, update, or insertion) or modifications at different levels (e.g., bit, value, tuple, or attribute). Such operations range from modifying individual bits to transforming values using different units of measurement (e.g., recording temperatures in Fahrenheit instead of Celsius) or truncating decimal values [5]. Another category of attacks targets fingerprinted data and is commonly known as collusion attacks. These attacks combine different copies of the same data to compromise fingerprint identification. The simplest form of collusion attack is known as the *mix and match* strategy. More advanced collusion attacks operate at the bit level, requiring higher computational resources but enabling more precise manipulation to compromise stored marks. This class of collusion is typically referred to as the *majority attack* [24, 25].

Malicious operations may also involve higher-level logic beyond simple updates or value modifications. These include false ownership claims carried out through *additive attacks* or *invertibility attacks*. *Additive attacks* consist of embedding additional watermarks into already watermarked data, thereby creating doubts about the true data owner. *Invertibility attacks* aim to identify fake watermarks by detecting regular bit patterns present in the data [16]. Achieving robustness alone is insufficient to counter such malicious operations. A watermarking technique must provide guarantees beyond watermark persistence under data updates. In particular, it is crucial to implement strong security measures that , like robustness, are aligned with the challenges faced and the goals of the technique.

Other attacks depend on the structure used to store the data and how watermark synchronization is performed by the technique. A particular case is the *subset reverse order attack*, which is applied to database relations by changing the order of tuples and attributes, exploiting the fact that relational structures do not enforce a fixed ordering of data. For techniques that rely on sequential embedding, such attacks compromise watermark detection. Another malicious operation is the brute force attack, in which the attacker attempts to overwrite or extract the watermark by guessing the values of private parameters [26].

In the case of images, attacks can be broadly categorized into signal-processing and geometric ones. Signal-processing attacks utilize benign operations such as compression,

filtering, printing, and scanning, or more malicious techniques, such as noise addition, to compromise watermark detection. Geometric attacks are further divided into global and local ones. Global geometric attacks primarily involve rotation, scaling, and translation, whereas local geometric attacks target specific areas within an image through operations such as random bending transformations, cropping, and row/column deletion. More recently, a new class of malicious operations referred to as deep attacks, has emerged. These attacks leverage deep learning techniques from the fields of computer vision and image processing to undermine watermark detection [27].

Despite the challenges posed by malicious operations, it is important to consider benign updates that may degrade watermark quality when implementing a watermarking technique. While editing operations are less frequent for images or text than for relational data, remote sensing workflows often involve generating new content through updates and modifications, which may compromise data embedded in the original source.

2.3 Related work

The use of advanced programming systems has proven effective for remote sensing applications, either when focusing on increasing information retrieval effectiveness, such as in [28], where the authors proposed an approach based on ontologies, or when focusing on improving the accuracy of generated information, such as in [29], where the authors proposed a framework for increasing weather forecasting accuracy. In particular, watermarking techniques have proven successful for ownership protection and tampering detection, among other applications, in different digital assets such as multimedia, relational data, compiled code, and documents. Nevertheless, their application to EO data, by exploiting the relationships among the different data types used to represent EO information, remains very limited.

Despite their success, image watermarking techniques are unfit for safeguarding images containing EO data due to differences in the frequency of updates, structure, and processing between these digital assets. EO images are typically associated with files containing essential processing data, making any distortion introduced by watermark embedding potentially compromise the integrity of these links. Furthermore, pixels in EO images encode values beyond standard image representation, corresponding to the intensity of radiation reflected by the observed objects within the wavelength ranges to which the sensors are sensitive. While colors in satellite images are often assigned to display content in a visually friendly manner to the HVS, they may not always represent the true colors of the objects displayed [30].

2.3.1 Integrity and authenticity control

As previously mentioned, fragile techniques are used for integrity and authenticity validation. However, techniques that do not introduce distortion during watermark synchronization, also defined as zero-watermarking, are particularly suitable for these objectives. Recently, the combination of zero-watermarking with blockchain technologies has enabled researchers to achieve strong results in verifying and preserving the quality of digital assets.

Generally, watermarking techniques associated with EO data target satellite imagery. Some approaches, such as those proposed in [31–34], are designed for tampering detection, making them fragile and unsuitable for ownership or provenance protection.

In the techniques proposed in [31, 32], the authors introduce a semi-fragile approach for tampering detection in multi-band images (e.g., multi and hyperspectral images). They used tree-structured vector quantization to consolidate pixel signature processing, thereby eliminating the need to handle each band separately. Conversely, in [33], the authors focus on the ciphertext data of remote sensing images and address the challenge of achieving a substantial watermark capacity, enabling broader application of the technique. This research direction is particularly relevant given the critical importance of securely transmitting and linking EO content with ciphertext data. In [34], the authors proposed a blind and fragile watermarking approach based on regions of interest in satellite images. This method enables both integrity verification of the watermarked image and localization of intentional or unintentional tampering, thereby supporting secure transmission of this type of digital asset.

Several blockchain-based approaches have been presented in [35–38]. In particular, the method proposed in [35] aims to eliminate dependence on trusted third parties (TTP). It leverages the multi-band features of satellite images to generate a zero-watermark, which is then stored on a blockchain by a zero-watermark registration system. In [36], the authors propose a technique based on a stacked denoising autoencoder (SDAE) to extract deep and robust features from local square feature regions for watermark construction.

In [37], the authors focus on integrity authentication by combining blockchain with perceptual hashing, leveraging its potential as a non-destructive data integrity protection technique. This approach also exploits the multi-band features of remote sensing images. Finally, in [38], the authors introduce an approach for spatial index verification. Their method linearizes the spatial information of remote sensing images and integrates it with log-structured merge (LSM) trees to enable efficient image retrieval and verification.

2.3.2 Ownership and provenance protection

Most watermarking techniques developed for protecting EO data are classified as robust and primarily focus on safeguarding copyright and provenance. In [39, 40], the authors introduce approaches designed to protect EO digital assets beyond conventional image-based methods. For instance, in [39], the authors propose a scheme known as TrajGuard, which aims to re-identify GPS trajectories to prevent the dissemination of tampered data. Such tampering could lead to misinformation and misallocation of resources in road safety monitoring, traffic management, and various mobility services. Similarly, in [40], the authors introduce a scheme called W-trace, which focuses on identifying the provenance of GPS trajectories to prevent unauthorized parties from impersonating providers. While these schemes effectively protect EO data, they are constrained to GPS trajectories and remain susceptible to attacks that alter GPS points, commonly referred to as trajectory resampling [41].

Another example is the work presented in [42], in which the authors combine the lifting wavelet transform [43] with a partial Hadamard matrix [44], utilizing a ternary sequence as a watermark to increase the watermark capacity. This approach aims to enhance reconstruction accuracy when adopting compressive sensing [45]. Similarly, in [46], the authors propose a robust, non-blind watermarking technique based on the quaternion wavelet transform [47] and tensor decomposition [48].

In [49], the authors propose a technique for copyright protection of satellite images that applies image degradation and restoration methods for watermark embedding and extraction. Similarly, in [50, 51], watermark synchronization is performed on remote-sensing images using the discrete wavelet transform. In [52], the authors introduce a real-time, cheating-immune secret-sharing approach for the secure transmission of remote sensing images. In [53], the authors propose a robust scheme for protecting landslide image transmission, in which watermark embedding is carried out in the wavelet domain. This approach focuses on a specific type of digital asset and targets landslides as the primary objects of interest.

Zero-watermarking and reversible approaches have also been developed for ownership and copyright protection. For example, in [54], the authors focus on protecting remote sensing images and perform watermark synchronization using the Discrete Cosine Transform (DCT) and the Discrete Fourier Transform (DFT). Another group of techniques is described in [55–58], where the authors propose several schemes for copyright protection of vector maps and vector geographical databases. In [59], the authors present a watermarking framework designed specifically for electromagnetic systems, particularly for applications such as radar detection. They introduce a dual-function radar and communication (DFRC) waveform method that exploits the peak sidelobe level (PSL) of the autocorrelation function (ACF). Information embedding is achieved via the DFT, ensuring robustness against signal distortions and noise introduced during transmission.

Furthermore, in [60], the authors propose a granular content distribution scheme to preserve the privacy of remote sensing data obtained from Internet of Things (IoT) devices. Their scheme emphasizes computational efficiency, addressing resource-intensive challenges commonly encountered in remote sensing applications. By distributing content at a granular level, the approach aims to minimize the risk of data breaches while optimizing resource utilization in IoT environments. Additionally, in [61], the authors introduce a frequency-domain watermarking technique tailored for protecting vector geographic data. This technique enhances the randomness of embedding positions in the frequency domain, thereby improving the watermark robustness against various attacks, including geometric transformations, cropping, simplification, and coordinate point editing.

Finally, within the category of relational data protection approaches, in [62], the authors propose a distortion-free watermarking scheme that provides high capacity while preserving data quality. Also, in [63], the authors presente a lossless semi-fragile watermarking scheme based on prediction-error expansion, which enables differentiation between legal and malicious modifications.

2.3.3 Research gap to be addressed

Advances in the related work contribute significantly to the field of data security, offering important solutions for safeguarding sensitive information in electromagnetic systems, remote sensing applications, and geographic data protection scenarios. Nevertheless, existing techniques generally focus on specific data types, such as images, and do not always analyze how watermark embedding affects the outcomes of remote sensing algorithms. Although robustness is often achieved, demonstrating that such schemes can be applied without affecting the practical scenarios for which the data are generated remains challenging.

Most watermarking techniques developed for EO data protection focus on images and overlook other data structures used to store content generated and managed by remote

sensing processes. Furthermore, they are typically designed for ownership protection and tampering detection, often neglecting provenance identification. By considering multiple data structures and their interconnections, together with dimensions such as regions of interest and observation periods, it is possible to increase watermark capacity while enhancing theability to protect digital assets.

Beyond watermarking-based approaches, most existing EO provenance solutions rely on external mechanisms such as metadata annotations, audit logs, or blockchain-backed registries. While effective in controlled environments, these approaches depend on trusted infrastructures and can lose their binding to the data when content is transformed, partially disseminated, or detached from its original repository. In contrast, the proposed framework embeds provenance evidence directly within the EO data through self-recalibrated watermark synchronization, allowing provenance verification to persist across benign updates, redistribution, and heterogeneous data representations.

Beyond watermarking, data provenance is commonly represented using structured metadata models and lineage graphs, often following general standards such as W3C PROV [64] and domain-specific lineage elements (e.g., ISO LI_Lineage) adopted in Earth observation data management systems [65]. These approaches are effective when provenance metadata and workflow logs remain attached to the dataset throughout processing and distribution, including database/pipeline-level extraction from query event logs as in modern governance systems [66]. A distinct class of solutions uses blockchain-based infrastructures to provide tamper-evident provenance records in distributed EO environments [65]. However, all these approaches bind provenance evidence externally, via metadata records, logs, or ledgers. In contrast, our scheme binds provenance evidence internally to EO data representations through watermark synchronization and self-recalibration, allowing provenance validation to persist even when data are redistributed or subjected to bounded benign modifications. A comparative summary of these approaches and their respective assumptions is provided in Table 5, highlighting the complementary role of embedded watermarking relative to existing provenance protection mechanisms.

Quantitative comparisons with non-embedded provenance approaches (e.g., provenance metadata models such as W3C PROV, log/workflow provenance extraction, or blockchain-backed registries) are not directly applicable, since these mechanisms record provenance externally and do not operate on embedded carriers or introduce measurable data distortion in the EO content. For this reason, we provide a qualitative comparison of assumptions and complementary strengths in Table 5. In contrast, embedded watermarking schemes can be compared on distortion and robustness metrics, but they typically target a single data type and do not address multi-type repositories with self-recalibrated horizontal synchronization, which is the focus of this work.

3 Feature-based EO watermarking

The watermarking scheme proposed in this work focuses on safeguarding the provenance of EO data. Our approach takes into account the different structures used to store these digital assets and therefore requires support for various data types, such as images, relational data, vectors, and text. To this end, it is necessary to design an engine capable of handling different content and establishing meaningful connections among them.

Table 5 Comparison of representative provenance protection approaches in Earth observation and data-intensive systems

Approach / Work	Primary mechanism	Where provenance resides	Robust to data modification?	Main strengths	Limitations w.r.t. EO data provenance
W3C PROV model [64]	Standardized provenance metadata and relations	External metadata documents	No	Interoperable, expressive provenance representation across systems	Provenance can be detached from data; does not enforce data-level binding or survive redistribution
Blockchain-based EO provenance [65]	Distributed ledger with off-chain EO data references	External ledger + metadata	Partially	Tamper-evident records; suitable for distributed governance	Provenance depends on continued linkage to the ledger; indirect binding to EO data
General blockchain traceability frameworks [67]	Blockchain + smart contracts (optional ZK proofs)	External ledger	Partially	Auditability and access control in decentralized settings	System-dependent; provenance not intrinsically bound to data content
Pipeline and database provenance extraction [66]	Query logs and workflow event tracking	Platform-level governance layer	No	Captures detailed derivation and processing history	Effective only within managed pipelines; provenance lost outside the platform
Training data provenance verification for ML [68]	Statistical verification of data influence	External verification procedure	Not applicable	Addresses ownership and attribution of training data	Orthogonal problem; not designed for EO dataset provenance
Vector geographic data watermarking [69]	Embedded watermark with reversibility or authentication	Inside the data	Yes (bounded)	Data-level binding; supports integrity and authentication	Typically single data type; not designed for multi-type EO repositories
This work	Embedded watermark with vertical and horizontal synchronization	Inside the data (repository carriers)	Yes (bounded + recalibration)	Provenance bound to EO data; supports heterogeneous data types and redistribution	Requires normalization and meaningful fact versions; depends on distortion-tolerant carriers

Our proposal provides strong resilience against attacks aimed at removing watermarks or compromising their synchronization, considering the threats that provenance preservation encounters. It is classified as robust, and its architecture incorporates features that enhance both applicability and completeness.

3.1 EO data guardianship: Understanding ownership, authorship, and provenance

This section first defines the term "watermarking technique" for protecting EO data provenance. It then highlights the differences between this concept and others commonly utilized in the context of robust watermarking. Accordingly, the section formally defines authorship, ownership, and provenance.

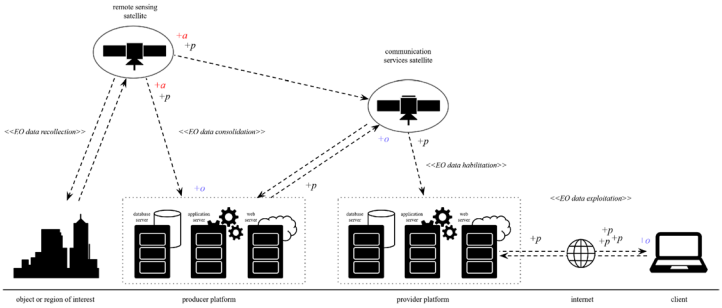


Fig. 5 Life cycle of EO data

Figure 5 provides an overview of the stages involved in EO data generation and handling. It depicts the stages at which signal traces can be introduced to identify provenance ($+p$), authorship ($+a$), and ownership ($+o$). In general, remote sensing satellite sensors target specific objects or regions of interest. The manner in which content is captured and transmitted depends on the sensor type and the remote sensing principles applied (i.e., , passive vs. active sensing, etc...). Each remote sensing satellite is associated with a producer platform that hosts the servers used to process, store, and provide the content[8]. EO data producers may also rely on conventional communication satellites for data transmission.

Providers access EO data via communication satellites and process the content through their platforms. Clients can access EO data providers through the Internet[9]. Furthermore, s client may generate EO data products from data previously obtained from providers and make them publicly available on the Internet for use by third parties and other clients.

We analyze the challenges faced by EO data products by focusing on EO data in general, rather than on a specific type of digital asset used for a given EO content representation. The primary focus of this work is provenance protection, which is more central to our study than ownership and authorship. The following sections describe each of these concepts in detail.

3.1.1 Provenance

Provenance refers to information that identifies the origin of data and its lineage. It plays a crucial role in tracing the path EO data packages take from the source sensor to their final provider, or to an intermediate provider prior to acquisition by a specific client.

EO data is typically accompanied by metadata detailing the sensors used for content production and additional information required by remote sensing algorithms to process it. However, this metadata is often unprotected and vulnerable to tampering. Moreover, there is a lack of standard procedures and additional information to document the lineage data may undergo as it passes through various sensors, storage devices, and providers. Consequently, ascertaining the reliability of provenance information associated with the data presents a significant challenge.

[8] Here, the term remote sensing satellites refers to all types of satellites used for remote sensing purposes, including relay, imaging, and electric reconnaissance satellites.

[9] The use of the Internet includes connections to interconnected computing systems established through various means, including communication satellites.

In this work, provenance is addressed from a technical perspective, focusing on data-level traceability, integrity, and origin verification. Legal, institutional, and standardization frameworks are complementary to this approach and can leverage the proposed mechanism but are outside the scope of the present study.

3.1.2 Authorship

Authorship pertains to the rights of content creators, in this case, in the context of EO data. In terms of the data lifecycle, authorship has a narrower scope than provenance, as it is established during the initial data generation phase, but remains relevant throughout subsequent phases. However, authorship faces vulnerabilities similar to provenance, as EO data authorship is similarly linked or contained. Therefore, the readily accessible nature of authorship information complicates efforts to prevent its overwriting, tampering, or elimination.

3.1.3 Ownership or copyright

On the other hand, ownership (a.k.a copyright) is intended to protect intellectual property by focusing on the rights of content owners rather than creators (although the owner and the creator roles may sometimes overlap). Copyright typically relies on the good faith of the data possessor and serves as a mechanism to prevent unauthorized use of content for personal gain. It can be enforced through tools that provide additional evidence of the true content owner. Accordingly, users found to be in violation of copyright regulations may be subject to penalties proportional to the nature of the infringement.

Digital watermarking is a widely recognized approach for implementing ownership protection mechanisms. However, due to the high complexity and volume of EO data and the diverse phases and providers involved, it is challenging to develop a system capable of harmonizing copyright management across the varied policies established by each provider.

3.2 Varying features of interest

Our watermarking scheme follows the architecture presented in Section 2.2.1. The main principle of our approach is to embed a robust watermark, taking advantage of the cover that using different types of digital assets may offer for representing EO content and the common dimensions that exist when linking content from different providers or captured with different devices or sensors. The proposed scheme manages a watermark that encodes provenance information and is built throughout the life cycle of the EO data. We xploit the ability to derive different feature representations from the same data to select the cover dynamically, varying the elements used to embed the marks. As a result, the induced distortion does not interfere with the outcome of remote sensing algorithms.

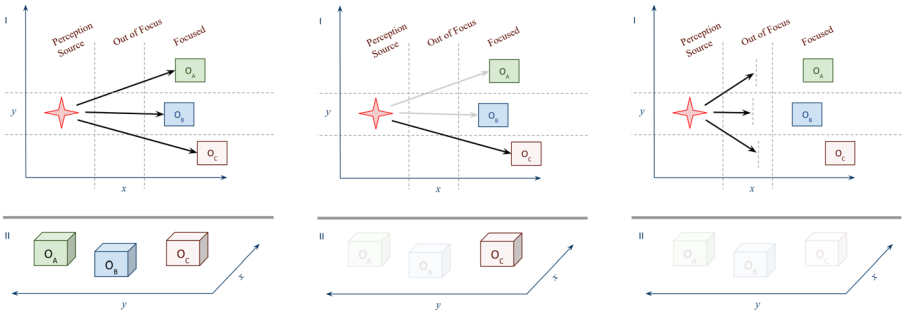
The main idea is based on different data packages (e.g., pictures) of the same region containing information from various objects. Although all objects are represented in each package, some are better focused than others, depending on their relevance to the analysis. Under this premise, at least momentarily, the remaining content of the package is temporarily disregarded. Figure 6 illustrates this case.

(a) Focusing the withered flower.

(b) Focusing the closets yellow flower.

(c) Focusing the background.

Fig. 6 Picture of the same region of interest, focusing on different objects. The item the process focused on is contained inside the red polygon



(a) Focalization of all elements ($\alpha = 3$).

(b) Focalization of one element ($\alpha = 1$).

(c) Out of focus perception ($\alpha = 0$).

Fig. 7 Principle of selection of carriers, simulating classification through localization of different objects ($\varkappa \leq 3$)

Formalizing the previous idea, having all attributes from a data set containing EO information, the focus varies for those representing the main interest, either for the current application or the watermark synchronization. Notice that the interest is inverse, considering that those used for watermarking are of interest for the embedding but should be less relevant for the remote sensing application (and vice-versa).

Figure 7 illustrates cases in which the number of observed objects (denoted by $\varkappa$) remains constant while the focus varies. The number of targeted objects may change even after focusing on a region containing different elements. Therefore, two variables are required to manage this feature. In the figure, a black arrow indicates the objects in focus, while a light-gray arrow indicates those out of focus. The number of objects focused is denoted by α. Figure 7a presents the case in which all elements are in focus, formally described by $\alpha = 3$. Figure 7b shows the case $\alpha = 1$, in which O_C is the only object observed in detail. Finally, Fig. 7c shows the case in which all objects are out of focus, described by $\alpha = 0$.

Formally, the number of focused objects must be less than or equal to the number of observed objects (i.e., $\alpha \leq \varkappa$). The criteria for selecting an object may vary , including its relevance to the remote sensing algorithm and its distortion tolerance for the watermarking scheme. Thus, our scheme must support registering different criteria for watermark synchronization. Another key issue is implementing a mechanism that enables dynamic refocusing of observed objects so that the set of objects involved in the watermark synchronization can be redefined at any time in response to changes in observation interest.

3.3 EO data integration

To change the focus on objects, it is first important to register all data for the observed regions, adding details of all objects, so that they can later be identified according to functional requirements. Given the use of different EO data sources and common values such as region of interest or space and observation time, we consider a multidimensional structure as the main data repository.

3.3.1 Multidimensional repository module

The module managing the multidimensional repository consists of the joiner, the EO data systemizer, and the repository itself. First, the joiner receives content from different providers. It then passes control to the EO data systemizer. The systemizer proceeds with data normalization whenever possible to define the links among normalized values. Finally, the facts are corrected. After the new data are consolidated, they are committed to the repository for subsequent management through watermarking processes.

Note that all types of digital assets involved in EO content representation can be stored in the repository after appropriate normalization, linking, and fact correction. Nevertheless, when needed, the scheme must guarantee reconstruction of the original data items without loss of quality despite watermarking processes. Accordingly, it must not alter the outcome of remote sensing algorithms or any other downstream applications.

Figure 8 depicts an integrated view of the multidimensional repository module architecture. The joiner considers common values defined as dimensions such as space and time. A detailed description of the data integration engine is provided in the following section.

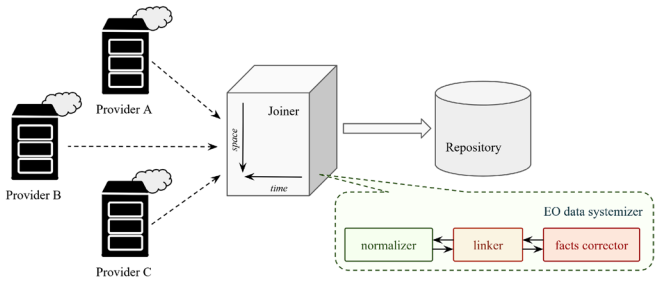


Fig. 8 Data integration engine

3.3.2 Management of dimension and facts

The EO repository consists of a multidimensional structure (a.k.a data warehouse or cube) composed of dimensions and facts. It is not meant to support data updates but rather to provide a space that can be used to protect EO data provenance. This is an important feature that contributes to the performance of our proposal and is described in the following sections.

The repository dimensions are values intersected to register other values of particular interest (i.e., the facts). Common dimensions for EO data are the region of interest and the observation time. Figure 9 presents an example of the same region of interest (north of Lake Mead, USA) observed during the month of January of four different years. The figure clearly shows variations in the lake's water level and drought conditions.

In our work, we integrate EO data based on the premise that data obtained from different sources can be linked through common dimension values, as in [2]. The Joiner takes each composed data (defined as CD) and uses the EO data systemizer to transform it into normalized data (denoted by ND). Dimensions and facts are then derived from ND. Dimensions are managed by the *DimMgr* module, which inserts them into the repository as long as they are not yet stored. We denote dimensions as $D_i : i \in [0, |\mathrm{D}| - 1]$, where D is the set of dimensions detected by the *Splitter* (see Fig. 10).

After processing the dimensions in ND, the engine analyzes the facts using the *FactsMgr* module. Facts are denoted as $F_j : j \in [0, |\mathrm{F}| - 1]$, where F is the set of facts. Since facts are linked to dimensions, the engine must verify whether each fact and its associated dimensions are already present in the repository. The engine uses the function GET( ) to check facts and dimensions stored in the repository. The function also takes a second parameter consisting of a threshold (denoted by δ) applied to facts.

The value of δ is used to seek the same values possible while considering different ones as long as they are inside the given threshold. By definition, the underlying fact is considered the same, while value differences allow the embedding of other instances. Different values of the fact are used to check its authenticity, perform correction when needed, and extend the fact's potential for generating external calibration values and embedding more than one mark in it while making possible self-correction of mark embedding. Under this criterion, multiple values may correspond to the same fact, being formally defined as fact versions.

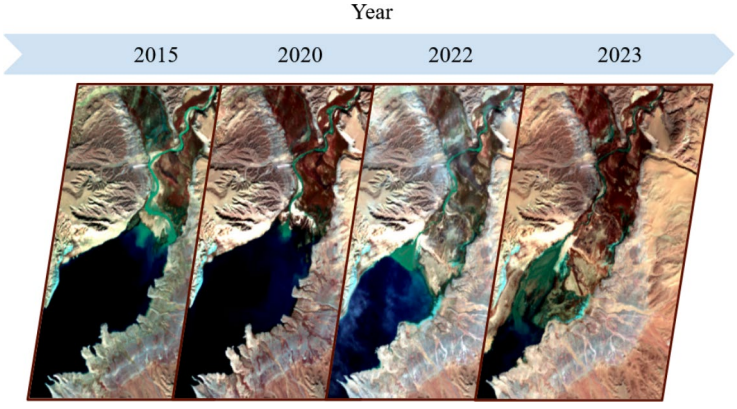


Fig. 9 Example of time and space dimensions. Analysis of a region of interest using remote sensing temporal resolution [2]

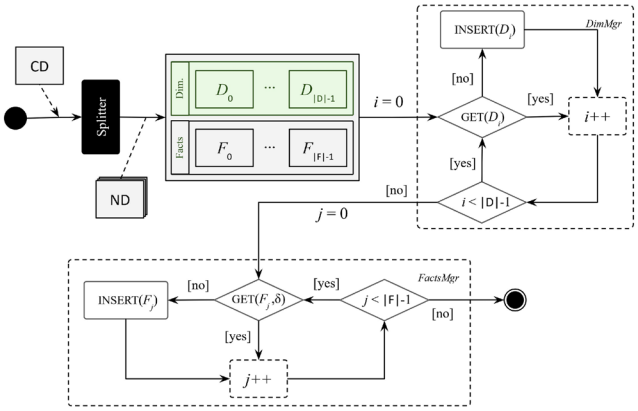


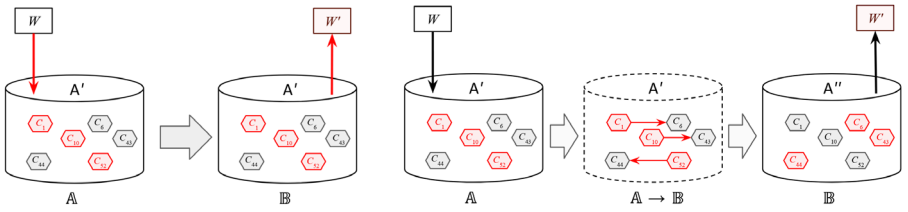
Fig. 10 Managing dimensions and facts (content linking and correction) [2]

3.4 Transversal and multidimensional watermarking

Compared to traditional watermarking, the repository's multidimensional structure makes it possible to boost watermark capacity and increase the amount of content available for synchronization. After consolidating the data in the repository, the watermarking scheme can exploit both dimensions and facts. Furthermore, marks can be moved across facts and their versions, enriching watermarking processes by enabling a transversal direction in addition to the vertical one used in traditional schemes. The multidimensional organization of the repository is a structural prerequisite for the proposed watermarking scheme. By organizing heterogeneous EO data through shared dimensions such as space and time, it enables watermark carriers to be selected and relocated across different facts and fact versions independently of their underlying data types. This capability is essential for the transversal (horizontal) synchronization mechanism, which would not be feasible in a flat or single-type data model. Carrier relocation during horizontal synchronization is driven by key-dependent pseudo-random selection based on virtual primary keys and cryptographic hash functions. As a result, relocation patterns are computationally unpredictable to adversaries without knowledge of the secret key and are not externally observable through the exposed EO data.

Figure 11 highlights the differences between traditional watermark synchronization, where the marks are embedded and extracted, and the transversal one proposed in this work. Stage $\mathbb{A}$ of the digital asset identifies the carriers selected after watermark embedding (formally identified as *first carriers*). On the other hand, stage $\mathbb{B}$ identifies the carriers right before watermark extraction. The transversal synchronization is characterized by the movement of marks across carriers after the watermark is embedded. This approach must not interfere with the basic embedding and extraction processes. In Fig. 11, the movements of interest are highlighted with red arrows, and a set of generic carriers (C_1, C_6, C_{10}, C_{43}, C_{44}, and C_{52}) is illustrated inside the digital asset A. The carriers containing the marks are highlighted in red. The figure denotes a transitional stage of the digital asset from stage $\mathbb{A}$ to stage $\mathbb{B}$, in which some marks move to different carriers.

The vertical synchronization is characterized by $\mathbb{A} \equiv \mathbb{B}$, whereas horizontal synchronization is characterized by $\mathbb{A} \not\equiv \mathbb{B}$. The symbols $\equiv$ and $\not\equiv$ refer to the equivalent



(a) Vertical synchronization (traditional approach). (b) Horizontal synchronization (transversal approach).

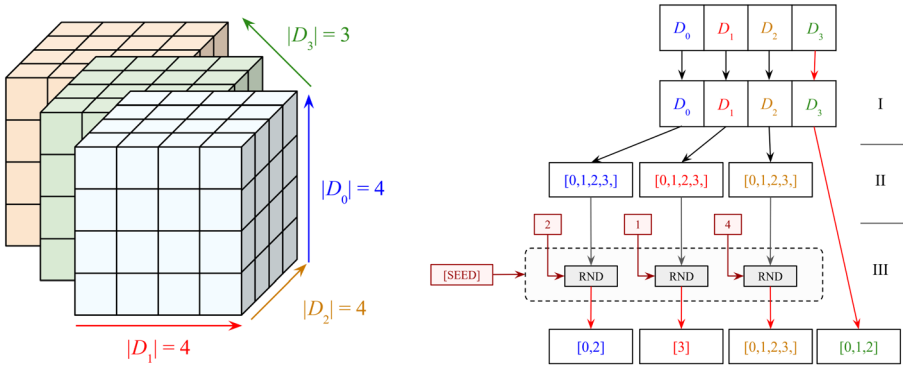
Fig. 11 Directions involved in watermark synchronization (red arrows highlight the process of interest for the synchronization featured on each figure

and not equivalent relationship, respectively. The relationship between stages is expressed in terms of equivalency, considering that the content in the digital asset may vary, and the comparison is only in terms of elements selected as carriers. Furthermore, transversal watermarking involves moving marks through the digital asset's structure, not necessarily during watermark embedding or extraction.

3.4.1 Multidimensional synchronization engine

The repository's multidimensional structure increases synchronization complexity and contributes to higher watermark capacity. Multidimensional databases enable synchronization over multiple dimensions. Nevertheless, once the set of dimensions is defined, it typically remains fixed. Therefore, we combine multidimensional databases with NoSQL technology to provide flexibility in the number of dimensions. NoSQL databases enable horizontal scalability; thus, if newly ingested data requires additional dimensions, our scheme makes it possible.

Figure 12a shows an example of a multidimensional database composed of four dimensions (D_0, D_1, D_2, and D_3), each highlighted with a different color. Using this structure, Fig. 12b illustrates how vertical synchronization is carried out. The dimensions



(a) Example of multidimensional structure (four dimensions). (b) Dimension values selection for facts mapping in the watermark embedding.

Fig. 12 Example of multidimensional structure and dimensions reduction and combination

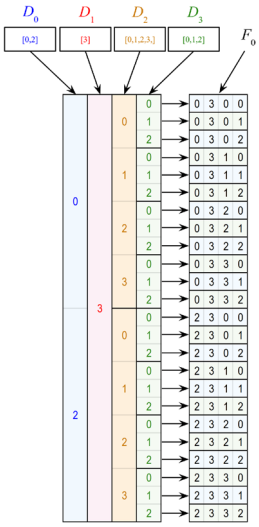


Fig. 13 Map of facts entry generated from the example of Fig. 12a

can be reduced by pseudo-randomly selecting some values per set, whereas all values can be considered for others. In the figure, all values of D_3 are selected, whereas the remaining dimensions are reduced via subset selection. A red arrow identifies the final values. The values for the remaining dimensions are pseudo-randomly selected using the function RND(SEED,D,κ), which takes as input a seed, the set of values of dimension D, and a constant κ indicating the number of elements to consider from the set.

Once the sets of selected dimensions are reduced, all values are combined to obtain the facts considered carriers. The combination of all dimensions identifies the facts' position. Their entry is depicted in Fig. 13. Once the fact entry is selected, multiple versions per fact can be considered.

The first operation obtains the dimensions to consider from the digital asset. This is performed by the process `dim_def( )` depicted in Algorithm 1. It takes as input the secret cryptography key K_S, the digital asset given by the repository A, the filter $\mathcal{X}$ used to reduce the dimensions, and an array of integers N indicating the number of elements per dimension to consider. The process uses the function $\Gamma(\cdot)$ to generate the set D containing all dimensions from the dataset (see line 3 of Algorithm 1). Then, the function $\Lambda(\cdot)$ applies the filter $\mathcal{X}$ to reduce the values considered for each dimension. It takes as input N and a set of seeds previously generated using a VPK scheme, implementing process III of Fig. 12b. As a result, the subset D is obtained and returned by the process (see lines 4 and 5).

Algorithm 1 Selection of dimensions using dim_def( ) procedure.

```
1 Input: K_S, A, X, N
2 Output: D
3 D ← Γ(A)                          // extraction of dimensions
4 SEEDS ← VPK(K_S, A)               // generation of seeds set
5 D ← Λ(D, X, N, SEEDS)             // dimensions filtering
```

Figure 14 depicts the structure of D, a matrix containing all combinations of the dimensions considered after the filtering. Each linear array containing the combinations of all dif-

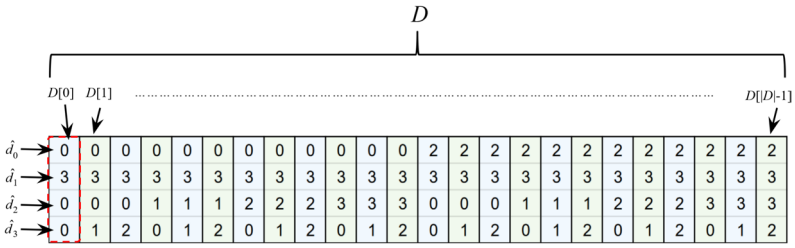


Fig. 14 Matrix representing the combination of selected dimension's values

ferent dimensions is identified as $D[i] : 0 \leq i \leq (|D| - 1)$, and each element of a particular array $D[i]$ is denoted as $\hat{d}_j : 0 \leq j \leq (|D[i]| - 1)$. It can be seen as a rotation of the matrix previously obtained in Fig. 13.

Algorithm 2 presents a detailed description of the vertical synchronization embedding. The process requires the secret cryptography key K_S, the digital asset A, the watermark W containing provenance information, the number of most significant bits (*msb*) considered for each analyzed fact β, the number of less significant bits(*lsb*) available for mark embedding ξ, the filter $\mathcal{X}$, the array of integers N, and the fact version fraction φ, which allows controlling the number of versions considered for each fact entry. The algorithm first executes `dim_def( )` to obtain D (see line 2) and then combines the filtered dimensions to obtain the positions of the facts used as carriers.

For each array $D[i]$, the set d is computed using all its items $\hat{d}$ (see lines 3 to 6). Whenever a new array $D[i]$ is analyzed, d is set to null to avoid combining elements from different arrays (see line 4). After obtaining d, all fact versions are extracted from the cell and stored in the set F using the function `get_facts( )` (see line 7). The versions of interest are extracted from F and stored in $\mathcal{F}$ using the function `facts_filter( )` (see line 8).

Algorithm 2 Definition of emb_vertical_sync( ) procedure.

1 **Input:** K_S, A, W, β, ξ, $\mathcal{X}$, N, φ
2 $D \leftarrow$ `dim_def`(K_S, A, $\mathcal{X}$, N)
3 **for** $i = 0$; $i <$ |D|; $i++$ **do**
4 $d \leftarrow$ `null`
5 **foreach** $\hat{d} \in D[i]$ **do**
6 d.`add`($\hat{d}$) // see Section 3.4.2
7 F $\leftarrow$ `get_facts`(A, d) // extraction of all fact's versions
8 $\mathcal{F} \leftarrow$ `facts_filter`(F, φ, β, d, 0) // see Algorithm 3
9 $K_C = H(\mathcal{F}, \beta, d)$ // cell's VPK generation
10 $f = H_f(K_C, \mathcal{F})$
11 $K_F = H(f, \beta, d)$ // fact's version VPK generation
12 $b_\beta = H_\beta(K_F, f, \beta)$ // selection of *msb* value
13 $p_\xi = H_\xi(K_F, \xi)$ // selection of *lsb* position
14 $p_{\overline{m}} = H_{\overline{m}}(K_F, w \leftarrow |W|)$ // meta-mark position's selection
15 $\overline{m} = W[p_{\overline{m}}]$ // meta-mark selection
16 $m = \overline{m} \oplus b_\beta$ // mark generation
17 $f.b_V \leftarrow [f]_2[p_\xi]$ // save original bit
18 $[f]_2[p_\xi] \leftarrow m$ // mark embedding
19 $f.c_V \leftarrow 1$ // flag busy carrier

After obtaining the allowed versions of the current fact, the VPK K_C is computed to identify the set $\mathcal{F}$ (see line 9). Next, in line 10, one version f is selected from $\mathcal{F}$ using the function H_f, which is built based on the pseudo-code $pos \leftarrow K_C \bmod |\mathcal{F}|$, where *pos* is the position of the selected version in $\mathcal{F}$. Then, the VPK of f is generated and stored in K_F using the one-way hash function $H(\cdot)$, which takes as input f, β, and d (see line 11). Then, one of the most significant bits from the fact and one less significant bit position are *pseudo-randomly* selected and stored into b_β and p_ξ, respectively, using the same principles as for function H_f (see lines 12 and 13). Next, a meta-mark is selected from the watermark and used to generate the mark by *XOR-ing* it with b_β (see lines 14 to 16). Finally, the *lsb* value being replaced is stored into $f.b_V$ and the mark is stored at position p_ξ, generating a new version of f (see lines 17 and 18). The current mark embedding concludes by flagging the carrier to guarantee its consideration by horizontal synchronization when required (see line 19).

3.4.2 Selection of dimensions and facts

The structure storing the facts is a cell of the multidimensional database shown in Fig. 12a, accessed by selecting a value per dimension and using it as index to point to facts. Figure 15 presents its structure in detail. Given its appearance in the higher-level representation (block level **I** in the figure), we defined it as the *fork* structure, which is used to store a fact and its versions. Furthermore, a node identifying a fact version contains a set of values of critical relevance to our scheme. In the figure, a set of D dimensions are combined to obtain the entry of fact F_0, which is linked to a set of fact versions (F_1, F_2, F_3, ..., $F_{|F|}$). The number of dimensions and fact versions can vary depending on the data stored in the repository.

In Fig. 15, a red rectangle identifies locked values (often available for seed generation). On the other hand, fragments that tolerate distortion and are useful for mark embedding are identified by a green rectangle. The block depiction level **I** represents the combination of one value per dimension used to address the cell, which contains the fact entry identified as F_0. Depiction level **II** includes the set of versions denoted by $F_j : j \in [1, |F|]$, where F is the set of versions for the current fact. F_0 is linked to each version, and all versions are connected to another node containing two hash values C and S. The value of C is generated using the selected carriers and their current status (i.e., busy, available, or unavailable). On the other hand, S is computed from the fragments used as seeds in each node. Each node structure is shown in depiction level **III**.

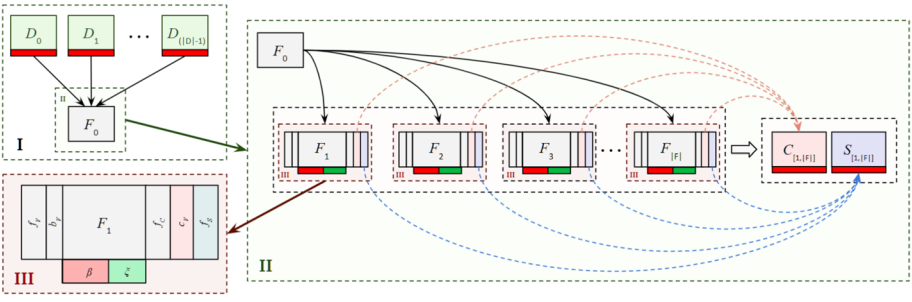


Fig. 15 *Fork* structure of a cell in the multidimensional repository utilized to obtain and store versions of facts

The node used to store a fact version is composed of the version value F_j (F_1 in the figure), fragments of locked values (in this case, β), and values available for mark embedding (in this case, ξ). The node also includes a flag denoting the availability of the version for watermark synchronization (denoted as f_V), a flag indicating whether the current version is available as a carrier (denoted as f_C), the attribute c_V, which stores the carrier status when the node is available, and the flag f_S, which specifies whether the node is available for seed generation. Another node attribute is b_V, which stores the original value of the bit selected to embed the mark. When the node is not selected for mark embedding, a random value can be assigned to b_V to increase chaos randomness and improve the scheme's robustness.

The function `set( )` in line 9 in Algorithm 2 is used to assign to *d* the values contained in the different dimensions. Thus, *d* stores a set of integers, which are used to intersect the values of dimensions pointing to a particular fact. For example, in Fig. 13, the first value of *d* is $\{\hat{d}_0 = 0, \hat{d}_1 = 3, \hat{d}_2 = 0, \hat{d}_3 = 0\}$. After selecting the dimension indeces, a cell is located, and the *fork* structure is used to select one or more versions for each fact and to use them as carriers. The function `facts_filter( )`, previously employed in line 11 of Algorithm 2, implements this task. Algorithm 3 presents details of its construction.

Algorithm 3 Definition of facts_filter( ) procedure.

```
1  Input: F, φ, β, d, ϵ
2  Output: F
3  foreach f ∈ F do
4      K_F = H(f, β, d)                          // fact's version VPK generation
5      if (K_F mod φ = 0) and (f.f_V = 1) then
6          if ϵ and f.f_C and f.c_V then
7              F.add(f)                          // fact's version marked
8          else
9              f.f_C ← 1
10             f.c_V ← 0
11             F.add(f)                          // fact's version candidate as carrier
```

The function takes as input the set of versions stored in the cell (denoted by F), the integer φ, called the fact version fraction, defined as $\varphi \in [1, |\mathrm{F}|]$ and used to control the number of versions *pseudo-randomly* considered as carriers. If $\varphi = 1$, all versions are selected. On the contrary, if $\varphi = |\mathrm{F}|$, only one fact version is used. Other inputs the algorithm is β, used to generate the VPK of the fact versions, *d*, which allows access to the set of dimension values pointing to the fact entry, and ϵ, which specifies whether the versions filtered are those available as carriers or those already containing marks. The function returns the set of values comprising the versions of facts considered carriers $\mathcal{F}$.

For each fact version stored in F, the algorithm generates its VPK, identified as K_F (see line 4). From F, only the fact versions fulfilling the condition $K_\mathrm{F} \bmod c = 0$ and having $f_V = 1$ are considered as carriers (see line 5). The values of f_V for each node are assigned during the embedding of facts in the repository. They can be pseudo-randomly assigned, increasing the entropy to the watermark synchronization.

When selecting new carriers, the parameter ϵ is set to 0, and the selected fact versions are identified by assigning $f_C = 1$, and flagging the carrier as available according to the expression $c_V = 0$ (see lines 8 to 11). On the contrary, when $\epsilon = 1$, the nodes already flagged with $f_C = 1$ and $c_V = 1$ are identified as active carriers and added to $\mathcal{F}$. In this case, the versions filtered are those classified as active carriers (see lines 6 and 7). Therefore, the parameter ϵ extends the functionality of the `facts_filter( )` function.

The values $f_C = 1$, and $c_V = 0$, along with the fact version value used to generate K_{F} and the fact version fraction φ, contribute to balancing and authenticating the number of versions involved in the watermark synchronization, as well as the role of carriers in the cell. This structure and approach directly contribute to increasing and managing the technique's robustness and capacity.

3.4.3 The map of carriers

Each component in the repository's structure has a different role in watermark synchronization. They can all be classified as carriers and synchronizers, according to the carrier concept. Together, along with the flag values, they comprise the carrier map of the watermarking scheme.

Synchronizers are, by definition, not carriers. They are used to generate seeds and virtual keys involved in watermark synchronization. As a general rule in this work, all dimensions are synchronizers. Furthermore, fact version nodes used as synchronizers are identified by $f_S = 1$.

On the other hand, carriers are the elements available for mark embedding. They are selected with the help of the synchronizers. A carrier can be active (i.e., carrying a mark) or inactive (i.e., available for mark embedding). Controlling these states can avoid overlapping mark embedding, therefore implementing the non-interference requirement (see Section 2.2.3). A fact version is considered a carrier if $f_C = 1$, and it is flagged as busy (already storing a mark) when $c_V = 1$. Available carriers are identified by $c_V = 0$. Regardless of the value of c_V, a version flagged with $f_C = 0$ cannot be considered a carrier.

Conceptually, a single cell can play both roles, since synchronizers are used to generate values and virtual keys involved in locating marks in carriers, and carriers are selected to contain the marks. The *msb* fragment of a fact version can be used as a synchronizer, whereas the *lsb* fragment of the same fact version can be used to act as a carrier.

By assigning $c_V = 1$ to a carrier once it is marked (see line 22 of Algorithm 2), the process marks the selected carrier as active, thereby avoiding collusion between carriers during embedding and extraction, even if they change over time due to horizontal synchronization.

The fact version values used to identify carriers and their states, synchronizers, and the content used to generate seeds and keys can be authenticated using the content stored in the same cell in nodes *C* and *S* (see Fig. 15).

3.4.4 Runtime recalibration module

The recalibration of watermark synchronization relocates some marks already stored in the repository through horizontal synchronization. Determining when and where to perform this process without colluding with vertical synchronization is critical for the design and

implementation of the watermarking scheme. We define the trigger as the component responsible for identifying the stages at which recalibration should start. In addition, a set of rules is required to determine which marks are relocated and where. Recalibration events are processed sequentially, and carrier availability is enforced through the status flags associated with each fact version, ensuring that no two watermark relocations can target the same carrier simultaneously.

Horizontal synchronization is primarily triggered when a new fact version is inserted into a cell or whenever an existing value is modified. Therefore, it relies on triggers that detect changes in the repository. The need to trigger this type of synchronization when a new fact version is inserted arises from the possibility of generating seeds or virtual keys that differ from those previously used for watermark embedding due to the added content. If those values change and the content stored in the *fork* is not *recalibrated*, noise may be introduced into the watermark during extraction, thereby risking its correct recognition.

On the other hand, by triggering the process whenever the data are modified, the data owner can customize horizontal synchronization. Given that the features of interest of an observed region may vary depending on a study's requirements, customizing the trigger is convenient, as it enables the redistribution of the distortion caused by embedding, thereby avoiding degradation of the quality of newly generated features derived from the data. Although distortion must remain within an allowable range, this functionality helps ensure reliable data processing outcomes.

Since horizontal synchronization is performed while the database is deployed and operative, we defined it as a runtime module. First, the trigger detects the cell that requires mark relocation. Then, the mark is extracted, and its original meta-mark is reconstructed. Next, a new location is selected, and the mark is regenerated using the previously reconstructed meta-mark. Finally, the mark is inserted at the new location. To evaluate the accuracy of this process, it is sufficient to compare the meta-mark and mark values used during embedding with those obtained during extraction, assuming recalibration took place in between (i.e., $\mathbb{A} \not\equiv \mathbb{B}$). In this way, collusion between vertical and horizontal synchronization can be detected.

To implement transversal synchronization, two new blocks are added to the block containing the chain of fact versions in the cell. These are the input and output blocks, denoted by B_I and B_O, respectively (see Fig. 16). Each block contains a flag f_A indicating the cell's request for the input or output of a mark. These blocks also include the position formed by the combination of dimension values from which the mark arrives or to which the mark is sent. They are defined as the transmitter and receiver addresses, identified as $\hat{d}_0, \hat{d}_1, \cdots, \hat{d}_{(|\mathrm{D}[i]|-1)}$, and are also contained in a set d.

Fig. 16 Input and output control blocks (B_I and B_O, respectively) in the cell for transversal synchronization management

Using as an example the movement of a mark through two cells, denoted as F_0^A and F_0^B, and assuming that the address $F_0^A.\mathrm{B}_O.d$ points to F_0^B, transversal synchronization starts when the trigger detects the condition $F_0^A.\mathrm{B}_O.f_A = 1$. The process then proceeds with mark extraction and attempts its insertion into F_0^B, which is performed only if $F_0^B.\mathrm{B}_I.f_A = 1$. Once the mark leaves the cell, the output of the fact is closed (in this case, $F_0^A.\mathrm{B}_O.f_A \leftarrow 0$). A mark can be moved through more than two cells if such movements are not restricted, provided that the flags of the cells located at the addresses specified in the input and output blocks indicate the possibility of mark emission and reception, respectively.

Algorithm 4 presents a detailed view of the initial call for transversal synchronization. It extracts dimensions from the digital asset, filters the dimension values, and begins combining them to create pointers for checking all cells in the multidimensional structure. In this case, the parameter θ is added to control the number of times the same mark can be moved through different carriers during horizontal synchronization. After computing the set *d*, line 7 calls the procedure `horiz_sync( )`, which is responsible for performing horizontal synchronization for the cell pointed to by *d* (see Algorithm 5).

Algorithm 4 Definition of sync_checker( ) procedure.

```
1 Input: K_S, A, β, ξ, 𝒳, N, φ, θ
2 D ← dim_def(K_S, A, 𝒳, N)
3 for i = 0; i < |D|; i++ do
4     d ← null
5     foreach d̂ ∈ D[i] do
6         d.add(d̂)                                   // see Section 3.4.2
7     horiz_sync(K_S, null, φ, d, A, β, ξ, θ)       // see Algorithm 5
```

Algorithm 5 provides a detailed definition of a recursive procedure. For this reason, it is separated from the initial operations responsible of extracting, filtering, and combining the dimension values (see Algorithm 4). The `horiz_sync( )` procedure takes as input the data owner's secret key K_S, the recurrence key K_θ, and other parameters previously introduced such as φ, *d*, A, β, and ξ. As mentioned earlier, the process uses the parameter θ to control the number of times the mark is moved during horizontal synchronization. This parameter is used to define the end of recursive calls to the process.

The first time `horiz_sync( )` is executed, K_θ is null. Then, as long as $\theta > 0$, the function continues its recursive execution (see line 2 of Algorithm 5). Using the parameter *d*, the first transmitter F_T is obtained (see line 3). If the output gate is available, the process then retrieves the active carriers from the current transmitter and stores them in $\mathcal{F}_T$ (see lines 4 and 5).

For each carrier in $\mathcal{F}_T$, the VPK K_T is computed (see line 7). If the process is performing horizontal synchronization for the first time, K_0 is used as the initial key K_0; otherwise, it is combined to K_S using a *pseudo-random* hash function, *xored* to K_θ and then assigned to K_0 (see lines 8 to 11). In this way, the initial key used to select the meta-mark from the watermark is preserved throughout the recursive execution, guaranteeing the selection of the same position for meta-mark restoration whenever required. The *msb* is then extracted, and the mark is obtained from the *lsb* position of the current carrier (see lines 12 to 14). The meta-mark is obtained by *xoring* the extracted *msb* and mark (see line 15).

After obtaining the meta-mark, the receiver pointed to by the current transmitter is obtained and stored in $\mathcal{F}_R$ (see lines 16 and 17). Then, if the input gate of the current receiver is open and its address matches the transmitter address, the process proceeds with the generation and embedding of the mark in one of its nodes (see line 18). This step is performed to double-check the authenticity of the link between the transmitter and the receiver.

Algorithm 5 Definition of horiz_sync( ) procedure.

```
1  Input: K_S, K_θ ← null, φ, d, A, β, ξ, θ
2  if θ > 0 then
3      F_T ← get_facts(A, d)                          // extraction of all fact's versions
4      if F_T.B_O.f_A then
5          F_T ← facts_filter(F_T, φ, β, d, 1)
6          foreach f_T ∈ F_T do
7              K_T = H(f_T, β, d)
8              if is_null(K_θ) then
9                  K_0 ← K_T
10             else
11                 K_0 ← K_θ ⊕ H_PRF(K_S, K_T)
12             b_β = H_β(K_T, f_T, β)
13             p_ξ = H_ξ(K_T, ξ)
14             m = [f_T]_2[p_ξ]
15             m̄ = m ⊕ b_β
16             d ← F_T.B_O.d
17             F_R ← get_facts(A, d)
18             if F_R.B_I.f_A and F_R.B_I.d = F_T.d then
19                 F_R ← facts_filter(F_R, φ, β, d, 0)
20                 K_C_R = H(F_R, β, d)
21                 f_R ← H_f(K_C_R, F_R)
22                 K_R = H(f_R, β, d)
23                 b_β = H_β(K_R, f_R, β)
24                 p_ξ = H_ξ(K_R, ξ)
25                 m = m̄ ⊕ b_β
26                 f_R.b_V ← [f_R]_2[p_ξ]                  // save original receiver bit
27                 [f_R]_2[p_ξ] ← m
                   // mark embedding
28                 f_R.c_V ← 1                                  // flagged as marked
29                 [f_T]_2[p_ξ] ← f_T.b_V                  // restore original transmitter bit
30                 f_T.c_V ← 0
31                 F_T.B_O.f_A ← 0
32                 F_R.B_I.f_A ← 0
33                 if θ − 1 > 0 then
34                     F_R.B_O.f_A ← 1
35                 K_θ ← H_PRF(K_S, K_R) ⊕ K_0
36                 horizontal_sync(K_S, K_θ, φ, d, A, β, ξ, θ − 1)
```

Once the receiver is obtained, mark embedding proceeds as in vertical synchronization, and the transmitter is released from its role by closing the output gate (see lines 19 to 32). Next, the process verifies that the current call is not the last recursive call and, if so, opens the gate for the current receiver, which will assume the transmitter role in the next recursive call (see lines 33 and 34). This is done because the final call of the function will close the output gate of the current cell. Subsequently, the key K_θ is constructed to contain the original key used for meta mark selection, and the function calls itself, now decreasing the value of θ in one unit (see lines 35 and 36). The recursive execution continues until $\theta = 0$.

3.4.5 Generation of the transmitters and receivers map

To create the map of transmitters and receivers, selecting dimension values that are distant from each other increases the likelihood of redistributing distortion across different features of interest during horizontal synchronization. Algorithm 6 details the procedure `transm_receiv_map( )`, which is responsible for establishing links between carriers for transversal synchronization. This procedure is designed to avoid collusion between embedding locations. Its outcome depends on appropriate parameter selection, which also helps prevent collusion between horizontal and vertical synchronization. The transmitter–receiver mapping is derived through a key-dependent *pseudo-random* process based on virtual primary keys and cryptographic hash functions, ensuring deterministic reconstruction while remaining computationally unpredictable to adversaries without knowledge of the secret key.

The process described in the algorithm creates a string that defines a path through different cells in the repository. It should be executed a sufficient number of times to generate multiple strings, one for each mark in W, while avoiding overlap among the elements involved in constructing other strings , in order to guarantee consistency and robustness. To create a single string, Algorithm 6 takes as input K_{S}, A, $\mathcal{X}$, and N, to select the dimension values to be considered form A, as described in previous algorithms. In addition, this process requires the number of links (a.k.a cells acting as transmitters and receivers) involved in the string, denoted by Θ.

Other parameters of particular interest for this procedure are I, v and V. They are all integer arrays of the same size as the number of dimensions in the digital asset. In the case of I, it initially contains the starting position for selecting the dimensions involved in the strings. On the other hand, v and V define the interval used to *pseudp-randomly* select, for each dimension, a value representing the cell playing e role in the string. For example, for the first dimension, a value can be selected within the interval defined by $\mathrm{v}[0]$ and $\mathrm{V}[0]$. More precisely, this interval is given by $[\mathrm{I}[0] + \mathrm{v}[0], \mathrm{I}[0] + \mathrm{V}[0]]$, considering the position contained in I.

Algorithm 6 Definition of transm_receiv_map( ) procedure.

```
1  Input: K_S, A, 𝒳, N, Θ, I, v, V
2  D ← dim_def(K_S, A, 𝒳, N)
3  F ← null
4  d_I ← null
5  for o = 0; o < Θ; o++ do
6      d ← null
7      for j = 0; j < |D[0]|; i++ do
8          I[j] ← H_RND(I[j], I[j] + v[j], I[j] + V[j])
9          d.add(D[j][I[j]])
10     if !is_null(F) then
11         if !F.B_I.f_A then
12             F.B_I.f_A ← 1
13             if o < (Θ − 1) then
14                 F.B_O.d ← d
15                 F.B_O.f_A ← 1
16             else
17                 F.B_O.d ← null
18                 F.B_O.f_A ← 0
19         d_I ← F.d
20     F ← get(A, d)
21     F.B_I.d ← d_I
```

First, the process defines the dimensions to work with (see line 2). Next, it initializes the cell and the input address of the cell being analyzed to null, defined as F and d_I, respectively (see lines 3 and 4). It then starts the Θ loops cycle to define the string (see line 5 to line 21). For each dimension, a *pseudo-random* value is selected using the function $H_{\mathrm{RND}}(\cdot)$ which takes as input the value contained in I as a seed and the interval of values corresponding to the given dimension (see lines 7 to 8). All selected values are overwritten in the corresponding position in I, since the process uses the previously selected value as the initial reference when choosing the next link in the string.

Each time a dimension is selected, it is added to *d*, which must contain all dimension values required to identify the cell (see line 9). In the first iteration, since F is null, the cell is assigned to F and the input address is set to null (see lines 20 and 21). In the second and subsequent iterations, F is no longer null, and lines 11 to 19 are executed. Only carriers not already assigned as inputs are considered (see line 11). Once selected, their inputs are flagged (see line 12), which helps avoid collusion with other cells already participating in different strings. For all the rest of the iterations except the last one, the output address of the cell is linked to the previously computed address, and the output gate is flagged as available (see lines 14 and 15). In contrast, for the last iteration, the output address is set to null, and the gate is flagged as closed (see lines 17 and 18). Every time the following address is computed, the address of the current cell is provided as the input so that it can be assigned to the next cell once obtained (see lines 19 to 21).

Once created, the maps of transmitters and receivers define the pathways for horizontal synchronization of marks. Each pathway constitutes a string of Θ links and $\Theta + 1$ nodes, formed by cells playing the roles of transmitters and receivers. Once a receiver stores a mark, it becomes the transmitter for the next iteration in the string. When horizontal synchronization is executed, the initial value of the parameter controlling recursion, θ, must satisfy the condition $\theta \leq \Theta$. During watermark synchronization, the authenticity of the

content stored in B_I and B_O is validated using the hash generated and stored in R, since this part of the cell is not involved in the generation of C and S. Details on the computation of R, C, and S are provided in Section 3.5.1.

3.4.6 Watermark detection

Watermark detection is a crucial part of vertical synchronization and the step that provides the final outcome regarding the validation of dataset provenance in this case. We implement watermark detection based on the Algorithm 7, which takes as input the data owner's secret key K_S, the allegedly watermarked digital asset A, the watermark size w, and the *msb* and *lsb* parameters denoted by β and ξ, respectively. In addition, the same filter used to reduce the dimensions $\mathcal{X}$, the array of integers N used to set the number of elements per dimension to, and the fact version fraction φ are required.

First, the majority voting matrix M is initialized (see line 3 of Algorithm 7). Since each meta-mark can be involved more than once in watermark synchronization, this matrix sets the different meta-marks for each position. Then, it evaluates their final value in the watermark by means of a majority vote. In this way, minor inconsistencies due to benign updates or malicious operations can be overcome. The majority voting matrix has the same size as the watermark w, but each position can store an array of integers of different sizes.

Algorithm 7 Definition of ext_vertical_sync( ) procedure.

1 **Input:** $K_\mathrm{S}, K_\theta \leftarrow$ `null`, A, $w, \beta, \xi, \mathcal{X}$, N, φ
2 **Output:** W
3 $M[w] = [\,]$ // initialization of the majority vote matrix M
4 $D \leftarrow$ `dim_def`$(K_\mathrm{S}, \mathsf{A}, \mathcal{X}, \mathsf{N})$
5 **for** $i = 0;\ i < |D|;\ i++$ **do**
6 $\quad d \leftarrow$ `null`
7 $\quad$ **foreach** $\hat{d} \in D[i]$ **do**
8 $\quad\quad d$.`add`$(\hat{d})$
9 $\quad \mathsf{F}_R \leftarrow$ `get_facts`(A, d)
10 $\quad \mathcal{F} \leftarrow$ `facts_filter`$(\mathsf{F}_R, \varphi, \beta, d, 1)$
11 $\quad K_\mathrm{C} = H(\mathcal{F}, \beta, d)$
12 $\quad f = H_f(K_\mathrm{C}, \mathcal{F})$
13 $\quad K_\mathrm{F} = H(f, \beta, d)$
14 $\quad$ **if** `is_null`(K_θ) **then**
15 $\quad\quad K_0 \leftarrow K_\mathrm{F}$
16 $\quad$ **else**
17 $\quad\quad K_0 \leftarrow K_\theta \oplus H_\mathrm{PRF}(K_\mathrm{S}, K_\mathrm{F})$
18 $\quad b_\beta = H_\beta(K_\mathrm{F}, f, \beta)$
19 $\quad p_\xi = H_\xi(K_\mathrm{F}, \xi)$
20 $\quad \mathsf{m} = [f]_2[p_\xi]$
21 $\quad \overline{\mathsf{m}} = \mathsf{m} \oplus b_\beta$
22 $\quad p_{\overline{\mathsf{m}}} = H_{\overline{\mathsf{m}}}(K_0, w)$ // selection of meta-mark position
23 $\quad M[p_{\overline{\mathsf{m}}}]$.add$(\overline{\mathsf{m}})$ // addition of the meta-mark value to M
24 $W \leftarrow$ maj_vot(M) // generation of W with majority voting

After detecting the dimensions and filtering their values (see line 4), dimension values are combined to identify the facts used to synchronize the watermark takes place (lines 5 to 8). The current dimension values are stored in *d* (see line 8), and the fact versions currently analyzed are obtained using the same function `get_fact( )` previously employed during embedding (line 9). Each fact version already marked as a carrier is obtained using `facts_filter( )` and stored in $\mathcal{F}$ (see line 10).

Next, the VPK identifying the cell carriers is obtained, a carrier is selected, and its VPK is also computed (see lines 11 to 13). Then, the initial key K_0 is obtained depending on the value of K_θ, provided as part of the process input (see lines 14 to 17). More precisely, K_θ is obtained from *R* if the mark has already been moved due to horizontal synchronization. Otherwise, K_θ remains `null`. Using K_{F}, the *msb* value and the *lsb* position used to extract the mark are obtained (see lines 18 and 19). The mark is extracted from the p_ξ*-th* bit of the selected fact version, and the meta-mark is generated by *XOR-ing* the selected *msb* and the detected mark (see lines 20 and 21).

Finally, the position to which the meta-mark belongs is generated (this time using K_0), and the meta-mark value is added to the list of values considered for that position on the majority voting matrix (lines 22 and 23). The use of K_0 instead of K_{F} is required because the key used to select the meta-mark from the watermark may change after horizontal synchronization. Therefore, a mechanism is needed to compute K_0 within a different cell containing the same mark.

After all dimension values and fact versions have been analyzed, majority voting is performed to obtain the final watermark signal from the majority voting matrix (line 24). Once the watermark is obtained and returned as the result of the algorithm execution, the process concludes.

3.5 Authenticity and distortion control module

The authenticity and distortion control module comprises a set of mechanisms and tools to preserve the quality of the repository data and the synchronization, which can be compromised as a result of attacks or synchronization tampering. This module includes the inclusion-exclusion mechanism to guarantee that some fact versions are available for mark embedding, whereas others are used to check the authenticity of the watermark. In this way, the scheme implements robustness against more advanced attacks, such as additive ones (see Section 2.2.6).

The distortion control module also implements a set of metrics to evaluate the distortion introduced during watermark embedding. This makes it possible to preserve the numerical distribution of the values stored in the carriers. The following sections describe the components of the distortion control module in detail.

3.5.1 Inclusion-exclusion mechanism

The inclusion-exclusion (I/E) mechanism double-checks the fact versions available for mark embedding and those used to validate the watermark's authenticity. It identifies carriers, synchronizers, and items not involved in the synchronization. This is the reason for its name; while some fact versions are considered for one task, others are excluded.

The existence of two roles in the *fork* structure nodes, together with the values of the flags (a.k.a f_V, f_C, c_V, and f_S), contributes to validating the authenticity of the watermarking scheme.

The I/E mechanism is responsible for generating *C* and *S* and for checking the authenticity of the relationships between carriers and *C*, as well as between synchronizers and *S* (see Fig. 15). It operates at the fact level, allowing the authenticity of each fact version to be checked, while authenticating other versions intentionally defined as fake. Each role is double-checked by the flags and the hashes generated from them.

A set of rules is added to the *fork* structure to implement the I/E mechanism (see Fig. 17). First, the order of fact versions is defined by creating a linked list in which each item is a fact version node. We add a new attribute to the nodes that points to the preceding node (for the first node, this attribute stores `null`). This modification to the *fork* structure is highlighted in the figure with a red arrow between two consecutive nodes.

Another modification to the *fork* structure is the addition of a node to store a hash value, alongside the nodes that store the hashes of carriers *C* and synchronizers *S*. The new node is identified as *R*, representing the hash of excluded fact versions (a.k.a ., the rest).

In the figure, we use light-green color to highlight included nodes (nodes F_1, F_2, F_4, and $F_{|F|}$), and light-red color to highlight excluded nodes (nodes F_3, and F_5). Included nodes are used to compute *C*, and *S*, whereas excluded nodes are used to compute *R*.

The generation of *C*, *S*, and *R* is performed using the flags defined in Section 3.4 (see block level **III** in Fig. 15). Nodes considered to generate *C* must satisfy the conditions $f_V = 1$ and $f_C = 1$. This means that all values available as carriers are used to generate *C*, independently of their current state, whether they are busy ($c_V = 1$) or available ($c_V = 0$). Therefore, nodes for which $f_C = 1$ are included, and *C* is generated using their *lsb* defined by ξ. Also, *C* is recomputed every time a new fact version with $f_C = 1$ is added to the *fork*, after changing a *lsb* as a result of receiving a new mark in the *fork* (vertical synchronization), or due to moving a mark among carriers within the fork (horizontal synchronization). To this end, a trigger that detects changes is implemented, so the process starts automatically whenever a value is updated.

Similarly, *S* is generated by considering nodes that satisfy the conditions $f_V = 1$ and $f_S = 1$, and the *msb* fragment of their values, defined by β. Ideally, *S* is recomputed every time a new node is inserted into the *fork* structure. It does not need to be recomputed when performing watermark synchronization. Nevertheless, its value can be generated to validate the authenticity and usability of the data contained in the cell.

Finally, nodes that satisfy the condition $f_V = 0$ are used to generate *R*. In this case, their entire value is used. Although they do not play a role in watermark synchronization, they can be used to detect whether any unauthorized operation has been performed on excluded nodes at any time. The regeneration of *R* is mainly performed every time a new value is added to the *fork* structure.

Algorithm 8 defines the computation of *C*, *S*, and *R*. It uses as input the secret key K_S, the set of current dimension values *d*, the digital asset A, the dimensions *D*, and the most significant and less significant values β and ξ, respectively. First, the hash values are initial-

ized (see line 3). Next, the process obtains the location of the node and starts analyzing all the facts contained in it (lines 4 and 5).

Algorithm 8 Hash value generation (used to compute C, S, and R).

```
1  Input: K_S, d, A, D, β, ξ
2  Output: C, S, R
3  C, S, R = 0
4  F ← get_facts(A, d)                          // extraction of all fact versions
5  foreach f ∈ F do
6      if f.f_V = 1 then
7          if f.f_C = 1 then
8              C ← C ∘ [f]_2[(|[f]_2| − ξ + 1)), (|[f]_2| − 1)]
9          if f.f_S = 1 then
10             S ← S ∘ [f]_2[0, (β − 1)]
11     else
12         R ← R ∘ [f]_2
13 C ← H_C(K_S, C)
14 S ← H_S(K_S, S)
15 R ← H_R(K_S, R)
```

Facts considered by the scheme are identified by $f.f_V = 1$ (see line 6). The binary value of the *lsb* fragment of those considered as carriers (busy or not) is joined to C (see lines 7 and 8). The notation $[f]_2$ denotes the binary representation of f, whereas the *lsb* fragment is defined as $[(|[f]_2| - \xi + 1)), (|[f]_2| - 1)]$, which corresponds to the last ξ bits of f. Next, the *msb* of the values considered for synchronization are joined to S (see lines 9 and 10). In this case, the *msb* corresponds to the first *beta* bits, identified as $[0, (\beta - 1)]$.

Values not involved in the synchronization are flagged with $f.f_V = 0$. These, also known as fake facts, are entirely used to generate R (see lines 11 and 12). After checking all fact versions, the hashes are computed using (K_S and the fragment obtained for each case. Each of C, S, and R is generated using a different hash function, namely H_C, H_S, and H_R, respectively (see lines 13 to 15). This method is used to generate the values stored in the hash nodes as well as the values to be compared with the nodes every time a change is detected in the cell.

In general, to increase the security of this mechanism, different secret key values can be used to generate C, S, and R. Each value is triggered to be committed and stored in the structure defined for hash storage in the node, and to check the authenticity of the values generated from those already defined on different occasions. Overall, C, S, and R are hard to forge, as a ttackers would need access to the secret keys to introduce any distortion without being detected.

3.5.2 Preservation of numerical distributions

Even though the changes performed to embed the marks are carried out in one of the *lsb* of carrier value, the distortion control module implements a set of metrics to preserve the numerical distribution of facts. We compare the distribution's mean and standard deviation before and after watermark embedding. Thus, we can identify the maximum distortion introduced in the data and prevent the embedding in cases that exceed the maximum tolerated changes.

We measure the distortion using (1) and (2) on each numerical attribute. In the equations, μ and σ represent the mean and standard deviation of the numerical distribution of the unwatermarked set of values, whereas μ'and σ'represent the mean and standard deviation of the new distribution obtained after watermark embedding, respectively.

The symbols M and Σ represent the absolute difference between the original and distorted means and standard deviations, respectively. Therefore, the closer the values of M and Σ are to zero, the better.

$$\mathrm{M} = |\mu - \mu| \tag{1}$$

$$\Sigma = |\sigma - \sigma| \tag{2}$$

To increase the watermark capacity, considering the extensive number of values in the repository, it is also possible to implement embedding while preserving data values by means of difference expansion. This approach also allows the data to be restored once the mark is placed in another carrier. This idea is inspired by the reversible technique proposed by Gupta et al. [70] and by Jawad & Khan in [71], based on the research by Alttar [72]. Nevertheless, we avoid implementing this approach in the current version of our proposal, as it would add complexity to the synchronization process, which is already stressed by the repository's structure.

3.6 Assumptions and operational scope

The proposed framework is designed under a set of explicit assumptions. First, fact versioning reflects naturally occurring alternative representations of the same EO fact, arising from multi-source acquisition, temporal updates, or processing pipelines; the framework does not require artificial version creation. Second, distortion-tolerant fields are selected from attributes that are not directly consumed by downstream remote sensing algorithms, ensuring that watermark embedding does not affect analytical outcomes. Third, repository normalization is assumed to be provenance-preserving: normalized data items retain source identifiers and are not merged across distinct origins, preventing ambiguity during provenance verification.

Many EO processing tasks rely on numerical derivations computed from the original data (e.g., index-based formulations or aggregate statistics) rather than on exact bit-level values. The watermarking scheme proposed in this work introduces bounded, statistically controlled distortion, as quantified by the distortion metrics reported above. Since the measured variations remain well below the tolerance thresholds typically associated with EO-derived numerical computations, the outputs of representative downstream processing operations remain stable when computed from watermarked data. As a result, the proposed watermarking mechanism preserves the usability of EO data for analytical workflows while enabling provenance verification at the data level.

Under these conditions, the framework's modular components operate deterministically and can be implemented and validated independently, supporting practical deployment and operational robustness.

4 Experimental results

This section presents the results of the experiments performed to validate our approach. We implemented our proposal using a client/server architecture. The client was developed using Java 1.8 as the programming language, with Eclipse 4.21 as the Integrated Development Environment (IDE). The server was based on the Oracle Database engine 19C, managed through Oracle SQL Developer 20.4. The data stored in the repository was obtained using the Google Earth Engine API 1.4.3 and Python 3.12.7, with Spyder 5.5.1 in the Anaconda 24.11.0 distribution. We conducted the experiments on Windows 10 Pro operating system in two different environments. The first environment (denoted as $\mathsf{E}-\mathsf{I}$) was a 2.11 GHz Intel i5-10210U PC with 16.0 GB of RAM. The second environment (denoted as $\mathsf{E}-\mathsf{II}$) consisted of a 4.20 GHz Intel i7-7700K PC with 32.0 GB of RAM.

4.1 Data collection and storage

Given the repository's critical role in the proper functioning of our proposal, it is important to describe the details of its physical model and the content stored in it that were used in the experiments. This section addresses both aspects.

4.1.1 Repository's physical model

The repository's physical model contains some features focused on the design of the cells containing the fact versions. Each version is considered a node, and the group of nodes corresponding to the same fact is contained within a cell (see Fig. 17). These design choices directly affect the implementation of our scheme and provide additional opportunities for data insertion and authentication.

The dimensions considered in our experiments are:

- `DIM_AOI`: Stores the areas of interest (AOI).
- `DIM_TIME`: Contains the time windows used to query the AOIs.
- `DIM_SOURCE`: Records the names and characteristics of the data sources.
- `DIM_METRIC`: Maintains the features analyzed for a given time and AOI.

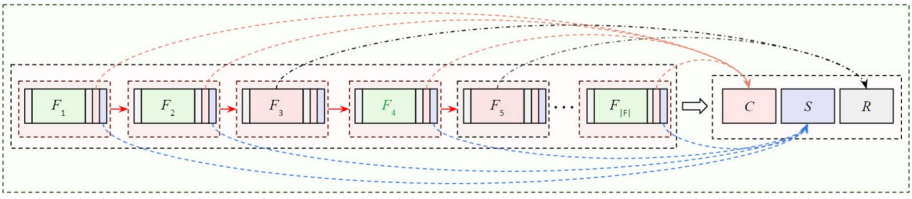


Fig. 17 Changes made to the *fork* structure for implementing the I/E mechanism

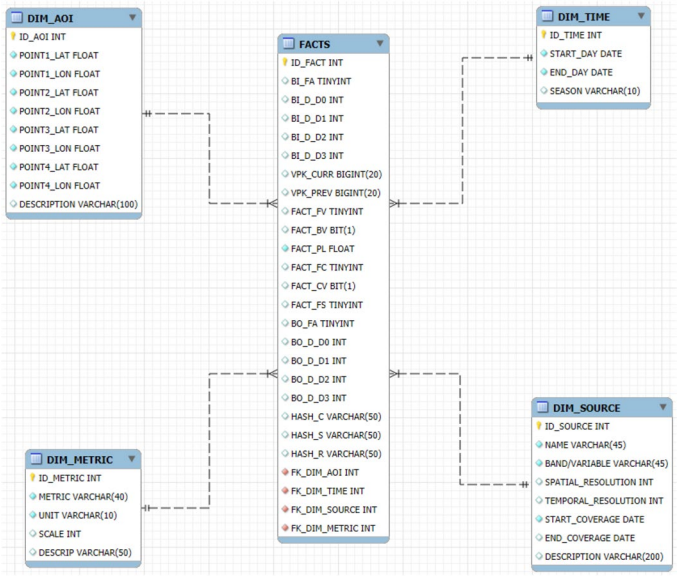


Fig. 18 Diagram of the repository's physical model composed of four dimensions and the table of facts in the center

Figure 18 presents the repository's physical model, depicting the dimensions and facts. The dimension `DIM_AOI` contains the primary key `ID_AOI`, which identifies the region, as well as the latitude and longitude of the four points defining a rectangle that delimits the region. It also contains an optional field to store a description.

The dimension `DIM_TIME` includes the primary key `ID_TIME` as well as the attributes `START_DATE` and `END_DATE`, which define the observation time window. Additionally, the optional field `SEASON` indicates the season of the year in which the data were collected.

The next dimension, `DIM_SOURCE`, contains `ID_SOURCE`, which identifies each source. It also includes the attribute `NAME`, which refers to the source queried to populate the repository, and `BAND/VARIABLE`, which specifies the queried data. This dimension also includes the attributes `SPATIAL_RESOLUTION` and `TEMPORAL_RESOLUTION`, which contain the source's spatial and temporal resolutions, respectively. Furthermore, the attribute `START_COVERAGE` stores the date when the sensor collecting the data became operational, and `END_COVERAGE` stores the last operational date, if applicable, or `null` otherwise. Finally, an additional field is provided to store a description of the source.

The last dimension, `DIM_METRIC`, stores the metrics being analyzed, which may be obtained from different sources. The attribute `ID_METRIC` identifies each metric, `METRIC` contains its name, and `UNIT` specifies its measurement unit. This dimension also contains the attribute `SCALE`, which stores the scale on which the metric is obtained, and `DESCRIP`, which provides a short description of the metric. These last two attributes are optional.

In addition to the dimensions, the repository includes a table , denoted `FACTS` (see Fig. 18). The structure of this table implements the logical design of a cell introduced in Figs. 15, 16, and 17. Facts are stored in a table with a one-to-many relationship to each dimension. Accordingly, `FACTS` contains four foreign keys, each referencing a value stored in the

corresponding dimension. The combination of these four attribute values must be unique; otherwise, the record is considered a duplicate.

We describe the attributes of FACTS in Table 6. The column ID_FACT represents the primary key that identifies each tuple. The attributes BI_FA, BI_D_D0, BI_D_D1, BI_D_D2, and BI_D_D3 correspond to the input block denoted as B_I in Fig. 16. Here, BI_FA is the active gate flag (i.e., $B_I.f_A$), while BI_D_D0, BI_D_D1, BI_D_D2, and BI_D_D3 represent the dimensions contained in $B_I.d$.

The VPKs VPK_CURR and VPK_PREV are used to link fact versions, as depicted in Fig. 17, making it possible to identify the fact pointing to the current one represented in the tuple. The attribute FACT_FV represents f_V, indicating whether the node is available for synchronization (see Fig. 15). Similarly, FACT_BV corresponds to b_V, storing the original value of the bit selected to store the mark (often used as a decoy). The actual value of the fact version is stored in FACT_PL01, referred to as the fact's payload.

Additionally, three boolean attributes, FACT_FC, FACT_CV, and FACT_FS, define f_C, c_V, and f_S, respectively. These correspond to the flag indicating the availability of the fact version as a carrier, the carrier status, and the availability of the fact version for seed generation (see Table 6).

According to Fig. 15, in the cell rpresentation, the rest of the fact version nodes follow after f_S. In the physical model of the repository, a single tuple is used to represent one fact version. Therefore, multiple tuples are required to represent an entire cell. This design choice is discussed in detail after all the attributes in the tuple have been described.

After FACT_FS, the attribute BO_FA indicates whether the output gate pointed to by the cell is active. This attribute corresponds to $B_O.f_A$, as shown in Fig. 16. The attributes BO_D_D0, BO_D_D1, BO_D_D2, and BO_D_D3 define the output address based on the output dimensions contained in $B_O.d$. Additionally, the attributes HASH_C, HASH_S, and HASH_R store the hash values corresponding to *C*, *S*, and *R*, respectively (see Fig. 17). Finally, the attributes FK_DIM1_AOI, FK_DIM2_TIME, FK_DIM3_SOURCE, and FK_DIM4_METRIC are the foreign keys linking FACTS to the dimensions DIM_AOI, DIM_TIME, DIM_SOURCE, and DIM_METRIC, respectively.

As previously mentioned, each tuple represents a fact version, and the elements of a version collectively define a node. Moreover, all versions of the same fact are contained within the same cell. Consequently, multiple tuples are required to represent a single cell. Figure 19 illustrates how different tuples define a cell. While some attributes repeat their values across tuples representing different versions within the same cell, others remain unique.

Attributes with repeating values define the cell status for synchronization, such as synchronization flags (highlighted by the red rectangles in Fig. 19). By contrast, attributes with unique values store fact versions, indicated by the green rectangles. In the figure, two groups of green rectangles represent two cells. Additionally, the group highlighted in brown corresponds to unique values used by the database schema. This is the particular case of the primary key ID_FACT.

Finally, an interesting case involves the values of the dimension contained within the blue rectangle (in this example, the values stored in BI_D_D2, BO_D_D2, and FK_DIM_SOURCE, which represent DIM_SOURCE). Since each tuple represents a fact version, each version is identified by referencing the same metric but using a different source while referring to the same area of interest and observation time. In other words, a fact version within a cell is identified by the same values of ID_AOI, ID_TIME, and ID_METRIC,

Table 6 Structure of the FACTS entity

Column	Type	Description
ID_FACT	NUMBER	Tuple ID
BI_FA	BOOLEAN	Input block (active gate flag)
BI_D_D0	NUMBER	Input block, address container (first dimension)
BI_D_D1	NUMBER	Input block, address container (second dimension)
BI_D_D2	NUMBER	Input block, address container (third dimension)
BI_D_D3	NUMBER	Input block, address container (fourth dimension)
VPK_CURR	NUMBER	Hash identifying the current fact version
VPK_PREV	NUMBER	Hash of the previous fact version
FACT_FV	BOOLEAN	Flag (fact version or node available for synchronization)
FACT_BV	BIT	Bit (original value of the marked bit)
FACT_PL01	NUMBER	Payload of the fact version (e.g., NDVI value)
FACT_FC	BOOLEAN	Flag indicating the availability of the version as a carrier
FACT_CV	BOOLEAN	Carrier status (0-available, 1-occupied)
FACT_FS	BOOLEAN	Flag indicating the availability of the version for seed generation
BO_FA	BOOLEAN	Output block (active gate flag)
BO_D_D0	NUMBER	Output block, address container (first dimension)
BO_D_D1	NUMBER	Output block, address container (second dimension)
BO_D_D2	NUMBER	Output block, address container (third dimension)
BO_D_D3	NUMBER	Output block, address container (fourth dimension)
HASH_C	NUMBER	Hash of carriers
HASH_S	NUMBER	Hash of synchronizers
HASH_R	NUMBER	Hash of remainder items (rests)
FK_DIM1_AOI	NUMBER	Entry address (first dimension)
FK_DIM2_TIME	NUMBER	Entry address (second dimension)
FK_DIM3_SOURCE	NUMBER	Entry address (third dimension)
FK_DIM4_METRIC	NUMBER	Entry address (fourth dimension)

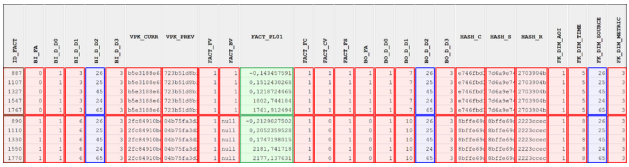


Fig. 19 Combination of tuples to define the cells containing fact versions. In this case, two cells are depicted

while varying `ID_SOURCE`. Of course, only sources providing the targeted facts can be considered. Therefore, although all dimensions theoretically share the same values when referring to a cell, the source dimension varies in the physical model to enable the collection of fact versions.

4.1.2 Data collection

We collected data for the AOIs defined in Fig. 20 to validate our approach. These AOIs are defined by regions located in or near the Italian Peninsula and exhibit diverse features, encompassing sea and land, flat terrain, and mountainous areas. This diversity allows us to include cases in which the metrics can be retrieved reliably as well as cases in which they are expected to be absent, potentially resulting in missing, inaccurate, or spurious values. In the figure, the regions are highlighted with red rectangles, with their coordinates displayed in small white text on a black background. The name of each region is shown inside a small label box. In total, we collected data for 11 areas.

Concerning `DIM_TIME`, we considered time windows representing the four seasons from 2020 to 2024. January, April, July, and October were selected as representative months for each season. Instead of collecting data daily, we computed the average of the selected metrics over each period. Figure 21 shows the periods.

Fig. 20 Areas of interest considered for collecting data in our repository (dimension `DIM_AOI`)

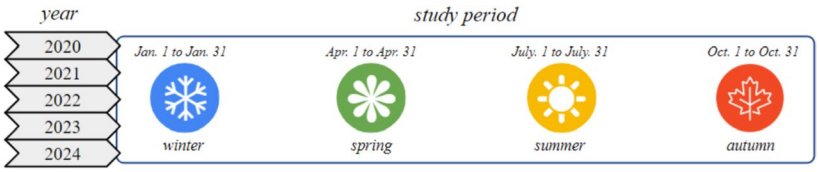


Fig. 21 Time period considered for data collection (dimension DIM_TIME)

Table 7 Link between the sources used from DIM_SOURCE and the metrics stored in DIM_METRIC

Source	Metric		
	NO_2	NDVI	EVI
COPERNICUS/S5P/OFFL/L3_NO2	✓	✗	✗
COPERNICUS/S2	✗	✓	✓
MODIS/006/MOD13Q1	✗	✓	✓
MODIS/061/MOD13Q1	✗	✗	✓
LANDSAT/LC08/C02/T1_L2	✗	✓	✓
LANDSAT/LC08/C02/T1_L2 (EVI Corrected)	✗	✗	✓

The next dimension is DIM_SOURCE, which stores the sources used to obtain data through Google Earth Engine and Python. Among the sources included in this dimension, Table 7 lists those queried, together with the corresponding metrics obtained and stored in the dimension DIM_METRIC. As a result, we obtained one fact version for nitrogen dioxide (NO_2), three fact versions for the Normalized Difference Vegetation Index (NDVI), and five for the Enhanced Vegetation Index (EVI). This variation in the number of fact versions per metric aligns with our goal of populating the repository to test the scheme under different scenarios.

In Table 7, the source COPERNICUS/S5P/OFFL/L3_NO2 refers to the Level 3 (L3) product of NO_2 measurements derived from the TROPOMI instrument aboard the Sentinel-5P satellite. This dataset provides global observations of NO_2 concentrations in the atmosphere, which are essential for monitoring air quality, atmospheric composition, and pollution levels. In contrast, COPERNICUS/S2 refers to data from the Sentinel-2 mission, part of the European Space Agency's Copernicus program. The Sentinel-2 satellites are designed for Earth observation, with a focus on land monitoring, agriculture, forestry, water bodies, and disaster management.

The source MODIS/006/MOD13Q1 refers to a Moderate Resolution Imaging Spectroradiometer (MODIS) product with a 1000 m spatial resolution, providing vegetation-related data, primarily NDVI and EVI, along with other vegetation parameters. Similarly, MODIS/061/MOD13Q1 refers to the MODIS product from the MOD13Q1 collection, with 061 indicating a more recent version that includes updates and improvements in processing and quality control compared to earlier versions.

Additionally, LANDSAT/LC08/C02/T1_L2 refers to Level-2 surface reflectance data from the Landsat 8 satellite (denoted as LC08), using the Collection 02 (C02) processing version and classified as a Tier 1 (T1) product. For LANDSAT/LC08/C02/T1_L2 (EVI Corrected), we apply a correction to the metric obtained from this source during Python querying, constraining EVI values to their expected domain, specifically [-1, 1].

4.2 Watermark synchronization

This section presents details of the watermark structure, the construction of the provenance signal stored in the watermark, and the vertical synchronization process. It is important to note that vertical and horizontal synchronization are distinct processes that must be carefully managed to prevent carrier collusion, since carrier statuses may vary with each synchronization iteration.

4.2.1 Watermark structure and construction

To build the provenance signal, all dimension values referencing the cells are considered. Specifically, we account for the AOI, the time when the content is acquired, the source providing the data, and the attributes being queried. We restrict dimension content to their primary keys to minimize the amount of data added to the watermark. Since we do not control content during its production, we also incorporate metadata fragments from the source. Such metadata may include information about the satellite and sensors used to collect the content, as well as other events that occurred during its capture or generation.

The watermark is generated in batches, each time focusing on the cells involved in the first vertical synchronization together with the selected metadata. We recommend reducing metadata volume by selecting a chunk of information small enough to encode distinctive values without introducing significant distortion during watermark embedding. It is also important to consider the volume of watermarked data, as a larger number of fact versions provides a broader cover for watermark embedding. The same principles introduced in [73] are applied here. The considered metadata include Unix timestamps (in milliseconds), indicating when the image associated with the queried content was captured.

Figure 22 illustrates how the watermark fragments are linked to the set of dimensions, the associated metadata (denoted as $\mathrm{M_d}$), and the corresponding cell. In this example, the cell used as a reference is F1, and the corresponding watermark fragment is denoted as W_{F1}. The figure also highlights the relationship between the embedded content and the fact versions. Note that the link between the watermark and the fact versions is represented by dashed lines, indicating that this relationship depends on the availability of fact versions for embedding, as specified by `FACT_FC` in each fact version node.

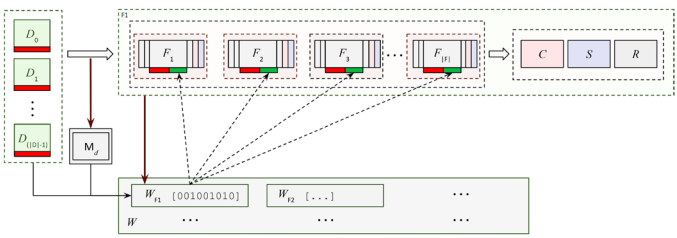


Fig. 22 Generic representation of the watermark format, its fragmentation, and links to dimensions, metadata, cells, and nodes containing the fact versions

4.2.2 Vertical synchronization execution

When vertical synchronization is executed for the first time, only a subset of cells is selected for watermark embedding. This is given by the subset of dimensions D so as to leave sufficient cover in the repository for horizontal synchronization, avoid overloading the process during watermark embedding, and prevent compromising data quality due to excessive distortion.

Before generating and embedding of the watermark, it is important to create the map of transmitters and receivers using Algorithm 6. Figure 23 depicts an example map, showing first the dimensions, highlighting those in light red as the initial entries used for cell selection. Above each dimensionthe figure reports the number of units separating the dimension values used as first entries (e.g., +2 for AOI). This separation is introduced to avoid collusion among carriers during horizontal synchronization. For the case of the source (depicted as dimension SRC. in the figure), no such selection is performed, reflecting the situation illustrated in Fig. 19, where different sources targeting the same metric are used to represent fact versions within the same cell. In Fig. 23, this is represented by a yellow cell containing the primary key values of the sources.

Each group of cells in Fig. 23 is defined as a block for synchronization. Each block presents the address fragment, defined by the combination of dimension values, and the versions fragment, determined by the source dimension values. The map is created by combining all valid combinations of dimension values, iterating over the key values of one dimension while keeping the others constant. For cases in which the source does not provide the metric represented by the address, the source key is depicted in red strikethrough text. After the initial blocks, the blocks considered at each iteration of horizontal synchronization are depicted as columns. Each block is labeled using a predefined format. For example, $\mathrm{B}^1 - \mathrm{II}$ denotes the second block of the first horizontal synchronization iteration associated with the initial block B.

The figure illustrates the case in which alternative links are created for the block B, as well as for the rest of the blocks linked to it at each iteration. Notice that the alternatives for the second block are always presented among the options for each iteration, up to $\mathrm{B}^4 - \mathrm{II}$. The case of $\mathrm{B}^4 - \mathrm{II}$ ends the possibilities for that path since that block is not valid given that the possible values for the dimension `TIME` for the block B are forced to remain between 5 and 8, using 9 to define the block C instead. In the figure, the blocks involved in the analyzed path are highlighted in a darker green color and linked using a green arrow.

Fig. 23 Map of transmitters and receivers with entries considered for each iteration. Each set of nodes depends on the cell selected for the synchronization in the previous step

4.2.3 Analysis of distortion and capacity

Once we created the map, we embedded the watermark using different data volumes and measured the resulting distortion. The amount of watermarked data varies with the cardinality of D. We analyzed different metrics to assess the impact of selecting fact versions linked to various facts. For embedding, we used different numbers of versions by varying φ. Table 8 presents the results from the lowest embedding capacity (given by $\varphi = 5$) to the highest (given by $\varphi = 1$). For each metric, the values of M and Σ indicate how the numerical distributions change after watermark embedding. The lowest values in each column are highlighted in red.

Although distortion varies across metrics when different numbers of fact versions are involved, it remains consistently low. Adjusting parameters and applying horizontal synchronization redistributes distortion across other facts. The original data can be restored using the remaining content stored in the cells.

Results in Table 8 highlight the availability of options for managing distortion, such as adjusting embedding parameter values, with the version fraction φ being particularly relevant. Nevertheless, as the repository size increases, so does the potential to increase watermark capacity without compromising data quality. This is especially important given that the repository functions as a data warehouse rather than a transactional database. Figure 24 summarizes the capacity for each metric across different values of φ. The more fact versions selected, the higher the capacity. This effect is amplified by the presence of sources that allow querying multiple versions of a particular fact, as is the case for `EVI`.

Table 8 Distortion caused due to watermark embedding using different metrics and version fractions

Metric	NO_2		NDVI		EVI	
	M	Σ	M	Σ	M	Σ
$\varphi = 5$	1.90×10^{-11}	3.26×10^{-11}	2.54×10^{-4}	2.08×10^{-4}	2.93×10^{-3}	9.93×10^{-5}
$\varphi = 3$	8.95×10^{-13}	1.90×10^{-4}	6.36×10^{-4}	5.24×10^{-4}	3.11×10^{-11}	5.62×10^{-3}
$\varphi = 1$	6.10×10^{-12}	2.34×10^{-11}	2.01×10^{-3}	1.66×10^{-3}	1.74×10^{-2}	5.90×10^{-4}

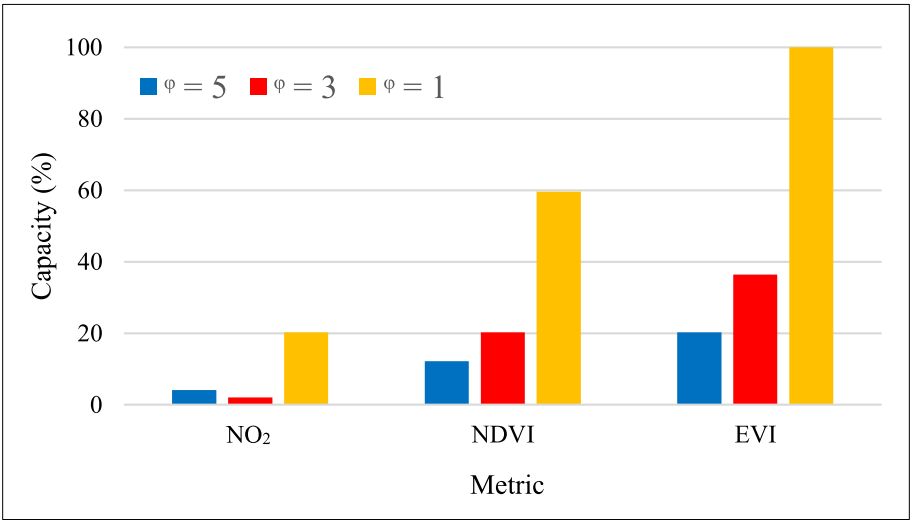
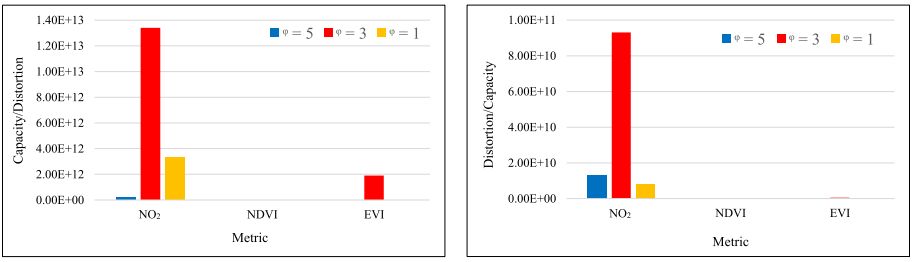


Fig. 24 Recorded changes in watermark capacity



(a) Capacity vs. distortion. (b) Distortion vs. capacity.

Fig. 25 Rates depicting the performance of proportions between capacity and distortion

Higher capacity often comes at the cost of increased distortion. Figure 25 illustrates the ratios obtained by combining capacity and distortion to determine which metrics and version fraction values are most suitable. Representing capacity as c and distortion as d, Fig. 25a shows the c/d ratio, while Fig. 25b presents the $\mathrm{d}^{-1}/\mathrm{c}$ ratio, since lower distortion is preferable. Both ratios reach their highest values when using the $\mathtt{NO}_2$ metric with $\varphi = 3$, reflecting a favorable balance between introduced distortion and the achieved capacity.

4.3 Accuracy of carriers detection

Watermark recognition relies on accurate carrier detection, which may be affected by horizontal synchronization. However, by defining the transmitter-receiver map under specific conditions, we prevent carrier collisions regardless of horizontal synchronization performance. To verify carrier authenticity, we use gate flag values and data stored in the cell, ensuring that only consistent values are considered, thereby preventing collusion.

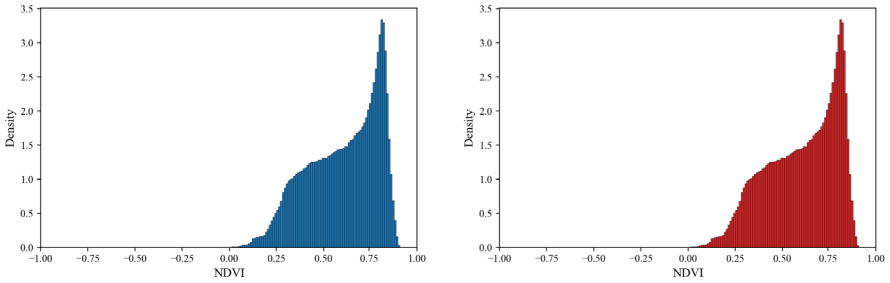
If values do not match the hashes stored in the corresponding cell field, they can be ignored or restored when possible. However, once a cell is deemed inauthentic, the objective is met, and the content is reported as tampered with.

In our experiments, the predefined dimension values and the distances within them in the transmitter-receiver map effectively prevented collusion, achieving 100% carrier detection accuracy in all cases.

4.4 Impact on representative EO-derived products

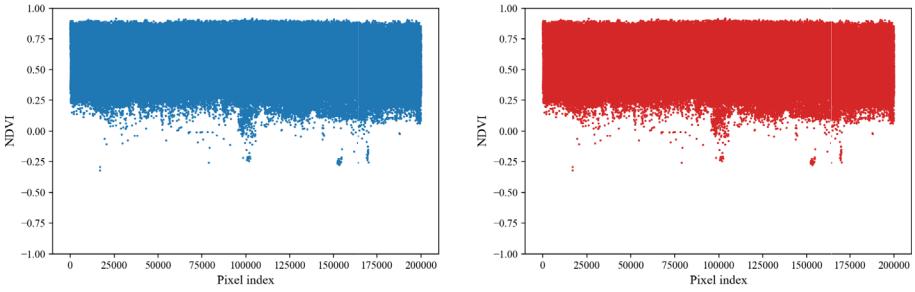
To assess the impact of watermark embedding on downstream EO usage, we evaluated the stability of the NDVI as a representative EO-derived product. NDVI values were computed from the original and watermarked data. The comparison shows that numerical differences remain negligible and well within tolerances typically accepted in EO analysis, indicating that the bounded distortion introduced by watermark embedding does not affect the usability of derived EO products.

For this experiment, we selected NDVI values derived from satellite observations over the Po Valley for the period of summer 2024. The embedding strategy targeted NDVI values using a selection factor of 1 out of every 20 fact versions, resulting in sparse and controlled modifications. Figures 26 and 27 summarize the impact of the watermarking process on



(a) NDVI distribution before watermark embedding. (b) NDVI distribution after watermark embedding.

Fig. 26 Impact of watermark embedding on NDVI values over the Po Valley AOI (NDVI distribution)



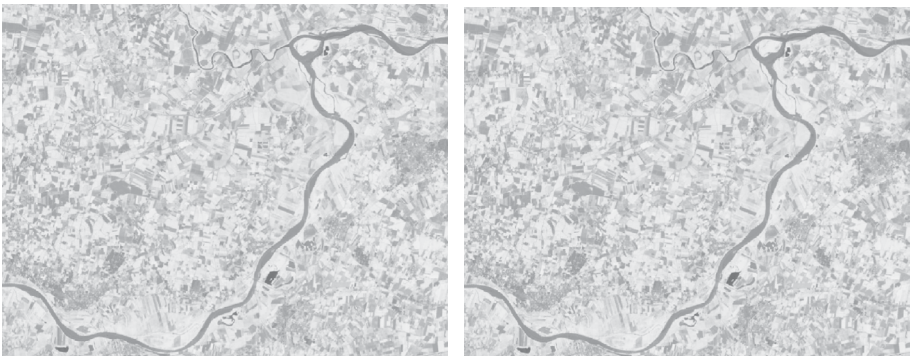
(a) NDVI values before watermark embedding. (b) NDVI values after watermark embedding.

Fig. 27 Pixel-wise distribution of NDVI values before and after watermark embedding

NDVI data quality, combining a global distributional analysis (see Fig. 26) with a pixel-level fidelity assessment (see Fig. 27).

Figure 26 shows the probability density distributions of NDVI values before and after watermark embedding. The two distributions are virtually indistinguishable across the entire NDVI range, indicating that the watermarking process does not alter the global statistical properties of the NDVI field. No shifts in mean, variance, or distribution shape are observed, and the tails corresponding to low and high vegetation vigor remain unchanged. This demonstrates that the embedding mechanism is statistically neutral and does not bias vegetation index measurements. From a remote sensing perspective, the preservation of the NDVI distribution indicates that higher-level analyses, such as vegetation classification, phenological assessment, or temporal trend analysis, are unaffected by the watermark.

Figure 27 present pixel-wise scatter representations of NDVI values before and after watermark embedding. The two plots exhibit nearly identical structures across the full range of pixel indices, with consistent value ranges, dispersion, and density patterns. No visually discernible differences are observed between the distributions, indicating that individual NDVI values are preserved with high fidelity following watermark insertion. Local variability, extrema, and overall dispersion are maintained, and no nonlinear distortions or scale-dependent effects are introduced. This pixel-level consistency demonstrates that the watermarking mechanism does not degrade data quality and preserves the analytical and operational value of NDVI products for scientific remote sensing applications.



(a) Image containing original NDVI values. (b) NDVI image with watermarked NDVI values.

Fig. 28 NDVI images of the Po Valley AOI before and after watermark embedding

In addition to the statistical and pixel-wise analyses presented in the previous figures, a direct visual comparison of the NDVI images obtained from a fragment of the selected AOI before and after watermark embedding is shown in Fig. 28. No visible differences can be discerned between the two images at the spatial or radiometric level. Vegetation patterns, spatial gradients, and high- and low-NDVI regions are preserved without perceptible alteration. This qualitative assessment confirms the quantitative results obtained in Figs. 26 and 27, demonstrating that the watermark embedding process introduces no visually detectable artifacts and does not compromise the interpretability of NDVI products. Consequently, the proposed watermarking approach preserves both the analytical and operational value of remotely sensed NDVI data.

4.5 Robustness under common attacks

We conducted a set of experiments to evaluate the robustness of our approach under common data manipulation operations. Although traditional data warehouses are typically optimized for append-only workloads, real EO repositories often undergo complex transformations, updates, and partial removals during integration, curation, and dissemination processes. For this reason, and to stress the robustness of the proposed synchronization mechanism, we consider three classes of operations commonly associated with transactional data management: content deletion, content update, and content insertion.

For these experiments, robustness is assessed in terms of the capacity of the detected watermark, expressed as the percentage of correctly recovered number of marks relative to the reference watermark obtained from the unmodified repository. In the following, watermark robustness is quantified using the detected watermark capacity, defined as the ratio (in percentage) between correctly recovered watermark symbols and the total number of embedded symbols, measured relative to the unmodified repository baseline. In line with the principles of fragile watermarking, any deviation from the initial watermark capacity is interpreted as an indication of potential data tampering. This criterion is particularly relevant for deletion and update operations, where data integrity violations are expected to be detected rather than tolerated.

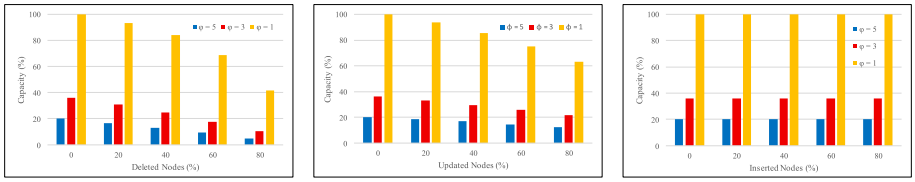
It is important to note that the results reported here correspond to a best-case attack scenario, in which only the values directly containing marks are affected. In practice, random deletion or modification attacks are also likely to impact additional elements involved in the synchronization process, such as transmitters and receivers, gate values, and consistency hashes. In such cases, the detected watermark capacity would not improve relative to the values reported here, but would typically decrease further, reinforcing the evidence of tampering.

Figure 29 summarizes the results of these experiments using the EVI metric, evaluated for three different values of the synchronization parameter φ ($\varphi = 5$, $\varphi = 3$, and $\varphi = 1$). The watermark capacity observed when no operation is applied (0% affected content) is consistent with the results previously reported in Fig. 24. Subsequently, the proportion of affected content is progressively increased to 20%, 40%, 60%, and 80%, and watermark detection is performed after each operation to assess the degradation relative to the unmodified case.

The first experiment considers content deletion. As shown in Fig. 29a, the detected watermark capacity decreases monotonically as the fraction of deleted content increases. For deletion attacks, the reduction in detected watermark capacity corresponds to an effective false negative event with respect to the original watermark, and its magnitude reflects the severity of the attack. Due to the fragile nature of the watermark, any deviation from the initial capacity immediately signals potential tampering. This behavior is consistent across all tested values of φ, with more pronounced degradation observed for configurations involving a larger number of synchronized cells.

A similar trend is observed for content update attacks (see Fig. 29b). As the percentage of updated cells increases, the quality of the recovered watermark progressively decreases. Compared to deletion, updates generally cause slightly less damage to the detected watermark capacity; however, the recovered values never match the initial capacity, which again provides clear evidence of data modification. As in the deletion case, watermark capacity degradation under update attacks reflects increasing false negative behavior as the proportion of modified content grows. The influence of the synchronization parameter φ is particularly visible in this case: configurations with smaller φ values exhibit a slower degradation for low attack intensities (e.g., between 0% and 20%), indicating reduced sensitivity when fewer cells participate in synchronization. Smaller values of φ reduce the number of synchronized carriers, resulting in slower degradation for low attack intensities but increasing the risk of delayed tampering detection. This highlights an inherent trade-off between robustness and detection sensitivity controlled by φ.

Finally, Fig. 29c reports the results for content insertion, which represents the most common operation in data warehouse environments. In this case, since newly inserted



(a) Content deletion (b) Content update (c) Content insertion

Fig. 29 Robustness of the synchronized watermark under common attack scenarios

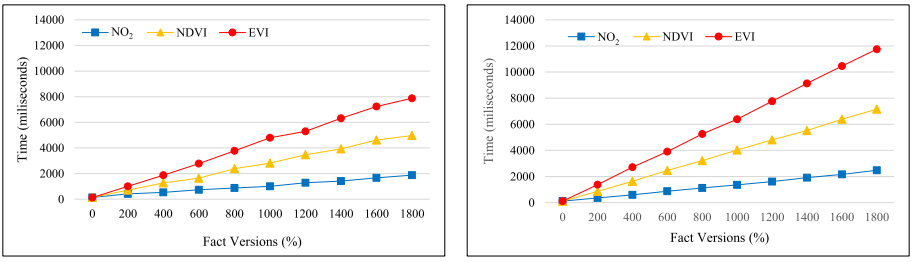
content is not associated with existing transmitter–receiver mappings or gate values, the detected watermark capacity remains unchanged. This behavior is expected and confirms that the insertion of new data does not compromise the integrity of the existing watermark. Since watermark detection in the unmodified repository yields stable, non-zero capacity values and no watermark is detected in non-watermarked data, false positive events are not observed in the evaluated scenarios, while false negatives increase monotonically with attack strength. Moreover, if incremental watermarking were enabled, newly inserted content could contribute additional watermark carriers, potentially increasing overall detection capacity and strengthening future tampering detection.

From a detection perspective, the observed degradation trends under increasing attack strength can be interpreted in terms of false negative behavior, where valid watermarks become undetectable once integrity violations exceed the tolerated thresholds. This behavior is consistent with the fragile–robust hybrid objective of the proposed framework, in which loss of detectability is treated as evidence of compromised provenance rather than as a failure of the scheme. False positives are negligible by construction, as watermark synchronization and detection are deterministic and key-dependent, making successful detection in unwatermarked data computationally infeasible without knowledge of the secret key.

4.6 Evaluation of performance and complexity

By considering different data volumes, we evaluated the scalability and complexity of our approach. We conducted a series of experiments in which we repeatedly resynchronized the watermark while increasing the number of fact versions by 200% relative to the initial count. The new values were randomly generated within the domain of existing versions in the repository. To maintain consistency, we grouped the new versions by metric according to the criteria in Table 7. Figure 30 summarizes the results for environments $\mathsf{E}-\mathsf{I}$ and $\mathsf{E}-\mathsf{II}$.

The results in both figures illustrate that, despite the complex structure of the cells stored in the repository, the approach maintains linear complexity, demonstrating its practicability. In general, synchronization occurs quickly due to the initially low volume of stored data. However, as the number of tuples increases linearly, synchronization time also rises steadily, regardless of the environment used to run the watermark services.



(a) Environment $\mathsf{E}-\mathsf{I}$ (b) Environment $\mathsf{E}-\mathsf{II}$

Fig. 30 Performance recorded for environments $\mathsf{E}-\mathsf{I}$, and $\mathsf{E}-\mathsf{II}$, during watermark synchronization when the number of versions experiences a linear increment of 200% each time

4.7 Limits and threats

We validated our approach through experiments based on gathering values of the same metrics from different data sources. We included multiple metrics and areas of interest to analyze how our approach performs in extreme cases. A significant limitation is that our proposal depends on data availability for a given AOI in a given observation period. The greater the number of fact versions, the higher the robustness and accuracy of our approach. On the other hand, performance is expected to degrade when only a few fact versions are available, due to the limited amount of content available for execution. This limitation will affect it, particularly if applied out of context. Considering our repository as a data warehouse, it is intended to grow over time, increasing data volume and thereby improving the scheme's performance.

In terms of complexity, watermarking experiences a linear cost. Therefore, adding new dimensions increases the cost, but in a predictable manner. The inclusion of additional content also contributes to increasing unpredictability for robustness. As data volume grows, it becomes more feasible to optimize the scheme, reducing parameters by fixing the values of some dimensions in the repository and reducing synchronization costs by simulating OLAP slicing. External provenance mechanisms (e.g., PROV metadata, pipeline logs, or blockchain registries) are complementary to our approach and may be preferable for strongly transformative operations that generate new derived products (e.g., band math or data assimilation).

4.8 Applicability of watermark-based provenance under EO data operations

The robustness experiments presented in this section evaluate the behavior of the proposed framework under content deletion, modification, and insertion at the repository level. To contextualize these results with respect to real EO data workflows, it is important to analyze how watermark-based provenance behaves under common EO data operations, and to clarify when embedded provenance is most effective compared to alternative provenance mechanisms.

Table 9 summarizes the applicability of the proposed watermarking approach under representative EO operations frequently encountered in data access, processing, and dissemination pipelines. Operations such as subsetting and filtering (e.g., spatial or temporal selection) and format conversion typically preserve a sufficient portion of the embedded carriers, allowing watermark detection to remain effective provided that the affected content does not exceed the tolerated distortion bounds. In these cases, watermark-based provenance offers the advantage of remaining bound to the data even after redistribution or partial release, complementing workflow-level provenance records.

It is important to clarify the interpretation of the row "Band math & indices (e.g., NDVI, EVI)" in Table 9 in relation to the experimental results presented in Section 4.4. The assessment reported in the table refers to provenance persistence when watermarking is applied to original source data (e.g., reflectance bands) and downstream products are subsequently generated through band-math operations. In such cases, derived products do not necessarily preserve the original watermark carriers, which motivates the relatively low applicability rating. By contrast, the experiments in Section 4.4 evaluate data-quality

Table 9 Applicability of watermark-based provenance under common EO data operations

EO operation	Typical effect on data	Watermark-based provenance	Rationale	Better-suited complementary mechanisms
Subset & filtering (AOI, time, variables)	Partial data removal	Medium–High	Detection remains possible if a sufficient number of watermark carriers are preserved within the selected subset	Query lineage, workflow and access logs
Format conversion (NetCDF, HDF, GeoTIFF)	Representation change	High (value-preserving)	Embedded provenance persists when numeric values are preserved within tolerated distortion bounds	Standardized metadata, FAIR-compliant provenance records
Projection & resampling	Systematic numerical modification	Medium	Watermark capacity degrades with increasing resampling magnitude, enabling tampering indication under bounded transformations	Pipeline provenance, signed derived products
Band math & indices (e.g., NDVI, EVI)	Nonlinear derivation of new variables	Low for original data	Derived products invalidate original watermark carriers; watermarking is better applied directly to derived outputs	Workflow provenance graphs, reproducibility records
Data fusion / assimilation	Multi-source blending	Low for source-level attribution	Source contributions become entangled, limiting carrier preservation across fusion steps	Provenance graphs, governance infrastructures

stability when the derived product itself (NDVI values) is selected as the watermark carrier. The two analyses therefore address complementary aspects: Table 9 concerns provenance survivability across transformations, while Section 4.4 demonstrates that watermark embedding does not compromise the usability of EO-derived products when applied at the appropriate stage of the processing chain.

Operations involving projection changes and resampling introduce systematic numerical modifications that may progressively degrade watermark capacity, depending on the magnitude of the transformation. While watermark detection may still indicate tampering under bounded resampling, such operations are more naturally complemented by pipeline-level provenance records that explicitly document transformation steps.

More strongly transformative operations, such as band arithmetic and index computation or data fusion and assimilation, generate derived products in which the original watermark carriers may no longer be preserved. In these scenarios, provenance is more reliably captured through workflow provenance graphs, execution logs, or signed derived products, while watermarking can be applied at the level of the derived outputs if persistent data-level binding is required.

Overall, this analysis highlights that watermark-based provenance is most effective for operations that preserve data semantics within bounded distortion, while external provenance mechanisms provide complementary strengths for complex derivations. The proposed framework is therefore best viewed as an enabling data-level provenance layer that integrates naturally with existing EO pipeline and governance infrastructures.

5 Conclusions

In this work, we presented a watermarking technique for tracking EO data copies, enabling the validation of their provenance and authenticity. Our approach preserves data usability, does not constrain data management, and does not interfere with downstream applications. We introduced a scheme that allows the relocation of existing marks within the data, compensating for the distortion introduced during watermark embedding. This extends traditional synchronization by introducing a new concept to the field: horizontal synchronization.

Moreover, we ensure that distortion remains within the acceptable range for carrier values. This enhances the scheme's robustness and capacity, outperforming other methods by leveraging a multidimensional repository that integrates attributes from different data types. While our approach was designed with EO data in mind, it can also protect other content that can be represented as fact versions within multidimensional structures.

For future work, we aim to incorporate an engine that considers data semantics both before watermark embedding and during resynchronization. This would not only preserve semantics but also leverage it as an additional criterion for carrier selection.

Author Contributions M. L. P. G. implemented and tested the proposal, collected the data, and wrote the manuscript. A. C. worked on the research's conceptualization, formalization, and supervision. All authors reviewed the manuscript.

Funding This study was partially funded by the European Union - NextGenerationEU, in the framework of the iNEST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS_00000043 – CUP H43C22000540006), SERICS (PE00000014 - CUP H73C2200089001), and PADS4Health (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.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

1. Harris R, Baumann I (2015) Open data policies and satellite earth observation. Space Policy 32:44–53. https://doi.org/10.1016/j.spacepol.2015.01.001
2. Pérez Gort ML, Cortesi A (2024) A fragile watermarking approach for earth observation data integrity protection. In: Cortesi A (ed) Space Data Management. Studies in Big Data, 41:47–67. Springer Singapore. https://doi.org/10.1007/978-981-97-0041-7_3
3. Brown JF, Wardlow BD, Tadesse T, Hayes MJ, Reed BC (2008) The vegetation drought response index (vegdri): A new integrated approach for monitoring drought stress in vegetation. GIScience & Remote Sensing 45(1):16–46. https://doi.org/10.2747/1548-1603.45.1.16
4. Mullapudi A, Vibhute AD, Mali S, Patil CH (2022) A review of agricultural drought assessment with remote sensing data: methods issues challenges and opportunities. Appl Geomat, pp 1–13. https://doi.org/10.1007/s12518-022-00484-6
5. Halder R, Pal S, Cortesi A (2010) Watermarking techniques for relational databases: Survey classification and comparison. J Univers Comput Sci 16(21):3164–3190. https://doi.org/10.3217/jucs-016-21-3164
6. Cox I, Miller M, Bloom J, Fridrich J, Kalker T (2007) Digital Watermarking and Steganography. The Morgan Kaufmann series in multimedia information and systems. Morgan kaufmann Burlington MA. https://doi.org/10.1016/B978-0-12-372585-1.X5001-3

7. Pérez Gort ML, Olliaro M, Cortesi A (2021) A quantile-based watermarking approach for distortion minimization. In: International symposium on foundations and practice of security, pp 162–176. https://doi.org/10.1007/978-3-031-08147-7_11. Springer
8. Prasad S, Pal AK (2020) A secure fragile watermarking scheme for protecting integrity of digital images. Iran J Sci Technol Trans Electr Eng, 44:703–727. https://doi.org/10.1007/s40998-019-00275-7. Springer
9. Rani S, Halder R (2022) Comparative analysis of relational database watermarking techniques: An empirical study. IEEE Access. 10:27970–27989. https://doi.org/10.1109/ACCESS.2022.3157866
10. Agrawal R, Kiernan J (2002) Watermarking relational databases. In: VLDB'02: Proceedings of the 28th international conference on very large databases pp 155–166. https://doi.org/10.1016/B978-1558608869-6/50022-6. Elsevier
11. Pérez Gort ML, 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. https://doi.org/10.1145/3082031.3083241
12. 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 Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2022.3194191
13. Lin C-C, Nguyen T-S, Chang C-C (2021) Lrw-crdb: Lossless robust watermarking scheme for categorical relational databases. Symmetry 13(11):2191. https://doi.org/10.3390/sym13112191
14. Agrawal R, Haas PJ, Kiernan J (2003) Watermarking relational data: framework, algorithms and analysis. The VLDB journal, 12:157–169. https://doi.org/10.1007/s00778-003-0097-x
15. Pérez Gort ML, Cortesi A (2025) A robust scheme for securing relational data incremental watermarking. International Journal of Information Management Data Insights 5(1):100320. https://doi.org/10.1016/j.jjimei.2025.100320
16. Pérez Gort ML, Olliaro M, Feregrino-Uribe C, Cortesi A (2019) Preventing additive attacks to relational database watermarking. In: Research and practical issues of enterprise information systems: 13th IFIP WG 8.9 International Conference CONFENIS 2019 Prague Czech Republic December 16–17. Proceedings 13 pp 131–140. https://doi.org/10.1007/978-3-030-37632-1_12. Springer
17. Date CJ (2003) An Introduction to Database Systems 8th edn, USA. https://www.pearson.com/en-us/subject-catalog/p/introduction-to-database-systems-an/P200000003393/9780321197849
18. 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. https://doi.org/10.1145/947380.947398
19. Pérez Gort ML, Díaz EA, Uribe CF (2017) A highly-reliable virtual primary key scheme for relational database watermarking techniques. In: 2017 international conference on computational science and computational intelligence (CSCI) pp 55–60. https://doi.org/10.1109/CSCI.2017.10. IEEE
20. Pérez Gort ML, 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, 138:112770. https://doi.org/10.1016/j.eswa.2019.06.058
21. Liang T, Zhao Y, Wang H, Cai Z, Wang Z, Wang W, Liu C (2024) Frbbm-scheme: A flexible ratio virtual primary key generation approach based on binary matching. In: International conference on intelligent information processing, pp 438–452. https://doi.org/10.1007/978-3-031-57808-3_32. Springer
22. Yang K, Yuan S, Yu J, Wang Y, Yang T, Chen C (2024) Don't abandon the primary key: A high-synchronization and robust virtual primary key scheme for watermarking relational databases. In: International conference on information and communications security, pp 289–309. https://doi.org/10.1007/978-981-97-8801-9_15. Springer
23. Che X, Akbari M, Li S, Yue D, Zhang Y, Chu L (2025) Primary key free watermarking for numerical tabular datasets in machine learning. In: International conference on pattern recognition pp 254–270. https://doi.org/10.1007/978-3-031-78119-3_18. Springer
24. Li M, Chang H, Xiang Y, An D (2020) A novel anti-collusion audio fingerprinting scheme based on fourier coefficients reversing. IEEE Signal Processing Letters, 27:1794–1798. https://doi.org/10.1109/LSP.2020.3028037
25. Joudeh B, Skoric B (2022) Collusion-resistant fingerprinting of parallel content channels. In: Proceedings of the 2022 ACM workshop on information hiding and multimedia security, pp 81–89. https://doi.org/10.1145/3531536.3532953
26. Pérez Gort ML, Olliaro M, Cortesi A (2023) Relational data watermarking resilience to brute force attacks in untrusted environments. Expert Syst Appl, 212:118713. https://doi.org/10.1016/j.eswa.2022.118713
27. Wan W, Wang J, Zhang Y, Li J, Yu H, Sun J (2022) A comprehensive survey on robust image watermarking. Neurocomput, 488:226–247. https://doi.org/10.1016/j.neucom.2022.02.083
28. Liu W, Gu H, Peng C, Cheng D (2010) Ontology-based retrieval of geographic information. In: 2010 18th international conference on geoinformatics, pp 1–6. IEEE Beijing China. https://doi.org/10.1109/GEOINFORMATICS.2010.5567612

29. Zhang C, Zhou W (2025) Physics-aware dual-branch architectures for accurate weather predictions. GeoInformatica, pp 1–26. https://doi.org/10.1007/s10707-025-00537-z
30. Conway ED (1997) An Introduction to Satellite Image Interpretation, First edition edn. Johns Hopkins University Press 2715 North Charles Street Baltimore Maryland 21218-4363. https://doi.org/10.56021/9780801855764
31. Serra-Ruiz J, Qureshi A, Megías D (2019) Entropy-based semi-fragile watermarking of remote sensing images in the wavelet domain. Entropy 21(9):847. https://doi.org/10.3390/e21090847
32. Serra-Ruiz J, Megias D (2011) A novel semi-fragile forensic watermarking scheme for remote sensing images. Int J Remote Sens 32(19):5583–5606. https://doi.org/10.1080/01431161.2010.507256
33. Jiang L, Zheng H, Zhao C (2021) A fragile watermarking in ciphertext domain based on multi-permutation superposition coding for remote sensing image. In: 2021 IEEE international geoscience and remote sensing symposium IGARSS, pp 5664–5667. https://doi.org/10.1109/IGARSS47720.2021.9553872 . IEEE
34. Shivani S, Sharma S, Saxena N (2023) An efficient fragile watermarking scheme for tamper localization in satellite images. Comput Electric Eng 109:108783. https://doi.org/10.1016/j.compeleceng.2023.108783
35. Xu D, Zhu C, Ren N (2022) A zero-watermark algorithm for copyright protection of remote sensing image based on blockchain. In: 2022 international conference on blockchain technology and information security (ICBCTIS), pp 111–116. https://doi.org/10.1109/ICBCTIS55569.2022.00036. IEEE
36. Xu D, Ren N, Zhu C (2023) High-resolution remote sensing image zero-watermarking algorithm based on blockchain and sdae. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. https://doi.org/10.1109/JSTARS.2023.3329022
37. Xu D, Ren N, Zhu C (2023) Integrity authentication based on blockchain and perceptual hash for remote-sensing imagery. Remote Sensing 15(19):4860. https://doi.org/10.3390/rs15194860
38. Liu Y, Chang Y (2024) Blockchain-based method for spatial retrieval and verification of remote sensing images. Sensors 24(7):2078. https://doi.org/10.3390/s24072078
39. Pan Z, Bao J, Zhang W, Yu Y, Zheng Y (2019) Trajguard: A comprehensive trajectory copyright protection scheme. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 3060–3070. https://doi.org/10.1145/3292500.3330685
40. Dadwal R, Funke T, Nüsken M, Demidova E (2022) W-trace: robust and effective watermarking for gps trajectories. In: Proceedings of the 30th international conference on advances in geographic information systems, pp 1–4. https://doi.org/10.1145/3557915.3561474
41. Schestakov S, Gottschalk S, Funke T, Demidova E (2024) Re-trace: Re-identification of modified gps trajectories. ACM Transactions on Spatial Algorithms and Systems. https://doi.org/10.1145/3643680
42. Tong D, Ren N, Zhu C (2019) Secure and robust watermarking algorithm for remote sensing images based on compressive sensing. Multimed Tools Appl, 78:16053–16076. https://doi.org/10.1007/s11042-018-7014-1
43. Li Y, Via BK, Li Y (2020) Lifting wavelet transform for vis-nir spectral data optimization to predict wood density. Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, 240:118566. https://doi.org/10.1016/j.saa.2020.118566
44. Goldberger A, Strassler Y (2022) A practical algorithm for completing half-hadamard matrices using lll. J Algebraic Combin 55(1):217–244. https://doi.org/10.1007/s10801-021-01077-z
45. Rani M, Dhok SB, Deshmukh RB (2018) A systematic review of compressive sensing: Concepts implementations and applications. IEEE access, 6:4875–4894. https://doi.org/10.1109/ACCESS.2018.2793851
46. Li D, Che X, Luo W, Hu Y, Wang Y, Yu Z, Yuan L (2019) Digital watermarking scheme for colour remote sensing image based on quaternion wavelet transform and tensor decomposition. Mathematical Methods in the Applied Sciences 42(14):4664–4678. https://doi.org/10.1002/mma.5668
47. Fletcher P, Sangwine SJ (2017) The development of the quaternion wavelet transform. Signal Proc 136:2–15. https://doi.org/10.1016/j.sigpro.2016.12.025
48. Sidiropoulos ND, De Lathauwer L, Fu X, Huang K, Papalexakis EE, Faloutsos C (2017) Tensor decomposition for signal processing and machine learning. IEEE Trans Signal Process 65(13):3551–3582. https://doi.org/10.1109/TSP.2017.2690524
49. Zhu P, Jiang Z, Zhang J, Zhang Y, Wu P (2021) Remote sensing image watermarking based on motion blur degeneration and restoration model. Optik 248:168018. https://doi.org/10.1016/j.ijleo.2021.168018
50. Yuan G, Hao Q (2020) Digital watermarking secure scheme for remote sensing image protection. China communications 17(4):88–98. https://doi.org/10.23919/JCC.2020.04.009
51. Chen W, Zhu C, Ren N, Seppänen T, Keskinarkaus A (2020) Screen-cam robust and blind watermarking for tile satellite images. IEEE Access 8:125274–125294. https://doi.org/10.1109/ACCESS.2020.3007689
52. Shivani S, Patel SC, Arora V, Sharma B, Jolfaei A, Srivastava G (2021) Real-time cheating immune secret sharing for remote sensing images. J Real-Time Image Proc 18:1493–1508. https://doi.org/10.1007/s11554-020-01005-7

53. Mohan A, Anand A, Singh AK, Dwivedi R, Kumar B (2021) Selective encryption and optimization based watermarking for robust transmission of landslide images. Comput Electric Eng 95:107385. https://doi.org/10.1016/j.compeleceng.2021.107385
54. Xing S, Li TY, Liang J (2021) A zero-watermark hybrid algorithm for remote sensing images based on dct and dft. In: Journal of Physics: Conference Series, 1952:022049. https://doi.org/10.1088/1742-6596/1952/2/022049. IOP Publishing
55. Lafaye J, Béguec J, Gross-Amblard D, Ruas A (2012) Blind and squaring-resistant watermarking of vectorial building layers. GeoInformatica 16(2):245–279. https://doi.org/10.1007/s10707-011-0133-8
56. Hou X, Min L, Yang H (2018) A reversible watermarking scheme for vector maps based on multilevel histogram modification. Symmetry 10(9):397. https://doi.org/10.3390/sym10090397
57. Ren N, Zhao Y, Zhu C, Zhou Q, Xu D (2021) Copyright protection based on zero watermarking and blockchain for vector maps. ISPRS Int J Geo Inf 10(5):294. https://doi.org/10.3390/ijgi10050294
58. Ren N, Zhao M, Zhu C, Sun X, Zhao Y (2021) Commutative encryption and watermarking based on svd for secure gis vector data. Earth Sci Info 14:2249–2263. https://doi.org/10.1007/s12145-021-00684-5
59. Yang J, Tan Y, Yu X, Cui G, Zhang D (2022) Waveform design for watermark framework based dfrc system with application on joint sar imaging and communication. IEEE Trans Geosci Remote Sens 61:1–14. https://doi.org/10.1109/TGRS.2022.3232528
60. Zhang X, Zhang G, Huang X, Poslad S (2022) Granular content distribution for iot remote sensing data supporting privacy preservation. Remote Sensing 14(21):5574. https://doi.org/10.3390/rs14215574
61. Qu C, Xi X, Du J, Wu T (2022) Robust watermarking scheme for vector geographic data based on the ratio invariance of dwt-csvd coefficients. ISPRS Int J Geo Inf 11(12):583. https://doi.org/10.3390/ijgi11120583
62. 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 (ICCCN) pp 1–6. https://doi.org/10.1109/ICCCN61486.2024.10637516. IEEE
63. Hamadou A, Hassane AAI, Naroua H et al (2024) Reversible semi-fragile watermarking technique for integrity control of relational database. Eng 16(9):309–323. https://doi.org/10.4236/eng.2024.169023
64. Missier P, Belhajjame K, Cheney J (2013) The w3c prov family of specifications for modelling provenance metadata. In: Proceedings of the 16th international conference on extending database technology, pp 773–776. https://doi.org/10.1145/2452376.2452478
65. Zhang F, Wang Z, Guo R, Qu G (2024) Earth observation data provenance: A blockchain-based solution. IEEE Trans Industr Inf 20(7):9548–9556. https://doi.org/10.1109/TII.2024.3383511
66. Psallidas F, Agrawal A, Sugunan C, Ibrahim K, Karanasos K, Camacho-Rodríguez J, Floratou A, Curino C, Ramakrishnan R (2023) Oneprovenance: Efficient extraction of dynamic coarse-grained provenance from database query event logs. Proceedings of the VLDB Endowment 16(12):3662–3675
67. Li Q, Zhou Q (2024) Design of blockchain traceability mechanism for data privacy protection. In: Proceedings of the 2024 2nd international conference on internet of things and cloud computing technology, pp 312–316. https://doi.org/10.1145/3702879.3702932
68. Xie Y, Song J, Wang H, Song M (2025) Training data provenance verification: Did your model use synthetic data from my generative model for training? In: Proceedings of the computer vision and pattern recognition conference, pp 23817–23827. https://doi.org/10.1109/CVPR52734.2025.02218
69. Gao L-M, Wang Q, Zhang L (2025) A high-precision verifiable watermarking scheme for vector geographic data using difference expansion and metadata restoration. Symmetry 17(11):1849. https://doi.org/10.3390/sym17111849
70. Gupta G, Pieprzyk J (2009) Database relation watermarking resilient against secondary watermarking attacks. In: International conference on information systems security, pp 222–236. https://doi.org/10.1007/978-3-642-10772-6_17. Springer
71. Jawad K, Khan A (2013) Genetic algorithm and difference expansion based reversible watermarking for relational databases. J Syst Softw 86(11):2742–2753. https://doi.org/10.1016/j.jss.2013.06.023
72. Alattar AM (2004) Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans Image Process 13(8):1147–1156. https://doi.org/10.1109/TIP.2004.828418
73. Pérez Gort ML, Olliaro M, Cortesi A (2024) Study of the watermark source's topology role on relational data watermarking robustness. IEEE Access, 12:25857–25875. https://doi.org/10.1109/ACCESS.2024.3364760