CERIAS Tech Report 2004-39

RIGHTS ASSESSMENT FOR DISCRETE DIGITAL DATA

by Radu Sion

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

RIGHTS ASSESSMENT FOR DISCRETE DIGITAL DATA

A Thesis

Submitted to the Faculty

of

Purdue University

by

Radu Sion

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2004

To Amy Brown.

ACKNOWLEDGMENTS

I want to start by thanking both my advisers. *Mike* was always available, a good friend, with amazing insights and advice (often on personal topics) for a green bean as myself. Be it 2 am or 2 pm I knew he was going to be there with a good word or a sharp research remark. Through his ways he made me experience and understand respect and restraint. I owe much of the enjoyment I get out of research writing to *Sunil* and his many great eye-opening perspectives, including on social dimensions of academia and the much debated peer-review process. It was invaluable to experience his entire tenure process parallel my graduate studies and succeed. I was extremely lucky to have the academic parents I had. I never had to worry about funding, going to (often expensive) conferences and organizing my own schedule. Thank you. Growing up now, "away from home", I will miss you.

I want to thank *Amy Brown*. For being. *Bogdan Carbunar*, my long-term apartment mate and good friend. *Melissa Dyehouse* for allowing me to teach her skiing and putting up with me for weeks through Europe. *Daniel Aliaga* for the racquetball, mountain biking and car races. *Nicoleta Neagu*, for being there. *Ion Constantinescu*, for shelter, skiing and personality. *Mirela Mustata* for driving through rain, snow and mud for me, in times when I needed a friend. *Jens Palsberg* for the great comedy club experiences and friendship. *Irina Athanasiu* for Pizza Hut. *Murat Kantargioglu*, for Tomatina. *Ladislau Boloni* for latex (the digital kind). *Ton Kalker* for the insights into issues of high dimensionality of usability spaces and friendship. *Dr. Gorman*, clearly the most important person in our department, for great conversations during empty campus holiday weeks, for advice, help and insights into the intricacies of academic bureaucracy. *Renate Mallus* for dancing and smiling. The friends from the Vienna Coffee-shop including *Olga*, *Shared*, *Jacques*, *Mercan* and *Umut*. For games, fun and a great atmosphere.

I would also like to thank my teachers at Purdue. *Doug Comer* for his PhD topic generator and his great TCP/IP class. *Susanne Hambrusch* for the great advice on the association between flower-power and academics. The members of my PhD committee (*Jens Palsberg*, *Elisa Bertino* and *Samuel Wagstaff*) for useful advice and insights. *Debbie Frantz* and the entire staff at CERIAS for great chats and help with speedy travel reimbursements. *Dan Marinescu* for guidance and help in my first year at Purdue. *Eugene Spafford* (*Spaf*) for a great research environment at CERIAS.

I thank my parents for coming up with the idea of having me, my father (1935-2003) for teaching me the power of work and persistence. There are creations out there that become relevant after they cease to exist, often leaving a bitter taste of things that could've been on a backdrop of things that actually were. My mother, for too many things, including reading the Dialogues of Socrates to a three and a half year old version of me as bed-time stories.

I want to thank all of the above and those many others that I ought to but probably forgot due to being stressed and growing older. My graduate student years were the most amazing and fun years in my life so far and you all were a big part of this experience. Thank you.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

ABSTRACT

Sion, Radu. Ph.D., Purdue University, August, 2004. Rights Assessment for Discrete Digital Data. Major Professors: Mikhail Atallah and Sunil Prabhakar.

Zero-cost verbatim digital copies, possibly one of the main features of the Information Age threatens to become one if its significant road-blocks, as more and more information is processed in fast, distributed environments. The ability to produce duplicates of digital Works at almost no cost can now be misused for illicit profit. This mandates mechanisms for effective rights assessment and protection.

One such mechanism is based on *Information Hiding*. By concealing a resilient rights holder identity "signature" (watermark) within the digital Work(s) to be protected, Information Hiding for Rights Assessment (Watermarking) enables ulterior court-time proofs associating particular Works with their respective rights holders.

One main challenge is the fact that altering the Work in the process of hiding information could possibly destroy its value. Additionally, one has to be concerned with a malicious adversary ("Mallory"), with major incentives to remove or alter the watermark beyond detection (thus disabling the ability for court-time proofs) without destroying the value of the Work (potential for illicit profit).

In this work we show that Information Hiding can be deployed as an effective tool for Rights Assessment for discrete digital data. We explore a wide range of discrete data domains, including numeric and categorical relational data, discrete sensor streams and semi-structured aggregates. We then prove that there are inherent limits to applying it effectively in hostile environments, limits illustrated best by a trade-off between the ability to "convince" in court and at the same time survive malicious adversaries.

1 INTRODUCTION

Information, as an expression of knowledge, is probably the most valuable asset of humanity today. By enabling relatively cost-free, fast, and accurate access channels to information in digital form, computers have radically changed the way we think and express ideas. As increasing amounts of it are produced, packaged and delivered in digital form in a fast, networked environment, one of its main features threatens to become its worst enemy: zero-cost verbatim copies. The inherent ability to produce duplicates of digital Works at virtually no cost can now be misused e.g., for illicit profit (see Figure 1.2). This dramatically increases the requirement for effective rights assessment and protection mechanisms.

Different avenues are available, each with its advantages and drawbacks. Enforcement by legal means is usually ineffective, unless augmented by a digital counterpart such as Information Hiding. *Digital Watermarking* as a method of Rights Assessment deploys Information Hiding to conceal an indelible "rights witness" ("rights signature", watermark) within the digital Work to be protected (see Figure 1.1). The soundness of such a method relies on the assumption that altering the Work in the process of hiding the mark does not destroy the value of the Work, and that it is difficult for a malicious adversary ("Mallory") to remove or alter the mark beyond detection without destroying the value of the Work. The ability to resist attacks from such an adversary (mostly aiming at removing the embedded watermark) is one of the major concerns in the design of a sound watermarking solution.

There exists a multitude of semantic frameworks for discrete information processing and distribution. Each distinct data domain would benefit from the availability of a suitable watermarking solution. With the notable exception of software watermarking [1], the overwhelming majority of research efforts [2] [3] have been invested in the frameworks of signal processing and multimedia Works (e.g., images, video

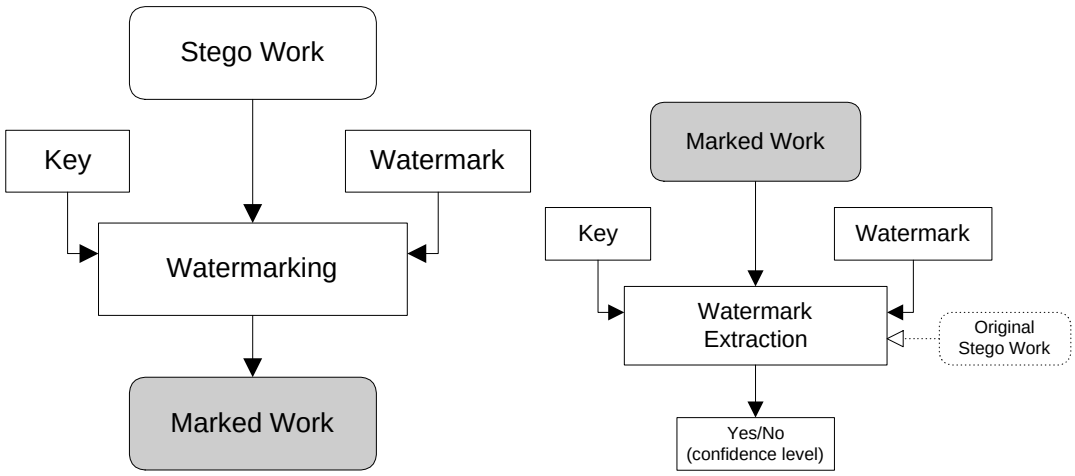


Figure 1.1. Introduction: (a) *Digital Watermarking* conceals an indelible "rights witness" ("rights signature", watermark) within the digital Work to be protected. (b) In court, a detection process is deployed to prove the existence of this "witness" beyond reasonable doubt (confidence level) and thus assess ownership.

and audio). In this dissertation, we analyze information hiding as a rights assessment tool for discrete digital data types such as relational and time-series data. In this framework we propose a theoretical model and ask: are there any limitations to what watermarking can do? What are these and when can they be reached? (Chapters 2 and 7) We then design and analyze watermarking solutions for (i) numeric sets and relational data (Chapter 3), (ii) categorical data (Chapter 4), (iii) discrete sensor streams (Chapter 5) and (iv) semi-structures (Chapter 6).

1.1 Deployment Scenario

How does the ability to prove rights in court relate to our final desiderata, namely to *protect* those rights? Why not simply publish a digest of the Works to be protected in a newspaper, just before releasing them, enabling us to prove later on in court that at least they were in our possession at the time of publication. In the following we address these and other related issues.

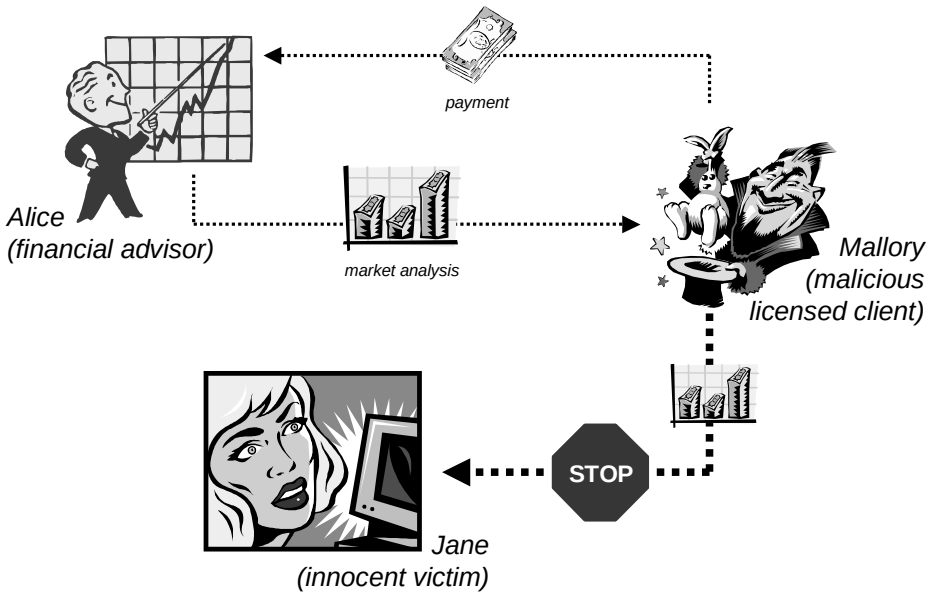


Figure 1.2. Introduction: Rights Assessment is useful when valuable content is to be sold/outsourced to potentially un-trusted parties, even if rightfully licensed.

1.1.1 Rights Protection through Assessment

The ability to prove/assess rights convincingly in court constitutes a deterrent to Mallory. It thus becomes a tool for rights protection if counter-incentives and legal consequences are set high enough. But because information hiding does not provide means of actual access control, the question of rights protection still remains. *How* are rights protected here?

It is intuitive that such a method would only work if the rightful rights-holder (Alice) actually knows about Mallory's misbehavior **and** is able to prove to the court that: (i) Mallory possesses a certain Work X and (ii) X contains a "convincing" (e.g., very rare with respect to the space of all considered similar Works) and "relevant" (e.g., a string stating "(c) by Alice") watermark.

What watermarking does not offer is a direct deterrent. If Alice does not have the knowledge of Mallory's illicit possession of the Work and/or if it is impossible to actually prove this possession in court beyond reasonable doubt, then watermarking cannot be deployed directly to prevent Mallory.

If, however, Information Hiding is aided by additional access control level levers, it can become very effective.

For example, if in order to derive value from the given Work (e.g., watch a video tape), Mallory has to deploy a known mechanism (e.g., use video player), information hiding could be deployed to enable such a proof of possession, as follows. One simple example would involve modifying the video player so as to detect the existence of a watermark and match it with a set of credentials and/or "viewing tickets" (that can be purchased) associated with the player's owner. If no match is found, the tape is simply not played back.

This is just one of many scenarios where watermarking can be deployed in conjunction with other technologies to aid in managing and protecting digital rights. Of course this scenario is simplistic and relies on the assumption that the cost of reverse engineering this process is far higher than the potential derived illicit gain. However this is essential in that it illustrates the game theoretic nature at the heart of the watermarking proposition and of information security in general.

Another example application of Resilient Information Hiding as a tool aiding rights management, would be its deployment to "track" license violators by hiding a specific mark inside the Work, a mark that uniquely identifies the party it was sold/outsourced to (fingerprinting). If the Work would then be found in the public domain, that mark could be used to assess the source of the leak (see Figure 1.3).

Watermarking is a game with two adversaries, Mallory and Alice. At stake lies the value inherent in a certain Work X, over which Alice owns certain rights. When Alice releases X she deploys watermarking for the purpose of ensuring that one of the following holds:

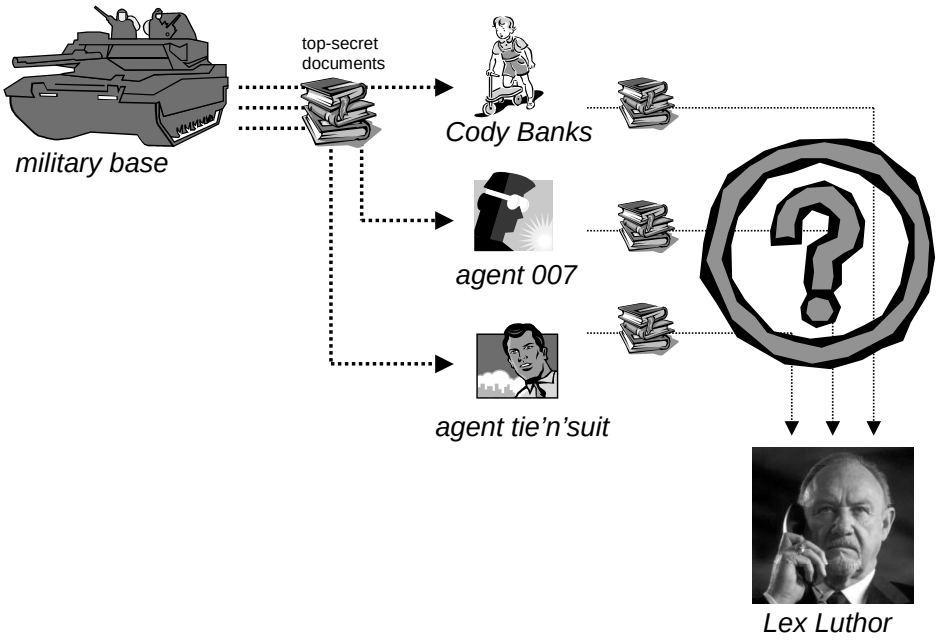


Figure 1.3. Introduction: A scenario where resilient information hiding for fingerprinting might reveal which secret agent leaked secret documents to Lex Luthor.

- she can always prove rights in court over any copy or valuable derivate of X (e.g., segment)
- any existing derivate Y of X, for which she cannot prove rights, does not preserve any significant value (derived from the value in X)
- the cost to produce such an un-watermarked (for which she cannot prove rights) derivate Y of X that is still valuable (with respect to X) is higher than its value

1.1.2 Information Hiding vs. Newspaper Digests

Apparently Alice could simply publish a (e.g., cryptographic) digest of X in a newspaper, thus being able to at least claim a time stamp of possession of X

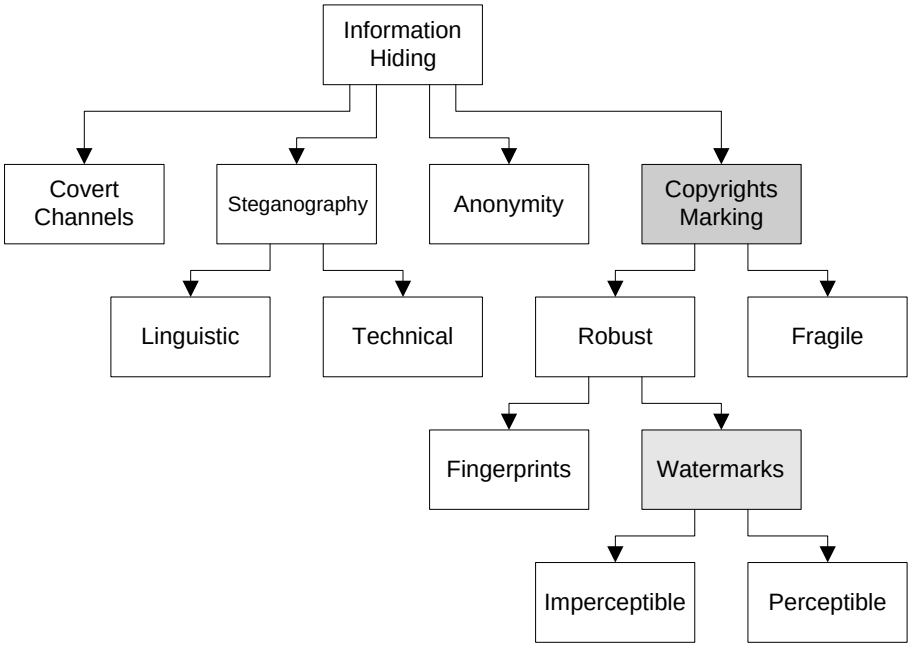


Figure 1.4. Introduction: Information Hiding classification according to Petitcolas et al [4]

later on. Why not deploy this as a rights assessment tool instead of information hiding? There are many reasons why it would not work, including (i) scalability issues associated with the need for a trusted third party (newspaper), (ii) the cost of publishing a digest for each released Work, (iii) scenarios when the fact that the Work is watermarked should be kept secret (stealthiness) etc.

Maybe the most important reason is that Mallory can now claim that his ownership of the Work precedes X's publication date, and that Alice simply (modified it and) published a digest. It would then be up to the court to decide if Mallory is to be believed or not, hardly an encouraging scenario for Alice. This could work if there existed a mechanism for the mandatory publication of digests for each and every valuable Work, again probably impractical due to both costs and lack of scalability.

It becomes clear that deploying such aids as rights assessment tools makes sense only in the case of the Work being of value only un-modified. In other words if it does not tolerate any changes (without losing its value) and Mallory is caught

in possession of an identical copy, Alice can successfully prove in court that she possessed the original at the time of its publication, but she cannot prove more.

Now, considering that, in the case of watermarking, the assumption is that, no matter how small, there are modifications allowed to the Works to be protected, in some sense the two approaches complement each other. If no modifications are allowed, then a third-party "newspaper" service might work for providing a time-stamp type of ownership proof that can be used in court.

1.2 Watermarking vs. Watermarking

In existing research, the term "watermarking" denotes the use of information hiding techniques to (also) assess digital rights, overwhelmingly focused in the broader frameworks of signal processing and multimedia Works.

In this dissertation we (appropriately) re-use this term, to denote the same concept of using Information Hiding to provide proofs of rights. However this brings about the question of the specifics of the relationship between the actual research challenges and techniques deployed in both frameworks. Because, while the terms might be identical, the associated model, challenges and techniques are different, almost orthogonal: whereas in the signal processing case there usually exists a large noise bandwidth, due to the fact that the final data consumer is likely human (with associated limitations of the sensory system), in the case of discrete data types this cannot be assumed and data quality assessment needs to be closely tied with the actual watermarking process (see Section 2.2).

Another important differentiating focus in our research is the emphasis on the actual ability to convince in court as a success metric, unlike most approaches in the signal processing realm, that centered on bandwidth. We believe that, while bandwidth is a relevant related metric, it does not consider important additional issues such as malicious transforms and removal attacks. We are not as concerned with packing a lot of rights assessment information (i.e., watermark bits) in the

Works to be protected, as we are concerned with being able to both survive removal attacks and convince in court. We explore this more in Chapter 7 (and [5], [6]).

Maybe the most important difference between the two domains is that, while in a majority of watermarking solutions in the multimedia framework, the main domain transforms are signal processing primitives (e.g., Works are mainly considered as being compositions of signals rather than strings of bits), in our case data types are mostly discrete and are not naturally handled as continuous signals. Additionally, while (for example) discrete versions of frequency transforms can be deployed as primitives in information encoding for digital images [2], the basis for doing so is the fact that, although digitized (thus in discrete format), images are at the core defined by a composition of light reflection signals and are consumed as such (by the final human consumer). By contrast, arbitrary discrete data (e.g., categorical data) is naturally discrete [1] and often to be ingested by a highly sensitive processing component (e.g., a computer rather than a perceptual system tolerant of distortions).

Thus, while the term "watermarking" will be used throughout this dissertation to denote the process of deploying information hiding for the purpose of rights assessment, in terms of actual models, challenges and techniques, it is to be distinguished from its use in the broader domain of signal processing and multimedia. And while similarities are always to be found (e.g., "Gaussian noise addition" in the multimedia case equates to a "random un-informed attack" in the discrete data case) we do not believe that "everything is a signal" for the purpose of rights assessment. Comparing efforts in the two domains can often result in comparing apples to oranges.

1.3 Summary of Contributions

The main contributions of this dissertation include: a theoretical model for rights assessment through information hiding for discrete digital data (Chapters 2 and 7), the design and analysis of watermarking solutions for numeric sets and relational

[1] Unless we consider quantum states and uncertainty arising in the spin of the electrons flowing through the silicon.

data (Chapter 3), categorical data (Chapter 4), discrete sensor streams (Chapter 5) and semi-structures (Chapter 6).

1.3.1 Model

In Chapter 2 (and [5], [6]) we introduce a model for watermarking. We define fundamental concepts including: *usability domain* - a set of functionals quantifying a digital Work's value in terms of its specific use; *watermark* - an induced property of a watermarked Work O', so rare, that if we consider any other Work O'', "close-enough" to the original Work O, the probability that O'' exhibits the same property can be upper-bounded; watermark *vulnerability* - the ability of an attack to succeed against a watermarking scheme. One fundamental difference between watermarking and generic data hiding resides in the main applicability and descriptions of the two domains. Data hiding in general, and covert communication in particular, aims at enabling Alice and Bob to exchange messages in a manner as resilient and stealthy as possible, through a medium controlled by evil Mallory. Digital watermarking is deployed in court by Alice to prove rights over a given Work, usually in a scenario where Mallory benefits from using/selling that very same Work or maliciously modified versions of it. In digital watermarking, the actual value to be protected lies in the Works themselves whereas information hiding usually makes use of them as simple value "transporters". Rights assessment can be achieved by demonstrating that a particular Work exhibits a rare property (read "hidden message" or "watermark"), usually known only to Alice (with the aid of a "secret" - read "watermarking key"). For court convince-ability purposes this property needs to be so rare that if one considers any other random Work "similar enough" to the one in question, this property is "very improbable" to be present (i.e., very unlikely to arise fortuitously). This defines a main difference from steganography: for its purpose, the specifics of the property (e.g., watermark message) are irrelevant as long as Alice can prove "convincingly" it is she who embedded/induced it to the original (non-

watermarked) Work. Thus, in watermarking the emphasis is on "detection" rather than "extraction". Extraction of a watermark is usually a part of the detection but just complements the process up to the extent of increasing the ability to convince in court.

1.3.2 Numeric Relational Data

In Chapter 3 (and [7], [8], [9]) we introduce a solution for relational database content rights protection through watermarking. Rights protection for relational data is of ever increasing interest, especially considering areas where sensitive, valuable content is to be outsourced. A good example is a data mining application, where data is sold in pieces to parties specialized in mining it. Our solution addresses important attacks, such as subset selection, linear data changes, random alteration attacks, and data loss. We introduce wmdb.*, a proof-of-concept implementation and its application to real-life data, namely in watermarking the outsourced Wal-Mart sales data available at our institute.

The main challenges in this new domain derive from the fact that, since the associated data types do not have fixed, well defined semantics (as compared to multimedia) and may be designed for machine ingestion, identifying the available "bandwidth" for watermarking becomes as important as the actual encoding algorithms. Remember that one of the desiderata of watermarking is to insert an indelible mark in the object such that the insertion of the mark does not destroy the value of the object. Clearly, the notion of value or utility of the object is central to the watermarking process. This is closely related to the type of data and its intended use. For example, in the case of software the value may be in ensuring equivalent computation, and for text it may be in conveying the same meaning (i.e., synonym substitution is acceptable). Similarly, for a collection of numbers, the utility of the data may lie in the actual or the relative values of the numbers, or in their distribution (e.g., normal with a certain mean and variance). Because, one can always

identify some use of the data that would be affected by even a minor change to any portion of it, it becomes necessary that the intended purpose of the data to be preserved is identified and integrated in the watermarking process.

Our solution starts by receiving as user input a reference to the relational data to be rights-protected, a watermark to be embedded as a copyright proof, a secret key used to protect the embedding, and a set of data quality constraints to be preserved in the result. It then proceeds to watermark the data while continuously assessing data quality, potentially backtracking and undoing undesirable alterations that do not preserve data quality. Watermark embedding consists of two main parts: in the first stage, the input data set is securely partitioned into subsets of items; the second stage then encodes one bit of the watermark into each subset. If more subsets (than watermark bits) are available, error correction is deployed to result in an increasingly resilient embedding. The algorithms prove to be resilient to important classes of attacks, including subset selection, linear data changes, and random alterations.

The system design, (including the mechanisms evaluating data quality constraints through plugins), is outlined in Figure 1.5 (a). To exemplify the resilience of the method (e.g., to random alterations), in Figure 1.5 (b), a comparison is made between the case of uniformly distributed (i.e., values are altered randomly between 100% and 120% of their original value) and fixed alterations (i.e., values are increased by exactly 20%). In the case of fixed alterations the behavior demonstrates the self-healing ability of our method: as more and more of the tuples (past the 50% mark) are altered linearly, the watermark distortion decreases. When over 95% of the data is modified consistently and linearly, the watermark suffers only 7% alterations.

Another important experiment analyzes the ability to preserve classes in the resulting watermarked Work. Classification is extremely relevant in areas such as data mining, and we envision that many of the actual deployment scenarios for our relational watermarking application will require classification preservation. Classification preservation deals with the problem of propagation of the classes occurring in the original (input) data in the watermarked (output) version of the data. It

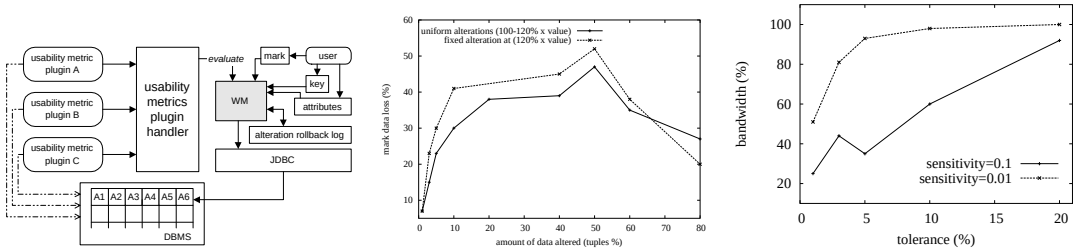


Figure 1.5. Introduction: Relational Data with Numeric Types – (a) The **wmdb.*** package. (b) Random attack (non-zero average) on a normally distributed data set. (c) Impact of classification preservation on the available watermarking bandwidth.

provides thus the assurance that the watermarked version still contains most (or within a certain allowed percentage) of the original classes. Figure 1.5 (c) depicts how classification can be preserved while making optimal use of the available bandwidth. For example, up to 90% of the underlying bandwidth can become available for watermark encoding with a restrictive 6% classification preservation goodness.

These results confirm the adaptability of our watermarking algorithm. As classification tolerance is increased, the application adapts and makes use of an increased available bandwidth for watermark encoding. This also show that classification preservation is compatible with our distribution-based encoding method, an important point to be made, considering the wide range of data-mining applications that could naturally benefit from watermarking ability.

The main contributions in this chapter include: (i) a resilient watermarking method for numeric relational data, (ii) a technique for enabling user-level run-time control over properties that are to be preserved as well as the degree of change introduced, (iii) a complete, user-friendly implementation for numeric relational data, and (iv) the deployment of the implementation on real data, in watermarking the Wal-Mart Sales Database and the analysis thereof.

1.3.3 Categorical Data

In Chapter 4 (and [10], [11]) we introduce a novel method of watermarking relational data with categorical types. We discover new watermark embedding channels and design novel watermark encoding algorithms. We analyze important theoretical bounds including mark vulnerability. While fully preserving data quality requirements, our solution survives important attacks, such as subset selection and random alterations. Mark detection is fully "blind" in that it does not require the original data, an important characteristic especially in the case of massive data. We propose various improvements and alternative encoding methods. We perform validation experiments by watermarking the outsourced Wal-Mart sales data available at our institute. We prove (experimentally and by analysis) our solution to be extremely resilient to both alteration and data loss attacks, for example tolerating up to 80% data loss with a watermark alteration of only 25%.

Important new challenges are associated with this domain. One cannot rely on "small" alterations to the data in the embedding process. Any alteration may be significant. The discrete characteristics of the data require discovery of fundamentally new bandwidth channels and associated encoding algorithms. Our method proves to be resilient to important attacks, including subset selection and random alterations.

Our solution starts by discovering two domain-specific watermark embedding channels, namely (i) the *inter-attribute associations* and (ii) the *value occurrence frequency-transform* (attribute frequency histogram). Next, embedding methods to resiliently hide information in these channels are designed. The main method starts with an initial user-level assessment step in which a set of attributes to be watermarked are selected. Next, watermark encoding proceeds for each attribute pair (K, A) in the considered attribute set, by selecting a subset of "fit" tuples (determined directly by the association between A and K). These tuples are then considered for mark encoding. Mark encoding alters the tuple's value according to secret criteria that induces a statistical bias in the distribution for that tuple's altered

value. The mark decoding process relies on discovering this induced statistical bias. Yet another embedding method is available to counter extreme vertical partitioning attacks in which only a single attribute A is preserved in the result. If, intuitively, for massive data sets, the number of possible discrete values for A is much smaller than the data set size, then A contains many duplicate values. There is probably very little value associated with knowing the set of possible values of A. The main value in this scenario (in Mallory's eyes) is (arguably) to be found in one of the only remaining characteristic properties, namely the value occurrence frequency distribution for each possible value of A. If we could devise an alternative watermark encoding method for this set then we would be able to associate rights also to this aspect of the data, thus surviving this extreme partitioning attack. In Chapter 3 we introduce a watermarking method for numeric sets that is able to minimize the absolute data alteration in terms of distance from the original data set. We propose to apply this method here to embed a mark in the occurrence frequency distribution domain. One concern we should consider is the fact that in the categorical domain we are usually interested in minimizing the *number* of data items altered whereas in the numeric domain we aim to minimize the absolute data change. It is fortunate that, because we now have numeric values modeling occurrence frequency, a solution minimizing absolute data change in this (frequency) domain naturally minimizes the *number* of items altered in the categorical value domain.

The experimental results include an analysis of the relationship between the amount of alterations required in the watermarking phase and a minimum guaranteed watermark resilience. It can be seen in Figure 1.6 (a) that with a decreasing number of encoding alterations (decreasing e) the vulnerability to random alteration attacks increases accordingly. This illustrates the trade-off between the requirement to be resilient and the preservation of data quality (e.g., fewer alterations). An experiment analyzing resilience to data loss is depicted in Figure 1.6 (b). We observe here the compensating effect of error correction. Compared to data alteration attacks, the watermark survives even better with respect to attack size (in this case data loss).

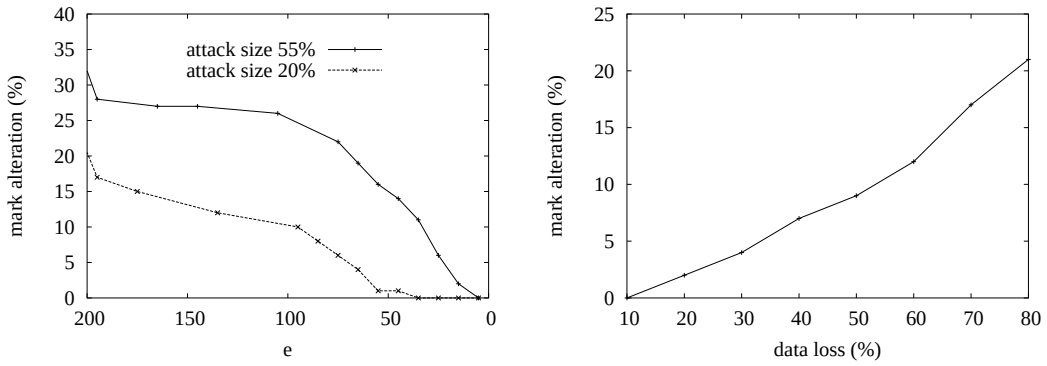


Figure 1.6. Introduction: Relational Data with Categorical Types – (a) More available bandwidth (decreasing e) results in a higher attack resilience. (b) The watermark degrades almost linearly with increasing data loss.

The main contributions of this effort include: (i) the proposal and definition of the problem of watermarking categorical data, (ii) the discovery and analysis of new associated watermark embedding channels (iii) the design of novel encoding algorithms and (iv) their experimental analysis.

1.3.4 Sensor Streams

Today's world of increasingly dynamic computing environments naturally results in more and more data being available as fast streams. Applications such as stock market analysis, environmental sensing, web clicks and intrusion detection are just a few of the examples where valuable data is streamed to its consumer. Often, streaming information is available on the basis of a non-exclusive, single-use customer license. One major concern, especially given the digital nature of the valuable stream, is the ability to easily record and potentially "re-play" parts of it in the future. If there is value associated with such future re-plays, it could constitute enough incentive for a malicious customer (Mallory) to record and duplicate segments of

data, subsequently re-selling them for profit. Being able to protect against such infringements becomes a necessity.

In Chapter 5 (and [12]) we introduce the issue of rights protection for streaming discrete (sensor) data through watermarking. This is a novel problem with many associated challenges including: the inability to perform multiple-pass random accesses to the entire data set, the requirement to be fast enough to keep up with the incoming stream rate, to survive instances of extreme sparse sampling and summarizations, while at the same time keeping data alterations within allowable bounds. We propose a solution and analyze its resilience to various types of attacks as well as expected domain-specific alterations, such as sampling and summarization. We implement a proof of concept software (wms.*) and perform experiments to assess these resilience levels in practice. Our method proves to be well suited for this new domain. For example, we can recover an over 97% confidence watermark from a sampled (e.g., less than 8%) stream. Similarly, our encoding ensures survival to stream summarization (e.g., 20%) and random alteration attacks with very high confidence levels, often above 99%.

A set of novel challenges present themselves in this domain. Any stream processing performed is necessarily both time and space bound. The time bounds derive from the fact that the processing has to keep up with incoming data. The space bounds are referring to the finiteness of any storage mechanism, when compared with the virtually infinite nature of streaming data. At the same time, any quality preservation constraints can be formulated only in terms of the current available data window; including any history information will come at the expense of being unable to store as much new incoming data. Moreover, the effectiveness of any rights protection method is directly related to its ability to survive legitimate domain specific transformations as well as malicious attacks. In this framework we deal with the following: (A1) summarization, (A2) sampling, (A3) segmentation (we would like to be able to recover a watermark from a finite segment of data drawn from the stream), (A4) scaling (there might be value in actual *data trends*, that Mallory could

still exploit, by scaling the initial values), (A5) addition of stream values and (A6) random alterations.

At an overview level, watermark embedding proceeds as follows: (a) first a set of "major" data extremes (actual stream max/min values) are identified in the data stream, extremes that feature the property that they (or a majority thereof) can be recovered after a suite of considered alterations (possibly attacks) such as (random) sampling and summarization. Next (b) a certain criteria is used to select some of these extremes as recipients for parts of the watermark. Finally (c), the selected ones are used to define subsets of items considered for 1-bit watermark embedding of bits of the global watermark. The fact that these extremes can be recovered ensures a consistent overlap (or even complete identity) between the recovered subsets and the original ones (in the un-altered data). In the watermark detection process (d) *all* the extremes in the stream are identified and the selection criteria in step (b) above is used once again to identify potential watermark recipients. For each selected extreme, (e) its corresponding 1-bit watermark is extracted and ultimately the global watermark is gradually re-constructed, by possibly also using an error correction mechanism. In summary, one of the main ideas behind our solution is the use of extreme values in the stream's evolution as watermark bit-carriers. The intuition here lies in the fact that much of the stream value lies in its fluctuating behavior (and associated extremes), more likely to survive value-preserving, domain-specific transforms.

We performed experiments on watermark survival to a variety of transformations, including random alterations and combined sampling and summarization. In Figure 1.7 (a), random alterations are illustrated. Naturally, an increasing level of distortion results in decreasing detection. Nevertheless, for 50% of the data altered within 10% of the original value, we still detect a watermark bias of roughly 25 bits, yielding a very convincing false-positive rate of less than "one in thirty million". In Figure 1.7 (b) we outline the impact of a *combined* transformation (sampling and summarization) on the watermark embedding. Because of the nature of both

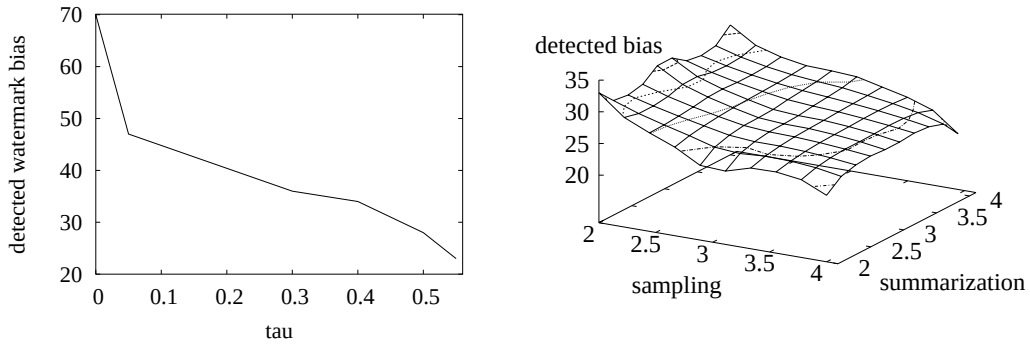


Figure 1.7. Introduction: Discrete Streaming Data – (a) Watermark survival to epsilon-attacks. (b) Watermark survival to combined sampling and summarization.

transformations and of the resilience featured in each case, the combination seems to be survived well. For example, 25% sampling, followed by 25% summarization still yields a watermark bias of up to 20, corresponding to a favorable, low false-positive rate of "one in a million".

The main contributions in this chapter include: (i) the proposal and definition of the problem of watermarking discrete sensor streams, (ii) the discovery and analysis of new watermark embedding channels for such data, (iii) the design of novel associated encoding algorithms, (iv) a proof of concept implementation of the algorithms and (v) their experimental evaluation. The algorithms introduced here prove to be resilient to important domain-specific classes of attacks, including stream re-sampling, summarization (replacing a stream portion by its average value) and random changes.

1.3.5 Abstract Structures

While most existing work on watermarking is about specific kinds of media (in fact most papers were about image watermarking), in this part of our research, rather than dealing with single media at a time (image, audio, video), we explore a more

general notion of a modern document. Generalized documents are aggregates of multiple types of content – a document is a structured aggregation of many types of data, and the information content of the document is as much in the structure (the graph) as in the nodes (that contain particular data types). The fact that nodes in semi-structures are value-carrying, means that a watermarking algorithm can make use of their encoding capacity by using traditional watermarking, but the graph that "glues" these together is another central element of the watermarking process. We propose to combine and use these two facets (structural and node-content) to provide a solution for the watermarking of aggregates.

In Chapter 6 (and [13]) we discuss the watermarking of abstract structured aggregates of multiple types of content, such as multi-type/media documents. These *semi-structures* can usually be represented as graphs and are characterized by value lying both in the structure *and* in the individual nodes. Example instances include XML documents, complex web content, workflow and planning descriptions. We propose a scheme for watermarking abstract semi-structures and discuss its resilience with respect to attacks. While content-specific watermarking deals with the issue of protecting the value in the structure's nodes, protecting the value pertaining to the structure itself is a new, distinct challenge. Nodes in semi-structures are value-carrying, thus a watermarking algorithm could make use of their encoding capacity by using traditional watermarking. For example if a node contains an image then image watermarking algorithms can be deployed for that node to encode parts of the global watermark. Also, given the intrinsic value attached to it, the graph that "glues" these nodes together becomes in itself a central element of the watermarking process that makes use of these two value facets, structural and node-content.

Multiple challenges are encountered in this framework, mostly derived from the requirement to survive domain-specific transformations and likely attacks by Mallory, including: elimination of value-"insignificant" nodes (A1), elimination of inter-node relations (A2), value preserving graph partitioning into independent usable partitions (A3), modification of node content, within usability vicinity (A4), addition of value

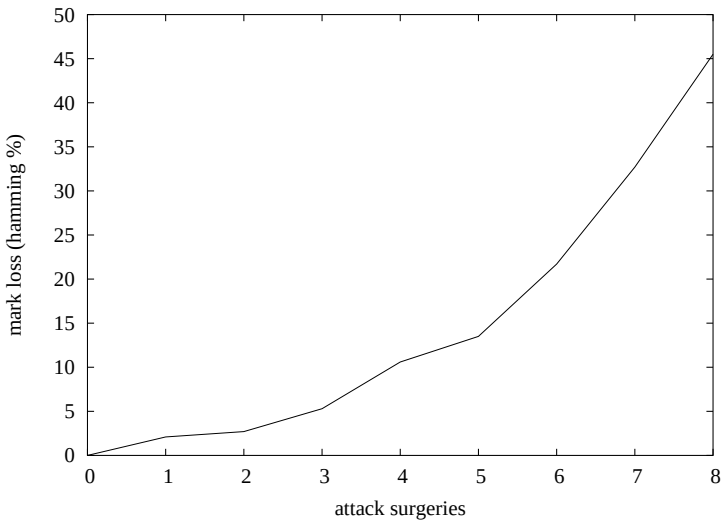


Figure 1.8. Introduction: Semi-structured Aggregates – Averaged watermark loss over 10 runs of an 8 bit watermark embedded into an arbitrary 32 node graph with 64 edges. Surgery attacks are applied randomly (node removals 60%, link addition 20%, link removal 20%). The labeling scheme was trained for 3 surgeries.

insignificant nodes (A5). Our solution is based on a canonical labeling algorithm that self-adjusts to the specifics of the content. Labeling is tolerant to a significant number of graph attacks ("surgeries") and relies on a complex "training" phase at embedding time in which it reaches an optimal stability point with respect to these attacks. We perform attack experiments on the introduced algorithms under different conditions with very encouraging results. In Figure 1.8 we show the watermark behavior to data alteration in the case of a random artificially generated structure with 32 nodes and 64 edges. The embedded watermark is 8 bits long. The labeling scheme was trained for 3 surgeries. As the number of attack surgeries increases, the watermark degrades slightly. The results are averaged over 10 runs on the same graph with different random attacks. When 8 attack surgeries are applied to the graph we can still recover $60 - 65\%$ of the watermark. One has to consider also the fact that an attacker is bound not to modify the structure beyond distortion limits.

1.3.6 Limits

The main desiderata and features of watermarking in a signal processing/multimedia framework, as outlined in [4] include: it should not degrade the perceived quality of the marked Work; the ability to detect the presence/content of a watermark should require the knowledge of a secret (key); different watermarks in the same Work should not interfere with each other; collusion attacks should not be possible; the watermark should survive any value-preserving transformation.

A common un-proved consensus has been implicitly assumed, namely that watermarking indeed lives up to its claimed features. [2, 3, 14] present excellent area surveys as well as comprehensive examples of algorithms for watermarking (mainly) multi-media Works. We know now that arbitrary large collusion attacks cannot be defeated against [15]. Moreover, while most watermarking algorithms prove to be safe against a considered set of value-preserving transformations (e.g., JPEG compression) they certainly fail with respect to many others. This shortcoming can be directly traced back to the relativity of the "value" and "quality" concepts. Several (mostly experimental) efforts explored the ability to analyze and quantify the "goodness" of watermarking applications, resulting in various watermark benchmarking "suites" mainly for multimedia (i.e., images). Additional research [16–18] aimed at analyzing concepts such as available bandwidth in the broader area of information hiding from a signal-processing, information-theoretic perspective, focusing mainly on various multimedia techniques. One particular question becomes of interest, namely: *Are there theoretically assessable bounds on watermark vulnerability with respect to an arbitrary watermarking method?* In other words, what is the inherent safety/vulnerability of a generic (i.e., with a minimum amount of assumptions, without considering implementation particularities) watermarking algorithm? An answer to this question might derive real-life recommendations for fine-tuning actual algorithms to increase their marking resilience.

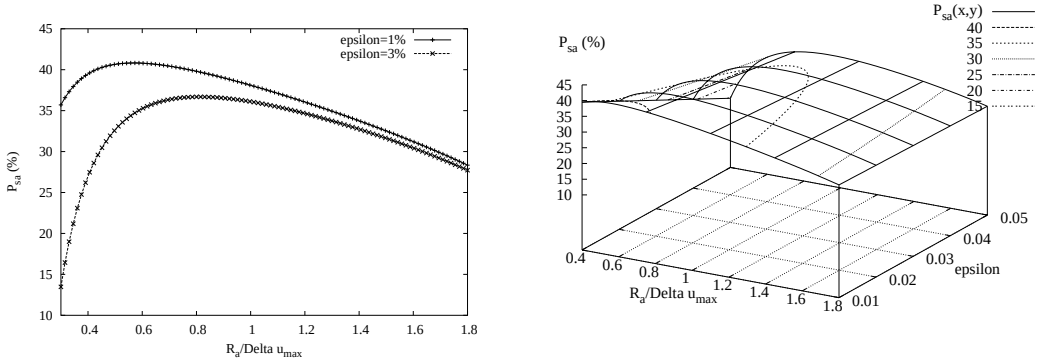


Figure 1.9. Introduction: Model of Watermarking – (a) No matter how sophisticated the watermarking method, there exists a random attack with a success probability of 33% and above (although we might not know what the attack is). It can be seen that a more court convincing ϵ_w value yields an even higher upper bound on attack success probability (2D cut through (b)). (b) The 3D evolution of the probability of a successful attack.

In Chapter 7 (and [6]) we explore these and other issues in the broader dissertation framework, for a broad class of watermarking algorithms. We use the previously introduced model to assess watermarking resilience and bounds. While we believe it generalizes to a much larger class of algorithms, the quantitative part of our analysis is done within a well-defined algorithmic class framework, namely our research in watermarking numeric relational data (see Chapter 3). We discover that indeed there exist such limitations. More specifically, we identify an important *convince-ability trade-off*: the more "convincing" in court a watermarking method is, the higher the probability of success of a perfect attack. We further derive the *watermarking optimality principle* that states that the vulnerability of a watermarking scheme (in our considered class) is likely minimized when it yields watermarked results on the boundary of the maximum allowable usability vicinity of the original un-watermarked Works.

From Mallory's perspective this is good news. It turns out that it *is* possible to defeat watermarking algorithms with a surprisingly high success rate, without any additional (insider's) knowledge. This is an inherent limitation of watermarking in general. Any additional knowledge can only improve on this probability. This is the case even if these algorithms conform to the optimality principle. Also, there seems to exist a "sweet spot" in which the probability of a successful attack is maximized. Mallory could make use of this by fine-tuning.

In summary, in this chapter we identify and analyze inherent limitations of watermarking, including the trade-off between two important watermarking properties: *being suitably "convincing" in court* while at the same time *surviving a set of attacks*. In the attempt to become as court convincing as possible, a watermarking application becomes more fragile to attacks aimed at removing the watermark, while preserving the value of the Work. It thus becomes necessarily characterized by a significant non-zero probability of being successfully attacked. We discovered an optimality principle (quantified and proved for a broad class of algorithms) that postulates the minimization of vulnerability in specific data points.

2 MODEL OF WATERMARKING (PART ONE)

The main purpose of Digital Watermarking is to protect a certain content from unauthorized duplication and distribution by enabling provable ownership over the content. It has traditionally [2] [3] [14] [19] relied upon the availability of a large noise domain within which the Work (data object) can be altered while retaining its essential properties. For example, the least significant bits of image pixels can be arbitrarily altered with little impact on the visual quality of the image (as perceived by a human). In fact, much of the "bandwidth" for inserting watermarks (such as in the least significant bits) is due to the inability of the human sensory system (especially sight and hearing) to detect certain changes.

Although a considerable amount of research effort has been invested in the problem of watermarking multimedia data (images, video and audio), there is relatively little work on watermarking other types of data such as text, software, and algorithms. Since these data types often have very well defined semantics and quality constraints, and may be designed for machine ingestion, the identification of the available "bandwidth" for watermarking is as important a challenge as the algorithms for inserting the watermarks themselves.

While a general background of watermarking has been provided above, in this chapter we are introducing a more formal model for watermarking for discrete data types. We propose, define and explore concepts such as "watermark", "watermarking algorithm", "watermarking attack", "usability spaces" etc. In Chapter 7 this model is reasoned about and important principles are derived.

2.1 Model and Definitions.

Let $\mathbb{D}$ be the domain of all possible Works to be considered for watermarking [1]. The value associated with such Works is owned by a given rights holder (possibly the Work creator). Information Hiding (for rights protection) tries to protect the association between the value carrying Work and the identity of its rights holder.

Considering any reasonable security assumptions and attacks, it becomes clear that a correct watermarking algorithm has to assure that the domain of all possible watermarked data Works (i.e., results from watermarking Works in $\mathbb{D}$) should be a subset of $\mathbb{D}$. For simplicity we assume that any considered algorithm produces watermarked Works only in $\mathbb{D}$ or that $\mathbb{D}$ is simply the union of all the closures over $\mathbb{D}$ of all resulting watermarked Works from considered algorithms.

More specifically, watermarking deploys *information hiding* techniques in order to embed a rights "witness" *within* the Work to be considered. The embedding process usually distorts/changes the original Work. The level of distortion introduced is usually measured within a model of tolerable change from a data consumer perspective. For example if the final data consumer is to be a human, and the data domain is digital multimedia, the Human Sensory System's limitations are defining allowable distortion bounds in the watermarking framework. If the Work is a JPEG image, the allowable distortion could be defined by the human eye limitations.

Works can exhibit different value levels when put to different uses. Thus we need a way to express the different associated values of Works, for different use case scenarios. We capture this insight through the concept of *usability domains*.

Usability Domain: A *usability domain* is defined as a set of functionals, $e = \{f | f : \mathbb{D} \rightarrow [0, 1]\}$, quantifying intrinsic data value in terms of its specific use. In a real world algorithm the considered usability domain is constructed by mapping real world properties to actual parameterized functionals in e (for example the set of Discrete Cosine Transform coefficients considered for alteration in a possible image

[1] For example in case of digital media Works we can simply assume that $\mathbb{D}$ is the set of all variable sized strings over $\mathbb{B} = \{0, 1\}$.

watermarking scenario). Also often $|e| = 1$, that is, a domain contains only one significant function of usability. Notation: Let the set of all usability domains be $\mathbb{U}$.

Usability: The *usability* metric of a Work $O \in \mathbb{D}$ corresponding to a domain $e \in \mathbb{U}$ ($e = \{f_1, f_2, ..., f_q\}$) is defined as $u(O, e)$ where $u : \mathbb{D} \times \mathbb{U} \rightarrow [0, 1]$. $u(O, e)$ is a combination of all the elements of e. For simplicity we will assume that $|e| = 1$. In this case we define $e = \{u\}$.

The concept of usability enables the definition of a certain threshold below which the Work is not usable anymore in the given domain. In other words, it "lost its value" to an unacceptable degree. The notion of usability is related to *distortion*. A highly distorted Work (e.g., as result of watermark embedding or attacks) will likely suffer a drop in its distortion domain usability. For simplicity, in the following we consider a single usability domain $e \in \mathbb{U}$, unless otherwise specified.

In real life settings each usability domain is characterized uniquely by the set of alterable data items (or degrees of freedom) that a Work features for the purpose of watermarking. For example in the case of image frequency transform encodings such as the DCT transform [2], the main usability domain (associated with the human sensory system) is uniquely identified by the set of DCT coefficients considered in the watermarking process. This defines a multi-dimensional usability "space" in which each point uniquely identifies a Work in $\mathbb{D}$ and watermarking becomes a simple translation from a point O to the watermarked version O' (see Figure 2.1 (a)).

Change in Usability: The *difference in usability* is defined as $\Delta u : \mathbb{D} \times \mathbb{D} \times \mathbb{U} \rightarrow [-1, 1]$, where $\Delta u(O_1, O_2, e) = u(O_1, e) - u(O_2, e)$.

Usability Vicinity: Let $v \in \mathbb{U}$ be a usability domain and a maximum allowed change in usability Δu_{max}. We say that element $X \in \mathbb{D}$ is *in the radius* Δu_{max} *usability vicinity of* $O \in \mathbb{D}$ *with respect to* v [2] iff $\Delta u(O, X, v) < \Delta u_{max}$. The concept of a usability vicinity is basically expressing how far can one go in altering the considered Work in the process of watermarking.

[2] And vice-versa. It is commutative.

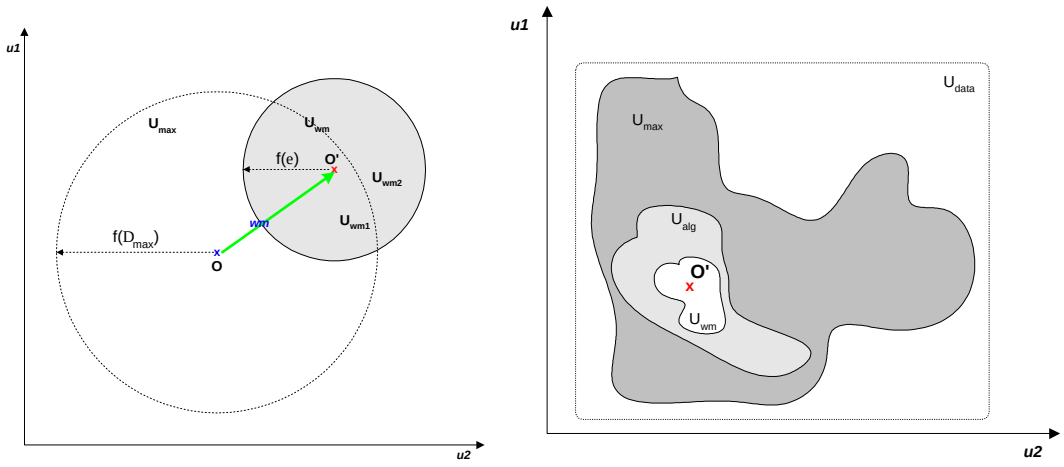


Figure 2.1. Model of Watermarking: (a) A 2-dimensional view of a usability space. A point uniquely identifies a Work in $\mathbb{D}$ (e.g., coordinates in this space are DCT coefficients considered for watermark embedding). Watermarking results in a point O', a "watermarked" version of O and is naturally represented as a transform in the usability space. (b) Usability vicinities of a certain Work $O \in \mathbb{D}$ for a given marking algorithm. U_{data} is defined by the actual data type of the usability metrics. U_{max} is the maximal allowable usability vicinity with respect to the associated usability domain(s) (e.g., Human Visual System). The results (U_{alg}) of a valid marking algorithm with respect to a given Work and all other possible inputs should be contained within the maximal allowable usability vicinity (U_{max}) of the Work. U_{wm} is determined by ϵ_w.

Note that the usability vicinity of a certain Work $O \in \mathbb{D}$ with respect to a considered usability domain defines actually the set of possible watermarked versions of O with respect to it and Δu_{max}. In the usability space in Figure 2.1 (a) this vicinity translates into an shape "around" the original Work O, labeled U_{max}. Thus, the Work and any minor altered version of it (within usability vicinity bounds) can be uniquely identified by its locality within the considered usability domain (i.e., by its usability "coordinates").

Watermark: Given an un-marked Work $O \in \mathbb{D}$ and the considered watermarked version of it, $O' \in \mathbb{D}$, where O' is within radius Δu_{max} usability vicinity

of O, a key $k \in \mathbb{K}$ [3] , a usability domain $v \in \mathbb{U}$ and a maximum allowed change in usability Δu_{max}, a *watermark* can be asserted by a special property functional $w : \mathbb{D} \times \mathbb{K} \rightarrow \mathbb{B}$, defined by the following: $w(O, k) = 0$, $w(O', k) = 1$ and there exists $\epsilon_w \in (0, 1)$ such that we have

$$P(w(x, k) = 1, \forall x \in \mathbb{D}, x \neq O', x \in U_{max} | w(O', k) = 1) < \epsilon_w$$

Notation: Let $\mathbb{W}_{\mathbb{D}}$ be the set of all w over a given $\mathbb{D}$.

In plain words, a watermark is defined as a special, induced (through watermarking) property (w) of a watermarked Work $O' \in \mathbb{D}$, so rare, that if we consider any other Work $x \in \mathbb{D}$, "close-enough" to the original Work O, the probability that x exhibits the same property can be upper-bounded. This is derived from the requirement to be enough "court-convincing" by bounding the rate of false-positives.

In Figure 2.1 (a) this probability maps into an area "around" the watermarked Work O', labeled U_{wm}, zone in which the points directly correspond to Works that exhibit the watermark property (detection area). Of particular interest then becomes the intersection between U_{max} and U_{wm}, an area bounded by the watermarking construction (i.e., ϵ_w). This area defines the Works that are both valuable/usable (with respect to the original Work) and watermarked. Intuitively, one main challenge of watermarking becomes to find/derive O' such that, given only O' it will be very hard for an attacker (e.g., Mallory) to determine an $x \neq O'$ inside the usability vicinity of O, for which $w(x, k) = 0$.

In real life, w is likely a combination of the actual watermark detection method and any and all encoding parameters and secret keys required to apply the method on the watermarked result to yield the embedded watermark (see Figure 2.2).

Note: Throughout this chapter, for illustrative purposes, the usability spaces are discussed in a 2-dimensional context. Real world usability spaces are multi-dimensional (e.g., $O(n^2)$ in case of DCT encoding with a DCT matrix of size $n \times n$). We also assume a continuous nature of the usability vicinity shapes and propose extensions later on, in Chapter 7.

[3] Where $\mathbb{K}$ is the set of secrets (i.e., keys) involved in the watermarking process.

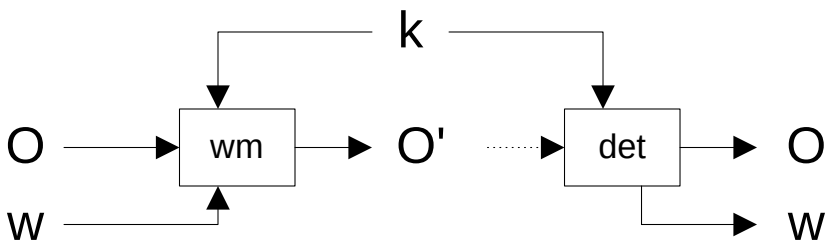


Figure 2.2. Model of Watermarking: In actuality, (symmetric) watermarking is based on the use of a common secret (key) k shared between the encoding and detection (e.g., in court) phases.

Algorithm: A *watermarking algorithm* can be defined as a functional $a : \mathbb{D} \times \mathbb{K} \rightarrow \mathbb{D} \times \mathbb{W}$, which, given as input an Work $O \in \mathbb{D}$ provides a watermarked version of the Work, O', and a property functional w that enables watermark detection. Notation: Let $\mathbb{A}_{\mathbb{D}}$ be the set of all a over a given $\mathbb{D}$.

In other words, a watermarking algorithm alters the original Work to produce its watermarked version. It also provides the required ability to recover the watermark (i.e., through w). Naturally, for a watermarking algorithm to be valid, its results, with respect to a given Work and all other possible inputs should be contained within the maximal allowable usability vicinity (U_{max}) of this Work, see Figure 2.1 (b).

Attack: Given a watermarking algorithm $a \in \mathbb{A}_{\mathbb{D}}$, a Work $O \in \mathbb{D}$ and its watermarked version $O' \in \mathbb{D}$, Δu_{max}, $\forall k \in \mathbb{K}$, a *successful watermarking removal attack* is defined by $z : \mathbb{D} \rightarrow \mathbb{D}$ such that $\Delta u(z(O'), O) < \Delta u_{max}$ and $w(z(O'), k) = 0$.

In other words, an attack tries to maintain the attacked watermarked Work within the usability vicinity of the original non-watermarked version, while removing the watermark [4]. Notation: Let $\mathbb{Z}_w$ be the set of all attacks z for a given w.

Vulnerability: It is certainly desirable to be able to quantify the ability of a given attack to succeed against a watermarking scheme. This can be used as a metric of "goodness" of a specific scheme. The probability that a *particular* attack

[4]We focus on the arguably most common class of attacks, having a final goal of removing the watermarking information while preserving the value.

succeeds is not easy to compute as it depends on a variety of parameters, including the potentially infinite set of input watermarked Works. Instead of focusing on specific algorithms, one solution is to assess bounds for a generic class.

We define the *single point vulnerability* of a certain watermarking method with respect to a given watermarked Work as the lower bound on the probability of success (e.g., watermark removal, resulting Work still "usable") of an attacker with no additional knowledge but the actual watermarked Work itself. That is, given a watermarked Work, this is the probability that an un-informed attack succeeds to remove the watermark and produce a result within the usability vicinity (U_{max}) of the original Work. It turns out that for a broad class of algorithms this can be quantified exactly. It provides a natural measure of watermark fragility.

We define the *algorithm vulnerability* of a certain watermarking method as the average of the single point vulnerability over the entire input space. Although this value presents a great importance in assessing the quality of a watermarking method, it is not trivial to compute, especially in cases with infinite input spaces. We can define a *k-sampling approximation* of it that can be determined by taking k random samples (i.e., Works to be watermarked) from the input space and computing the algorithm vulnerability over this limited subset.

Throughout this dissertation we often explore single point vulnerability through P_{sa}, the probability of an attack to succeed, given as input only the watermarked Work.

2.2 Consumer Driven Watermarking

Watermarking works by deploying resilient information hiding techniques to insert an indelible mark in the data such that (i) the insertion of the mark does not destroy the value of the data (i.e., the data is still useful for the *intended purpose*); and (ii) it is difficult for an adversary to remove or alter the mark beyond detection without destroying the value of the data. Clearly, the notion of value or utility of

the data is central to the watermarking process. This value is closely related to the type of data and its intended use. For example, in the case of software the value may be in ensuring equivalent computation, and for text it may be in conveying the same meaning (i.e., synonym substitution is acceptable). Similarly, for a collection of numbers, the utility of the data may lie in the actual values, in the relative values of the numbers, or in the distribution (e.g., normal with a certain mean).

An important point about watermarking should be noted. By its very nature, a watermark modifies the item being watermarked. If the data to be watermarked cannot be modified then a watermark cannot be inserted. The critical issue is not to avoid changing the data, but to limit the change to acceptable levels with respect to the intended use of the data.

Clearly, one can always identify some use of the data that is affected by even a minor change to the any portion of the data. It is therefore necessary that the intended purpose of the data that should be preserved be identified during watermarking. Thus there exists a trade-off between the desired level of marking resilience and resistance to attacks, and the ability to preserve data quality in the result (with respect to the original). We believe it is important that watermarking-related alterations to not interfere with known quality constraints (or other advertised guaranteed properties of the result). For a watermarking solution to be sound, it has to operate within these boundaries. It has to become aware and accommodate such a quality guarantee principle.

At the same time, the concept of value of resulting Works, is necessarily relative and largely influenced by each semantic context it appears in. For example, while a statistical analyst would be satisfied with a set of feature summarizations (e.g., average, higher-level moments) of a numeric data set Work, a data mining application may need a majority of the data items, for example to validate a classification hypothesis. On the other hand, intuitively, to one extreme, if the embedded watermark is to be very "strong" one can simply modify the entire Work such that a majority of sub-segments feature the watermark, but at the same time probably also destroying

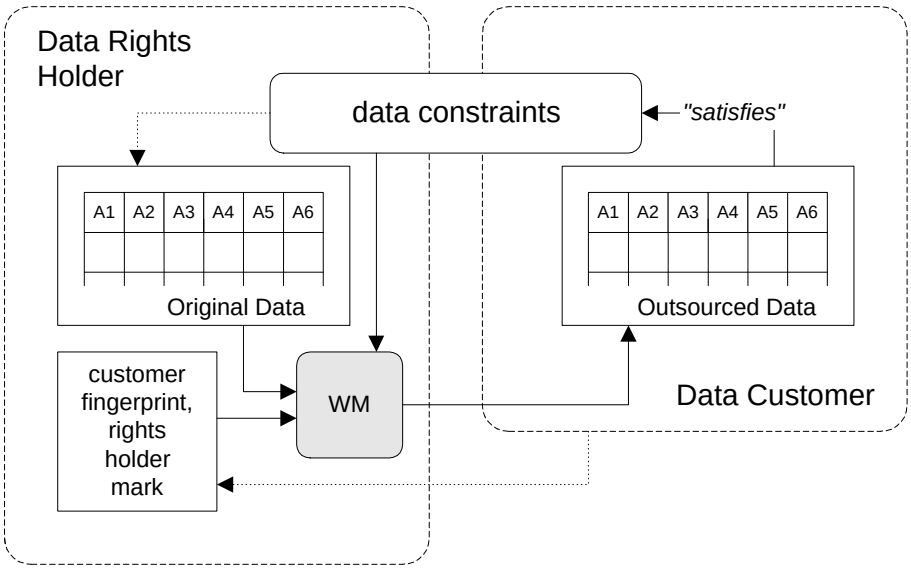


Figure 2.3. Model of Watermarking: In consumer-driven watermarking a set of data constraints are continuously evaluated in the encoding process to ensure quality of the result.

its actual value. If *any* alterations are allowed to the data, the available bandwidth is directly related to data entropy. As data quality requirements become increasingly restrictive, any applied watermark is necessarily more vulnerable. Often we can express the available bandwidth as an increasing function of allowed alterations. At the other extreme, a disproportionate concern with data quality will hinder most of the watermarking alterations, resulting in a weak, possibly non-existent embedding.

Thus, the process of watermarking can be expressed metaphorically as a game between the watermarker and Mallory. In this game, the watermarker and Mallory play against each other within subtle trade-off rules aimed at keeping the quality of the result within acceptable bounds. It is as if there exists an impartial referee (the data itself) moderating each and every "move". This is why we propose to make this "referee" an explicit part of the marking process. We call this paradigm *consumer driven watermarking*.

The solutions discussed in Chapters 3, 4 and 5 are all consumer driven enabled through feedback mechanisms (see Figure 2.3) that allow the watermarking process to "rollback" modifications that would violate quality constraints in the result on a step by step basis. This ensures the preservation of desired quality metrics with respect to the original un-watermarked input Work.

2.3 Steganography and Watermarking

At this point it might be helpful to outline that there exists a fundamental difference between watermarking and generic information hiding (steganography) from an application perspective (and associated challenges). Information hiding in general (covert communication in particular), aims to enable Alice and Bob to exchange messages in a manner as resilient and stealthy as possible, through a medium controlled by evil Mallory. On the other hand, digital watermarking (especially for rights assessment) is deployed by Alice to prove rights over a piece of data, to Jared the Judge, usually in the case when Tim the Thief [5] benefits from using/selling that very same piece of data or maliciously modified versions of it.

In digital watermarking, the actual value to be protected lies in the Works themselves whereas pure steganography usually makes use of them as simple value "transporters". Rights assessment can be achieved by demonstrating that a particular Work exhibits a rare property (read "hidden message" or "watermark"), usually known only to Alice (with the aid of a "secret" - read "watermarking key"). For court convince-ability purposes this property needs to be so rare that if one considers any other random Work "similar enough" to the one in question, this property is "very improbable" to apply (i.e., bound on false-positives). It also has to be relevant, in that it somehow ties to Alice (e.g., by featuring a bit string saying "(c) by Alice").

There is a threshold determining Jared's convince-ability related to the "very improbable" assessment. This defines a main difference from steganography: from

[5] Tim's middle name is *Mallory*.

Jared's perspective, specifics of the property (e.g., watermark message) are irrelevant as long as they link to Alice (e.g., by saying "(c) by Alice") and, she can prove "convincingly" it is she who embedded/induced it to the (non-watermarked) original.

It is to be stressed here this particularity of watermarking in our model. In watermarking the emphasis is on "detection" rather than "extraction". Extraction of a watermark (or bits of it) is usually a part of the detection process but just complements the process up to the extent of increasing the ability to convince in court. If recovering the watermark data in itself becomes more important than detecting the actual existence of it (i.e., "yes/no" answer) then (from an application point of view) this is a drift toward covert communication and pure information hiding (steganography).

2.4 Notations and Primitives

Throughout this dissertation we repeatedly use a set of notations and security primitives. In the following we summarize some of the more frequent ones. For any value (e.g., numeric) x let $b(x)$ be the number of bits required for its accurate representation and $msb(x, b)$ its most significant b bits. If $b(x) < b$ we left-pad x with $(b - b(x))$ zeroes to form a b-bit result. Similarly, $lsb(x, b)$ is used to denote the least significant b bits of x. If by wm we denote a watermark to be embedded, $wm[i]$ will then be the i-th bit of wm. Let $set\_bit(d, a, b)$ be a function that returns value d with the bit position a set to the truth-value of b. In any following mathematical expression let the symbol '&" signify a *bit-AND* operation. A special de-facto secure construct we are leveraging is the one-way cryptographic hash. If $crypto\_hash()$ is a cryptographic secure one-way hash, of interest are two of its properties: (i) it is computationally infeasible, for a given value V' to find a V such that $crypto\_hash(V) = V'$ (one-wayness), and (ii) changing even one bit of the hash input causes random changes to the output bits (i.e., roughly half of them change even if one bit of the input is flipped). Examples of potential candidates for $crypto\_hash()$

are the MD5 (used in the proof of concept implementation) or SHA hash. For more details on cryptographic hashes consult [20]. Let $H(V,k) = crypto\_hash(k;V;k)$ (where ";" denotes concatenation). We use crypto hashes in several scenarios, e.g., in fighting court-time exhaustive key search claims in Chapter 4.

2.5 Conclusions

In this chapter we proposed a model for watermarking discrete data types. We defined and explored associated constructs, e.g., "watermark", "watermarking algorithm", "watermarking attack", "usability spaces" etc. This served two purposes: (i) establishing a clear understanding of what watermarking is for the purpose of the following chapters in which different discrete data types are explored, and (ii), setting out the premises for Chapter 7 in which this model is reasoned about and associated principles and challenges of watermarking are derived and analyzed. Published research results of this work include [5].

3 RELATIONAL DATA WITH NUMERIC TYPES

In this chapter we introduce a solution for numeric relational data rights protection through watermarking.

3.1 Introduction

A majority of current database management systems are based on a relational data model [21]. A relational database is a data storage system in which relations between information items are explicitly specified. In such a model data is organized as "a number of differently sized *tables*" composed of "related" rows/columns. Thus, a table is a collection of *rows* or records and each row in a table contains the same *fields*. Certain fields may be designated as data *keys* (not to be confused with "cryptographic keys" – secrets usually aiding in security-related protocols and frameworks), which means that searches for specific values of that field will possibly deploy indexing to speed things up. Data is structured logically into valued *attributes*. From this perspective, a table is a collection of such attributes (the columns of the table) and models a *relation* among them. The data rows in the tables are also called *tuples*. Data in this model is manipulated using a *relational algebra*. Main operations in this algebra are set operations (e.g., union, intersection, Cartesian product), selection (of some tuples in tables) and projection (of some columns/attributes).

Rights protection for such data is important in scenarios where it is sensitive and valuable and about to be outsourced. A good example is a data mining application, where data is sold in pieces to parties specialized in mining it (e.g., sales patterns database, oil drilling data, financial data). Other scenarios involve for example online B2B interactions (e.g., airline reservation and scheduling portals) in which data is made available for direct, interactive use (see Figure 3.1). Given the nature of most

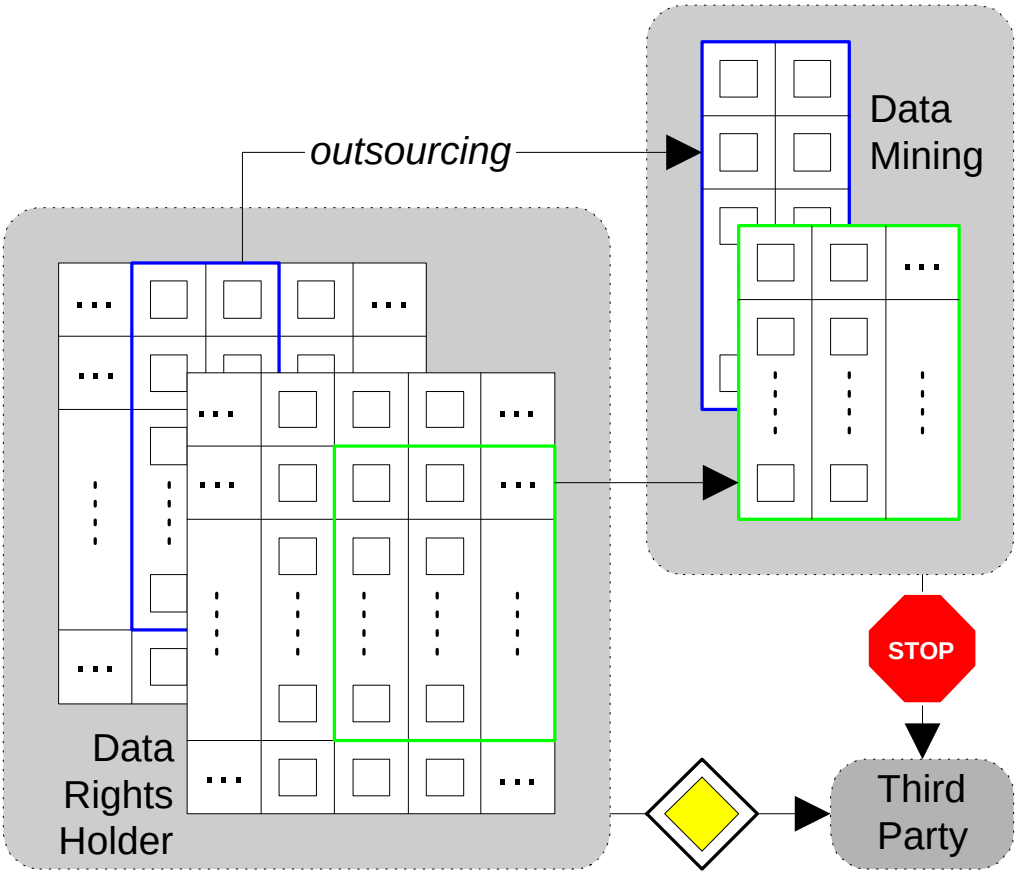


Figure 3.1. Relational Data: Rights assessment is important when valuable data is outsourced to a third party.

of the data, it is hard to associate rights of the originator over it. Watermarking can be used to solve this issue.

While extensive efforts have focused on various aspects of DBMS security, including access issues [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33], little has been done to enable the ability to assert rights over outsourced data. In this chapter we explore the issue of securing valuable outsourced numeric relational data through watermarking, enabling future court-proofs assessing proper rights over the content.

Our solution starts by receiving as user input a reference to the relational data to be protected, a watermark to be embedded as a copyright proof, a secret key used to protect the embedding and a set of data quality constraints to be preserved in the result. It then proceeds to watermark the data while continuously assessing data quality, potentially backtracking and rolling back undesirable alterations that do not preserve data quality. Watermark embedding is composed of two main parts: in the first stage, the input data set is securely partitioned into subsets of items; the second stage then encodes one bit of the watermark into each subset. If more subsets (than watermark bits) are available, error correction is deployed to result in an increasingly resilient embedding. The algorithms introduced here prove to be resilient to important classes of attacks, including subset selection, linear data changes and random item(s) alterations.

The chapter is structured as follows. Section 3.2 discusses the main challenges for watermarking relational databases. Section 3.3 introduces an initial idea to a primitive problem (watermarking numeric collections) to be used later in the global algorithm. Section 3.4 constructs a solution for relational databases by building upon the primitive building block introduced earlier. Various issues and algorithm extensions are discussed in Section 3.5. Section 3.6 presents implementation details as well as experiments and evaluations of the proposed watermarking technique on real outsourced Wal-mart warehouse data. Section 3.8 concludes.

3.2 Challenges

While research related to the issue of embedding information into a set of numbers [34], can be found (sometimes implicitly) in different frameworks, associated with various information hiding techniques (e.g., frequency domain embedding, DCT and Wavelet watermarking [2]), relational data presents a different set of challenges and associated constraints. These challenges are novel and directly related to the specifics of the domain, namely large sets of items organized in a relational frame-

work, with associated semantics to be preserved. This is not the case for multimedia (mostly time-series type of) data, where semantics are associated with the data stream only at a much higher composite level. For example in a multi-megabit audio channel of news broadcast, the semantics to be protected are likely to be in the broadcast speech text rather than directly in the underlying audio stream bits; thus a fundamentally different and broader noise band becomes available for watermark embedding, and with it different (possibly less accurate) encoding and evaluation methods. By contrast, the low noise bandwidth of major relational framework data uses (e.g., data-mining) require a different approach, taking a more careful look at the actual tolerated changes on the given data.

Whereas in the multimedia case, the data quality model is usually at best fuzzy, because of the relativity of any model of human perception, one solution here is to define the noise channel explicitly as part of the watermarking solution, in terms of required customer constraints to be preserved on the final data. At watermarking time, data quality can be continuously assessed as an intrinsic part of the marking algorithm in itself. In this respect we can claim that, as opposed to other watermarking algorithms in various domains (e.g., image watermarking), we maintain 100% of the associated data value with respect to a set of given required data "goodness" constraints. We believe this is an essential part of any watermarking application in this low-noise, high fragility domain of relational data, especially considering data-mining issues, such as classification and JOIN results preservation (see Section 3.6.2).

Additionally, the watermark *encoding* method needs to feature a design suited to the new constraints, namely the ability to survive a maximum level of attacks and at the same time accommodate the existence of required data "usability" conditions to be satisfied by the result. Our algorithm, deploying means for data distribution manipulation, encoding the actual information in distribution properties of the data rather than directly into the data itself, is best suited for its purpose, and almost optimally so. For, while allowing an adjustable degree of freedom in alteration

points selection, it provides at the same time a surprisingly high level of resilience as evidenced by our extensive validation experiments (see Section 3.6.2).

3.2.1 Available Bandwidth

An important first step in inserting a watermark into a relational database (and thereby altering it), is to identify changes that are acceptable. As was mentioned earlier, the acceptable nature and level of change is dependent upon the application for which the data is to be used.

With respect to particular data uses and metrics of quality, it is of utmost importance that *the watermarking process not interfere with the final data consumer requirements*. This is why these requirements need to be considered as an integral part of the watermarking process, providing a feedback loop, in assessing the quality of the final result (see consumer driven watermarking, Section 2.2).

In the following we define a functionality that will enable us to determine the watermarking result as being valuable and valid, within permitted/guaranteed error bounds. The available "bandwidth" for inserting the bits of the watermark text is therefore not defined directly. Instead we define allowable distortion bounds for the input data in terms of consumer-defined metrics. If the watermarked data satisfies the metrics, then the insertion of the watermark is considered to be successful. This quality assessment mechanism is part of the marking process.

Example One simple but relevant example is the *maximum allowable mean squared error* (MSE) case, in which the usability metrics are defined in terms of mean squared error tolerances as

$$(s_i - v_i)^2 < t_i, \forall i = 1, ..., n$$

and

$$\sum (s_i - v_i)^2 < t_{max}$$

where $\mathbb{S} = \{s_1, ..., s_n\} \subset \mathbb{R}$, is the data to be watermarked, $\mathbb{V} = \{v_1, ..., v_n\}$ is the result, $\mathbb{T} = \{t_1, ..., t_n\} \subset \mathbb{R}$ and $t_{max} \in \mathbb{R}$ define the guaranteed error bounds at data

distribution time. In other words $\mathbb{T}$ defines the allowable distortions for individual elements in terms of MSE and t_{max} its overall permissible value.

Database Semantics Specifying only allowable change limits on individual values and possibly an overall limit, fails to capture important semantic features associated with the data – especially if the data is structured. Consider for example, age data. While a small change to the age values may be acceptable, it may be critical that individuals that are younger than 21 remain so even after watermarking if the data will be used to determine behavior patterns for under-age drinking. Similarly, if the same data were to be used for identifying legal voters, the cut-off would be 18 years. Further still, for some other application it may be important that the relative ages (in terms of which one is younger) not change. Other examples of constraints include: (i) *uniqueness* – each value must be unique; (ii) *scale* – the ratio between any two number before and after the change must remain the same; and (iii) *classification* – the objects must remain in the same class (defined by a range of values) before and after the watermarking. As is clear from the above examples, simple bounds on the change of numerical values are often not enough.

Structured Data Structured collections, for example a collection of relations, present further constraints that must be adhered to by the watermarking algorithm. Consider a data warehouse organized using a standard Star schema with a fact table and several dimension tables. It is important that the key relationships be preserved by the watermarking algorithm. This is similar to the "Cascade on update" option for foreign keys in SQL and ensures that tuples that join before watermarking also join after watermarking. This requires that the new value for any attribute should be unique after the watermarking process. In other words, we want to preserve the relationship between the various tables. More generally, the relationship could be expressed in terms of an arbitrary join condition, not just a natural join. In addition to relationships between tuples, relational data may have constraints within tuples. For example, if a relation contains the start and end times of a web interaction, it

is important that each tuple satisfies the condition that the end time be later than the start time.

Also, an adversary attempting to destroy a watermark becomes much more effective if he can identify the values in which the watermark has been embedded. In addition to specifying properties of the data that should be preserved for usability, constraints can be used to prevent easy detection of watermark locations. For example a tuple with start time later than its corresponding end time, or a customer with age less than 12 years are very likely to be detected as resulting from watermarking.

3.2.2 Model of the Adversary

In order to be effective, the watermarking technique must be able to survive a wide variety of attacks. These attacks may be malicious with the explicit intent of removing the watermark, or may be the result of normal use of the data by the intended user.

A1. Subset Selection The attacker (Mallory) can randomly select and use a subset of the original data set that might still provide value for its intended purpose (subtractive attack).

A2. Subset Addition Mallory adds a set of numbers to the original set. This addition is not to significantly alter the useful (from the Mallory's perspective) properties of the initial set versus the resulting set.

A3. Subset Alteration Altering a subset of the items in the original data set such that there is still value associated with the resulting set. A special case needs to be outlined here, namely (**A3.a**) a linear transformation performed uniformly to all of the items. This is of particular interest as such a transformation preserves many data-mining related properties of the data, while actually altering it considerably, making it necessary to provide resilience against it.

Given the attacks above, several properties of a successful solution surface. For immunity against **A1**, the watermark has to be embedded in overall collection prop-

erties that survive subset selection (e.g., confidence intervals). If the assumption is made that the attack alterations do not destroy the value of the data, then **A3** should be defeat-able by embedding the primitive mark in resilient global data properties. As a special case, **A3.a** can be defeated by a preliminary normalization step in which a common divider to all the items is first identified and applied. For a given item X, for notation purposes we are going to denote this "normalized" version of it by $NORM(X)$. Since it adds new data to the set, defeating **A2** seems to be the most difficult task, as it implies the ability to identify potential uses of the data (that Mallory could benefit from).

3.3 Simplified Problem: Numeric Collections

This section deals with the foundations of a primitive numeric collection watermarking procedure that will be later deployed as a sub-routine in the main watermarking algorithm. Section 3.4 evolves this building block into a complete solution in the relational framework.

Let $\mathbb{S}$ be a set of n real numbers $\mathbb{S} = \{s_1, ..., s_n\} \subset \mathbb{R}$. Then, the **general simplified problem** of watermarking the set $\mathbb{S}$ can be defined as the problem of *finding a transformation from* $\mathbb{S}$ *to another item set* $\mathbb{V}$*, such that, given all possible imposed usability metrics sets* $\mathbb{G} = \cup G_i$ *for any and all subsets* $S_i \subset \mathbb{S}$*, that hold for* $\mathbb{S}$*, then, after the transformation yields* $\mathbb{V}$*, the metrics should hold also for* $\mathbb{V}$ [1]. We call $\mathbb{V}$ the "watermarked" version of $\mathbb{S}$. Thus $\mathbb{V} = \{v_1, ..., v_n\} \subset \mathbb{R}$ is the result of watermarking $\mathbb{S}$ by minor alterations to its content.

But how much of a change is to be allowed to the content? For a numeric collection, a natural starting point for defining the allowed change is to specify an absolute (or relative) change in value. For example, each value may be altered by no more than 0.0005 or 0.02%. Moreover a bound on the cummulative change may be specified. Our solution for the *simplified problem* consists of several steps.

[1] In other words, if $\mathbb{G}$ is given and holds for the initial input data, $\mathbb{S}$, then $\mathbb{G}$ should also hold for the resulting data $\mathbb{V}$.

First, we deploy a resilient method for item labeling, enabling the required ability to "recognize" initial items at watermarking detection time (i.e., after watermarking and/or attacks). In the next step we ensure attack survivability by "amplifying" the power of a given primitive watermarking method. The amplification effect is achieved by deploying secrets in the process of selecting the subsets to become input for the final stage, in which a primitive encoding method is deployed.

3.3.1 Solution Summary

A summary of the solution for the simplified problem reads as follows.

Encoding Phase: (E.1) Select a maximal number of unique, non-intersecting (see below) subsets of the original set, using a set of secrets, as described in Section 3.3.3. **(E.2)** For each considered subset, (E.2.1) embed a watermark bit into it using the encoding convention in Section 3.3.3 and (E.2.2) check for data usability bounds. If usability bounds are exceeded, (E.2.3) retry different encoding parameter variations or, if still no success, (E.2.3a) try to mark the subset as invalid (i.e., see encoding convention in Section 3.3.3), or if still no success (E.2.4) ignore the current set. [2] We repeat step E.2 until no more subsets are available for encoding. This results in multiple embeddings in the data.

Decoding Phase: (D.1) Using the secrets from step E.1, recover a majority of the subsets considered in E.1, (or all if no attacks were performed on the data). **(D.2)** For each considered subset, using the encoding convention in Section 3.3.3, recover the embedded bit value and re-construct watermarks. **(D.3)** The result of D.2 is a set of copies of the same watermark with various potential errors. This last step uses a set of error correcting mechanisms (e.g., majority voting schemes) to recover the highest likelihood initial mark.

[2] This leaves an invalid watermark bit encoded in the data that will be corrected by the deployed error correcting mechanisms (e.g., majority voting) at extraction time.

3.3.2 Selecting Subsets

Watermarking a collection of data items requires the ability to "recognize" (i.e., re-discover, at detection time) *most* of the items before and after watermarking and/or a security attack. In other words if an item was accessed/modified before watermarking, e.g., being identified with a certain label L, then hopefully at watermark detection time the same item is identified with the same label L or a known mapping to the new label. More generally, we would like to be able to identify *a majority of the initial elements of a subset* after watermarking and/or attacks. As we will see, our technique is resilient to "missing" a small number of items. For more details see 3.3.4.

Our solution is based on lexicographically sorting the items in the collection, sorting occurring based on a one-way, secretly keyed, cryptographic hash of the set of most significant bits (MSB) of the normalized (see Section 3.2.2) version of the items. The secret one-way hashing ensures that Mallory cannot possibly determine the ordering. In the next step (see Section 3.3.3), subset "chunks" of the items are selected based on this secret ordering. Chunk-boundaries ("subset markers") are then computed and stored for detection time (for a more in-depth discussion of subset markers see Section 3.4).

More formally, given a collection of items as above, $\mathbb{S} = \{s_1, ..., s_n\} \subset \mathbb{R}$, and a secret "sorting key" k_s, we induce a secret ordering on it by sorting according to a cryptographic keyed hash of the most significant bits of the normalized items. Thus we have: $index(s_i) = H(k_s, MSB(NORM(s_i)), k_s)$. The MSB space here is assumed to be a domain where minor changes on the collection items (changes that still satisfy the given required usability metrics) have a minimal impact on the MSB labels. This is true in many cases (as usually the usability metrics are related to preserving the "important" parts of the original data). If not suitable, a different labeling space can be envisioned, one where, as above, minor changes on the collection items have a minimal impact.

Note: In the relational data framework, the existence of a primary key associated with the given attribute to be watermarked can make it easier to impose a secret sorting. For more details see Section 3.4.

3.3.3 Amplifying Watermark Power

Current watermarking algorithms draw most of their court-persuasion power from a secret that controlled watermark embedding (i.e., watermarking key). Much of the attack immunity associated with a watermarking algorithm is based on this key and its level of secrecy. Given a weak partial marking technique (e.g., (re)setting a bit), a strong marking method can be derived by a method of "mark amplification" – repeatedly applying the weak technique in a keyed fashion on different parts of the data being watermarked.

Generic Solution. Let $\mathbb{K} = \{k_1, ..., k_m\}$ be a set of m keys of n bits each. We define $S_i = \{s_j \in \mathbb{S} | (k_i)_{bitj} = 1\}, i = 1, ..., m$. In other words each $S_i \subset \mathbb{S}$ is defined by selecting a subset of $\mathbb{S}$ fully determined by its corresponding key $k_i \in \mathbb{K}$.

The main purpose of this step is to amplify the power of the general watermark. The next step will simply consider each S_i to be marked separately by building on a simple watermarking method. The result will be at least an m-bit (i.e., $i = 1, ..., m$) overall watermark bandwidth (unless we consider multiple embeddings and majority voting, for error correcting purposes) in which each bit is embedded in each of the marked S_i (bit $w[i]$ of the watermark is embedded into S_i).

Note: In building the subsets S_i we do not consider elements s_j which are subject to very restrictive usability metrics (e.g., with corresponding $t_j = 0$, i.e., no available encoding bandwidth). If any of the considered keys are selecting one of those unalterable elements, we simply generate another key instead.

We presented the generic solution above for illustrative purposes. It works well for cases when exact item labeling is available and there are no concerns of attacks

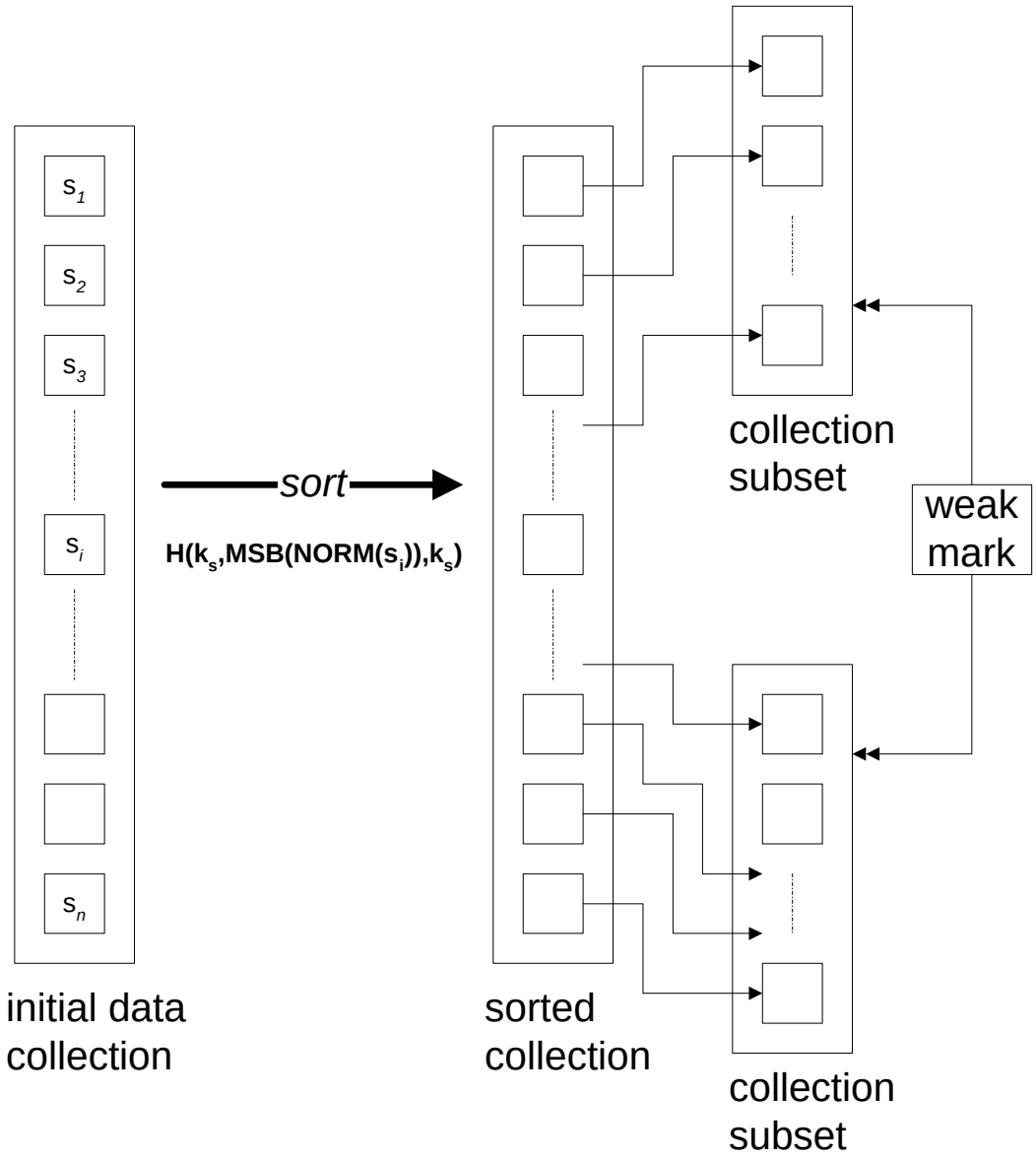


Figure 3.2. Relational Data with Numeric Types: Primitive Mark Power Amplification. Subset selection after sorting on keyed hash of the most significant bits (MSB) of the normalized data items. This enables recovery after various attacks, including re-shuffling/sorting and linear changes. The secrecy of the subsets to which the weak(er) encoding is applied provides a resilience amplification effect.

of the types **A2** and **A1** (i.e., subset addition, selection). The following idea takes also into account these concerns.

Actual Solution. Given a collection of items as above, $\mathbb{S} = \{s_1, ..., s_n\} \subset \mathbb{R}$, and a secret "sorting key" k_s, we first induce a secret ordering on it by sorting according to a cryptographic keyed hash of the most significant bits of the normalized items, e.g., $index(s_i) = H(k_s, MSB(NORM(s_i)), k_s)$. We then build the subsets, S_i, as "chunks" of items, a "chunk" being a set of adjacent items in the sorted version of the collection. This increases the ability to defeat different types of attacks including "cut" and/or "add" attacks (e.g., **A1** , **A2**), by "dispersing" their effect throughout the data, as a result of the secret ordering. Thus, if an attack removes 5% of the items, this will result in each subset S_i being roughly 5% smaller. If S_i is small enough and/or if the primitive watermarking method used to encode parts of the watermark (i.e., 1 bit) in S_i is made resilient to these kind of minor transformations (see Section 3.6.2) then the probability of survival of most of the embedded watermarks is accordingly higher (see Section 3.3.4). Additionally, in order to provide resilience to massive "cut" attacks, we will select the subset "chunks" to be of sizes equal to a given percent of the overall data set (i.e., not of fixed absolute sizes). This choice provides adaptability of our subset selection scheme to such attacks, assuring subsequent retrieval of the watermark even from, say, half of the original data. Thus, the main purpose of this step is to amplify the power of the general watermark. The next step will simply consider each S_i to be marked separately by building on a simple watermarking method. The result will be a m-bit (i.e., $i = 1, ..., m$) overall watermark bandwidth in which each bit is "hidden" in each of the marked S_i.

Embedding the Watermark

Once each of the to-be-watermarked secret (keyed) sets S_i are defined, the problem reduces to finding a reasonable, not-very-weak (i.e., better than "coin-flip", random occurrence) algorithm for watermarking a medium-sized set of numbers. While

the "amplification" step provides most of the hiding power of our application in the following we encode the watermark bits into the provided (secret) sub-sets.

A desired property of an encoding method is the ability to retrieve the encoded information ("blindly") without having the original data. This can be important especially in the case of very large dynamic databases (e.g., 4-5 TBytes of data) where data-mining portions were outsourced at various points in time. It is unreasonable to assume the requirement to store each outsourced copy of the original data. Our method satisfies this desiderata.

Single Bit Encoding

We now discuss how a single bit is encoded into a selected subset of the data. We are given S_i (i.e., one of the subsets secretly selected in the previous step) as well as the value of a watermark bit b that is to be encoded into S_i. Let $\mathbb{G}$ represent the set of user specified change tolerance, or usability metrics.

Let v_{false}, v_{true}, $c \in (0,1)$, $v_{false} < v_{true}$ be real numbers (e.g., $c = 90\%$, $v_{true} = 10\%$, $v_{false} = 7\%$). We call c a *confidence factor* and the interval (v_{false}, v_{true}) *confidence violators hysteresis*. These are values to be remembered also for watermark detection time. We can consider them as part of the encoding key.

Definition: Let $avg(S_i)$ and $\delta(S_i)$ be the average and standard deviation, respectively, of S_i. Given S_i and the real number $c \in (0,1)$ as above, we define $v_c(S_i)$ to be the *number of items of* S_i *that are greater than* $avg(S_i) + c \times \delta(S_i)$. We call $v_c(S_i)$ the number of positive "violators" of the c confidence over S_i, see Figure 3.4.

Mark encoding convention: Given S_i, c, v_{false} and v_{true} as above, we define $mark(S_i) \in \{true, false, invalid\}$ to be $true$ if $v_c(S_i) > (v_{true} \times |S_i|)$, $false$ if $v_c(S_i) < v_{false} \times |S_i|$ and $invalid$ if $v_c(S_i) \in (v_{false} \times |S_i|, v_{true} \times |S_i|)$.

In other words, the watermark is modeled by the percentage of positive "confidence violators" present in S_i for a given confidence factor c and confidence violators hysteresis (v_{false}, v_{true}). Encoding the single bit (see Figure3.3), b, into S_i is therefore

---

encode(bit, set, v_{false}, v_{true}, c)

compute $avg(set)$, $\delta(set)$

compute $v_c(set)$

if $v_c(set)$ **satisfies** desired bit value **return** *true*

if (bit)

compute $v_* \leftarrow v_{true} - v_c(set)$

alter v_* items close to the stddev boundary so that they become $> v_{true}$

else

(!bit) case is similar

compute $v_c(set)$

if $v_c(set)$ **satisfies** desired bit value **return** *true*

else rollback alterations (distribution shifted too much?)

return *false*

Figure 3.3. Relational Data with Numeric Types: Single Bit Encoding Algorithm (illustrative overview).

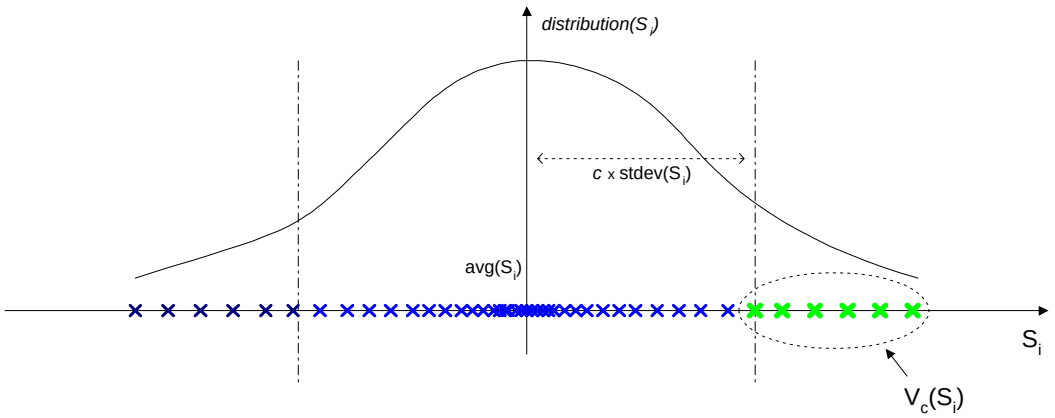


Figure 3.4. Relational Data with Numeric Types: Distribution of item set S_i. Encoding of the watermark bit relies on altering the size of the "positive violators" set, $v_c(S_i)$.

achieved by making minor changes to some of the data values in S_i such that the number of positive violators ($v_c(S_i)$)is either (a) less than $v_{false} \times |S_i|$ if $b = 0$, or (b) more than $v_{true} \times |S_i|$ if $b = 1$. Of course the changes made to the data must not violate the change tolerances, $\mathbb{G}$, specified by the user.

Note: Encoding the watermark bits into actual data distribution properties (as opposed to directly into the data itself) presents a set of advantages, the most important one being its increased resilience to various types of numeric attacks (see Section 3.6.2) as compared to the fragility of direct data domain encoding.

Performing the required item alterations while satisfying the given "usability" metrics (i.e., $\mathbb{G}$) is one of the remaining challenges. To do this, the algorithm deploys the primitive watermarking step (e.g., for S_i) and then checks for data usability with respect to $\mathbb{G}$. If the tolerances are exceeded it simply ignores S_i and considers the next secretly selected subset to encode the rest of the watermark. This will result in errors (misses) in the encoded marks but by deploying error correcting techniques (e.g., majority voting, see Figure 3.5 (b)) the errors are mostly eliminated.

A decision needs to be made regarding the size of the subsets selected in the amplification step (i.e., $|S_i|$). Given that our method embeds 1 bit per subset, a

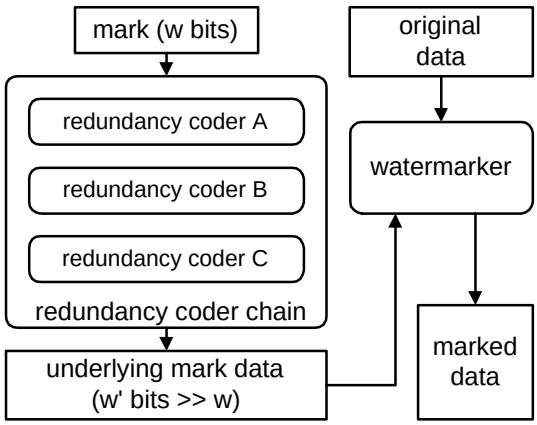
trade-off is to be observed between larger sets (tolerant to more data alteration attacks) but a small bandwidth, and smaller sets (more "brittle" to attacks) but a larger encoding bandwidth. This can and should be considered as a fine-tuning step for the particular data usability metrics provided. If those metrics are too restrictive, more items will be needed inside S_i to be able to encode one bit while still preserving required usability. On the other hand if the usability metrics are more on the relaxed side, S_i can be very small, sometimes even 10-15 items. This enables for more encoding bandwidth overall.

At watermark detection time, after recovering all the watermark copies from the given data ***majority voting*** over all the recovered watermark bits (or any other error correcting method for that matter, see Figure 3.5 (a)) can be employed in order to determine the most likely initial watermark bits.

Another interesting point to be made here is that (as outlined by the optimality principle in Chapter 7) bringing the watermarked data as close as possible to the allowable distortion bounds ("usability vicinity" limits) is of definite benefit in making the usability of the watermarked data as fragile as possible to any attack. An attacker with the intent of removing/altering the watermark is now faced with the fact that any further alterations performed have an increased likelihood of making the data invalid with respect to the guaranteed usability metrics [3], thus potentially removing its value. We integrated this idea also in our implementation. As watermark embedding progresses, a certain embedding aggressiveness factor increases, resulting in actual changes to the data to be performed more and more up to the permitted limit and not only as required.

Note: The *incremental* nature of the aggressiveness factor increase is required so as to make sure that at least several copies of the mark were embedded successfully. Getting aggressive too fast might not allow for entire mark copies to be embedded (while maintaining data usability).

[3]Because the watermarking process already altered the data up to its usability metrics limits.



bits	5	4	3	2	1	0
W_0	1	0	1	1	1	0
W_1	1	0	1	0	1	0
W_2	1	0	0	0	1	1
W_{result}	**1**	**0**	**1**	**0**	**1**	**0**

Figure 3.5. Relational Data with Numeric Types: (a) Different error correcting (wmdb. sys. RedundancyCoder) plugins can be added/removed at runtime in order to provide an increased level of resilience for the original watermark to be embedded. (b) Example of majority voting over three recovered watermark copies for a 6 bit sized original watermark.

3.3.4 Resilience Analysis

Maybe the most important resilience-revealing questions in evaluating watermarking algorithms, can be formulated as follows: *What is the probability of success of Mallory aiming at destroying at least one watermark bit, as a function of the amount of data damage (i.e., number of surgeries)?* The importance of an answer to the above stems from the immediate ability to compute resilience and attackability bounds of the watermarking algorithm by relating the required damage for a successful attack to the maximum permissible damage levels.

Let us naturally assume a primitive 1-bit encoding method for subsets (of subset size denoted by s) that is resilient to a minimum $s \times l$ random "surgeries" (data item removals and/or alterations), $l \in \mathbb{R}$. This resilience can be guaranteed by varying the encoding parameters presented in Section 3.3.3. For now we are going to assume

a most general scenario by not assigning values for l. Also let us consider an error correcting mechanism (e.g., majority voting) able to correct

$$e \times \frac{n}{s} \times \frac{1}{m}$$

bit errors where, as above, n is the number of total items in the input set, m is the bit-size of the watermark to be embedded and $e \in \mathbb{R}$, $e \geq 2$. In other words e naturally models the error correcting power proportional to the ratio of total available bandwidth to watermark size. In order to keep a maximum degree of generality, we are not assigning values for e at this point.

Let $P(s, a'')$ be the average success probability (i.e., actual bit-flip) of a random, a'' sized (i.e., a'' surgeries) attack on a 1-bit encoding subset of size s. The assumption of resilience to l surgeries of the subset encoding can be thus also expressed as

$$P(s, x) = 0, \forall x \leq l$$

First, let us compute the local (i.e., at subset level) amount of surgeries required in the case of an a-sized (i.e., a surgeries) global attack on the entire marking scheme. Because of the additional sorting and one-way hashing step (see Section 3.3.3), for illustrative purposes we introduce a simplifying assumption, namely that of a uniform distribution of all the surgeries among the individual subsets. That is:

$$a'' = a \times \frac{s}{n}$$

The probability of an a-sized attack affecting (e.g., flipping) exactly t bits in the underlying data bandwidth, before error correction, $P_t(s, a)$ is:

$$P_t(s, a) = C_{\frac{n}{s}}^{t} \times P(s, a'')^t \times (1 - P(s, a''))^{n-t} \tag{3.1}$$

Given our e-bit error correction ability, the probability that one watermark bit is altered by an a-sized attack becomes:

$$P_1(a) = 1 - \sum_{i=1}^{e}(P_i(s,a)) \tag{3.2}$$

Getting back to $P(s, a'')$ let us recollect the fact that it actually represents the average success probability (i.e., actual bit-flip) of a random, a'' sized (i.e., a'' surgeries) attack on a 1-bit encoding subset of size s. There also exists the assumption of resilience to l surgeries of the subset encoding.

Because the bit-encoding method is highly dependent on input data, its distribution and actual values of the involved encoding parameters (e.g., c, v_{false}, v_{true}), it becomes impossible to provide an exact, in-depth analysis of the actual value of $P(s, a'')$ for arbitrary input data. Given a certain fixed data set, it might be possible to actually exactly determine the value of $P(s, a'')$ but to no useful effect, as much of the power of the encoding lies in its ability to watermark arbitrary input.

Another method of analyzing $P(s, a'')$ could take the form of an experiment, sampling its value over a large number of potential different data inputs. We are proposing to pursue this avenue in future research. For the scope of the current analysis we are going to reasonably approximate $P(s, a'')$. Remember that we introduced the assumed average l tolerated surgeries per 1-bit encoding. We know that, on average, $P(s, x) = 0, \forall x \leq l$. Let us assume that $a'' > l$. Then we approximate

$$P(s, a'') = q \times \frac{a'' - l}{s}, \forall a'' \in (l, s)$$

where $q \in \mathbb{R}, q \geq 1$ is a input data characteristic normalization constant. Now we can write equation (3.1) as

$$P_t(s, a) = C^t_{\frac{n}{s}} \times (q \times \frac{a'' - l}{s})^t \times (1 - (q \times \frac{a'' - l}{s}))^{n-t} \tag{3.3}$$

For illustration purposes, by substituting $t = 1, n = 10000, s = 50, a = 1000, l = 4, q = 1$, and continuing the computation we obtain

$$P_1(50, 1000) = \frac{1}{200} \times \frac{1}{50} \times (\frac{49}{50})^{9999} \simeq 1.86 \times 10^{-92} \simeq 0 \tag{3.4}$$

There is a surprisingly low probability of destroying one bit in the underlying data by a 1000-sized attack on an input set of 10000 where the subsets are of size 50 and subset encoding is tolerant to at least 4 item-surgeries. In other words, for a 10000 tuples item set, an encoding with subsets of size 50 and an average 1-bit subset encoding tolerance to 6% data item losses (experimental results show much higher loss tolerance, see Section 3.6.2), this probability is surprisingly low, virtually zero.

Note: While the approximation introduced for $P(s, a'')$ is satisfying enough for real life uses, confirmed also in our experiments presented in Section 3.6.2, it might not be entirely accurate for generic data. Subject to future research is to theoretically determine the exact shape of $P(s, a'')$ (or the feasibility of the computation thereof) for generic normally distributed input data, for example.

3.4 The Relational Database

As discussed in Section 3.2, in the relational database setting it is essential to preserve structural and semantic properties of the data. Sometimes it is undesirable or even impossible to map higher level semantic constraints into low level (combined) change tolerances for individual tuples or attributes. It should be noted that not all constraints of the database need to be specified. For example, in certain scenarios, a practical approach would be to begin by specifying a mean square error bound on individual items. Further semantic or structural constraints that the final data consumer (user) would like to preserve can be added to these basic ones. The practically infinite nature of the set of these potential constraints that can be desired/imposed on a given data set makes it such that a different, more versatile, "data goodness" (i.e., semantically) assessment method is required. We propose a solution that handles each of these constraints that need to be preserved as an inherent component for the watermarking algorithm.

Constraints that arise from the schema (chiefly key constraints), can easily be specified in a form similar to (or derived from) SQL *create table* statements. In addition, integrity constraints (e.g., such as *end_time* being greater than *begin_time*) can be expressed. A tolerance (or usability metric) is specified for each constraint. The tolerance is the amount of change or violation of the constraint that is acceptable. This is an important parameter since it can be used to tailor the quality of the watermark (at the expense of greater change in the data). As mentioned earlier, if the tolerances are too low, it may not be possible to insert a watermark in the data.

In order to handle a very wide variety of constraints, our solutions allows various forms of expression, e.g., in terms of arbitrary SQL queries over the relations, with associated requirements (usability metric functions). For example, the requirement that the result of the join (natural or otherwise) of two relations does not change by more than 3% can be specified. Thus we can ensure that any changes made by the watermarking algorithm do not violate the required properties. Some representative examples of constraints are presented in Section 3.6.3 and were used to watermark real Wal-mart data warehouse data.

3.4.1 Algorithm

The algorithm outline for watermarking relational data proceeds as follows (see Figure 3.6): **(i)** User-defined queries and associated guaranteed query usability metrics and bounds are specified with respect to the given database. **(ii)** User input determines a set of attributes in the database considered for watermarking, possibly all. **(iii)** For each selected attribute we then deploy the simplified algorithm where in step E.2.2 instead of checking for local data usability the algorithm simply checks all global user-defined queries and usability bounds by execution.

An additional benefit of operating in the relational data domain is the ability to use the actual relation key in the secret subset selection procedure, instead of the proposed most significant bits of the data (i.e., watermarked attribute data). It is

highly unlikely that an attack will entirely change the database schema and replace the key attribute. Thus for most applications it might be a safe idea to use it (or it's MSB space), especially in cases where the actual data is subject to lax usability metrics (i.e., making the data MSB domain less reliable).

Subset Selection. Subset selection proceeds as follows. The input data tuples are sorted (lexicographically) on a secret keyed cryptographic hash of the primary key attribute K. Based on this hash of the primary key attribute value in each tuple, compose a criteria for selecting a set of "special" tuples such that they are uniformly distributed and average a total number of

$$e = \frac{length(attribute)}{subset\_size}$$

For example this criteria could be $H(K, key)) \bmod e = 0$. These special tuples are going to be used as subset "markers". Each subset is defined as the elements between two adjacent markers, having on average $subset\_size$ elements. The detection phase will then rely on this construction criteria to re-discover the subset markers.

Note: Alterations to the data (such as attacks) have a non-zero probability of destroying one of the markers, thus making recovery of the corresponding subset impossible. This will result in a one bit loss in the underlying data (before error correction), hopefully corrected by the error correction mechanisms. For a more in depth discussion see Section 3.5.

3.4.2 Embedding Optimizations

The embedding optimality of the solution presented above is dependent on a set of parameters such as c, $subset\_size$, v_{false}, v_{true} etc. These parameters define a space in which each point corresponds to a different embedding. Intuitively, the solution would benefit from a fine-tuning step in which a certain optimum can be identified in this space, for example, a set of values for which the encoding bandwidth is maximized. Subject to further research is determining potential shapes defined by optimal or close to optimal points in this space. What are some criteria that could

---

wm(attribute, wm_key, mark_data[], plugin_handler, db_primary_key, subset_size, v_{false}, v_{true}, c)

sorted_attribute ← **sort_on_normalized_crypto_hash**(wm_key,db_primary_key,wm_key)

for (i=0; i < $\frac{length(attribute)}{subset\_size}$;i++)

subset_bin ← **next** subset_size elements from sorted_attribute

compute rollback_data

encode(mark_data[i % mark_data.length], subset_bin, v_{false}, v_{true}, c)

propagate changes into attribute

if (not goodness_plugin_handler.isSatisfied(new_data,changes)) **then**

rollback rollback_data

continue

else

commit

map[i] = true

subset_boundaries[i] = subset_bin[0]

return map, subset_boundaries

Figure 3.6. Relational Data with Numeric Types: Watermark Embedding Algorithm (version using subset markers and detection maps shown).

```
det(attribute, wm_key, db_primary_key, subset_sz, v_false, v_true, c, map[], subset_boundaries[])
    sorted_attribute ← sort_on_normalized_crypto_hash(wm_key,db_primary_key,wm_key)
    read_pipe ← null
    do { tuple ← next_tuple(sorted_attribute) }
    until (exists idx such that (subset_boundaries[idx] == tuple))
    current_subset ← idx
    while (not(sorted_attribute.empty())) do
        do {
            tuple ← next_tuple(sorted_attribute)
            read_pipe = read_pipe.append(tuple)
        } until (exists idx such that (subset_boundaries[idx] == tuple))
        subset_bin ← (at most subset_sz elements from read_pipe, excluding last read)
        read_pipe.remove_all_remaining_elements_but_last_read()
        if (map[current_subset]) then
            mark_data[current_subset] ← decode (subset_bin, v_false, v_true, confidence)
            if (mark_data[current_subset] != DECODING_ERROR)
                    then map[current_subset] ← true
        current_subset ← idx
    return mark_data, map
```

Figure 3.7. Relational Data with Numeric Types: Watermark Detection Algorithm (version using subset markers and detection maps shown).

help in determining such a point? Furthermore, another optimization, time and storage permitting, would start by training the watermarking process to be resilient to a set of transformations expected from any potential attacker. The training process would first watermark the input data only to attack it afterward. If the post-attack recovered watermark is not of a satisfactory level, change the input parameters and restart the process. While this might yield considerably better embeddings it is obviously time-consuming by nature and can probably only be applied if time constraints are not an issue.

3.4.3 On-the-Fly Update-Ability

In most scenarios, watermarking outsourced relational content happens only once, at outsourcing time. The main purpose of watermarking in this framework is rights-protection and/or traitor tracing through fingerprinting. Thus, there seems to be little to be gained from an ability to watermark at runtime, in the presence of updates. Moreover, because watermarking inherently alters the data, it is unreasonable to assume that a certain party would keep an altered (i.e., watermarked) copy of the data as replacement for the original [4].

Nevertheless our solution naturally supports on-the-fly watermarking, especially in the presence of updates. Let us analyze several different update scenarios: (i) updates that add fresh tuples to the already watermarked data set, (ii) updates that remove tuples from the already watermarked data and (iii) updates that alter existing tuples. In each of the cases, we assume that the watermarking mechanism runs continuously as a dormant process and is notified for each update, having full control over the watermarked version of the relational data.

(iii) is naturally handled. As altering updates come in, the marking process lets all updates go through that are not altering the watermark. In other words, if an update is to alter a value that belongs to a certain subset $set \subset \mathbb{S}$, it first verifies

[4] After all, how could it generate the outsourced version at the time of outsourcing?

if the alteration is going to alter the $v_c(set)$ value. If this is the case, the marking process automatically re-constructs the set from the original data (also updating the new value), re-embeds the corresponding watermark bit and then updates the subset values in the watermarked version. Otherwise simply let the update go through.

Scenario (i) is similarly handled. If the added value is within a subset $set \subset \mathbb{S}$, the process verifies if this addition (which increases the set size) alters the $v_c(set)$ value. Then proceed as above. (i) becomes more challenging if we consider the insertion of new tuples containing primary key values that would qualify them as subset markers. This can be handled as follows: before inserting, a simple check is performed if indeed the new tuple could be mistaken for a marker. If this is the case, there are two options available. In option (A), the subset in which the tuple is to be inserted is split into two parts, each part being used independently as a subset in encoding one bit of the watermark. Another option (B) is to simply keep a list of such fake markers (together with the detection maps) at the detector's site, awaiting the detection process. At detection time, such markers are simply ignored. While (A) would result in producing a cleaner output, it presents the drawback of requiring a more complex management of bit embeddings. More specifically, these newly available, "out of bound", bit "slots" (split subsets) need to be managed in such a way as to not interfere with the already embedded bits (for efficiency purposes). This can be done by keeping a mapping between each subset marker and it's corresponding underlying watermarking data bit index. This scheme allows for more flexibility as subsequent data bits are to not be embedded in subsequent subsets anymore. On the other hand, (B) is more straightforward but does not make use of the newly available bandwidth.

(ii) is also challenging. Its main difficulty derives from the fact that some removed tuples could be actually subset markers. If a subset marker is removed, then in the detection phase, one bit in the underlying embedding bandwidth will be destroyed. This does not necessarily result in a watermark deterioration as this could hopefully be recovered in the error correction phase. Furthermore, an improvement dealing

with this scenario is the use of an embedding map, remembering exactly which subsets contain a watermark bit and which not (see Section 3.5.1). In the case of a marker removal, the embedding map bit for the subset corresponding to this marker can be reset to signal any future detection process that the marker was lost. Yet another idea would be to simply add a fake marker tuple (satisfying the marker criteria) at the right point in the data.

3.5 Discussion

3.5.1 Detection Maps

Storage space permitting, it might be helpful to store some information about the validity of subsets embeddings. In the detection phase, this information can be used to eliminate unusable bit encodings, in the case of invalid or un-marked subsets, thus increasing detection accuracy. At watermarking time (including on-the-fly phase), a bit for each encoding subset is maintained and updated by the marking process. This map of bits is then used in the detection process to avoid invalid subsets. If the bit is set, the subset is signaled as being valid, otherwise the detection process ignores the corresponding subset.

In the case of small to medium data sizes, and arguably for larger data sizes too, the detection map is not hard to store and "remember" (at the detector's site) for detection time. For example in Section 3.6.3, the embedding map used was only 2000 bits long, barely 1.5KBytes of data, certainly small enough to be stored together with the embedding key and additional watermarking parameters awaiting detection time. As a general rule, the detection map will be (naturally) $\frac{length(attribute)}{subset\_size}$ bits long.

3.5.2 Subset Markers

The use of subset markers (both stored and criteria-based) increases the ability to accurately re-construct the underlying partitions corresponding to the individual bit embeddings.

While experimental results show the real-life resilience of the embedding method, from a more theoretical viewpoint we would like to ask: (i) what is the likelihood of Mallory removing a subset marker? (ii) what is the impact of this likelihood on the final resulting watermark?

Let $P_m(s, a)$ be the probability that an a sized removal attack eliminates a subset marker in the resulting data. Naturally, for each subset (s items) there exists one subset marker. Thus each individual data removal has a probability of $\frac{1}{s}$ of succeeding in removing an actual subset marker; for $a < s$ we have

$$P_m(s, a) = \frac{a}{s}$$

In other words, for each "subset-worth" of data (s tuples) removed from the data, on average one subset marker is destroyed. An eliminated subset marker directly results in the inability to detect the corresponding subset (unless the adjacent markers are preserved in which case something can still be done), thus resulting in a lost bit in the underlying embedding (before error correction). This result is intuitive and encouraging because it shows a direct and linear behavior between data value degradation (tuples removal) and subset marker loss. Mallory has to remove at least this much data to be able to eliminate a marker. In the experiments we used an implementation deploying markers, thus the results include this probability of marker loss.

Another interesting issue related to using subset markers is that apparently the embedding method can get to a point where it is difficult to find subset marker selection criteria that would yield evenly sized sets in the data partitioning phase. This can happen when the selection criteria do not uniformly spread the selected markers over the input tuples. In the case of $H(K, key) \bmod e = 0$, this is the case

iff the $H(K, key)$ values are not uniformly distributed, which in turn can happen (because of the one-way randomized nature of the cryptographic hash) only if the K values are identical or partitioned in large chunks of values. But, the K values are by nature (primary key) different for each data tuple, thus the marker selection is naturally uniformly distributed and results in almost identical subset sizes.

3.5.3 Primary Key Dependence

In Section 3.3.2 we presented a subset selection idea for the case of the simplified problem. The solution was performing a lexicographical, secret, one-way sort on the most significant bit (MSB) space in the considered items, in order to then enable the selection of subsets. While this idea provides a certain level of self-containment and is well suited for the problem it was formulated in, where there exist no external aids in defining an ordering on the data, in the relational data framework, we chose to investigate the use of the primary key as such an aide.

Thus, the solution as such features a certain dependency on the primary database key. If attacks on the primary key occur, there are two potential options: (i) the use of the initial MSB space sort idea and/or (ii) an initial normalizing step that brings the primary key within a pre-defined range in which the subset selection step in Section 3.4.1 (i.e., $H(K', key) \bmod e = 0$) is performed on a MSB portion of the primary key K, i.e., $K' = MSB(K)$. This is to be subject to further investigation, hopefully resulting in primary key independence.

3.6 Experimental Results

This section presents our implementation and the experimental results of watermarking real-life, commercial, data, namely the Wal-Mart Sales relational database.

3.6.1 Implementation: wmdb.*

wmdb.* is our test-bed implementation of the algorithms presented in this chapter. It is written using the Java language and uses the JDBC api.

The package receives as input a watermark to be embedded, a secret key to be used for embedding, a set of relations/attributes to consider in watermarking as well as a set of external *usability plugin modules*. The role of the plugin modules is to allow user defined query metrics to be deployed and queried at run-time without recompilation and/or software restart [5]. The software uses those metrics to re-evaluate data usability after each atomic watermarking step as explained in Sections 3.3 and 3.4.

Once usability metrics are defined and all other parameters are in place, the watermarking module (see Figure 3.9) initiates the process of watermarking. An undo/rollback log is kept for each atomic step performed (i.e., 1-bit encoding) until data usability is assessed and confirmed (by querying the currently active usability plugins). This allows for rollbacks in the case when data quality is not preserved by the current atomic operation.

Watermark recovery takes as input the watermarking key used in embedding, the set of attributes known to contain the watermark as well as various other encoding specific parameters (see Figure 3.7). It recovers the set of watermark copies initially embedded. A final step of error correction (e.g., majority voting) over the recovered copies completes the recovery process.

3.6.2 Experiments

The Wal-Mart Sales Database contains most of the information regarding item sales in Wal-Mart stores nationwide. Its main value lies in the huge commercial potential deriving from mining buying patterns and association rules. In the following

[5]Usability metrics can be specified either as SQL queries, stored procedures or simple Java code inside the plug-in modules.

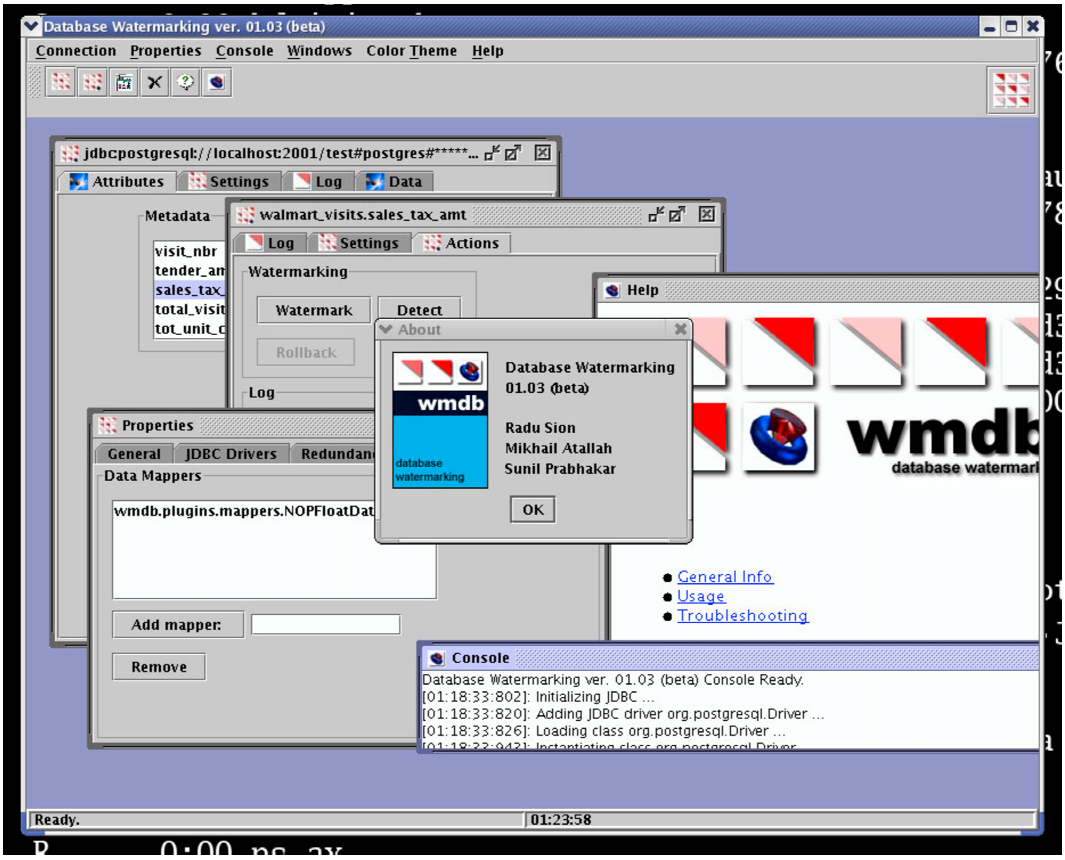


Figure 3.8. Relational Data with Numeric Types: The **wmdb.*** package. Application runtime snapshot.

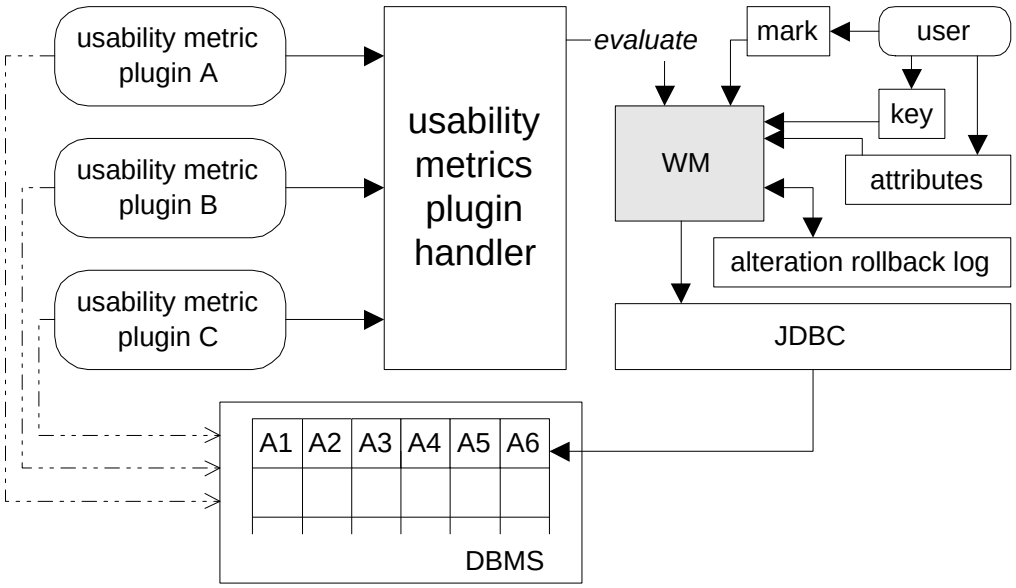


Figure 3.9. Relational Data with Numeric Types: The **wmdb.*** package. Overview.

we present some of our experiments using the wmdb.* package to watermark the Wal-Mart database.

Our experimental setup included access to the 4 TBytes Wal-mart data, (formerly) hosted on a NCR Teradata machine, one 1.8GHz CPU Linux box with Sun JDK 1.4 and 384MB RAM. A subset of the relational schema of the data is presented in the Appendix A. The amount of data available is enormous. For example, the *ItemScan* relation contains over 840 million tuples.

For testing purposes, we deployed our algorithm on a randomly selected subset of the original data (e.g., just a maximum of 141075 tuples for relation *UnivClassTables.StoreVisits*).

We assessed computation times and observed an intuitively (according to the O(n) nature of the algorithm) linear behavior, directly proportional with the input data size. Given the setup described above, in single-user mode, with a local database we obtained an average of around 350-400 tuples/second for watermark embedding, while detection turned out to be approximatively twice as fast. This occurs in the

non-optimized, interpreted Java proof-of-concept implementation. We expect major speedups (orders of magnitude) in a real life deployment version.

In the following we present experiments involving attacks (data loss, data alterations, linear changes, data resorting) as well as the evaluation of the available bandwidth in the presence of different data goodness metrics (tolerable absolute change and data classification preservation).

Data Loss Attacks ("Surgeries")

In this attack scenario, we study the distortion of the watermark as the input data is subjected to gradually increasing levels of data loss.

In Figure 3.10 (c) the analysis is performed repeatedly for single bit encoding using the "confidence-violators" encoding method outlined in Section 3.3.3. The results are then averaged over multiple runs. The "confidence-violators" primitive set encoding proves to be resilient to a considerable amount of randomly occurring uniformly distributed surgeries (i.e., item removals by Mallory, with no extra knowledge) before watermark alterations occur. Even then, there exists the ability to "trace" or approximate the original watermark to a certain degree (i.e., by trying to infer the original mark value from an invalid set). The set size considered was 35, experiments were performed on 30 different sets of close to normally distributed data. Other parameters for the experiment include: $v_{false} = 5\%, v_{true} = 9\%, c = 88\%$. The average behavior is plotted in the graphs. Up to 25% and above data loss was tolerated easily by the tested data, before mark alteration (i.e., bit-flip) occurred.

Figure 3.10 (a) and (b) depict more complex scenarios in which a real multi-bit watermark is embedded into a larger data set (both uniform (a) and normal distributions (b) were considered). The input data contained 8000 tuples, subset size was 30 and the considered watermark was 12 bits long. Other parameters: $v_{false} = 15\%, v_{true} = 35\%, c = 85\%$. This set is then subjected to various degrees of data loss and the watermark distortion is observed. The encoding method again proves to be

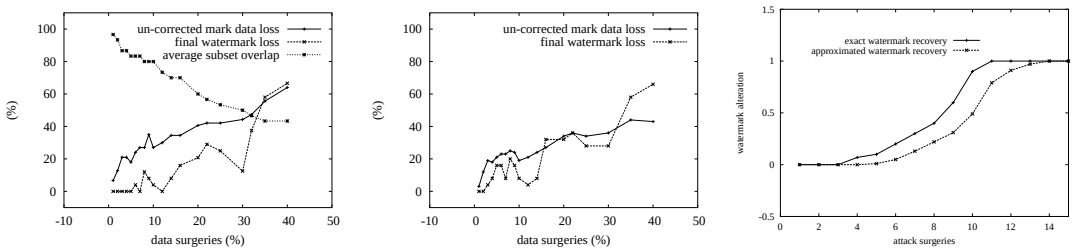


Figure 3.10. Relational Data with Numeric Types: Resilience to data surgeries (a) uniform distribution, (b) normal distribution, (c) single subset (1-bit) encoding

surprisingly resilient by allowing up to $45 - 50\%$ data loss while still $40 - 45\%$ of the watermark survives. Also, in (a), as data alteration increases, the subset (i.e., secretly selected for encoding 1-bit, see 3.3.3) overlap (i.e., the "resemblance" to the original content, the number of same elements in resulting subsets) degrades.

Note: Some of the figures presented in this section feature "spikes". This is a result of the adaptive data-dependent nature of the encoding. Different input data reacts differently to data surgeries (for example) and feature slightly varying behavior at distinct points. Averaging over multiple inputs provides a solution for this issue. Nevertheless, we believe that, while it might soften the spikes it would also (arguably) tone down distinct features for a given data set, features that inter-relate figures. Instead of focusing on local variations, the figures should be interpreted as an illustrative sample of the global governing trends.

Data Alteration Attacks (Epsilon-Attack)

Presented with the watermarked data Mallory is faced with two contradictory tasks: preserving the inherent value of the data while at the same time removing the hidden watermark. Given no knowledge of the secret watermarking key nor of the original data the only available choice is to attempt (minor) random data modifications in the hope that at some point the watermark will be destroyed. Because

the original data is unknown (thus also the current watermark-related distortion is unknown) it is impossible for Mallory to determine the real "minority" of changes he/she performs. In other words, because of the goal of preserving the data value, Mallory cannot afford to perform significant change to the data.

In this experiment we analyze the sensitivity of our watermarking scheme to randomly occurring changes, as a direct measure for watermark resilience. To do this, we define a transformation that modifies a percentage τ of the input data within certain bounds defined by two variables ϵ and μ. We called this transformation *epsilon-attack*. Epsilon-attacks can model any uninformed, random alteration – the only available attack alternative. A *normal* epsilon-attack modifies roughly $\frac{\tau}{2}$ percent of the input tuples by multiplication with $(1 + \mu + \epsilon)$ and the other $\frac{\tau}{2}$ percent by multiplication with $(1 + \mu - \epsilon)$. A *uniform altering* epsilon-attack modifies τ percent of the input tuples by multiplication with a uniformly distributed value in the $(1 + \mu - \epsilon, 1 + \mu + \epsilon)$ interval.

In Figure 3.12 (a), a comparison is made between the case of uniformly distributed (i.e., values are altered randomly between 100% and 120% of their original value) and fixed alterations (i.e., values are increased by exactly 20%). In the case of fixed alterations the behavior demonstrates the effectiveness of the encoding convention: as more and more of the tuples are altered linearly, the data distribution comes increasingly closer to the original shape. For example when 100% of the data is modified consistently and linearly the mark data suffers only 6% alterations. A peak around 50% data alterations can be observed indicating that an attack changing roughly 50% of the data might have a greater chance of success. This is also intuitively so (in the case of randomly distributed alterations) as a maximal change in distribution is expected naturally when close to half of the data set is skewed in the same "direction" (by addition or subtraction).

Parameter μ models the average of the data alteration distribution while ϵ controls its width. Naturally, a *zero-average* epsilon-attack ($\mu = 0$) is a transformation

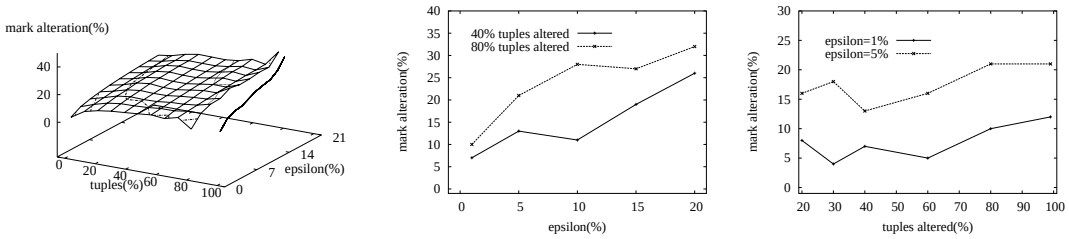


Figure 3.11. Relational Data with Numeric Types: Epsilon-attack (zero-average) on normally distributed data.

that modifies roughly $\frac{\tau}{2}$ percent of the input tuples by multiplication with $(1+\epsilon)$ and the other $\frac{\tau}{2}$ percent by multiplication with $(1-\epsilon)$.

Figure 3.11 presents the behavior of our encoding algorithm to this type of attack. This is particularly intriguing as it reveals clearly a special feature of the watermarking method: since the bit-encoding convention relies on altering the actual *distribution* of the data, it survives gracefully to any distribution-preserving transformation. Randomly changing the data, while it can definitely damage the watermark (e.g., especially when altering around 50% of the data, see Figure 3.12 (a)), proves to be, to a certain extent, distribution-preserving. A *zero-average epsilon-attack* is survived very well. For example, altering 80% of the input data within 20% of the original values still yields over 70% of the watermark.

Note: One could argue that, after all, if the watermark encoding relies too much on the distribution of the data, one successful attack could be the one that alters exactly this distribution. But this is not possible, as the power of the watermarking scheme lies not only in the distribution itself but also in the secrecy of the encoding subsets. In other words, *where* the bits are encoded (i.e., subsets, see Section 3.3) is as important as *how*. Altering global data characteristics would not only destroy probably much of the value of the data but, as shown above, achieve little in destroying the watermark.

In Figure 3.11 (a), as the percentage of tuples altered and the alteration factor goes up, so does the watermark distortion. Nevertheless, it turns out to be surprisingly resilient. For example, altering 100% of the data within 1% of the original values can yield a distortion as low as $5 - 6\%$ in the resulting watermark. The watermark distortion increases with increasing alteration factor (b) or percentage of data (c). Figure 3.11 (b) presents a comparison between the curves corresponding to the alteration of 40% of the tuples versus 80% of the tuples. Naturally the curve for the higher tuples percentage appears "above". In (c) a comparison is made between curves for the alteration factor 1% or 5%. The higher alteration curve is intuitively "above". Note that the curves are slightly increasing but not very steep: mark alteration is less dependent on the percentage of data altered than on the alteration factor (as seen in (b)). Thus the watermarking scheme proves a natural resilience to un-informed attacks (modeled by epsilon-attack transformations).

Data Quality (Goodness) Metrics

Here we analyze the impact of data goodness preservation on the available watermark encoding bandwidth. Intuitively the more restrictive data constraints one imposes, the less available bandwidth, as allowable data changes are directly impacted. We present two results. The first analyzed goodness metric is a commonly considered one, namely upper bounds imposed on the total and local tolerable absolute change (i.e., of the new data with respect to the original).

Note: An identical experimental result was obtained for a related metric, the maximum allowable mean squared error.

In Figure 3.12 (b), as data goodness metrics are increasingly restrictive, the available bandwidth (guaranteeing higher resilience) decreases. In the illustrated experiment, the allowed absolute change in the watermarked data (i.e., from the original) is decreased gradually (from 0.1% to 0.02%) and the decrease in available encoding bandwidth is observed (depicted as a percent of total potential bandwidth).

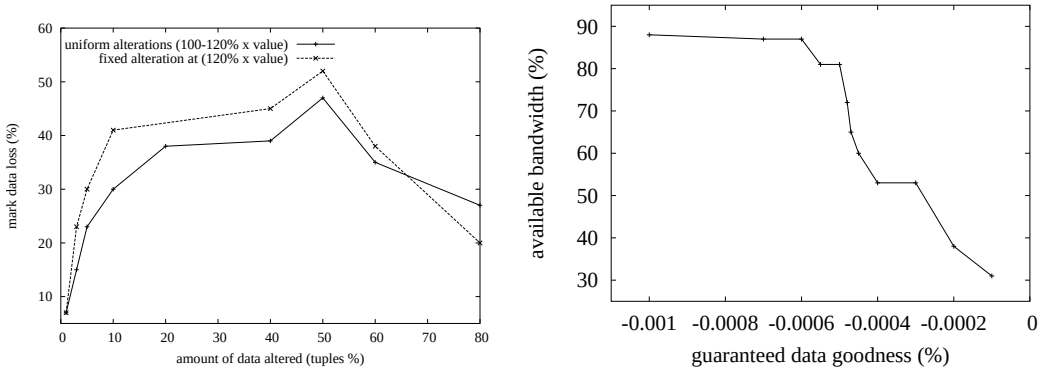


Figure 3.12. Relational Data with Numeric Types: (a) Epsilon-attack (non-zero average) on a normally distributed data set. (b) Impact of guaranteeing a Maximum Allowable Absolute Change on the available watermarking bandwidth.

The upper limit (approx. 90%) is inherently data-imposed and cannot be exceeded due to original data characteristics, making it the maximum attainable bandwidth.

Another important experiment analyzes a classification-preserving data goodness metric. Classification is extremely relevant in areas such as data mining and we envision that many of the actual deployment scenarios for our relational watermarking application will require classification preservation.

Classification preservation deals with the problem of propagation of the classes occurring in the original (input) data in the watermarked (output) version of the data. It provides thus the assurance that the watermarked version still contains most (or within a certain allowed percentage) of the original classes.

To perform the experiment, we designed and implemented a data classifier which allows for runtime fine-tuning of several important classification parameters such as the number of (synthetic) classes to be associated with a certain data set as well as the *sensitivity* of these classes. The *sensitivity* parameter can be illustrated best by example. Given a certain data set to be altered (e.g., watermarked) and an item X in this data set, the classification sensitivity models the amount of alterations X tolerates before it "jumps" out of its original class.

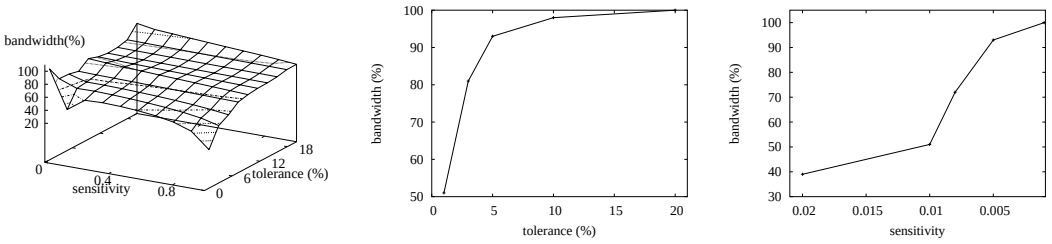


Figure 3.13. Relational Data with Numeric Types: Impact of a classification preservation on the available watermarking bandwidth.

Note: One different perspective on *sensitivity* can be obtained by linking it to the notion of classification selectivity. The more selective a classification is, the more *sensitive* its behavior.

The *tolerance* factor in Figure 3.13 represents the maximum tolerated classification distortion (i.e., percentage of class violators with respect to the original). In (a), as the classification tolerance and sensitivity go up, so does the available bandwidth. Figure 3.13 (b) shows how the watermarking algorithm adapts to an increasing data goodness tolerance (classification sensitivity 0.01). Figure 3.13 (c) depicts how for classification tolerance fixed at 1%, the sensitivity of the classification impacts directly the available bandwidth.

Depending on classification sensitivity (e.g., 0.01 in (b)), up to 90% of the underlying bandwidth can become available for watermark encoding with a restrictive 6% classification preservation goodness.

These results confirm the adaptability of our watermarking algorithm. As classification tolerance is increased, the application adapts and makes use of an increased available bandwidth for watermark encoding. This also show that classification preservation is compatible with our distribution-based encoding method, an important point to be made, considering the wide range of data-mining applications that could naturally benefit from watermarking ability.

3.6.3 Scenario: The Wal-Mart Sales Database

A set of more complex experiments performed are outlined in Section 3.6.2. Here we present an actual application of our watermarking software. The watermark considered to be hidden was the string "(C)walmart" (80 bits in 8bit/character encoding). Algorithm parameters were adjusted repeatedly in an attempt to maximize the number of embedded copies, finally establishing them as $c = 85\%, v_{false} = 15\%, v_{true} = 30\%$. The attributes watermarked were:

```
ItemScan.TotalScanAmount
ItemScan.UnitCostAmount
StoreVisits.TenderAmt
StoreVisits.TotalVisitAmt
StoreVisits.SalesTaxAmt
```

The size of the subsets considered was roughly 70 for a total of around 2000 available encoding bits in the TotalScanAmount attribute for example. We considered a set of usability metrics and associated queries, including the following:

(a) Intra-relational Consistency:

```
ItemScan.UnitCostAmount x ItemScan.ItemQuantity = ItemScan.TotalScanAmount
```

(b) Inter-relational Consistency:

```
StoreVisits.TotalVisitAmt < SUM(ItemScan.TotalScanAmount)
```

(c) General Attribute Constraints: MSE constraints for attribute StoreVisits. TotalVisitAmt: introduced normalized mean squared error should not exceed 1%.

(d) General SQL Constraints: e.g., (d.1) for each store and date, the number of sales after watermarking should not deviate more that 2% from the original data, (d.2) for the join between SV and IS on the *VisitNbr* attribute, a maximum number of 5% of the elements should be disturbed after watermarking. For example, the actual numeric value in (d.2) can be formulated as follows:

```
SELECT * AS J1 FROM ItemScanOrig, StoreVisitsOrig
WHERE ItemScanOrig.VisitNbr = StoreVisitsOrig.VisitNbr;
SELECT * AS J1 FROM ItemScanWM, StoreVisitsWM
WHERE ItemScanWM.VisitNbr = StoreVisitsWM.VisitNbr;
SELECT COUNT(*) FROM (
        (SELECT * FROM J1 EXCEPT SELECT * FROM J2)
        UNION
        (SELECT * FROM J2 EXCEPT SELECT * FROM J1))
```

In the working system, each of these metrics was represented by a separate usability metric plug-in, used in evaluating data usability after each atomic watermarking step (see Figure 3.9). For example, the usability metric module for (d.2) executes the above query and if the result exceeds a certain threshold, it simply returns false, denying the watermarking module the proposed modifications to the data. The watermarking module then rolls back those modifications and proposes (by backtracking) new ones. It eventually continues on to the next subset.

Deployment Issues

Some of the usability constraints above present a set of deployment challenges especially when implemented as usability plug-in modules. Whereas (a), (b) might be straight-forward to code, (c) presented some complications because of the need of maintaining original reference data in order to be able to compute MSE values. This was solved by creating an additional relation at run-time, used by the plug-in to keep original data that was altered in the watermarked version.

Step (d) proved to be the most challenging, particularly (d.2) because of the requirement to always compare JOINS on the original data to joins on the resulting data. We tried two approaches. In the first approach, the entire original data was duplicated temporarily and JOINS were dynamically performed at run-time. As discussed above, space and computation constraints are not of concern here if within reach, as this is done only once in the lifetime of the outsourced data. However, this soon proved to be infeasible, and computation-intensive, often causing JDBC buffer

related crashes and taking long times to execute. The second approach optimized the idea by keeping just a record of watermark-related alterations and then directly assessing their impact in the data JOIN result (i.e., determining whether a change in tuple X in table *SV* will affect an as yet un-accessed tuple in the JOIN result with table *IS*). This second approach, requiring less space and computation power, proved to work well.

Using the attribute *ItemScan.TotalScanAmount*, the watermark was embedded successfully roughly 21 times, leading to a good utilization (84%) of the potential encoding bandwidth of 2000 bits (see above). This allows for a highly accurate final majority voting step at mark retrieval/detection time (see Section 3.6.2 for attack and resilience experiments).

3.7 Related Work

With respect to directly related work, one simultaneous published related effort in the relational data framework [35] is available for comparison. Its main algorithm proceeds as follows. A subset of the initial data tuples are selected based on a secret criteria; for each tuple, a secret attribute and corresponding least significant (ξ) bit position are chosen. This bit position is then altered according to yet another secret criteria. The main assumption is, that changes can be made to any attribute in a tuple at any least significant ξ bit positions. At watermark detection time, the process will re-discover the watermarked tuples and, for each detected accurate encoding, become more "confident" of a true-positive detection.

There are many fundamental differences between this effort and our work, including: **(i)** In [35] there is no provision for multi-bit watermarks. **(ii)** Because the watermark is embedded in multiple attributes at the same time, vertical partitioning attacks become of concern. In a schema with a primary key and two attributes, removing one of the attributes will weaken the watermark embedding 50% (i.e., an amount proportional to the number of total attributes used in embedding). **(iii)**

The actual bit encoding of the watermark is naturally vulnerable to an entire set of trivial attacks in the numeric domain (e.g., linear changes, see below). **(iv)** Maybe the most important difference is the fact that our solution is built around a framework considering higher level semantics to be preserved in the original data. [35] only honors limits to fractional change in individual attribute values and there are no provisions for imposing any constraints specific to relational databases, such as preserving JOIN results, classifications, the relative values of attributes and other requirements such as outlined in Section 3.2. Our solution was designed around the concept of preservation of such higher level semantic constraints. All of these and ad-hoc specified SQL constraints can be preserved and honored in the result.

This is so because we believe that a sound and truly resilient watermarking method has to *start* by assessing the final purpose of the content to be watermarked, together with its associated allowable alteration limits. These limits are often times impossible to express as "least significant bit" constraints and require a higher level semantic expression power such as offered by the data goodness plugins. As outlined in Section 3.2.1 one of the main challenges of watermarking is the ability of the encoding method to not interfere with the final data uses. This is why **(v)** the assumption that the least significant ξ bits in any tuple can be altered, has limited applicability and is often plain wrong. It cannot be considered in many important applications such as data mining that require the preservation of classification. Consider a simple application where a relational data set is used in conjunction with a classifier clustering individuals into several categories based on age, e.g., "pre-school" (0-6 years), "child" (7-13), "teenager" (14-18), "young male" (19-21), "adult" (22+). Naturally, there are likely many scenarios in which any minor alteration to the data should *not* change a person's class, for various reasons (e.g., adult movie rentals). If the rights protection method deployed is not able to handle this semantic constraint, using the watermarked result can lead to potentially illegal situations in which a minor is able to purchase alcohol and/or rent adult movies. By randomly modifying least significant bits, changing an age of 20 into 21 becomes quite likely and pro-

duces a highly undesirable result. A higher level analogy can be constructed with the image watermarking framework, where the LSB approach to watermarking was among of the initial attempts and immediately proved its limits. LSB information hiding was immediately discarded as an effective technique for resilient watermarking [2] [19]. Modern media (e.g., sound, image) watermarking algorithms start from human perceptual models of (assuming this is the final consumer of the Works) and build on it by mapping the model into allowable tolerances for data changes. The watermark encoding needs to preserve data quality. Cross-domain experience is required to deploy the same paradigm also in the relational framework. Our approach builds on this experience.

(vi) In [35] resilience to true data alterations (e.g., linear changes to an arbitrary subset of the data, non-uniform scaling of all or part of the data, and epsilon-attacks) is not analyzed and the encoding method lacks fundamental provisions to resist such alterations, many of which would certainly preserve value in the result. Even minor-level epsilon-attacks such as the ones illustrated in Section 3.6.2 (where our encoding survives up to 97%) would entirely remove the mark. Consider for example the case of a data mining application aiming to discover association rules from a data set with a schema composed of a primary key and two numeric attributes. By multiplying all numeric values with e.g., 2 the resulting value bit strings are effectively shifted, resulting in a total loss of the original watermark. If the association rules are preserved in the result, the data is still valuable and Mallory can simply perform this attack on any and all suspected watermarked data sets and completely remove the watermark. Arguably a majority of associations are preserved if linear data changes are performed to the underlying data consistently.

Aside from the issues outlined above, [35] features also one difference that results in desirable properties of clear benefit. Partly because of the assumption of single bit sized watermarks, the encoding is independent of tuple ordering (if the primary key attribute is preserved and un-changed). The detection process is not required to reconstruct an actual watermark string but rather relies on the detection of a

statistical improbability in the result to return the one-bit watermark; thus data location-awareness is not necessary, a partial reason for tuple ordering independence. This independence results in two advantages of using an encoding such as in [35]: the ability **(vii)** to detect a watermark bias in small amounts of the data and, in certain scenarios, **(viii)** to handle multiple source data merging [6].

There surfaces a trade-off to be observed here. Handling multiple source data merges and resisting massive data loss attacks are desirable important properties. Preserving higher level semantics in the result and surviving value preserving data alteration attacks (e.g., classification preserving linear changes, random epsilon-attacks) are equally or even more important. Our solution balances the trade-off between the ability to resist data loss up to 60-70%, major value-preserving numeric attacks as well as preserve guaranteed levels of data quality in the result. It naturally handles a certain level of data merging (similar to data addition). Nevertheless an ability to handle increased merging levels could be obtained by designing an encoding scheme with all the advantages of our solution and independent of tuple ordering. We are currently investigating this issue.

Another interesting related research effort is to be found in [36] where the authors discuss theoretical links between query result preservation and associated allowable input data alterations.

3.8 Conclusions. Future Research.

In this chapter we introduced the problem of data security through watermarking in the framework of numeric relational data. We (a) designed a solution to a simplified version of our problem, namely watermarking a numeric collection by (a.i) defining a new suitable mark encoding method for numeric sets and (a.ii) building an algorithmic secure mapping (i.e., mark amplification) from a simple encoding method to a more complex watermarking algorithm, and (b) applied the concept to

[6] This is the case if several watermarked data sources are combined, and the primary key attribute that was used in the embeddings is preserved (e.g., as the join attribute) in the result.

numeric relational databases. We thus provided a solution for resiliently watermarking relational databases. We also developed a proof of concept implementation of our algorithms under the form of a Java software package, **wmdb.*** which we then used to watermark a commercial database, extensively used for data-mining in the area of customer trends and buying patterns. Published research results of this work include [7], [8], [9] and [37].

4 RELATIONAL DATA WITH CATEGORICAL TYPES

A novel method of rights protection for categorical data through watermarking is introduced in this chapter. New watermark embedding channels for relational data with categorical types are discovered and associated novel watermark encoding algorithms are proposed. While fully preserving data quality requirements, the introduced solution is designed to survive important attacks, such as subset selection and random alterations. Mark detection is fully "blind" in that it doesn't require the original data, an important characteristic especially in the case of massive data. Various improvements and alternative encoding methods are proposed and validation experiments on real-life data performed. Important theoretical bounds including mark vulnerability are analyzed. The method is proven (experimentally and by analysis) to be extremely resilient to both alteration and data loss attacks, for example tolerating up to 80% data loss with a watermark alteration of only 25%.

4.1 Introduction

Categorical data is data drawn from a discrete distribution (often with a finite domain). By definition, it is either non-ordered (nominal) such as gender or city, or ordered (ordinal) such as high, medium, or low temperatures.

While in Chapter 3 we explored the issue of watermarking *numeric* relational content, there are a multitude of applications that would benefit from a method of rights protection for categorical data types. In this chapter we propose and analyze this issue of rights protection for *categorical* relational content through watermarking.

Main challenges in this new domain derive from the fact that one cannot rely on "small" alterations to the data in the embedding process. Any alteration is going to necessarily be significant. This discrete characteristic of the data requires discovery

of fundamentally new bandwidth channels and associated encoding algorithms. Additionally, since the associated data types do not have fixed, well defined semantics (as compared to multimedia) and may be designed for machine ingestion, identifying the available "bandwidth" for watermarking becomes as important as the actual encoding algorithms.

Our solution proves to be resilient to various important classes of attacks, including subset selection and random item(s) alterations etc. The main contributions of this chapter include: (i) the proposal and definition of the problem of watermarking categorical data, (ii) the discovery and analysis of new watermark embedding channels for relational data with categorical types, (iii) the design of novel associated encoding algorithms.

The Chapter is structured as follows. In Section 4.2 we present our main data and adversary models. Section 4.3 introduces the main solution and outlines alternatives. Section 4.4 analyzes aspects of our algorithms and proposes improvements for particular scenarios. It also discusses the inherent algorithm vulnerability to data altering attacks. Section 4.5 outlines our experimental setup and results. Section 4.6 concludes.

4.2 Model

We choose to keep our model concise but representative. Our data schema includes a set of discrete attributes $\{A, B\}$ and a primary data key K, not necessarily discrete. Any attribute $X \in \{A, B\}$ can yield a value out of n_X possibilities. (e.g., city names, airline names). Thus our schema is (K,A,B).

4.2.1 Notation

Let the number of tuples in the database be N. For any categorical attribute X we naturally have $b(n_X) \leq b(X)$. Let $T_j(X)$ be the value of attribute X in tuple j. Let $\{a_1, ..., a_{n_A}\}$ the discrete potential values of attribute A. These are

distinct and can be sorted (e.g., by ASCII value). Let $f_A(a_j)$ be the normalized (to 1.0) occurrence frequency of value a_j in attribute A. $f_A(a_j)$ models the de-facto occurrence probability of value a_j in attribute A.

4.2.2 The Adversary

There is a set of attacks that can be performed by evil Mallory with the purpose of defeating the watermark while preserving the value in the data. Moreover these perceived attacks may be the result of normal use of the data by the intended user. In order to be effective, the watermarking technique has to consider these scenarios and be able to survive them. In Chapter 3 we outlined attacks in the numeric relational data framework. Here we discuss challenges specifically associated with categorical data types.

A1. Horizontal Data Partitioning Mallory can randomly select and use a subset of the original data set that might still provide value for its intended purpose.

A2. Subset Addition Mallory adds a set of tuples to the original data. This addition is not to significantly alter the useful (from Mallory's perspective) properties of the initial set versus the resulting set.

A3. Subset Alteration Altering a subset of the items in the original data set such that there is still value associated with the resulting set. In the categorical data framework, subset alteration is intuitively quite expensive from a data-value preservation perspective. One has also to take into account semantic consistency issues that become immediately visible because of the discrete nature of the data.

A4. Vertical Data Partitioning In this attack, a valuable subset of the attributes are selected (by vertical partitioning) by Mallory. The mark has to be able to survive this partitioning. The encoding method has to feature a certain attribute-level property that could be recovered in such a vertical partition of the data. We believe that while vertical data partitioning attacks are possible and also very likely in certain scenarios, often value is to be found in the association between

a set of relation attributes. These attributes are highly likely to survive such an attack, as the final goal of the attacker is to produce a still-valuable result.

A5. Attribute Remapping If data semantics allow it, re-mapping of relation attributes can amount to a powerful attack that should be carefully considered. In other words, if Mallory can find an, at least partial, value-preserving mapping from the original attribute data domain to a new domain, a watermark should hopefully survive such a transformation. The difficulty of this challenge is increased by the fact that there naturally are an infinity of transformations available for a specific data domain. Determining a value-yielding one is both data and consumer dependent. This is thus an intractable task for the generic case. One special case is primary key re-mapping. In Section 4.4.5 we discuss the particular case of bijective mappings.

Given the attacks above, several properties of a successful solution surface. For immunity against **A1**, the watermark has to be embedded in overall data properties that survive subset selection. If the assumption is made that the attack alterations do not destroy the value of the data, then **A3** should be defeat-able by embedding the primitive mark in resilient global data properties. Since it adds new data to the set, defeating **A2** seems to be the most difficult task, as it implies the ability to identify potential uses of the data (for the attacker). This is especially so in the case of categorical data where we suspect the main attack will focus not as much on expensive data alterations but more on data addition.

4.3 Categorical Data

The discrete nature of our data domain results in an inherent limitation in the associated entropy. In order to enable watermarking, we first aim to discover appropriate embedding channels. Then we propose new encoding methods, able to leverage the newly discovered bandwidth.

4.3.1 Challenges

Given our research in the numerical domain, the first impulse was to build an extension of it for non-numeric attributes. This would start by establishing a mapping between the non-numeric domain and a numeric one, followed by a translation of the input data A to a set of numbers N (after all, any data can be represented as a string of bits). In the next step, the numeric watermarking method is deployed on the translated data N and a watermarked version of it (N') is obtained. If the mapping features certain properties (e.g., has an inverse), the algorithm can then translate this watermarked version (N') back into the original data domain (according to the inverse mapping) and produce A', a watermarked version of A. The assumption here is that there exists a mapping that is stable and is suitable. For example in the case of A being an attribute containing multimedia JPEG images (possibly under the form of BLOB fields), this mapping might be exactly the DCT [1] (or a combination of the significant DCT coefficients). The detection algorithm will function similarly, by translating the suspected watermarked input data to the numeric domain (using the inverse transform) and deploying the numeric detection process on the translation.

Unfortunately, depending on the actual data domain, non-numeric relational data will feature a different set of data value and quality metrics, and associated uses. For many applications, this will make it difficult to directly apply the above idea. Let us consider for example the case of categorical data. Important new challenges are associated with watermarking in this domain. One cannot rely on "small" alterations to the data in the embedding process as any alteration is going to necessarily be significant. Changing DEPARTURE_CITY from "Chicago" to "Bucharest" is likely to affect the data quality of the result more then a simple change in a numeric domain. There are no "epsilon" changes in this domain. This discrete characteristic of the

[1] Discrete Cosine Transform. A frequency-domain transform used in the compression process of JPEG images, quantifying an image into a set of coefficients.

data requires discovery of fundamentally new bandwidth channels and associated encoding algorithms.

4.3.2 Bandwidth Channels

If the discrete attribute A has a finite set of possible values (n_A), unless this value is really high, the associated $log_2(n_A)$ bits entropy is not going to be enough for direct-domain embedding of a reasonable watermark length/convince-ability. For example in the case of a departure cities attribute, a value of $n_A = 16000$ is going to yield only 14 bits.

In the case of categorical data however (and not necessarily in any other continuous data domain) there exists a natural, solid semantic association between A, the rest of the schema's categorical attributes (e.g., B) and the data's primary key K. This association derives from the fact that in most cases there exists no concept of "minor" changes. *Any* change is going to necessarily be significant (e.g., change departure city from "Chicago" to "San Jose"). A comparatively large potential encoding bandwidth can be found in these *associations between categorical attributes* (including possibly the primary key). We propose to make use of it in our embedding. Additionally, while direct-domain embedding does not seem to have enough entropy potential, we will leverage a related dimension, the *value occurrence frequency-transform*, (attribute frequency histogram) as an additional (or alternate) encoding channel.

Our next objective is to provide an embedding method that is able to resiliently hide information in the attribute association outlined above (while preserving guaranteed data distortion bounds) and then, if necessary, augment it with a direct-domain watermark.

Note: Given the discrete nature of the data domain, as outlined above, any watermark-related data alteration is going to necessarily be significant, e.g., changing "Chicago O'Hare" into "Chicago Metropolitan". This is why, intuitively, one

would desire to minimize the number of such alterations while maximizing encoding resilience. Additionally, if there exists a certain distance metric in the attribute value domain, then another possible desiderata would be to upper bind some function of the distances of all performed alterations (see Section 4.4.7).

4.3.3 Algorithms

Surviving vertical partitioning attacks is important and requires a careful consideration of the attribute association used in the embedding process. Selecting the appropriate attributes is challenging as one has to determine many possible valuable features to be found in the data that would still be preserved after vertical partitioning. This is why we propose an initial user-level assessment step in which a set of attributes are selected that are likely to survive vertical partitioning attacks (see Section 4.3.4 for an extended discussion). In the extreme case (i), just one attribute and the primary key are going to survive. A milder alternative (ii) assumes that several (e.g., two) categorical attributes and the primary key survive the partitioning process. Apparently, a watermarking method for (i) presents the disadvantage of a direct primary key-dependency. In Section 4.3.4 we further expand on this.

Let us propose an encoding method for (i), in which we encode a watermark in the bandwidth derived from the association between the primary key and a categorical attribute A. In Section 4.4 we analyze (ii).

Mark Encoding

At mark encoding time we assume the following input: a relation with at least a categorical type attribute A (to be watermarked), a watermark wm and a set of secret keys (k_1, k_2) and other parameters (e.g., e) used in the embedding process. The algorithm starts by discovering a set of "fit" tuples determined directly by the association between A and the primary relation key K. These tuples are then considered for mark encoding.

wm_embed(K,A,wm,k_1,k_2,e,ECC)

$wm\_data \leftarrow ECC.encode(wm, wm.len)$

for $(j \leftarrow 1; j < N; j \leftarrow j + 1)$

if $(H(T_j(K), k_1)) \bmod e = 0)$ **then**

$t \leftarrow set\_bit(H(T_j(K), k_1), 0,$

$wm\_data[H(T_j(K), k_2)])$

$T_j(A) \leftarrow a_t$

wm_embed_alt(K,A,wm,k_1,e,ECC)

$wm\_data \leftarrow ECC.encode(wm, wm.len)$

$idx \leftarrow 0$

for $(j \leftarrow 1; j < N; j \leftarrow j + 1)$

if $(H(T_j(K), k_1) \bmod e = 0)$ **then**

$t \leftarrow set\_bit(H(T_j(K), k_1), 0, wm\_data[idx])$

$T_j(A) \leftarrow a_t$

$embedding\_map[T_j(K)] \leftarrow idx$

$idx \leftarrow idx + 1$

return $embedding\_map$

Figure 4.1. Relational Data with Categorical Types: (a) Embedding Algorithm (b) Alternative using embedding map (bit size adjustments omitted)

Step One. We say that a tuple T_i is "fit" for encoding iff $H(T_i(K), k_1) \bmod e = 0$, where e is an adjustable encoding parameter determining the percentage of considered tuples [2] and k_1 is a secret $max(b(N), b(A))$-bit key. In other words, a tuple is considered "fit" if its primary key value satisfies a certain secret criteria [3].

Note on Error Correction. Because often the available embedding bandwidth $\frac{N}{e}$ is greater than the watermark bit-size $|wm|$, we can afford the deployment of an error correcting code (ECC) that, upon embedding takes as input a desired watermark wm and produces as output a string of bits wm_data of length $\frac{N}{e}$ containing a redundant encoding of the watermark, tolerating a certain amount of bit-loss, $wm\_data = ECC.encode(wm, \frac{N}{e})$. At decoding time, the ECC takes as input (a potentially altered) wm_data and produces the (most likely) corresponding wm, $wm = ECC.decode(wm\_data, |wm|)$. There are a multitude of error

[2]The set of fit tuples contains roughly $\frac{N}{e}$ elements. The parameter e can be controlled at embedding time to adjust the trade-off between the level of data alteration and mark resilience. See Section 4.4.4 for a more detailed analysis.

[3]Similar criteria are found in various frameworks, such as [35].

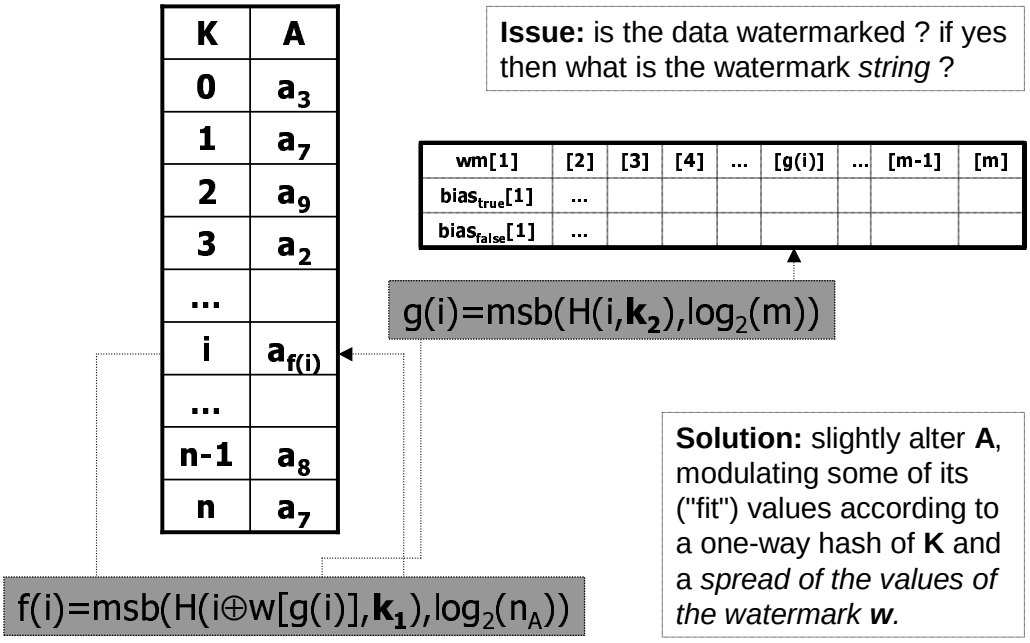


Figure 4.2. Relational Data with Categorical Types: Overview of multi-bit watermark encoding.

correcting codes to choose from. As this does not constitute the main contribution of this research, in our implementation we deploy majority voting codes. Let $wm\_data[i]$ be the i-th bit of wm_data. Thus, before embedding, our algorithm starts by deploying the error correcting code first to compute the bits to be embedded $wm\_data = ECC.encode(wm, \frac{N}{e})$.

Step Two. For each "fit" tuple T_i, we encode one bit by altering $T_i(A)$ to become $T_i(A) = a_t$ where

$$t = set\_bit(msb(H(T_i(K), k_1), b(n_A)), 0, wm\_data[msb(H(T_i(K), k_2), b(\frac{N}{e}))])$$

, where k_2 is a secret key $k_2 \neq k_1$. In other words, we are generating a secret value of $b(n_A)$ bits (depending on the primary key and k_1) and then force its least significant bit to a value according to a corresponding (random, depending on the primary key and k_2) position in wm_data.

Note: The use of a second different key here ensures that there is no correlation between the selected tuples for embedding (selected also by k_1) and the corresponding bit value positions in wm_data (selected by k_2). Such a correlation would potentially cause certain bits to be never considered in the embedding process. In summary, the new attribute value is selected by the secret key k_1, the associated relational primary key value and a corresponding bit from the watermark data wm_data.

The "fitness" selection step provides several advantages. On the one hand this ensures the secrecy and resilience of our method, on the other hand, it effectively "modulates" the watermark encoding process to the actual attribute-primary key association. Additionally, this is the place where the cryptographic safety of the hash one-wayness is leveraged to defeat court-time attacks in which Mallory claims that the data in dispute is not actually watermarked but that rather certain values for k_1,k_2 were searched for to yield the watermark.

Note: When computing t (i.e., selecting a new value for $T_i(A)$) there can be (arguably rare) cases when we select the same wm_data bit to embed. The pseudo-random nature of $H(T_i(K), k_2)$ guarantees on average that a large majority of the

bits in *wm_data* are going to be embedded at least once. The ulterior step of error correction can tolerate such small changes.

Alternately, we could keep an on-the-fly hash-table/mapping (with $(\frac{N}{e})$ entries, see Figures 4.1 (b) and 4.3 (b)) between $T_i(K)$ values and the actual considered bit index in *wm_data*. This mapping can be used at detection time to accurately detect *all wm_data* bits. In this case, also we do not require an extra watermark bit selection key (k_2). Although we use this alternative in our implementation, for simplicity and conciseness reasons we are not going to discuss it here.

The advantage of using $H(T_i(K), k_2)$ in selecting the *wm_data* bit to embed becomes clear when we discuss data loss alterations. Because the selected bit is directly related only to the *currently* considered tuple, this method naturally survives subset selection and data addition attacks. More on this in Section 4.5.

While it does a good job in watermark embedding, data alteration is an expensive operation because it effectively destroys valuable data. There are also other data transformations that we can make use of, each with a different degree of associated data distortion and benefits. For a discussion on an alternative (i.e., data addition) see Section 4.4.6.

Mark Decoding

In the decoding phase we assume the following input: the potentially watermarked data, the secret keys k_1, k_2 and e. We then use the same criteria for discovering "fit" tuples. That is, we say that a tuple T_i is "fit" for encoding iff $H(T_i(K), k_1)$ mod $e = 0$.

The first aim of the decoding algorithm is to discover the embedded *wm_data* bit string. For each "fit" tuple T_i, with $T_i(A) = a_t$, we set

$$wm\_data[msb(H(T_j(K), k_2), b(\frac{N}{e}))] = t\&1$$

wm_dec(K,A,k_1,k_2,e,ECC)
  for ($j \leftarrow 1$; $j < N$; $j \leftarrow j + 1$)
    if ($H(T_j(K), k_1) \bmod e = 0$) **then**
    determine t such that $T_j(A) = a_t$
    $wm\_data[msb(H(T_j(K), k_2), b(\frac{N}{e}))] = t\&1$
  $wm \leftarrow ECC.decode(wm\_data, wm.length)$
  return wm

wm_dec_alt(K,A,k_1,e,ECC,$embed\_map$)
  for ($j \leftarrow 1$; $j < N$; $j \leftarrow j + 1$)
    if ($H(T_j(K), msb(k, b(K))) \bmod e = 0$) **then**
    determine t such that $T_j(A) = a_t$
    $wm\_data[embed\_map[T_j(K)]] = t\&1$
  $wm \leftarrow ECC.decode(wm\_data, wm.length)$
  return wm

Figure 4.3. Relational Data with Categorical Types: (a) Decoding Algorithm (b) Alternative using embedding map

Once wm_data (possibly altered) is available, the error correcting mechanism is invoked to generate the ("closest", most likely) corresponding watermark $wm = ECC.decode(wm\_data, |wm|)$.

4.3.4 Multiple Attribute Embeddings

The above encoding method makes use of the bandwidth present in the association between the primary key and the categorical type attribute A. It does *not* touch the primary key attribute but rather relies on modulating A through minor alterations (and data additions, see Section 4.4.6).

In the following we extend this algorithm to provide more generality and resilience, in particular to attacks of the type A4 (vertical data partitions). In a possible attack scenario Mallory partitions the data in such a way as to preserve only two attributes and no primary key. Moreover, if one of the remaining attributes can act as a primary key, this partitioning results in no duplicates-related data loss (in the two attributes).

Defeating this scenario leads to a natural extension. Instead of relying on the association between the primary key and A, the extended algorithm considers *all*

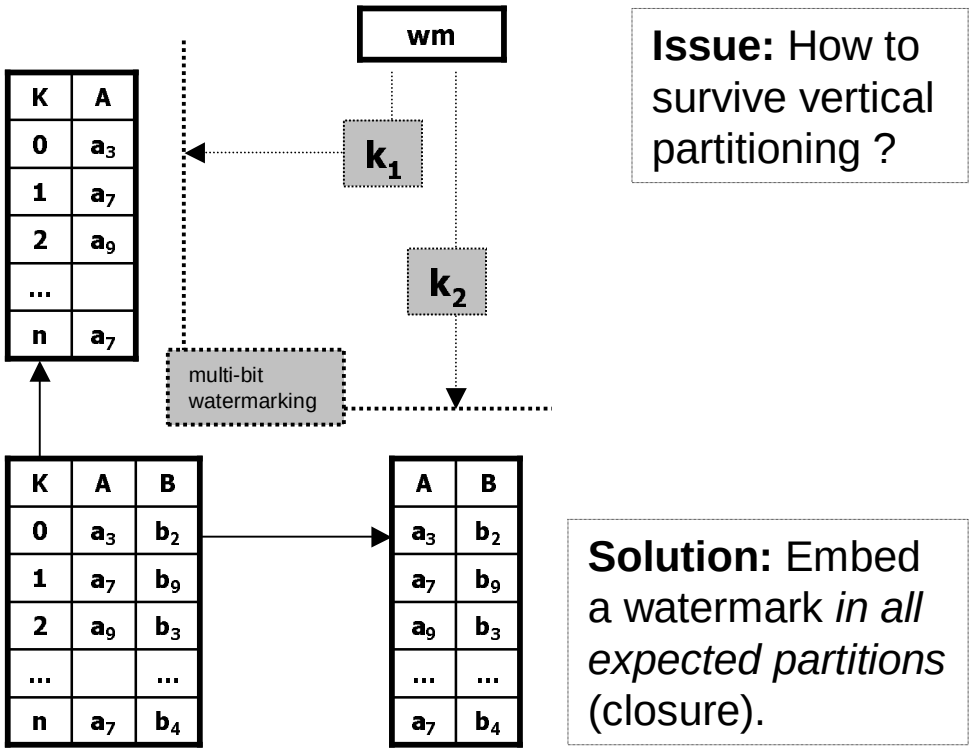


Figure 4.4. Relational Data with Categorical Types: Defeating vertical partitioning.

pairs [4] of attributes and embeds a watermark separately in *each* of these associations. In other words, if the original watermarking method read $mark(K, A)$ for a schema composed of the primary key K and A, in the case of a (K, A, B) schema we apply the watermark several times, for example $mark(K, A)$; $mark(K, B)$; $mark(A, B)$. In each case, we treat one of the attributes as a primary key (see Section 4.3.3), while maintaining the rest of the algorithm in place. This provides protection against A4 attacks and allows for more resilience in the rest of the scenarios (as there are more rights "witnesses" to testify). In addition, it effectively "breaks" the previous algorithm's dependency of the primary key.

Several issues need to be resolved. One apparent problem is the issue of interference. If we watermark the pair (K, A) and then aim to watermark (K, B) everything seems to work out fine as the modified attributes A, B are different. With the exception of semantic consistency issues that would need to be handled (as they would also be in the initial case, see Section 4.4) the two encodings seem to be independent. But in the case of additionally watermarking the pair (A, B), modifying B suddenly interferes with the modifications occurred in the (K, B) case.

Although the level of interference is likely to be very low [5], there exists a solution to this problem. Maintaining a hash-map at watermarking time, "remembering" modified tuples in each marking pass, allows the algorithm (extended accordingly) to avoid tuples and/or values that were already considered.

Additionally, when considering the association between two attributes A, B as an encoding channel for a watermark, if values in B were already altered during a previous encoding, instead of deploying $mark(A, B)$ (which would result in further alterations to B), we propose the deployment of $mark(B, A)$. While still encoding the mark in the association between A and B, by modifying A (assumed un-modified

[4] For simplicity we consider pairs for now, but believe that an arbitrary number of attributes could be considered.

[5] As the probability of the same tuple to be considered again in the second encoding is low, especially in large data sets, see Section 4.4.4 for a related analysis.

yet, otherwise it doesn't matter anyway) we effectively "spread" the watermark throughout the entire data, increasing its level of resilience.

Moreover, if data constraints allow, we propose watermarking each and every attribute pair by first building a closure for the set of attribute pairs over the entire schema that minimizes the number of encoding interferences while maximizing the number of pairs watermarked.

Note: The discrete nature of categorical attributes complicates the watermarking process of a pair (A, B) in which a categorical attribute A is used as a primary key (in the initial algorithm). In the extreme case, A can have just one possible value which would upset the "fit" tuple selection algorithm. It remains to be investigated if a pair-closure can be constructed over the schema such that no categorical attributes are going to be used as primary key place-holders. See Section 4.4.1 for a related analysis and extension.

4.4 Discussion

4.4.1 Correlation Attacks

The solution above features a particular issue of concern in certain cases of multi-attribute embeddings where two non-key attributes are used in the encoding, i.e., mark(A,B). Because of the correlation between the watermarking alteration (the newly selected value $T_i(B) = b_t$) and its actual location (determined by the fitness selection, $H(T_i(A), k_1)$ and e), sometimes Mallory can mount a special attack with the undesirable result of revealing some of the mark bit embedding locations. This occurs if the fitness criteria decides that a particular value of A yields a tuple fit and that value of A appears then in multiple (statistically significant) different tuples. This is possible only if A is not a primary key but rather another categorical attribute (with repeating duplicate values).

The attack then proceeds by first realizing that, despite the one-wayness of the deployed hash function $H()$, in fact, A is the only variable that determines *both*

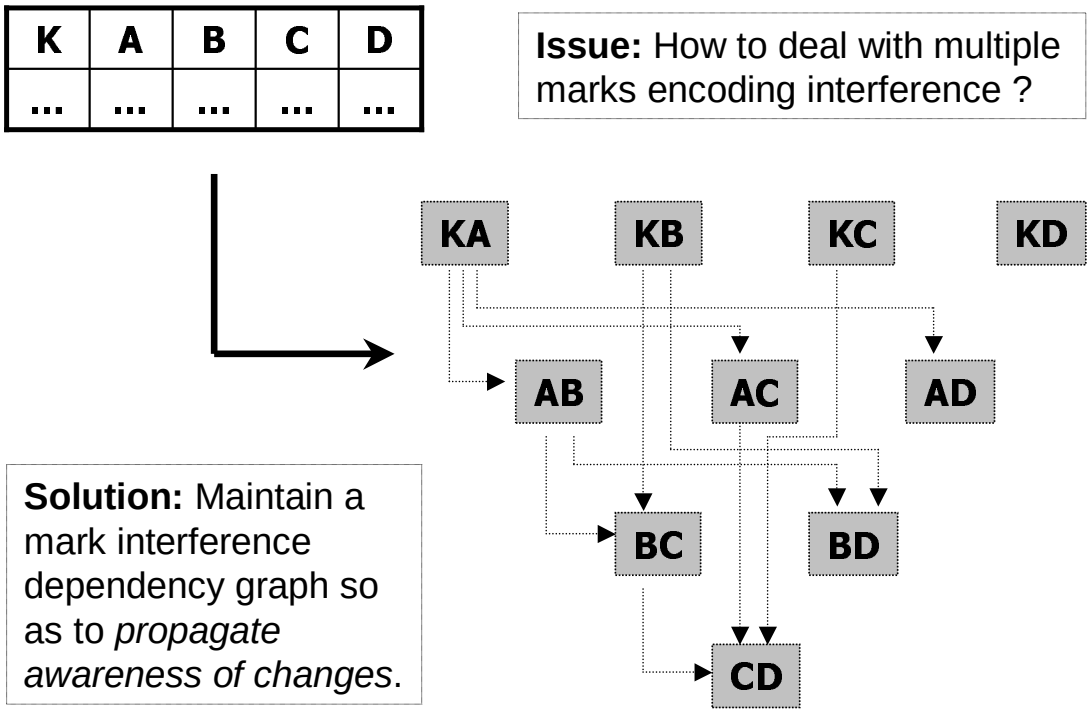


Figure 4.5. Relational Data with Categorical Types: Handling multiple marks interference.

the bit embedding location (tuple "fitness") *and* its value. If Mallory is able to detect this correlation for potential candidates (tuples), it would quickly lead to exposing some of the ones carrying a watermark bit. But how does he check for the correlation? Mallory can simply build a set of "hash buckets" for each separate value of A (yielding the same value of $H(T_i(A), k_1)$) and count (for all matching tuples) if there is a statistical bias for a certain value (e.g., "true") of the least significant bit of t (see Section 4.3.3). If such a bias is discovered, e.g., if a majority of LSB values are "true" then Mallory suspects (rightfully so) that the respective tuples are "fit" and a watermark bit of "true" (for example) is embedded in those locations. Mallory can now obliterate the embedding in these tuples by randomization, leading to a loss of the corresponding watermark bit.

In other words, if, as a result of the extension proposed in Section 4.3.4, two attributes A and B are used in a watermarking process, $mark(A, B)$, (and the data set contains many "fit" tuples with repeated values for attribute A), Mallory can discover the association between the individual unique values of A and the bit-embeddings in B. He can then use this discovered association to randomize the embeddings and effectively remove the corresponding watermark bits. Thus, the problem lies here in the correlation between the actual bit location and the bit value, correlation induced by the fact that a single variable (A) determines both of these and this variable can have repeated values for different tuples, allowing for a "bucket counting" attack as described above.

One solution to this issue would be to simply restrict the fitness selection criteria for tuples so as to only include the ones with attribute $T(A)$ values that do not have a significant number of repeats throughout the data.

A more radical idea would be to simply search the space of potential k_1 values until the fitness criteria results in selecting tuples with different values for A.

These solutions work only if A indeed does contain a significant number of non-repeating values in the data. In any case, this problem has the potential to introduce a significant reduction in available encoding bandwidth. In the extreme where A only

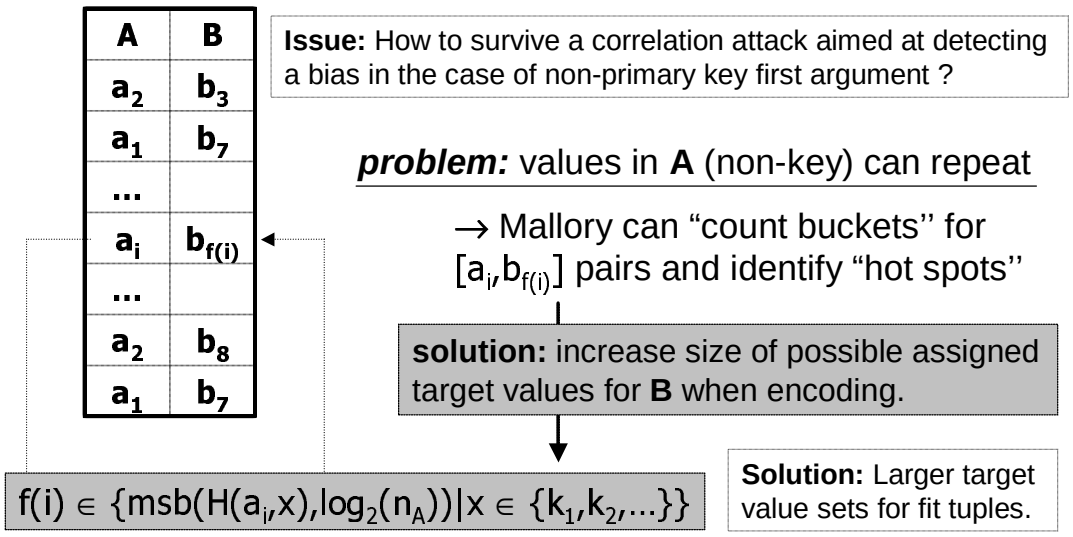


Figure 4.6. Relational Data with Categorical Types: Defeating correlation attacks.

contains a few unique values (e.g., true/false as in a binary attribute), it cannot be used as the first argument in a $mark(X, Y)$ operation.

Note: An interesting extreme scenario occurs in the case of binary attributes, i.e., attributes with only two possible values. If for example A can only take values in the $\{0, 1\}$ set (e.g., quite likely in many data mining sets), intuitively, it cannot be used as a first argument in an $mark(X, Y)$ operation (that is, it cannot play the pseudo primary key role) This is so due to the fact that the fitness selection criteria (controlled by X) is going to either (i) fail to discover any fit tuples (i.e., resulting in no available encoding bandwidth), (ii) possibly partition the entire data set into two, namely a subset of "fit" tuples (e.g., corresponding to the $T_i(A) = 1$ values) and the rest or (iii), deem all tuples fit in which case the entire data set is going to be altered in a potential watermarking operation. This case is interesting because it illustrates the impact of n_X (the cardinality of the first attribute's possible values set) on the $mark(X, Y)$ operation.

Multiple Embeddings

There exists a refinement that would overcome many of the above. What if the actual watermark were to consist of a combination of several different embeddings, each in turn being an encoding using a different k_1 value. While each of these "low impact" encodings would be weaker than the original solution, their combined "sum" can be made (arguably) arbitrarily strong(er), by increasing their number. At the same time correlation attacks would be defeated.

If for example we embed two watermarks with different keys k_1 and k_1', the correlation attack cannot be performed "across" the encodings, as the $H(T_i(A), k_1)$ and $H(T_i(A), k_1')$ values are not going to be consistent with each other, making "bucket counting" impossible.

One price to pay is the amount of computation required at each step to decode all of the potential watermarks. Another issue of (arguably) minor concern could

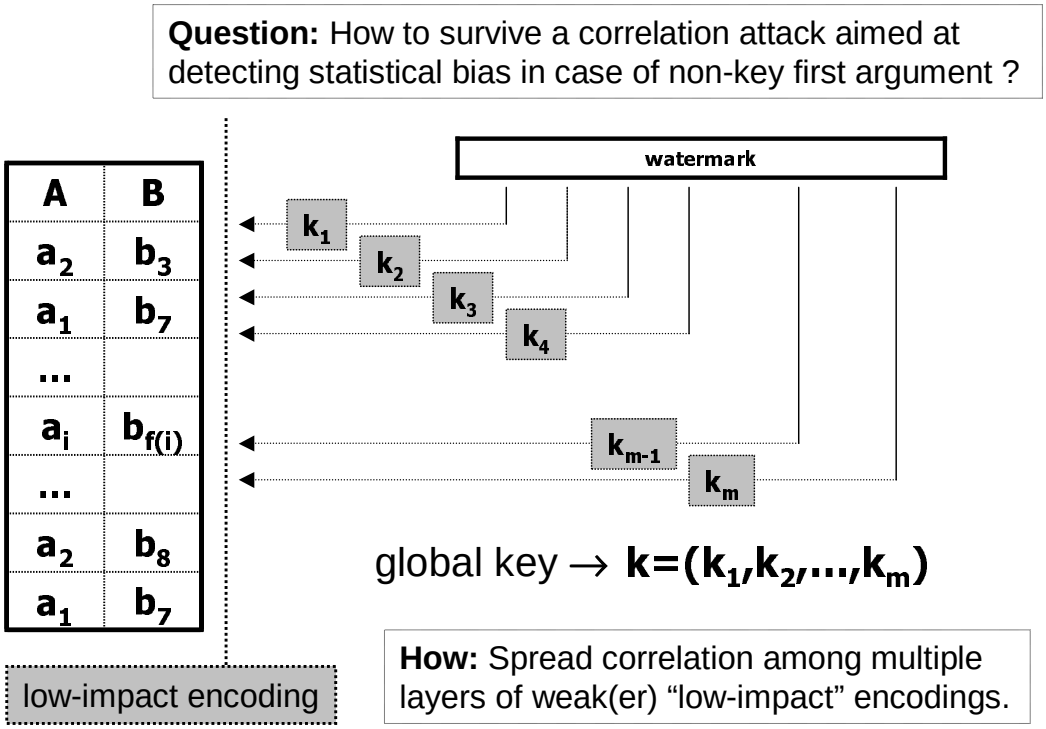


Figure 4.7. Relational Data with Categorical Types: Defeating correlation attacks revisited (multiple embeddings).

be the fact that the same tuple might be considered in multiple encodings, in which case some of these will suffer a mark loss. This is however of low probability due to the random nature of the cryptographic hashes used. Other provisions could be deployed to handle this situation.

4.4.2 On-the-fly Quality Assessment

In the relational framework it is important to preserve structural and semantic properties of the data. Because by its very nature, watermarking alters its input, we have to provide a mechanism ensuring that these alterations are not degrading the data beyond usability. Preserving data quality requires the ability to express and enforce data constraints. Sometimes it is undesirable or even impossible to directly map higher level semantic constraints into low level (combined) change tolerances for individual tuples or attributes [6]. The practically infinite set of potential semantic constraints that can be desired/imposed on a given data set makes it such that versatile, "data goodness" (i.e., semantically) assessment methods are required. Thus, we propose to extend the marking algorithm with semantic data constraints awareness.

We introduced and successfully analyzed this idea (an instance of *consumer driven* encoding) in Chapter 3. Each property of the database that needs to be preserved is written as a constraint on the allowable change to the dataset. The watermarking algorithm is then applied with these constraints as input and re-evaluates them continuously for each alteration. A backtrack log (see Figure 4.8) is kept to allow undo operations in case certain constraints are violated by the current watermarking step. Due to space considerations, we are not going to elaborate on this further. In the following we are going to focus on the encoding method itself.

[6]It should be noted that not all constraints of the database need to be specified. A practical approach would be to begin by specifying a upper bound on the percentage of allowable data alterations. Further semantic or structural constraints that the user would like to preserve can be added to these basic constraints.

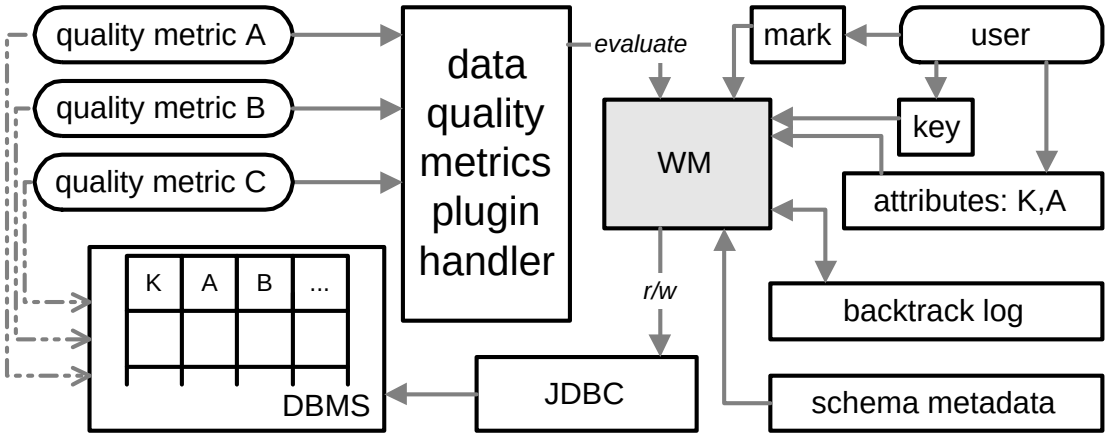


Figure 4.8. Relational Data with Categorical Types: Data quality is continuously evaluated. A backtrack log aids undo operations in cases where the watermark embedding would violate quality constraints (see also Chapter 3.

4.4.3 Vertical Partitioning Revisited

While most vertical partitioning attacks can be handled by a multiple attribute embedding solution as described in Section 4.3.4, consider an extreme vertical partitioning attack scenario in which Mallory only preserves a single (categorical) attribute A (a multi-set).

An intuitive assumption is that n_A (the number of possible values in A) is much smaller than N, thus A (by itself) is naturally containing many duplicate values. Because there is probably very little value associated with knowing the set of possible values of $\{a_1, ..., a_{n_A}\}$, the main market-able value of A (in Mallory's eyes) is (arguably) to be found in one of the only remaining characteristic properties, namely the value occurrence frequency distribution $[f_A(a_i)]_{i\in(1,n_A)}$. If we could devise an alternative watermark encoding method for this set we would be able to associate rights also to this aspect of the data, thus surviving this extreme partitioning attack.

Note: If the data value occurrences are uniformly distributed (often unlikely, imagine airport or product codes) distinguishing among these values will not work and (arguably) there is nothing one can do to watermark that result.

In Chapter 3 we introduced a watermarking method for numeric sets that is able to minimize the absolute data alteration in terms of distance from the original data set. We propose to apply this method here to embed a mark in the occurrence frequency distribution domain. One concern we should consider is the fact that in the categorical domain we are usually interested in minimizing the *number* of data items altered whereas in the numeric domain we aim to minimize the absolute data change. It is fortunate that, because $[f_A(a_i)]_{i\in(1,n_A)}$ are values modeling occurrence frequency, a solution minimizing absolute data change in this (frequency) domain naturally minimizes the *number* of items changed in the categorical value domain. Other concerns include issues such as multi-mark interference (with the other encodings), which can be solved by an approach similar to the one in Section 4.3.4 using

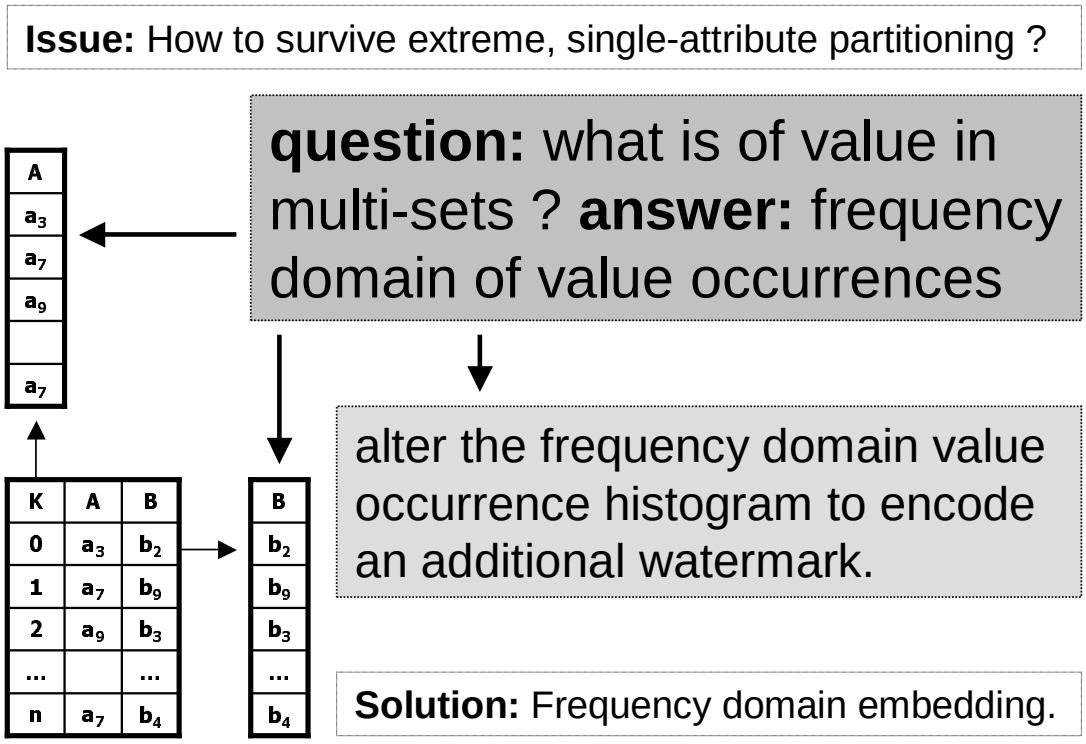


Figure 4.9. Relational Data with Categorical Types: Handling extreme multi-set partitioning.

embedding markers and/or deploying a mark interference graph/tree (see Figure 4.5) and watermarking in a bottom-up traversal order of it.

4.4.4 False Positives and Vulnerability to Attacks

In order to fight false-positive claims in court we ask: *What is the probability of a given watermark of length $|wm|$ to be detected in a random data set of size N?* The natural assumption is that $|wm| < \frac{N}{e}$ (enough bandwidth).

It can be easily proven that this probability is $(\frac{1}{2})^{|wm|}$. In case multiple embeddings are used (e.g., majority voting) and all available bits are utilized, this probability decreases even more to $(\frac{1}{2})^{\frac{N}{e}}$. For example, in the case of a data set with $N = 6000$ tuples and with $e = 60$, this probability is approximately 7.8×10^{-31}.

In the absence of additional information, Mallory, faced with the issue of destroying the watermark while preserving the value of the data, has only one alternative available, namely a random attack (here we discuss data alteration attacks). We ask: *what is the probability of success of such an attack?* In other words, if an attacker randomly alters a total number of a data tuples and succeeds in each case to flip the embedded watermark bit with a success rate p, *what is the probability of success of altering at least $r, r < a$ watermark bits in the result, $P(r, a)$?* This metric illustrates the relationship between attack vulnerability and embedding bandwidth. It can be shown that

$$P(r, a) = \sum_{i=r}^{a} \binom{a}{i} \times p^a \times (1 - p)^{a-i}$$

Remember that only every e-th tuple (on average) is watermarked, thus Mallory effectively attacks only an average of $\frac{a}{e}$ tuples actually watermarked. If $r > \frac{a}{e}$ then $P(r, a) = 0$. In the case of $r < \frac{a}{e}$ we have the corrected version

$$P(r, a) = \sum_{i=r}^{\binom{a}{e}} \binom{\frac{a}{e}}{i} \times p^i \times (1 - p)^{(\frac{a}{e})-i} \tag{4.1}$$

Consider $r = 15$, $p = 70\%$ (it is quite likely that when Mallory alters a watermarked tuple, it will destroy the embedded bit), $a = 1200$ (20% of the tuples are altered by the attacker[7], $|wm| = 10$ and $e = 60$ ($|wm\_data| = 100$).

Because we have an effectively binomial distribution experiment with $X_i = 1$, with probability p and $X_i = 0$, with probability $1 - p$. $E[X_i] = p$, $var(X_i) = E[X_i^2] - (E[X_i])^2 = ... = p \times (1 - p)$, by using the central limit theorem [38], we can derive that $f(\sum X_i)$, where

$$f(\sum X_i) = \frac{\sum X_i - \frac{a}{e} \times p}{\sqrt{\frac{a}{e} \times p \times (1 - p)}} \tag{4.2}$$

effectively behaves like a normal distribution $N(0, 1)$ (when $\frac{a}{e} \times p \geq 5$ and $\frac{a}{e} \times (1-p) \geq 5$). In other words, the probability that $(\sum X_i) > r$ (attack altering at least r bits) can be rewritten as the probability of $f(X_i) > f(r)$. Because of the normal behavior of $f(x)$ (we know $f(r)$) we can estimate this probability by normal distribution table lookup. Thus, we get $P(15, 1200) \approx 31.6\%$.

If we assume that the error correcting code tolerates an average of $t_{ecc} = 5\%$ alterations to the underlying data and that the alteration propagation is uniform and stable [8] then the final watermark is going to incur only an average fraction of

$$(\frac{r}{\frac{N}{e}} - t_{ecc}) \times \frac{|wm|}{|wm\_data|}$$

alteration. In our case this is only 1.0%, corresponding to an average of 1.0 bit in the watermark. *Thus in order to modify one bit in the watermark Mallory has to alter at least* 20% *of the data and even then has only a success rate of* 31.6% ! This analysis was done in a highly attack-favorable scenario in which error correction can only handle 5% alterations in *wm_data*.

Because data alteration is expensive, naturally we aim to minimize the number of altered tuples in the watermarking process. If we define attack vulnerability as the

[7]This is likely a highly value-damaging operation overall. Such an attack is unlikely because Mallory cannot afford destroying the data beyond use. We present it for illustration purposes.

[8]These are intuitive terms we use to denote the fact that if one bit in *wm_data* is altered above the t_{ecc} bound then a stable average of $\frac{|wm|}{|wm\_data|}$ are altered in the resulting error corrected watermark $wm = ECC.decode(wm\_data, |wm|)$.

probability $P(r,a)_1$ to succeed in altering one bit in the final watermark (wm), and the number of altered tuples is defined by the ratio $\frac{N}{e}$, we ask: *what is the relationship between the required number of fit tuple encodings (i.e., available bandwidth) and attack vulnerability?* In other words, what is the minimum number of alterations we have to allow (and perform) in the watermarking phase that would guarantee a certain upper bound on the overall attack vulnerability. While this relationship is somewhat defined by equation (4.1), we are interested here in an actual estimate for a likely scenario.

If we assume that Mallory cannot afford to modify more than 10% of the data items ($a = 600$) and we set a maximum tolerable threshold $\tau = 10\%$ for $P(r,a)_1$ ($P(r,a)_1 < \tau$), let us compute the minimum required e to guarantee these bounds (the other values are as above). By using equation (4.2) and doing a normal distribution table lookup we derive that (for $\tau = 10\%$) we have to satisfy

$$\frac{r - \frac{a}{e} \times p}{\sqrt{\frac{a}{e} \times p \times (1-p)}} = 1.28$$

which results in $e \approx 23$. In other words, we have to alter only $\approx 4.3\%$ of the data to guarantee these bounds !

4.4.5 Bijective Attribute Re-mapping

Consider the scenario of an attack in which the categorical attribute A is re-mapped through a bijective function to a new data domain. In other words, the $\{a_1, ..., a_{n_A}\}$ values are going to be mapped into a different set $\{a'_1, ..., a'_{n_A}\}$. The assumption here is that from Mallory's perspective, the re-mapped data still features enough value that can be banked upon [9].

The problem of remapping becomes clear in the mark detection phase when, after tuple fitness selection, the bit decoding mechanism will fail, being unable to determine t such that $T_j(A) = a_t$. It will instead determine a t value that maps to

[9]Even more, Mallory could sell a secret secure black-box "reverse mapper" together with the re-mapped data to third parties, still producing revenue.

the $\{a'_1, ..., a'_{n_A}\}$ value set. Thus our main challenge is to discover the mapping (or a major part of it) and apply its inverse in the detection phase.

Unless the items in the initial set $\{a_1, ..., a_{n_A}\}$ feature a peculiar distinguishing property, intuitively this task is impossible for the general case, as there are a large number of possible mappings. Nevertheless, over large data sets we argue that a such distinguishing property might exist, namely the value occurrence frequency for the items in $\{a_1, ..., a_{n_A}\}$. We propose to sample this frequency in the suspected (remapped) dataset and compare the resulting estimates ($E[f_A(a'_j)]_{j\in(1,n_A)}$) with the known occurrence frequencies ($(f_A(a_j))_{j\in(1,n_A)}$). Next, we sort both sets and associate items by comparing their values. For example if the closest value to $E[f_A(a'_i)]$ (in the set $E[f_A(a'_j)]_{j\in(1,n_A)}$) is $f_A(a_j)$ (in the set $(f_A(a_j))_{j\in(1,n_A)}$), then we add $i \rightarrow j$ to the inverse mapping to be used at watermark decoding time.

4.4.6 Data Addition

While it does a good job in watermark embedding, data alteration is an expensive operation because it effectively destroys valuable data. There are also other data transformations that we can make use of, each with a different degree of associated data distortion and benefits. In particular, *data addition* seems to be a promising candidate. Intuitively it features a much lower data distortion rate (no actual alterations) and thus presents potentially higher benefits. On the other hand there likely exists an upper bound on the number of tuples that can be added to the data. Let p_{add} be the upper bound on the allowed additional percentage of tuples to be added. We propose that in addition to the initial data-altering step, we artificially "inject" watermarked tuples that conform to the "fitness" criteria (while conforming to the overall data distribution, in order to preserve stealthiness).

But isn't data addition of "fit" tuples inhibited by the one-way nature of the used cryptographic hash? Not exactly. Because e effectively "reduces" the fitness criteria testing space to a cardinality of e, we can afford to massively produce random tuple

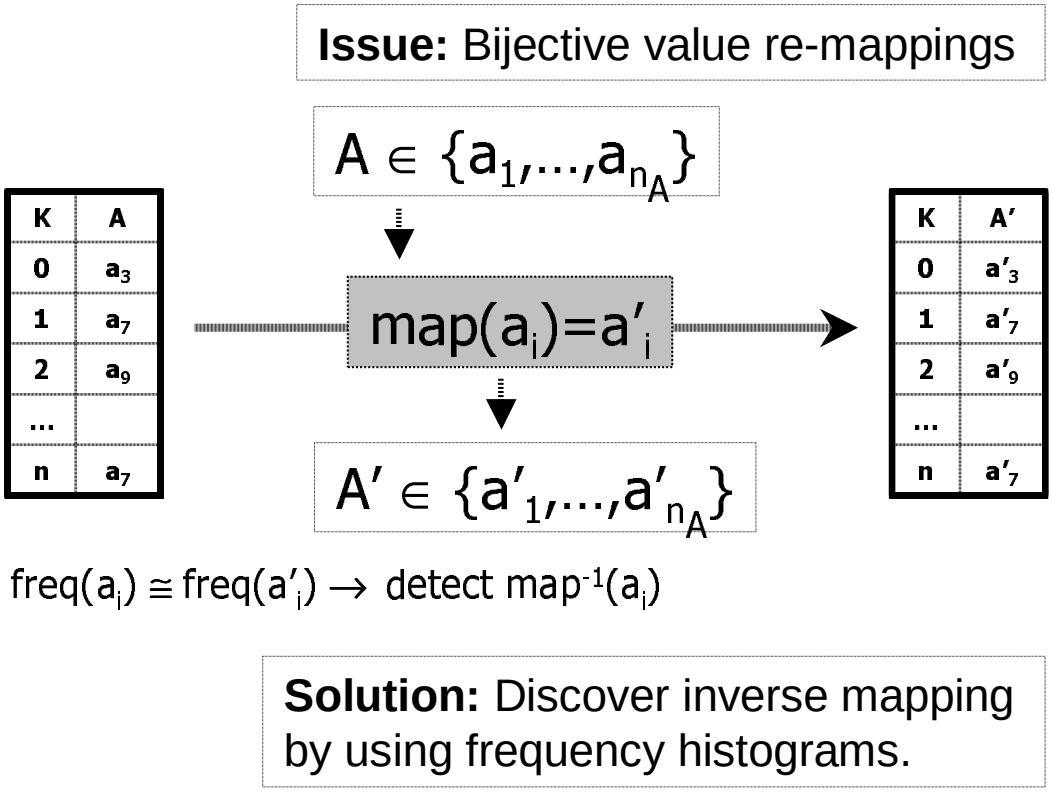


Figure 4.10. Relational Data with Categorical Types: Handling attribute remapping.

values (within the appropriate attribute data domain) and test for "fitness". On average (depending on the randomness of the tuple producing mechanism), one in every e tuples should conform (as the values are evaluated modulo e).

If a percentage of p_{add} artificially produced tuples are to be added to the data, the watermark is effectively enforced with an additional $p_{add} \times N$ bits. See Section 4.4.4 for an analysis on the impact of watermark bits on the encoding resilience.

4.4.7 Minimizing Alteration Distance

An interesting problem to consider is the case when, for a given "fit" tuple, certain alterations would be preferred to others. For example, if the given attribute represents airport names, intuitively, it is likely that an alteration changing "Chicago, O'Hare" into "Las Vegas" produces more damage overall than one that would result in "Chicago, Metro". In other words, what if there exists a certain distance metric model for the values within a categorical attribute and the encoding is to minimize a (e.g.,) sum (for each change in the data) of these associated alteration distances.

This scenario can be easily dealt with through the design of a data quality plugin that continuously evaluates the amount of damage already performed and allows only the alterations that conform. However such a solution suffers from several issues. For one, upon encountering a "fit" tuple that doesn't conform to this quality metric, the only alternative available to the data quality plugin is to simply veto the proposed modification. Depending on the restrictiveness of the desired alteration distance upper bound, this will possibly yield only few tuples that can be used in the marking process, thus resulting in a reduced bandwidth. Additionally, a condition like the one above (sum of all alteration distances to not exceed maximum) is not easy to implement so as to result in a resilient embedding. A trivial implementation would simply allow all alterations until the sum exceeds a certain upper bound. But in this case, if the data is read sequentially, this will result in a data set watermarked only in a portion at its beginning, after which most of the alterations will be denied.

A smarter version would be to "spread" the allowed modifications throughout the data, which in turn would require the data quality plugin to allow encoding only to some of the good "fit" tuples, to "save" some of the allowed alteration distance for future ones. This would hopefully result in the coverage of the entire data.

But what if we could modify the encoding method so as to naturally accommodate such a case? We propose to modify the tuple alteration criteria to result not in one value for t in the selection of $T_i(A) = a_t$ (see Section 4.3.3) but rather in an entire set of $\gamma < n_A$ potential candidate values. Let us define

$$t' = set\_bit(msb(H(T_i(K), k_1), b(n_A - \gamma)), 0, wm\_data[msb(H(T_i(K), k_2), b(\frac{N}{e}))])$$

and

$$S_{t'} = \{a_t | msb(t, n_A - \gamma) = msb(t', n_A - \gamma)\}$$

Then, at each encoding step for a fit tuple i, its new value $T_i(A)$ is selected from $S_{t'}$ so as to minimize the alteration distance.

In other words, we "divide" the set of potential discrete values for A into $\frac{n_A}{\gamma}$ subsets of γ elements (sharing the first $b(n_A - \gamma)$ most significant bits). Each t' value then selects a certain subset $S_{t'}$, and the final corresponding $T_i(A)$ is constructed by selecting the "closest" (in terms of the above discussed alteration distance metric) data value in $S_{t'}$.

The number and size of the subsets are controlled through a choice of an appropriate γ. To the extreme, if $\gamma = n_A - 1$, only two such subsets exist (i.e., one subset with values having the most significant bit of their index "true" and the other subset with the rest).

4.4.8 Blindness, Incremental Updates and Streams

Our watermarking method is blind, in that it doesn't require the original data in the detection process. This is important, because it is un-realistic to assume the original data available after a longer time elapses, especially in the case of massive data sets.

The method supports incremental updates naturally. As updates occur to the data, the resulting tuples can be evaluated on the fly for "fitness" and watermarked accordingly. Therefore, the encoding also (gracefully) handles data streaming in which tuples are available only in a one-pass streaming model with limited storage space and processing power. This makes it ideal for a transparent deployment scenario in which a black box "sits" at the data "exit" point and continuously watermarks outgoing information.

4.4.9 Multi-Layer Self-Reinforcing Marks

The above solution (and any symmetric/single-key watermarking method) is vulnerable to a scenario in which Mallory (who might have participated in a public court hearing) finds out the key that was used to watermark a given Work. He can then use the key to remove the watermark and have illicit access to the original version.

Efforts in asymmetric multi-media watermarking [39] [40] [41] [42] [43] [44] [45] deploy different keys for detection (public) and embedding (secret). The design of an asymmetric version of our solution is to be subject to future research. Now however, we propose a draft idea that seems to handle this scenario reasonably well.

The idea is to simply embed multiple weak watermarks with different secret keys and reveal in court only a certain subset of these, enough to satisfy the convinceability requirements. Having these keys would only enable the removal of the corresponding watermarks and nothing more. The data will still feature the remaining ones, hopefully enough for the next court hearing instance.

Yet another idea would be to embed multiple self-reinforcing pairs of watermarks $(w_1, w_2)_i$ with different keys $(k_1, k_2)_i$ such that altering w_2 will result in enforcing w_1. The feasibility, details and benefits of such a method are to be subject to further investigation.

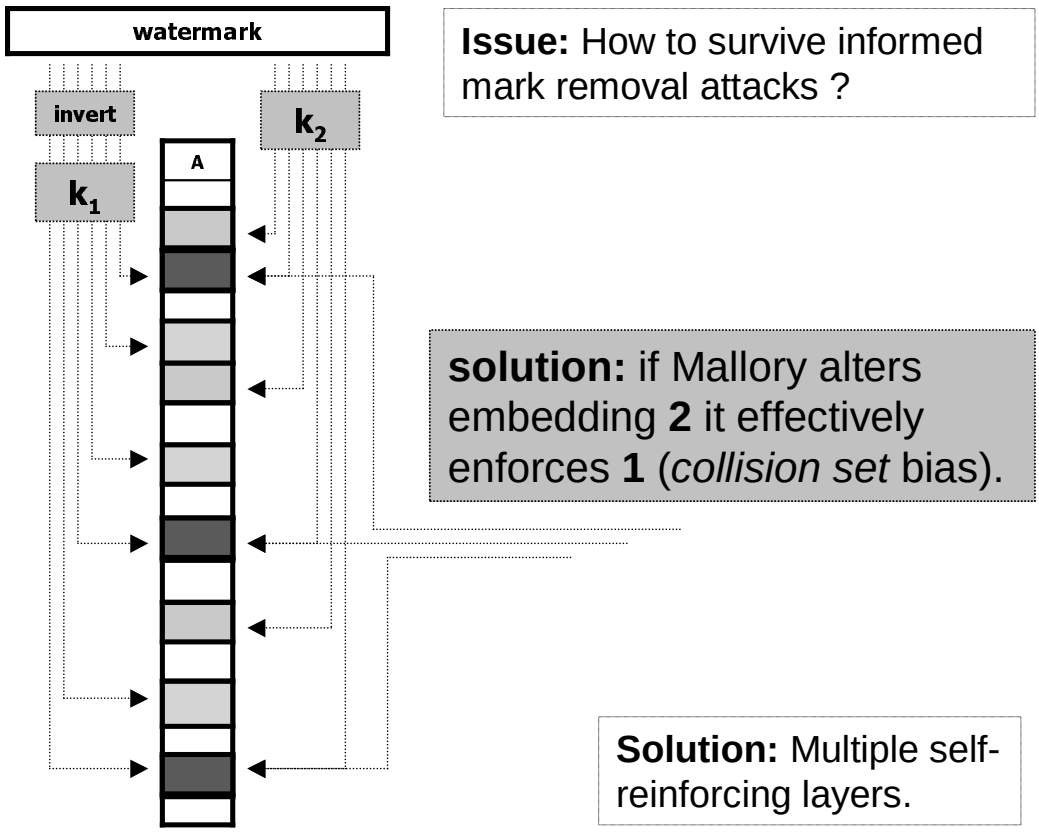


Figure 4.11. Relational Data with Categorical Types: Handling an informed Mallory.

4.5 Experiments

We implemented a Java proof-of-concept of the watermarking algorithm and deployed it on categorical attributes in the Wal-Mart Sales Database. The Wal-Mart Sales Database contains most of the information regarding item sales in Wal-Mart stores nationwide. In the following we present some of our experiments in watermarking categorical attributes within this database. Our experimental setup included access to the 4 TBytes Wal-mart data, (formerly) hosted on a NCR Teradata machine, one 1.6GHz CPU Linux box with Sun JDK 1.4 and 384MB RAM. The amount of data available is enormous. For example, the *ItemScan* relation contains over 840 million tuples. For testing purposes, we deployed our algorithm on a randomly selected subset of size equal to a small percentage of the original data size (e.g., just a maximum of 141000 tuples for relation *UnivClassTables.ItemScan*). The relational schema included the attributes:

```
Visit_Nbr INTEGER PRIMARY KEY
Item_Nbr INTEGER NOT NULL
```

To illustrate and test our watermarking algorithm we chose Item_Nbr, a categorical attribute, uniquely identifying a finite set of products. The watermark considered was 10 bits long, all the presented data is the result of an averaging process with 15 passes (each seeded with a different key), aimed at smoothing out data-dependent biases and singularities.

In the first experiment we analyzed the behavior of the embedded watermark in the presence of massive data alterations. As the attack size grows (random alterations to the data), the watermark distortion increases. The error correction mechanism (majority voting in this case) does a good job in error recovery. This is particularly so in the case of random alterations to the underlying data, the only available data altering attack option as discussed in Section 4.4.4. Figure 4.12 (a), depicts this phenomena for two values of e.

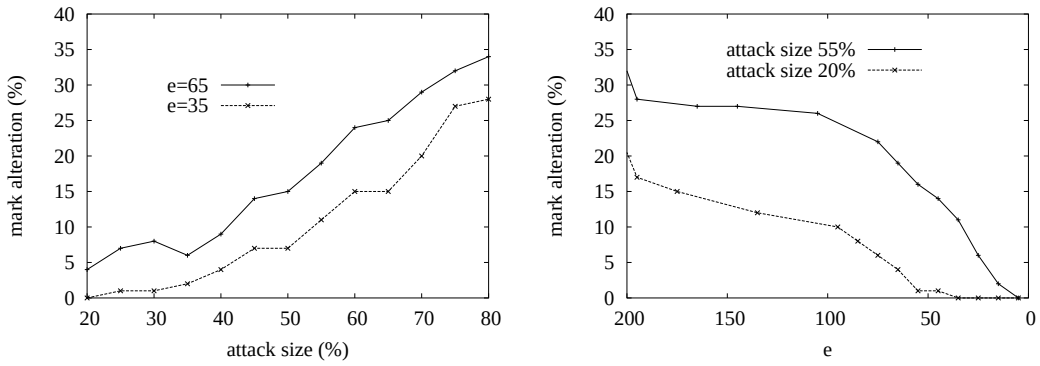


Figure 4.12. Relational Data with Categorical Types: (a) The watermark degrades gracefully with increasing attack size ($e = 65$). (b) More available bandwidth (decreasing e) results in a higher attack resilience.

The next experiment aims at exploring the relationship between the amount of data modifications required in the watermarking phase and a minimum guaranteed watermark resilience. It can be seen in Figure 4.12 (b) that as e increases (decreasing number of encoding modifications) the vulnerability to random alteration attacks increases accordingly. This illustrates the trade-off between the requirement to be resilient and the preservation of data quality (e.g., fewer alterations).

Figure 4.13 (a) represents the composite surface for both experiments.

A very interesting phenomena can be seen here. The surface presents some "crevasse" like shapes (also to be found in Figure 4.12 (b)) at the boundaries. We believe these to be the result of a less than perfect implementation of the MD5 hash used which results in a bias in the fitness selection process.

An experiment analyzing resilience to data loss is depicted in Figure 4.13 (b). We observe here the compensating effect of error correction. Compared to data alteration attacks, the watermark survives even better with respect to the attack size (in this case loss of data).

In Figure 4.14 various aspects of the implementation execution times are illustrated. In (a) it becomes clear that there seems to be a minimal dependency of e of

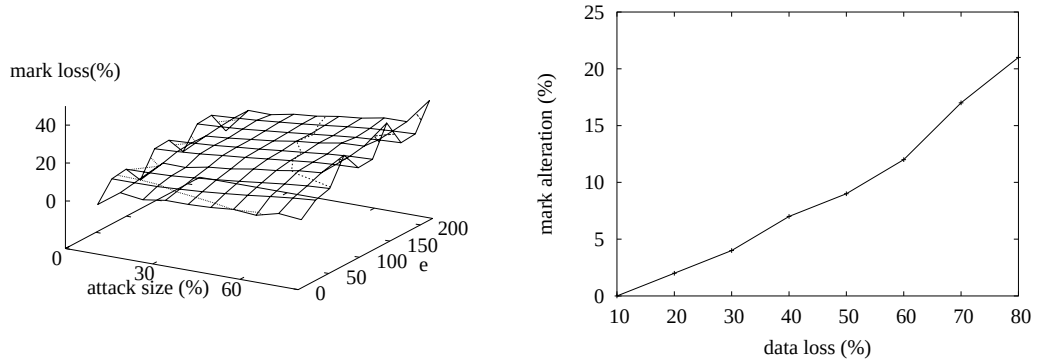


Figure 4.13. Relational Data with Categorical Types: (a) The watermark alteration surface with varying c (watermark modifications) and attack size. Note the lower-left to upper-right tilt. (b) The watermark degrades almost linearly with increasing data loss.

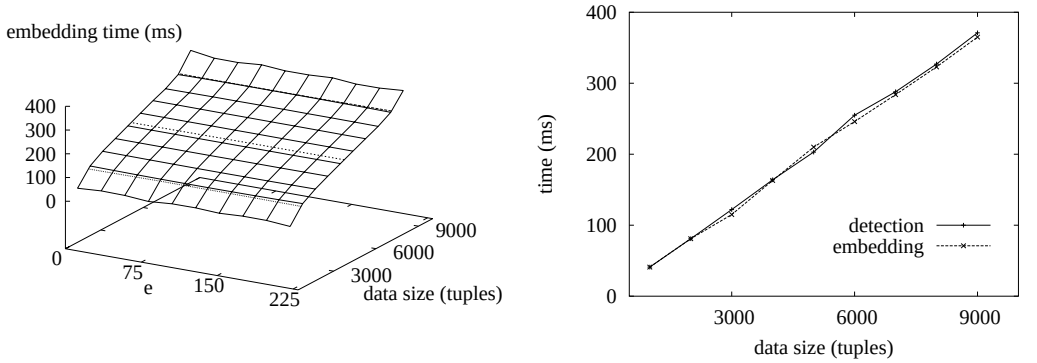


Figure 4.14. Relational Data with Categorical Types: (a) Embedding time dependency as a function of e and N. (b) Detection time requirements are similar to embedding and linear in the size of the data.

the embedding (detection graph is virtually identical, thus omitted) times. Execution time seems to be mainly linear in the data size, as also expected. In (b) it can be seen that detection yields almost identical times as the embedding process. The linear dependency is clear. An average of 25K tuples can be processed per second by our proof of concept implementation. We expect a speed-up of orders of magnitude in an optimized industry-level version.

4.6 Conclusions

In this chapter we defined the problem of watermarking categorical data. We proposed a solution and analyzed it both in theory and in practice. We outlined a set of extensions (e.g., an alternative for occurrence frequency encoding to survive extreme vertical partitioning attacks) and discussed main associated attacks and challenges. We implemented a proof-of-concept for our algorithm and deployed it in experiments on real Wal-Mart sales data. Our method proves (experimentally and by analysis) to be extremely resilient to both alteration and data loss attacks, for example tolerating up to 80% data loss with a watermark alteration of only 25%. Published research results of this work include [10] and [11].

Various issues remain to be explored. Additive watermark attacks need to be analyzed and handled. Also, while the concept of on-the-fly quality assessment (see Section 4.4.2) has a good potential to function well, as confirmed also by experiments in Chapter 3, another interesting avenue for further research would be to augment the encoding method with direct awareness of semantic consistency (e.g., classification and association rules). This would likely result in an increase in available encoding bandwidth, thus in a higher encoding resilience. One idea would be to define a generic language (possibly subset of SQL) able to naturally express such constraints and their propagation at embedding time.

5 DISCRETE STREAMING DATA. SENSOR STREAMS.

Today's world of increasingly dynamic environments naturally results in more and more data being available as fast streams. Applications such as stock market analysis, environmental sensing, web clicks and intrusion detection are just a few of the examples where valuable data is streamed. Often, streaming information is offered on the basis of a non-exclusive, single-use customer license. One major concern, especially given the digital nature of the valuable stream, is the ability to easily record and potentially "re-play" parts of it in the future. If there is value associated with such future re-plays, it could constitute enough incentive for a malicious customer (Mallory) to record and duplicate data segments, subsequently re-selling them for profit. Being able to protect against such infringements becomes a necessity.

In this chapter we introduce the issue of rights protection for discrete streaming data through watermarking. This is a novel problem with many associated challenges including: operating in a finite window, single-pass, (possibly) high-speed streaming model, surviving natural domain specific transforms and attacks (e.g.,extreme sparse sampling and summarizations), while at the same time keeping data alterations within allowable bounds. We propose a solution and analyze its resilience to various types of attacks as well as some of the important expected domain-specific transforms, such as sampling and summarization. We implement a proof of concept software (wms.*) and perform experiments on real sensor data from the NASA Infrared Telescope Facility at the University of Hawaii, to assess encoding resilience levels in practice. Our solution proves to be well suited for this new domain. For example, we can recover an over 97% confidence watermark from a highly downsampled (e.g., less than 8%) stream or survive stream summarization (e.g., 20%) and random alteration attacks with very high confidence levels, often above 99%.

5.1 Introduction

In this chapter we introduce and study the problem of watermarking discrete (sensor) streams data, which to the best of our knowledge, has not been addressed. Streaming data sources represent an important class of emerging applications [46] [47]. These applications produce a virtually endless stream of data that is too large to be stored directly. Examples include output from environmental sensors such as temperature, pressure, brightness readings, stock prices etc. Recent efforts in the broader area of streaming data deal with the database challenges of its management [48] [49] [50] [51].

Our work on discrete/itemized data types (e.g. Chapters 3, 4) and related efforts [35], all rely upon the availability of the entire dataset during the watermarking process. While this is generally a reasonable assumption, it does not hold true for the case of streaming data [46]; since the streamed data is typically available as soon as it is generated, it is desirable that the watermarking process be applied immediately on subsets of the data. Additionally, the attack and transformation models in existing research does not apply here. For example a process of summarization would defeat any of the above schemes. Yet another difference from previous research is the lack of a "primary key" reference data set, an essential, required, part in both Chapter 3 and [35]. Due to these differences, earlier work on watermarking relational data sets is not applicable to streams.

But why is watermarking streaming data important? Couldn't we simply watermark the data once it is stored? This surely would work and enable rights protection for the stored result. But it would not deter a malicious customer (Mallory), with direct stream access, to duplicate segments of the stream and re-sell them or simply re-stream the data for profit. The main rights protection scenario here (see Figure 5.1) is to prevent exactly such leaks from a licensed customer.

Our contributions include (i) the proposal and definition of the problem of watermarking sensor streams, (ii) the discovery and analysis of new watermark embedding

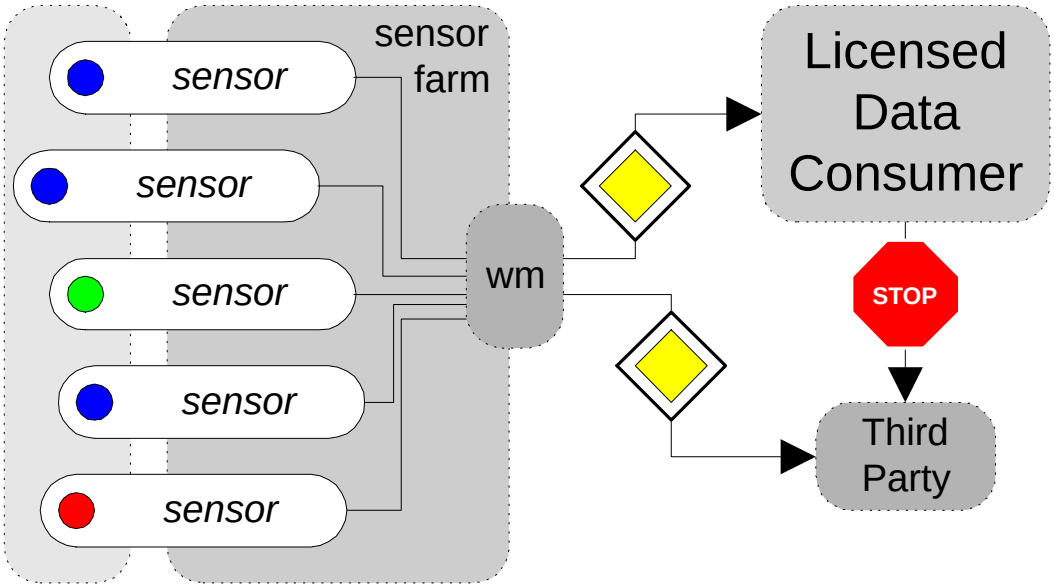


Figure 5.1. Discrete Streaming Data: Sensor Streams Watermarking Scenario.

channels for such data, (iii) the design of novel associated encoding algorithms, (iv) a proof of concept implementation of the algorithms and (v) their experimental evaluation. The algorithms introduced here prove to be resilient to important domain-specific classes of attacks, including stream re-sampling, summarization (replacing a stream portion by its average value) and random changes. For example, sampling the data stream down to less than 8% still yields a court-time confidence of watermark embedding of over 97%. Summarization (e.g., 20%) and random data alterations are also survived very well, often with a false-positive detection probability of under 1%.

The chapter is structured as follows. Section 5.2 outlines the major challenges in this new domain. It proposes an appropriate data and transform model, discusses associated attacks and overviews related work. In Section 5.3 an initial solution is provided. Further resilience-enhancing improvements and attack handling capabilities are gradually introduced in Section 5.4. Section 5.5 analyzes the ability to convince in court to survive attacks and natural domain transformations. Section

5.6 presents **wms.***, a proof-of-concept java implementation of our solution; our experimental setup and results are introduced. Section 5.7 concludes.

5.2 Challenges

5.2.1 The Adversary

As outlined above, the nature of most "fast" time-series data applications imposes a set of strict requirements on any on-the-fly data processing method, such as watermarking. For one, it has to be able to keep up with the incoming data rate and, the fact that only a finite window of memory (e.g., of size ϖ, see below) is available for processing makes certain history-dependent computations difficult or simply impossible. At the same time, metrics of quality can only be handled within this space; any preservation constraints can be formulated only in terms of the current available data window. Including any history information will come at the expense of being unable to store as much new incoming data. In summary, the nature of this new domain is such that only a limited amount of time is available to be spent in processing each incomming data item and only a limited number of such items can be considered at a time (limited window).

Moreover, the effectiveness of any rights protection method is directly related to its ability to deal with normal domain specific transformations as well as malicious attacks. There are several transforms relevant in a streaming scenario, including the following: (A1) summarization, (A2) sampling, (A3) segmentation (we would like to be able to recover a watermark from a finite segment of data drawn from the stream), (A4) linear changes [1] (there might be value in actual *data trends*, that Mallory could still exploit, by scaling the initial values), (A5) addition of stream values and (A6) random alterations.

While we discuss most of these and other attacks in the next sections, let us note here that a scaling attack (A4) can be handled by an initial normalization step,

[1] Taken care of by the initial normalization step.

e.g., yielding values in the $(-0.5, 0.5)$ interval. If the data distribution is assumed to be known, normalization can also be easily performed at detection time. If data distribution is not known, then we propose an initial "discovery" run in which data is simply read and a reference data distribution is constructed and updated on the fly. This will yield a certain data-dependent inaccuracy in the initial phases of detection but will likely quickly converge as more data is read. If detection is performed offline on a static segment of data, normalization is eased by the ability to read the data multiple times. In the following, unless specified otherwise, we consider this normalization step to have been performed, yielding a normalized version of the stream, with values in the interval $(-0.5, 0.5)$. To survive sampling and other minor stream transformations, several improvements to the normalization process are proposed in Section 5.3.2. with respect to (A5), Mallory is bound to add only a limited amount of data (in order to preserve the value in the original stream) and these new values are to be drawn from a similar data distribution, lest they become easy to identify in the detection process as not conforming to the known original distribution. Also, it can be seen that (A6) is naturally modeled by a combination of (A2) and (A5).

Apparently, data re-sorting might be also of concern as an attack. At closer inspection however, if value is to be found in the stream, it is assumed to lie in two aspects of it: the data values and their relative ordering. In other words, in most applications, a recorded stream (even sampled) is (arguably) only valuable if its replay is preserving the relative ordering of the values (with exception of some extreme cases). Re-ordering the sequence of values in the stream is going to significantly alter its core value. For example consider the case of stock market data. If the evolution of a given stock is modeled by a stream of values, a recording of it is only valuable if the sequence ordering is preserved. Also, significant on-the-fly data re-sorting, is simply not possible given the finite processing window and speed assumptions. In this chapter we consider data re-sorting to significantly alter the core value of the

data set, not a successful attack choice Mallory would consider. Our method does however handle minor data re-sorting gracefully.

5.2.2 Model

For the purpose of simplicity let us define a data stream as an (almost) infinite timed sequence of ($x[t]$) values "produced" by a set of data sources of a particular type (e.g., temperature sensors, stock market data). $x[t]$ is a notation for the value yielded by our source(s) at time t. Unless specified otherwise, lets denote a stream as $(x[], \varsigma)$ where ς is the number of incoming data values per time unit (*data rate*) [2].

Note: While a time-stamp t can be assigned naturally to each and every data value when produced by a data source, it often becomes irrelevant after such domain-specific transformations as sampling and summarization which destroy the exact association between the value $x[t]$ and the time it was initially generated, t. Thus, the notation $x[t]$ is merely used to distinguish separate values in the stream and is not intended for suggesting the preservation of the time-stamp-value in the resulting stream (which is ultimately just a sequence of values).

Any stream processing is necessarily both time and space bound. The time bounds derive from the fact that it has to keep up with incoming data. We are going to model the space bound by the concept of a window of size ϖ. At each given point in time, no more than ϖ of the stream ($x[t]$) values (or equivalent amounts of arbitrary data) can be stored locally, at the processing point. Unless specified otherwise, as more incoming data becomes available, the default behavior of the window model is to "push" older items out (i.e., to be transmitted further, out of the processing facility) and "shift" the entire window (e.g., to the right) to free up space for new entries.

For simplicity, without sacrificing generality, for the remainder of the chapter we are going to assume the stream values being normalized in the interval $(-0.5, +0.5)$.

[2]The proposed solution does not rely on any characteristic of the actual stream data rate. For space and simplicity purposes in this chapter we are discussing streams with fixed data rates.

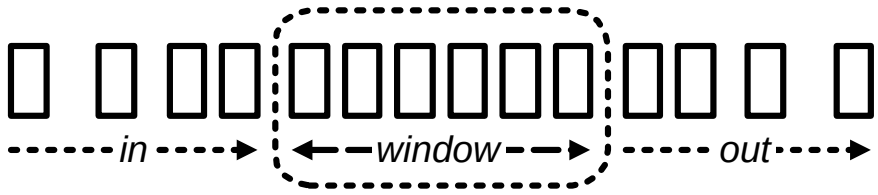


Figure 5.2. Discrete Streaming Data: Stream Processing is necessarily bound in both time (stream rate) and space (window).

This assumption does not need to hold in general but instead just simplifies the task of understanding the algorithms.

For the purpose of the current framework, we define the *uniform random sampling* of degree χ of a stream $(x[], \varsigma)$ as another stream $(x'[], \varsigma')$ with

$$\varsigma' = \frac{\varsigma}{\chi}$$

such that for each sample data item $x'[t]$, there exists a contiguous subset of $(x[])$, $(x[t_1], x[t_2])$ such that

$$x'[t] \in (x[t_1], x[t_2])$$

and

$$\{x'[t-1], x'[t+1]\} \nsubseteq (x[t_1], x[t_2])$$

and t is uniformly distributed in (t_1, t_2). In other words, it is constructed by randomly choosing one value out of every χ values in the original. A subtle variation of *uniform random sampling* is the case when $x'[t]$ is not randomly chosen but rather always the first element in it's corresponding χ sized subset (e.g., $t = t_1$). We call this *fixed random sampling* of degree χ.

We define the *summarization* of degree ν of a stream $(x[], \varsigma)$ as another stream $(x'[], \varsigma')$ with

$$\varsigma' = \frac{\varsigma}{\nu}$$

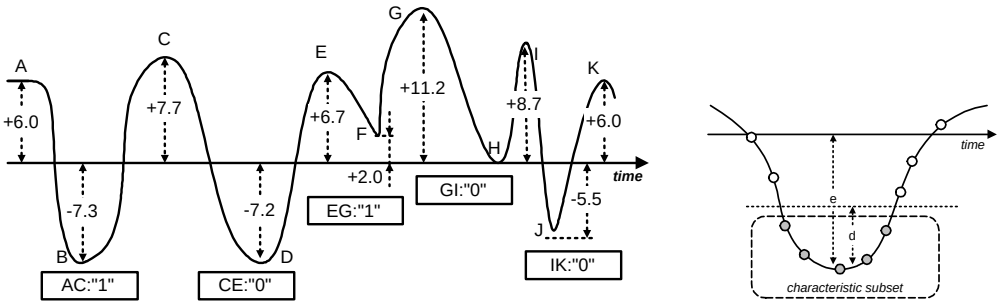


Figure 5.3. Discrete Streaming Data: (a) A sample stream. If all the extremes are considered to be major, then the resulting label bits for K are shown (for $\varrho = 2$) (b) δ-Radius characteristic subset of extreme η.

such that for each two adjacent sample data items $x'_1[t], x'_2[t+\nu]$, there exist two contiguous, adjacent, non-overlapping ν-sized subsets of $(x[])$, $(x[t-\nu+1], x[t-\nu+2], ..., x[t])$, $(x[t+1], x[t+2], ..., x[t+\nu])$ such that

$$x'_1[t] = \frac{\sum_{i\in(1,\nu)} x[t-\nu+i]}{\nu}$$

and

$$x'_2[t+\nu] = \frac{\sum_{i\in(1,\nu)} x[t+i]}{\nu}$$

. In other words, for a continuous chunk of ν elements from the original stream summarization outputs its average.

Note: Various other similar aggregates could be envisioned here (e.g., min/max, most likely value). We believe that in the current scope, considering averaging summarization is both illustrative and qualitatively identical and does not complicate the analysis too much

We define an *extreme* η in a stream simply as either a local minimum or local maximum value. We define the extreme's *characteristic subset of radius* δ, noted $\Xi(\eta, \delta)$ (see Figure 5.3 (b)), as the subset of stream items forming complete "chunks", immediately adjacent to η and conforming to the following criteria: item i, with value

$v_i \in \Xi(\eta, \delta)$ iff $|\eta - v_i| < \delta$ and all the items "between" i and the extreme η, also belong to $\Xi(\eta, \delta)$.

A *major extreme* of degree χ and radius δ is defined as an extreme η such that at least one item in $\Xi(\eta, \delta)$ can be found in *any* uniform random sampling of degree χ of $(x[])$ (i.e., some items in $\Xi(\eta, \delta)$ "survive" sampling of χ degree). For example, in Figure 5.3 (a), intuitively, it seems likely that extremes such as F,I and J have a smaller chance of surviving sampling than C,E or G. This is so because of the temporal shape of the stream's evolution. C,E,G seem to yield characteristic subsets much "fatter" than F,I,J. Intuitively, δ needs to be chosen such that the characteristic subsets are of an average size greater than χ (to handle a sampling of degree χ).

To model the "fluctuating" nature of a stream, let $\varepsilon(\chi, \delta)$ be the average number of stream data items encountered/read per *major extreme* (i.e., before encountering a major extreme) of degree χ and radius δ. $\frac{1}{\varepsilon(\chi,\delta)}$ defines the average "frequency of major extremes" in terms of the number of observed data items.

5.2.3 Related Work

Could existing results in non-media data sets watermarking such as relational data, discussed in Chapters 3 and 4 or related work [35] (discussed in Section 3.7), be adapted to the new domain? These solutions requires access to the entire data set in an almost random access model, which is certainly not possible here at embedding time. Also, these efforts seem to make extensive use of the existence of a primary key (or an additional attribute, e.g., in Chapter 4), thus rendering a direct adaptation impossible. Moreover, the expected attacks and transformations are different. For example a process of summarization would defeat any of the above schemes. Nevertheless it might be worth noting that, if a primary key is assumed to exist, e.g., if there is a guarantee that the time-stamp information for each stream value is going to be preserved in the result, then both the bit alteration method proposed in [35] (for numeric types; see Section 3.7) and the solution in Chapter 4 (for discrete data)

could be adapted to work on a single attribute, namely the stream value. The result would likely be resilient to (time-stamp preserving) sampling, but fail with respect to any other attack or transformation.

But what about multimedia watermarking? Given the "streaming" nature of our data, would it not be possible to simply adapt an existing audio (or media) watermarking algorithm [2] [19] [52] [53] [54] since audio data is also an example of a data stream? In other words, why is our problem different? While there seem to be similarities between watermarking audio and sensor data for example, at a closer inspection these similarities prove to be merely superficial. A multitude of differences are to be found between the two frameworks mainly deriving from different data models, associated semantic scopes and the itemized nature of sensor stream data.

In theory, a sensor stream could be viewed as an audio signal for example and processed as such. However, for all practical purposes such an approach would not suit reality and/or often yield undesired results. For example, while in sensor data streams, summarization and sampling are routinely expected natural operations, audio streams are not to be summarized, and sampling in the audio domain entails an entirely different semantic. Summarization for example would not be survived by any of the existing results. Moreover, data quality to be preserved in audio streaming is usually related to the human auditory system and its limitations. Any watermark-related alteration can be induced as long as the stream still "sounds" good. In the case of sensor streams (e.g., temperature) on the other hand, many scenarios involve widely different quality metrics, that often need to also consider overall stream characteristics [3].

In summary, while experiences in the multi-media domain are valuable, due to the nature of this new application domain, a solution for watermarking discrete sensor streams needs to naturally handle attacks and transformations such as the ones outlined in Section 5.2.1.

[3]e.g., the total alteration introduced per data item should not exceed a certain threshold.

5.3 An Initial Solution

This Section outlines the main solution and then gradually improves it to a more robust and resilient version, by identifying and fixing potential flaws.

5.3.1 Overview

The first issue to be considered when watermarking in such a framework are the data assumptions that the detection process is expected to handle. More specific, are we concerned with (i) an on-the-fly streaming detection process or (ii) the ability to detect a watermark offline, in a static "chunk" of data (with associated multiple-pass, random access), likely a subset of the original stream? These two different scenarios apparently feature distinct challenges. Intuitively a watermarking solution for (ii) could potentially yield an increased detection accuracy (with respect to the same amount of data), due to the ability to repeatedly iterate on the entire data set, without restrictive time bounds. Because any on-the-fly solution can be directly applied to (ii), for the time being let us consider a solution for (i). In Section 5.4.8 we analyze the offline case.

At an overview level, watermark embedding proceeds as follows: (a) first a set of "major" extremes (actual stream items) are identified in the data stream, extremes that feature the property that they (or a majority thereof) can be recovered after a suite of considered alterations (possibly attacks) such as (random) sampling and summarization. Next, (b) a certain criterion is used to select some of these extremes as recipients for parts of the watermark. Finally (c), the selected ones are used to define subsets of items considered for 1-bit watermark embedding of bits of the global watermark. The fact that these extremes can be recovered ensures a consistent overlap (or even complete identity) between the recovered subsets and the original ones (in the un-altered data). In the watermark detection process (d) *all* the extremes in the stream are identified and the selection criteria in step (b) above is used once again to identify potential watermark recipients. For each selected extreme, (e) its

corresponding 1-bit watermark is extracted and ultimately the global watermark is gradually re-constructed, by possibly also using error correction.

Thus, one of the main insights behind our solution is the use of extreme values in the stream's evolution as watermark bit-carriers. The intuition here lies in the fact that much of the stream value lies in exactly its fluctuating behavior and the associated extremes, likely to survive value-preserving, domain-specific transforms.

5.3.2 Embedding

Using the notation in Section 5.2.2, let $\alpha, \beta \in \mathbb{Z}$ such that $\alpha + \beta \leq b(x[])$, where $b(x[])$ is the bit-size of the values in the considered stream $(x[])$. Let χ be a secret integer and $\delta \in (0, 1)$ chosen such that

$$\delta < 2^{(b(x[])-\alpha)}$$

(i.e., all elements within a characteristic subset $\Xi(\eta, \delta)$ have the same most significant α bits). $\alpha, \beta, \delta, \chi$ are secret. We use the term "advance the window" to denote reading in more new data items while discarding old ones.

In the **initial step** of our embedding algorithm we first identify the first major extreme of degree χ and radius δ in the current window. The assumption here is that there exists a major extreme in the current window. If this is not the case, we can simply advance the window until we find one. The "majority" of an extreme can be easily evaluated by comparing the size of its characteristic subset $\Xi(\eta, \delta)$ with the sampling degree χ. The characteristic subset containing at least χ elements guarantees that in a random sampling of degree χ, at least one of those elements is going to survive. If no major extremes can be found for given δ and χ values, one could consider instead extremes with characteristic subsets smaller than χ that guarantee an acceptable chance (e.g., 70%) of survival in case of sampling (i.e., $\frac{subset\_size}{\chi} > 70\%$?).

Note: δ and the desired values for χ can be adjusted such that eventually (in the extreme) all characteristic subsets feature enough elements to survive a sampling of

wm_embed(δ,α,β,wm,k_1,ϕ)
  while ($true$) **do**
    $\eta \leftarrow$ first major extreme in win[]
    compute $\Xi(\eta, \delta)$
    $i \leftarrow H(msb(\eta, \alpha), k_1) \text{mod} \phi$
    if $i \leq b(wm)$ **then**
      $bit \leftarrow H(msb(\eta, \alpha), k_1) \text{mod} \beta$
      foreach $v \in \Xi(\eta, \delta)$ **do**
        $v[bit - 1] \leftarrow false$
        $v[bit] \leftarrow wm[i]$
        $v[bit + 1] \leftarrow false$
    advance win[] past η

Figure 5.4. Discrete Streaming Data: Initial Embedding Algorithm

degree χ. We should not forget though that we also aim to minimize the amount of change introduced. Thus an ideal choice for δ would yield just enough major extremes with characteristic subsets large enough to survive the required level of sampling but no more. This is a fine data dependent trade-off that needs to be considered in practice.

Once a major extreme (η) is identified in the current window, in the **second step**, a *selection criterion* is used to determine whether η is going to be used in the embedding process. If

$$H(msb(\eta, \alpha), k_1) mod \phi = i$$

and

$$i \leq b(wm)$$

then η is considered for embedding bit i of the watermark, $wm[i]$. $\phi \in (b(wm), b(wm)+k_2)$ ($k_2 > 0$) is a secret unsigned integer fixed at embedding time, ensuring that only a limited number (a ratio of $\frac{b(wm)}{\phi}$) of these major extremes are going to be selected for embedding. We used a similar "fitness" selection criteria in Chapter 4. Its power derives strength from both the one-wayness and randomness properties of the deployed one-way cryptographic hash, forcing Mallory into a "guessing" position with respect to watermark encoding location. The reason behind the use of the most significant bits of η in the above formula, is resilience to minor alterations and errors due to sampling. As discussed above, the assumption is that

$$msb(x, \alpha) = msb(\eta, \alpha), \forall x \in \Xi(\eta, \delta)$$

If η is the result of the previous selection step, in the **third step** we embed bit $wm[i]$ into $\Xi(\eta)$. This is done by first, selecting a certain bit position

$$bit = H(msb(\eta, \alpha), k_1) mod \beta$$

for embedding. Next, for each value $v \in \Xi(\eta, \delta)$ and in η itself, that bit position is set to $wm[i]$ and the adjacent bits are set to false (to prevent overflow in case of summarization). In other words $v[bit-1] = false$, $v[bit] = wm[i]$ and $v[bit+1] = false$. The reasoning behind modifying an entire subset of items ($\Xi(\eta, \delta)$) is to survive summarizations. This is the case if the bit encoding is such that the average of any combination of ($\nu < |\Xi(\eta)|$ or less) items in $\Xi(\eta, \delta)$, would preserve the embedded bit. It is easy to show that this is indeed the case. Finally, the window is advanced past η and the process re-starts.

5.3.3 Detection

We are going to illustrate a specific flavor of the detection process, namely the case when majority voting is deployed as an error correction mechanism.

In the detection process the watermark is gradually reconstructed as more and more of the stream data is processed. The reconstruction process relies on an array

of majority voting "buckets" as follows. For each bit $wm[i]$ in the original watermark wm, let $wm[i]^T$ and $wm[i]^F$ be "buckets" (unsigned integers) which are incremented accordingly each time we recover a corresponding true/false bit $wm^{det}[i]$ from the stream. In other words, if the detection process yields at some point $wm^{det}[i] = false$, then the $wm[i]^F$ value is incremented. Similarly, for $wm^{det}[i] = true$, $wm[i]^T$ is incremented. In the end, the actual $wm[i]$ will be estimated by the difference between $wm[i]^T$ and $wm[i]^F$, i.e., if

$$wm[i]^T - wm[i]^F > \upsilon$$

then the estimated value for this particular bit becomes $wm^{est}[i] = true$ (and conversely if $wm[i]^F - wm[i]^T > \upsilon$ then $wm^{est}[i] = false$) where $\upsilon > 0$. If detection would be applied on random, un-watermarked data, the probability of detecting $wm^{det}[i] = false$ would equal the probability of $wm^{det}[i] = true$, thus yielding virtually identical (υ is used to distinguish this exact case) values for $wm[i]^T$ and $wm[i]^F$. In this case, $wm^{est}[i]$ would be un-defined, thus the data considered un-watermarked. The watermark effectively lies in a statistical bias in the $true/false$ distribution for each bit encoding.

Detection starts by identifying the first extreme η in the current window. The selection criteria deployed in the embedding phase is tested on η. If

$$H(msb(\eta, \alpha), k_1) mod \phi = i$$

and $i \leq b(wm)$, then η was likely used in embedding bit i of the watermark, $wm[i]$. This bit is then extracted from bit-position $H(msb(\eta, \alpha), k_1)$ mod β and depending on its value, the corresponding bucket $wm[i]^T$ or $wm[i]^F$ is incremented. Finally, the window is advanced past η and the process re-starts. It is to be noted that, because of the infinite nature of the stream, detection is a continuous process. This is why it is enclosed in a **while** loop. At the same time it shares the $wm[]$ array with the watermark reconstruction process (**wm_construct()**).

The detection process does not consider only "major" extremes but rather any and all extremes that can be identified in the stream. The reason behind this is

```
wm_detect(δ,α,β,wm,k1,φ)
 while (true) do
  η ← first extreme in win[]
  i ←H(msb(η,α),k1)mod φ
  if i ≤b(wm) then
   bit ←H(msb(η,α),k1)mod β
   if (η[bit]==true) then
    wm[i]^T ←wm[i]^T + 1
   else
    wm[i]^F ←wm[i]^F + 1
  advance win[] past η
```

```
wm_construct(wm[]^T,wm[]^F,υ)
 for (i ← 0;i < b(wm);i ← i + 1)
  if (wm[i]^T − wm[i]^F > υ) then
   wm[i] ←true
  else
  if (wm[i]^F − wm[i]^T > υ) then
   wm[i] ←false
  else
   wm[i] ←undefined
 return wm[]
```

Figure 5.5. Discrete Streaming Data: Initial Detection Algorithm

the fact that the stream could have been subjected to sampling (A2) and/or summarization (A1) in the meantime. Considering "major" extremes only and their corresponding characteristic subsets in the embedding phase was a means to ensure survival to exactly such transformations. Nevertheless, the detection process apparently suffers now from the fact that it also considers extremes that were potentially not watermarked in the first place, possibly yielding false watermark readings. At a deeper insight, it becomes clear that this does not constitute a problem. As the watermark reconstruction problem relies on a statistical bias and as this bias is zero in the case of random data (as discussed above), introducing new, random, unwatermarked data points into the detection does not affect the watermark-induced bias at all. This is yet another reason why this embedding will prove resilience to data addition (A5).

5.4 Improvements

Various resilience enhancing improvements are possible with respect to the initial solution introduced above. These are discussed here.

5.4.1 Defeating Correlation Detection

One particular issue of concern in the above solution is the fact that because there exists a correlation between the watermarking alteration (the $wm[i]$ bit) and its actual location (determined by $H(msb(\eta, \alpha), k_1)))$, Mallory can mount a special attack with the undesirable result of revealing the mark embedding locations. The attack proceeds by first realizing that, despite the one-wayness of the deployed hash function $H()$, in fact, η is the only variable that determines *both* the bit embedding location as well as its value. Mallory can now simply build a set of "hash buckets" for each separate value of $msb(\eta, \alpha)$ (if α is secret the job becomes harder but not impossible) and count, for each extreme η encountered, which of the lower β bits of η is set (resp. reset) more often. For each η for which a bias in a bit position is discovered, that particular bit position is considered mark-carrying and randomized.

Thus, the problem lies here in the correlation between the actual bit location and the bit value, correlation induced by the fact that a single variable (η) determines both of these. A fix could possibly rely on a separate source of information to determine the location of the embedded bit, independently of the bit value. Also, this source of information would need to be consistently recoverable at detection time. For example, if time-stamp information would be assumed available, i.e., if all the processing and the attacks on the data stream could be assumed to preserve the time-stamp to value association, then the actual time-stamp would present an ideal candidate, effectively labeling each and every stream extreme uniquely while at the same time not being correlated (directly) to their values. This unique label could then be used in computing the bit position for embedding. In the selection of the bit embedding location, instead of using $bit = H(msb(\eta, \alpha), k_1) \bmod \beta$ which yields a

result correlated to the actual embedded bit value ($wm[i]$, where $i = H(msb(\eta, \alpha), k_1)$ mod ϕ) we propose to use

$$bit = H(msb(label(\eta), \alpha), k_1) mod \beta$$

where $label(\eta)$ is the (virtually) unique label of extreme η. A labeling scheme like this would make "bucket counting" attacks impossible. In our model however, timestamps are not assumed to be preserved. Can we envision a different labeling scheme (at least) for extremes, that would survive the attacks and transformations outlined in Section 5.2.1? We propose to build it from scratch.

Because the data can be subject to both sampling and summarization and we would like to enable watermark detection also from a finite segment of the data (see Section 5.2.1), this task becomes especially challenging. Sampling and summarization are already survived (by design) by the extremes selected using the "majority" criteria in Section 5.2.2. We could maybe make use of this fact in the labeling scheme.

One of the challenging aspects of such a labeling scheme becomes clear when one considers data segmentation. To support segmentation, it needs to function based solely on information available close (in terms of stream location) to the considered to-be-labeled extreme. Also, labels computed at detection time from potential segments of sampled and/or summarized data, need to (at least) converge to the original ones, as more and more watermarked data is available. Let λ be the (secret) bit length of the labels resulting in our labeling scheme. Let $\varrho > 1$ be a (secret) unsigned integer. We propose the following labeling scheme: given two extremes i and a subsequent $i + \varrho$, we define $label\_bit(i, i + \varrho) = true$ iff

$$msb(abs(val(i)), \alpha) < msb(abs(val(i + \varrho)), \alpha) \tag{5.1}$$

and $false$ otherwise. We define the label for extreme $i + \lambda$, $label(val(i + \lambda))$ as the bit string composed of the concatenation of "1" (binary true) followed by each and every $label\_bit(j, j + \varrho)$ in ascending order of $j \in (i - \varrho, i + \lambda - \varrho)$. In other

words, an extreme is labeled by a certain differential interpretation of some of the preceding extreme values, e.g., in Figure 5.3 (a), the label for extreme **K** becomes "110100" ($\varrho = 2$). The main role of ϱ's secrecy is to hide the actual labeling scheme locations from a potential attacker, making a random-alteration attack necessarily more damaging to the value of the data, thus increasingly un-successful. To illustrate this, consider for example the case where Mallory knows that $\varrho = 2$. Now all it needs to do is alter any and only two successive extremes (in any continuous chunk of 2λ extremes), just enough to flip one label bit. But now, if ϱ is secret, Mallory *has* to alter a larger, arbitrary number of successive extremes. Further improvements are discussed in Section 5.4.9.

Before going any further, let us analyze what happens if an important extreme is "lost", e.g., if one extreme i is altered so much that its α most significant bits flip inequality (5.1), corrupting its corresponding label bit. What happens is in fact not too damaging: labels that were constructed using this particular extreme will be corrupted, until the detection process encounters again a continuous sequence of extremes not altered beyond recognition. But Mallory cannot afford altering extremes to such extents, and the secrecy of ϱ makes a random alteration attack the only choice.

In summary, the main purpose of such a labeling scheme is to ensure that Mallory cannot mount the "bucket counting" type of statistical analysis attack as outlined above. Different labels for adjacent extremes together with the use of one-way hashing completely defeat such an attack. The labeling scheme provides an independent, un-correlated source of information for determining the bit position to be altered. Remember that our ability to survive "bucket counting" type of attacks was dependent on the labels being un-correlated with respect to the actual extreme values, while at the same time being virtually unique for each extreme.

5.4.2 Repeating Labels

But the finite nature of the considered bit size of the label poses a certain problem in this respect by necessarily allowing for duplicates (e.g., in the optimal case only due to "wrap-around" of the λ-sized space) if the considered data segment is small. For example if $\lambda = 10$ and we label 2000 extremes, on average, if we are lucky we will have each label repeated only roughly twice. A more complex analysis needs to also include data-time behavior, e.g., what is the likelihood of low to high vs. high to low transitions, given the considered ϱ? If there is a bias in this data behavior then the resulting labels are going to contain possibly more one-bits than zeroes etc. Nevertheless, in summary our problem is now that, because some labels might repeat themselves, an unfortunate circumstance could make it such that enough data for a particular label becomes available for Mallory to mount yet again a "bucket counting" attack.

There are two fixes for the above issue: (i) the selected size of the considered labels could be kept secret, within a certain range (e.g., $\lambda \in (10, 20)$) – there is a trade-off here between the ability to converge in case of data loss and a higher λ value, but for $\lambda = 20$ and $\varrho = 3$ for example, roughly 3 million extremes need to pass by before a label is going to be repeated; (ii) once the un-correlated nature of the labels has been established by their independent information source, we can re-consider the use of the most significant bits of the extreme values. If we re-define the labels as a concatenation between the initial $label\_bit(j, j+1)$-derived labels bit string and $msb(abs(val(i)), \alpha)$ we significantly decrease the probability of duplicates.

5.4.3 Reconstructing Labels

Labeling, while providing a defense for the correlation attack, introduces the requirement to be able to identify major extremes at detection times, possibly in a summarized and/or sampled stream. This becomes a challenge as the definition of "major" does not make sense anymore in the context of a sampled version of the

original stream. We propose the following solution. In a first stage, the degree of the transformation performed is determined. In a second stage, the definition of majority of an extreme is updated to reflect the fact that the considered stream is already transformed. A major extreme of degree χ and radius δ in the original stream $(x[], \varsigma)$, becomes a major extreme of degree $\frac{\chi}{\gamma}$ and radius δ in the transformed stream $(x'[], \frac{\varsigma}{\gamma})$, where γ is the degree of the transformation (e.g., summarization, sampling) applied to $(x[], \varsigma)$. Once we know γ identifying major extremes in the transformed stream is simply a matter of considering this updated definition. But how do we determine γ? In a dynamic stream, with consistent stream data rates, γ can be determined by simply dividing the original stream rate to the current (transformed) stream rate, $\gamma = \frac{\varsigma}{\varsigma'}$. The more challenging scenario is to determine the value of γ corresponding to a (possibly transformed) stream $(x'[], \varsigma')$ for which only a segment is available. In other words, given a certain segment of a transformed stream $(x'[], \varsigma')$, corresponding to an original stream $(x[], \varsigma)$, how do we determine the degree of the transform(s) applied to $(x[], \varsigma)$? A reasonable assumption that can be made is that the transform was applied uniformly to the entire stream. In this case, one solution would start by preserving some information about the initial stream, namely the average size of the characteristic subsets of extremes, for a given δ. Then, in the transformed segment, extremes are identified and their average characteristic subset size for the same δ is computed. It is to be expected (arguably) that in a transformed (sampled and/or summarized) stream these sizes would shrink according to the actual transform degree. Dividing the original average characteristic subset size by the sampled stream average would thus yield an estimate of the transform degree γ. In our proof of concept implementation this method is used successfully.

5.4.4 Hysteresis

The labeling features yet another interesting challenge. While ϱ's secrecy indeed makes it more difficult on Mallory to precisely alter extremes so as to flip label bits,

what is to stop him from still altering a large number of consecutive extremes with the same purpose? This attack is likely not of much concern as the assumption is that Mallory cannot afford such modifications throughout the data as the required modifications to flip several consecutive bits are likely quite significant. Unfavorable data distribution and data semantics preservation are further arguments that Mallory would not be able to deploy such an attack.

Nevertheless, a solution is available and we propose its use. It proceeds by changing the labeling scheme as follows: given two extremes i and $i + \varrho$, we define $label\_bit(i, i + \varrho) = true$ iff

$$(msb(abs(val(i))) - msb(abs(val(i + \varrho)))) < \iota^- < 0$$

and $label\_bit(i, i + \varrho) = false$ iff

$$0 < \iota^+ < (msb(abs(val(i))) - msb(abs(val(i + \varrho))))$$

As can be seen, these new formulas induce a hysteresis (defined by (ι^-, ι^+)). Now Mallory is not only presented with the dilemma of which extremes to alter but also unable to determine what the minimum change is that would flip the label's corresponding bit.

5.4.5 Defeating Bias Detection

But what prevents Mallory from identifying all the major extremes for which there exists a majority of (possibly all) items in the characteristic subset with a certain bit position set to the same identical value? These extremes would then be (rightfully so) considered watermark carrying and Mallory could mount a simple attack of randomizing those bit positions. This attack threatens the validity of the entire watermarking scheme. How can we fix this while surviving summarization? Remember that the main reason behind embedding the same bit multiple times at the same position in different items in the characteristic subset was directly mandated by the requirement to survive summarization. We propose a new approach that

survives summarization and results in alterations effectively appearing random to the eyes of an attacker. Let $\Xi(\eta, \delta) = \{x_1, x_2, ..., x_a\}$. For each $i \leq j \in [1, a]$, let

$$m_{ij} = \frac{\sum_{u \in [i,j]} x_u}{|j - i + 1|}$$

Then we define the *characteristic subset bit encoding convention* as follows: (i) we say that a bit value of "true" is embedded in $\Xi(\eta, \delta)$ iff

$$\forall (j, i), lsb(H(lsb(m_{ij}, \beta), label(\eta)), \zeta) = 2^{\zeta} - 1$$

Similarly, (ii) we say that "false" is embedded iff $\forall j, i$ we have

$$lsb(H(lsb(m_{ij}, \beta), label(\eta)), \zeta) = 0$$

where $\zeta > 0$ is a secret fixed at embedding time. The embedding method simply alters the least significant β bits in the values in $\Xi(\eta, \delta)$ until the criteria is satisfied for the desired $wm[i]$ bit value. It is to be noted that these alterations should aim to minimize the Euclidean distance (or possibly any other distance metric) from the point defined by $\{x_1, x_2, ..., x_a\}$. We call this a "multi-hash encoding".

The use of m_{ij} ensures survival to summarization, while the cryptographic hash provides the appearance of randomness. But is it feasible to assume that one could find such a point in the a-dimensional space defined by the items in $\Xi(\eta, \delta)$? How many computations are required to at least find one? For each item in $\Xi(\eta, \delta)$ we consider its β least significant bits, thus we effectively operate over an input space of $a\beta$ bits. There are $\frac{a(a+1)}{2}$ possible m_{ij} averages (including all $m_{ii} = x_i$ values). For each we consider the last ζ bits of its hash, effectively getting an output space of $\zeta \frac{a(a+1)}{2}$ bits. The probability that a desired pattern occurs in this space is then

$$2^{-\zeta \frac{a(a+1)}{2}}$$

Thus, on average, the expected number of configurations in the input space that would need to be tested in an exhaustive search before yielding one that results in the desired output, is $2^{\zeta \frac{a(a+1)}{2}}$. For example if $\zeta = 1$ and $a = 5$ we have 2^{15}, that

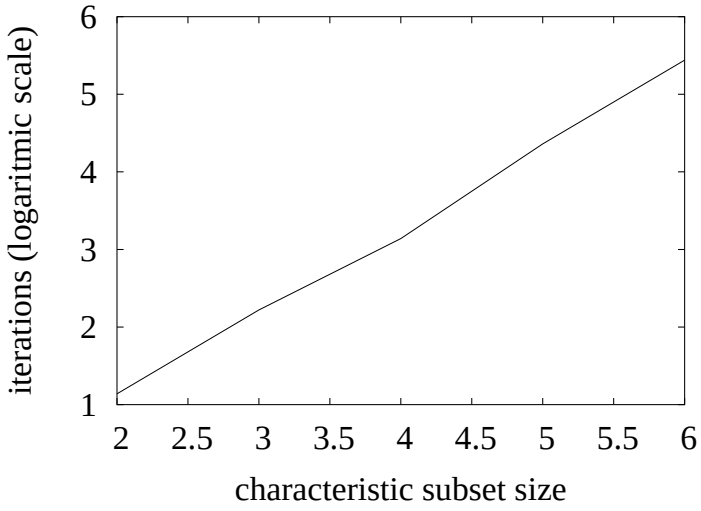


Figure 5.6. Discrete Streaming Data: Average exhaustive search iterations required in computing the closest point that satisfies the characteristic subset bit encoding convention (logarithmic scale).

is, approx. 32,000 computations would need to be performed (for each considered major extreme in the window). See Section 5.6.4 for an experimental analysis. We confirmed these results also experimentally.

In Figure 5.6 we illustrate an experiment in which an un-optimized exhaustive search was deployed. The exponential nature of the required amount of computation becomes clear by the linear behavior in the logarithmic graph.

If enough computation power is available with respect to the incoming stream data rate, larger values for ζ and a could be handled, resulting in an increased level of court-time persuasiveness. Nevertheless, given the exponential nature of the increase in required computations for an increasing number of items in the characteristic subset, it is probably not likely to be able to exhaustively handle subsets with more than $8 - 10$ items efficiently. While out of the scope of the current chapter, the design and use of efficient pruned-space algorithms would be required to significantly reduce these requirements. Alternately, we could deploy a computation-reducing technique that limits the number of m_{ij} averages for which (i) or (ii) needs to hold in the

subset bit encoding convention above. In other words, the search process (in the $\{x_1, x_2, ..., x_a\}$ space) will be stopped once a certain number of the m_{ij} averages feature the desired encoding convention ((i) or (ii)). We call these m_{ij} values "active". The resulting decrease in required computation time comes at the expense of decreased resilience to transforms. More specifically, the fact that the bit-embedding can only be "seen" through a limited number of "good" m_{ij}'s (which feature the appropriate subset bit encoding) makes it such that detecting the corresponding watermark bit in a transformed stream will fail if the stream does not contain at least one of the active m_{ij} values.

If such a reducing technique is applied, a desired property would be the ability to survive to as many levels of summarization as possible. Thus, after ensuring the subset bit encoding convention for every m_{ii} (original items, so as to survive also sampling), we propose to "divide" the remaining computing cycles so as to enable a non-zero probability of bit detection for any degree of summarization. This would be achieved, if for any considered summarization degree ν to be survived, there would exist at least one m_{ij} with $|j - i| = \nu$ (ensuring a non-zero probability of this average to appear in a ν-degree summarized stream) that allows the extraction of the associated watermark bit.

Also, an (arguably) fast(er) encoding than the use of cryptographic hashes above could be adapted from [34]. The method works by altering the β least significant bits until every one of the longest k pre-fixes of the whole value (most significant bits included), when treated as an integer, becomes a quadratic residue modulo a secret large prime, for embedding a 'true" value and a quadratic non-residue modulo the secret prime for embedding a 'false" value.

5.4.6 On-the-Fly Quality Assessment

As discussed in Section 2.2, in any watermarking framework, it is important to preserve structural and semantic properties of the watermarked data. Because by its

very nature, watermarking alters its input, one has to provide a mechanism ensuring that these alterations do not degrade the data beyond usability. Preserving data quality also requires the ability to express and enforce data constraints. Sometimes it is undesirable or even impossible to directly map higher level semantic constraints into low level (combined) change tolerances for individual data items. The practically infinite set of semantic constraints that can be desired of a given data set makes it such that a versatile "data goodness" (i.e., semantically) assessment method is required We propose to augment our sensor stream marking algorithm with such semantic constraints awareness. Each data property that needs to be preserved is written as a constraint on the allowable change to the dataset, the watermarking process is then applied with these constraints as input and re-evaluates them continuously for each alteration (consumer driven encoding). An "undo" log (quite like the "rollback" log in Chapter 3) is kept to allow undo operations in case certain constraints are violated by the current watermarking step (see Figure 5.7). The new challenges in this framework are related to the fact that now, due to storage limitations, any data quality preservation constraints can only be formulated in terms of the current available data window. Likely only few window slots can be used to store data aggregates, possibly including some history information to be used in the quality evaluation process but this will all come at the expense of being unable to store and process as much new incoming data.

5.4.7 Finite Window

With respect to finite space constraints, we believe the solution is an ideal fit for an on-the-fly finite-window processing model. The only requirements are: (i) to be able to detect at least one major extreme at a time for each window, (ii) to be able to fit its characteristic subset (or parts thereof) within the same window and (iii) to have enough remaining space to store some insignificant amounts of information such as the past $\varrho\lambda$ encountered major extremes. Even if the stream behavior is such that

entire characteristic subsets (if it is just one of them, we can simply ignore it, we are concerned here with a majority of the extremes) cannot fit in the current window, the embedding gracefully handles subsets with fewer elements. Yet another solution would be to adjust δ so as to result in smaller characteristic subsets. Nevertheless, we believe this is (arguably) not a concern as most likely the window would contain severals extremes.

5.4.8 Offline Detection

As outlined above, the detection process is designed to function on-the-fly, in one pass over the data and compute the statistical bias for the embedded watermark bits. Time and storage space permitting, would a offline detection process possibly yield more accuracy? In other words, could there be any advantages to having more memory (e.g., $2 \times \varpi$) and unlimited amounts of time in the detection process? The answer is no. The only improvement that could be achieved would be in the normalization process. If the actual data distribution is not known, on-the-fly normalization (as discussed in Section 5.2.1) suffers from the need to perform an initial (non-detection) "discovery" run in which (hopefully) enough data is seen so as to construct a reasonable accurate reference data distribution. Some of the data read in this process would be lost for detection purposes due to storage space limitations. In the offline detection scenario, if multiple-pass access is assumed, this data can be used in detection, effectively enforcing the overall watermark.

5.4.9 Labeling Made Safer

The safety of the labeling process with respect to an attack in which Mallory purposefully alters previous extreme values adjacent to a considered extreme (in the hope of flipping one bit in the corresponding label), could be improved as follows. Instead of using ϱ as a sequential "step" factor in selecting some previous extremes to construct the current extreme's (η) label bits, we could use $H(msb(\eta, \alpha), k_1)$ as a bit-

mask, to select a subset of the past extremes to define the label. For example out of the past 20 extremes we select 10 to be used in the 10-bit label computation, selection based on the last 20 bits of $H(msb(\eta, \alpha), k_1)$ (if a bit in the bit-mask is "true", the corresponding past extreme value is used in the label computation). This process yields both the benefits of shorter labels (more resilient overall, see Section 5.6) and forcing Mallory to consider all 20 bits (instead of 10) in his alteration attack, likely significantly more damaging to the data. For example, in Figure 5.3 (a), if we have the last 5 bits of $H(msb(val(K), \alpha), k_1)$ equal "01101" then the 4-bit label of extreme K would be "1010".

Yet another resilience enhancing idea for labeling would be the use of multiple labels instead of just one, labels constructed using several different subsets of previously seen extremes. Then embedding/detection proceed by enforcing the bit encoding convention considering both labels.

5.4.10 Summarization Revisited

Massive summarization is often used in scenarios involving storage and processing of streaming data. Summarization can be viewed as a normalized integration process. High summarization degrees (ν) are likely destroying much of the high frequency domain in the original stream. Often there exists a trade-off between preserving data of high-granularity in the recent past and of increasingly lower granularity in the distant past. The watermarking solution introduced here survives summarization very well up to high degrees. However, naturally, distant past data, if summarized to a higher degree would yield a more degraded version of the watermark than recent data. One solution to this issue would be to embed multiple layers of watermarks for different ν values, e.g., one layer for the low frequency domain (i.e., small ν values) and another layer for the high frequency domain (i.e., higher ν values). This would ensure an increasing accuracy on detection for both higher and lower degrees.

5.5 Analysis

In this Section we analyze the ability of our method to convince in court, survive attacks and transforms.

Court-convinceability can be naturally expressed as follows: given a one bit (e.g., true) watermark, what is the probability of false positives (P_{fp}) for the watermark encoding? In other words, we ask: *What is the probability of a one-bit (true) watermark to be detected in a random data stream?* If this probability is low enough, then a positive detection would constitute a strong proof of rights, with a "confidence" of $1-P_{fp}$. Here we define confidence as the probability that a given detected watermark was indeed purposefully embedded in the data by the rights owner.

Using the notation in Section 5.4.5, for each considered extreme η, the occurrence probability of a "good" corresponding m_{ij} (i.e., encoding "true" with respect to the bit encoding convention) in a random stream is naturally $\frac{1}{2}$, because of the cryptographic hash used in the encoding. There are $\frac{a(a+1)}{2}$ possible m_{ij} averages (including all $m_{ii} = x_i$ values). Because for each we consider the last ζ bits of its hash, we effectively have an output space of $\zeta\frac{a(a+1)}{2}$ bits. Thus the probability of the bit "true" being encoded consistently by all of these becomes (per extreme)

$$2^{-\zeta\frac{a(a+1)}{2}}$$

Now, for each $\varepsilon(\chi, \delta)$ items there is a potential major extreme recipient of a one-bit encoding. Out of these how many are actually selected for encoding? As discussed in Section 5.3.2 only a fraction of $\frac{1}{\phi}$ (because now $b(wm) = 1$) of them are actually selected for embedding. Thus if ς is the stream data rate, we can determine the relationship between the time elapsed since we started reading the incoming stream (t) and the reached level of persuasiveness, as follows.

If $\varepsilon(\chi, \delta)$ models the average number of items that need to be read before a major extreme is encountered, then $\frac{\varepsilon(\chi,\delta)}{\varsigma}$ represents the average time-interval "between" major extremes. But only $\frac{1}{\phi}$ of the major extremes are selected for embedding, and so

the time-interval between two major extremes that encode the watermark is $\frac{\phi\varepsilon(\chi,\delta)}{\varsigma}$, thus the number of extremes that we are likely to see in a time interval of size t, is

$$\frac{t\varsigma}{\phi\varepsilon(\chi,\delta)}$$

As discussed above, each major extreme has an associated probability of false positives of $2^{-\zeta\frac{a(a+1)}{2}}$, thus if we discover a consistent pattern of embedding in a time interval t, the probability of a false-positive becomes

$$P_{fp}(t) = (2^{-\zeta\frac{a(a+1)}{2}})^{\frac{t\varsigma}{\phi\varepsilon(\chi,\delta)}}$$

For example if $\zeta = 1$, $a = 5$, $\varsigma = 100Hz$, $\phi = 20\%$, $\varepsilon(\chi,\delta) = 50$, after detecting a bit "true" for only $t = 2$ seconds we have

$$P_{fp}(2) = (2^{-15})^{20} \approx 0$$

and an associated proof of rights, with a confidence of close to 100%. Even, at the limit, when due to transforms such as sampling and summarization, for each extreme, only one single m_{ij} average survives and the probability of false positives for each extreme becomes only $\frac{1}{2}$, $P_{fp}(2)$ becomes roughly only "one in a million". Thus, the persuasion power of our method quickly converges to a comfortable level. In Section 5.6 we provide experimental results for watermark resilience to various transforms, including random attacks.

Next we explore a theoretical analysis of the vulnerability of our scheme under the following attack model: Mallory starts to modify randomly every a_1-th ($a_1 > 1$) extreme (η) in such a way as to alter a ratio of $a_2 \in (0,1)$ of the items in the extreme's characteristic subset of radius a_3, $\Xi(\eta, a_3)$. (Thus, on average, Mallory alters only one in every $a_1' = a_1\phi$ bit-carrying extremes).

The assumption here is that these alterations do not impact the associated labeling scheme, in other words, they don't change the "greater than" relationship between extremes used in the labeling process. An extension considering this case is out of the current limited-space scope. To strengthen our derived bounds, we

are going to focus directly on a more challenging, "informed", Mallory, aware of the characteristic subset radius used at encoding time. In other words, we assume that $a_3 = \delta$ is known to Mallory, see Section 5.3.2.

We propose two ways to analyze this vulnerability: (i) looking at how much an attack "weakens" the encoding, i.e., how many of the *active* m_{ij} values are actually destroyed divided by the total number of active ones (making it thus proportionally harder to detect a watermark in court) and (ii) what is the probability that *all* of the active ones are obliterated? It is easy to see that, for a given extreme η, for which $\Xi(\eta, a_3) = \{x_1, x_2, ..., x_a\}$ the number of corresponding m_{ij} values altered is

$$c_m = \frac{1}{2} a a_2 (2a - a a_2 + 1)$$

Now, for (i), the "weakening" of the encoding can be defined as

$$c_m \times \frac{2}{a(a+1)}$$

which is the ratio of m_{ij} values that are altered from the total number of potential active ones for each altered extreme. Because one in every $a_1' = a_1 \phi$ bit-carrying extremes gets impacted, the overall "weakening" factor can be defined as

$$a_1 \times c_m \times \frac{2}{a(a+1)}$$

To answer (ii), we first model this scenario by a sampling experiment without replacement. In this experiment, $x + t, t > 0$ balls are randomly removed from a bowl with a total of y balls. The question answered is: if the bowl contained exactly x black balls what is the probability that the $x + t$ removals emptied the bowl of all of them. It can be shown that this is

$$P(x + t, x, y) = \frac{\binom{y-x}{t}}{\binom{y}{x+t}} \tag{5.2}$$

In our model $(x + t) = c_m$, $y = a(a+1)\frac{1}{2}$ and if $x = a_4 y$ (a_4 is the ratio of active m_{ij} values) we can compute the probability that *all* of them are altered.

Thus, for each attacked extreme we have a non-zero probability of altering all active m_{ij} values and removing the corresponding watermark bit. Next we ask, how

do these alterations impact our ability to convince in court and detect a watermark bias in the resulting data? Because the alteration is necessarily random (the randomness of the one-way hashes in the encoding in Section 5.4.5 guarantee this) we can model the attack as essentially a random noise addition attack. Evaluating the resilience of any watermark bias becomes now a matter of asking how many of the embeddings actually survive until detection time. Are there enough of them to actually convincingly reconstruct the multi-bit watermark after error correction? At the beginning of the section we looked at how the watermark bias becomes more convincing in time (and seen data). Loosing a fraction of the mark bit encoding extremes can be in fact seen as a reduction of the ϕ value (see Section 5.3.2). If, for each of the $a_1' = a_1\phi$ bit carrying extremes that are altered by Mallory, the attack success probability is given by $P(x+t, x, y)$ (equation 5.2) we can perform a similar reasoning with a new

$$\phi' = \phi + a_1' \times P(x+t, x, y)$$

What now happens is that the persuasiveness (court-time convince-ability) converges proportionally slower. In other words, we need to see $a_1 \times P(x+t, x, y)$ more stream data to be able to provide an equally convincing proof in court.

For example, for $a_1 = 5$, $a = 6$, $a_4 = 50\%$, $a_2 = 50\%$ we get the average probability $P(15, 10, 21) \approx 0.85\%$ of a complete alteration of all the active m_{ij} values at each extreme. This effectively translates in the need to see only an average of $a_1 \times P(x+t, x, y) \approx 4.25\%$ more data to be equally convincing at detection.

But how does our encoding handle transforms? By construction it certainly survives sampling (A2) up to a degree of $\chi_{max} = |\Xi(\eta, \delta)|$. Indeed this is so if at least one element in the characteristic subset of η is to be found in a sampling of degree χ_{max}. This element can be used in the detection process to recover the corresponding watermark bit for η. Higher degrees of sampling are also quite likely to be survived as there is a non-zero probability of elements in $\Xi(\eta, \delta)$ to be in the sampled stream even for $\chi > \chi_{max}$. This is experimentally analyzed in Section 5.6.

Summarization (A1) up to a degree of $\nu_{max} = |\Xi(\eta, \delta)|$ is also handled well by design, for example due to the use of m_{ij} in the bit-encoding procedure illustrated in Section 5.4.5. Any summarization of a degree $\nu \leq \nu_{max}$ naturally results in at least one of the m_{ij} averages being in the summarized stream. Even in the initial algorithm, the bit encoding pattern used on the elements in the characteristic subset ensured survival of the pattern in the process of averaging (thus surviving summarization) within the subset. Summarization is experimentally analyzed in Section 5.6.

But how well is segmentation (A3) survived? More specifically, what is the minimum size of a stream segment from which we are able to recover the watermark? For simplicity let us assume a one-bit watermark, i.e., $b(wm) = 1$. In the following we are trying to determine the minimum required size of a contiguous watermarked stream segment that would enable a proof more "convincing" than a coin-flip stating that a watermark is embedded in the data. This proof would be obtained if we can correctly detect at least two consistent bits (equal to $wm[0]$) from two different extremes found in the segment. In that case, the probability of a false-positive becomes lower than a random coin-flip. But what is the minimum amount of data we need to see to be able to decode two bits? In the best case, the two extremes are adjacent and we need to see enough data to build correct labels for those two extremes. To build the labels correctly, we need to have seen all the previous $\lambda\varrho$ major extremes correctly. Further qualitative analysis must be data dependent, for example if the fluctuating nature of the stream features a major extreme of degree χ and radius δ for every $\varepsilon(\chi, \delta)$ data items, then the minimum required size of a segment enabling watermark detection is $\varepsilon(\chi, \delta)\lambda\varrho$.

5.6 Experimental Results

We implemented **wms.*** a Java proof-of-concept of the watermarking solution. Our experimental setup included one 1.8GHz CPU Linux box with Sun JDK 1.4 and 384MB RAM.

We also implemented a temperature sensor synthetic data stream generator with controllable parameters, including the ability to adjust the data stream distribution, fluctuating behavior (e.g., $\varepsilon(\chi, \delta)$) and rate ($\varsigma$). This sensor was used in the initial design phase of some of our experiments because of the ability to produce various fine-tuned data inputs impacting specific strengths of the encoding.

We explored experiment scenarios modeling both the behavior of sub-systems such as the on-the-fly labeling module as well as the overall watermark resilience. Synthetic (temperature sensor model) and real-world data was used in our evaluation.

Because, as discussed in Section 5.3.3, watermark encoding relies on altering a certain secret statistical bias within the data, when we present resilience results we refer to the ability to detect and reconstruct this bias as an overall measure of encoding performance. In this case, the notion of a "watermark bias" refers to the number of instances of *active* extremes for which the characteristic subset bit encoding (see Section 5.4.5), survives with a positive true-bit embedding bias.

Note: With respect to court-time confidence, for example, a detected watermark bias of 10 yields a false-positive probability of $\frac{1}{2^{10}}$, and an associated proof of rights with a confidence of roughly 99.9%, as discussed in Section 5.5.

Unless specified otherwise, the experimental results presented here refer to an underlying normalized stream with values distributed normally with a mean of 0 and a standard deviation of 0.5. The fluctuating behavior of the stream was determined by an average $\varepsilon(\chi, \delta) = 100$ (100 items per each major extreme) and $\varsigma = 100Hz$ (100 items per second). Other parameters include: $\phi = 3$, $\alpha = 16$, $\beta = 16$, $\upsilon = 2$, k_1 was chosen by a random number generator. Whenever exact quantitative results are shown, they refer to a data set drawn from about 50 seconds of stream data

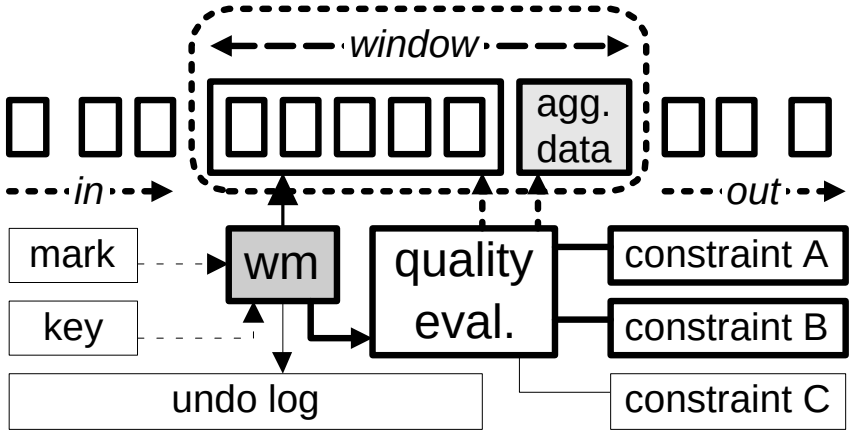


Figure 5.7. Discrete Streaming Data: Overview of proof of concept implementation.

(i.e., roughly 5000 data values). Additionally, when experiments were performed on real-life test data this is specified in the figure captions. The real life data sets [55] were obtained from the environmental monitors of the NASA Infrared Telescope on the summit of Mauna Kea, at the University of Hawaii. They represent multiple sets of once-every-two-minutes environmental sensor (i.e., temperature) readings at various telescope site locations. The reference data set used refers to 30 days worth of data from the month of September 2003, totaling a number of 21630 temperature readings (with values on the Celsius scale roughly between 0 and 35 degrees).

Some of the figures presented in this Section feature a "spikey" behavior. This is a result of the adaptive data-dependent nature of the encoding. Different input data sets react differently to sampling for example, yielding slightly varying behavior at distinct points. Averaging over multiple inputs would provide a solution for this issue. Nevertheless, we believe that, while it might soften the spikes it would also (arguably) tone down distinct features for a given data set, features that inter-relate figures. Instead of focusing on local variations, the figures should be interpreted as qualitative samples of global governing trends.

5.6.1 Random Alterations

In Chapter 3 we defined the *epsilon-attack* in the relational data framework, a transformation that modifies a percentage τ of the input data values within certain bounds defined by two variables ϵ (amplitude of alteration) and μ (mean of alteration). Epsilon-attacks can model any uninformed, random alteration – often the only available attack alternative. A *uniform altering* epsilon-attack (as defined in Chapter 3) modifies τ percent of the input tuples by multiplication with a uniformly distributed value in the $(1 - \epsilon + \mu, 1 + \epsilon + \mu)$ interval. We believe this attack closely resembles (A6), a very likely combination of (A5) and (A2). In Figures 5.8 and 5.9 ($\mu = 0$) we analyze the sensitivity of both our labeling module and overall watermarking scheme to such randomly occurring changes, as direct measures for

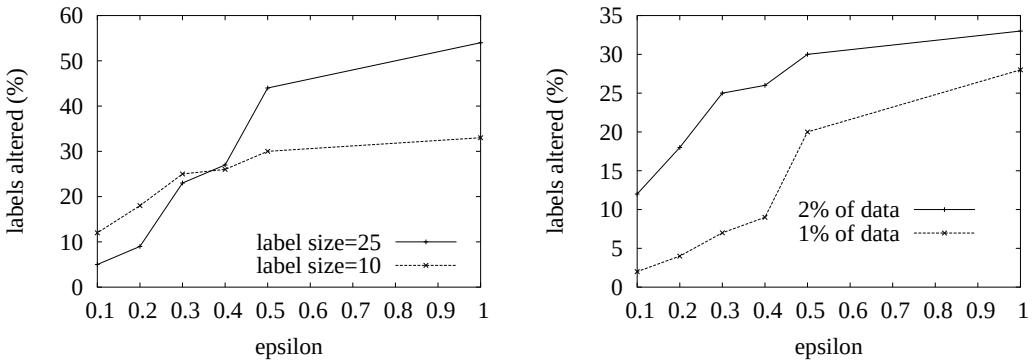


Figure 5.8. Discrete Streaming Data: Label alteration for increasingly aggressive uniform altering epsilon attacks. (a) Different label bit sizes shown. A smaller label size seems to survive better. (b) Different altered data percentages shown.

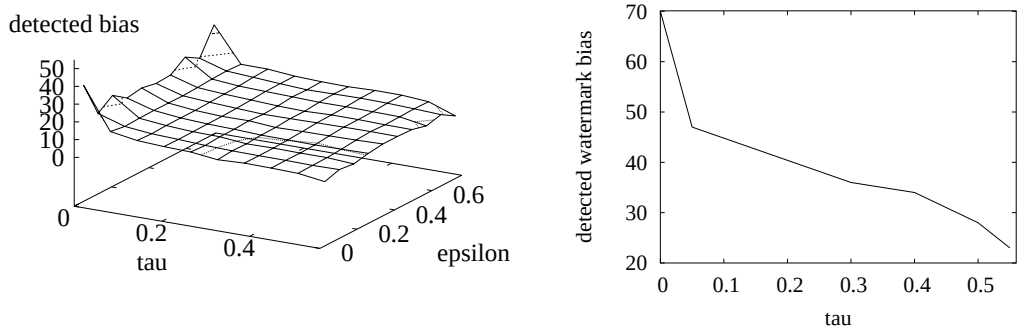


Figure 5.9. Discrete Streaming Data: Watermark survival to epsilon-attacks. (a) Naturally, increasing τ and ϵ values result in a decreasing watermark bias. (b) Same shown for $\epsilon = 10\%$. (real data)

encoding resilience. In Figure 5.8 (a), label alteration increases with an increasing degree of data change. Smaller label bit sizes seem to better survive such an attack. In Figure 5.8 (b), as the percentage of altered data items increases, the labeling scheme naturally degrades.

In Figure 5.9, an embedded watermark (bias) is detected in a randomly altered stream. Naturally, an increasing distortion results in a decreasing bias detection.

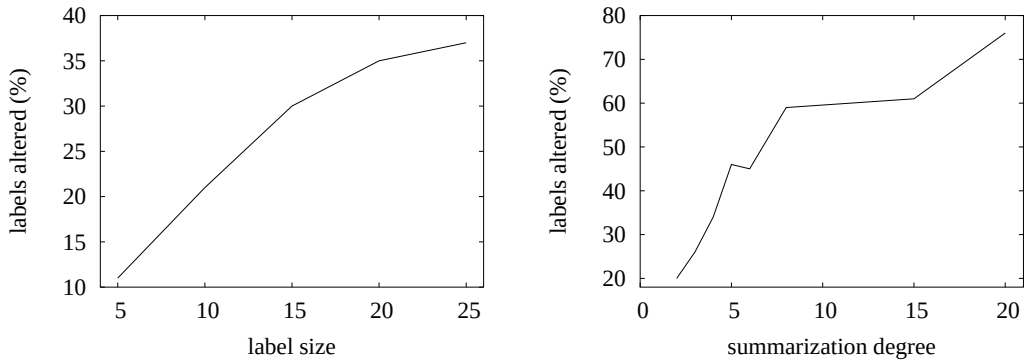


Figure 5.10. Discrete Streaming Data: (a) Label resilience under sampling conditions. A higher label bit-size naturally yields an increased fragility to sampling. (b) Label alteration for summarization of increasing degree.

Nevertheless, it is to be noted that the encoding scheme proves to be quite resilient by design, for example for $\tau = 50\%$ of the data altered within $\epsilon = 10\%$ (Figure 5.9 (b)), the detected bias is still above 25, yielding a false-positive rate of less than "one in thirty million".

5.6.2 Sampling and Summarization

The ability to survive summarization (A1) and sampling (A2) is of extreme importance as both are expected attacks. In Figure 5.10 the labeling algorithm is evaluated with respect to (a) sampling and (b) summarization. Intuitively, a higher label bit-size results in increased fragility to sampling (shown is a sampling degree of 3). Summarization seems to be naturally survived by our design. For example, a summarization of the data down to 5% ($\nu = 20$) still preserves over 20% of the original label values, thus conferring a strong back-bone to watermark embedding.

The behavior of the watermark encoding algorithm to sampling and summarization is outlined in Figure 5.11. The natural strength of the bit encoding convention is clearly illustrated here. Both transformations are survived extremely well.

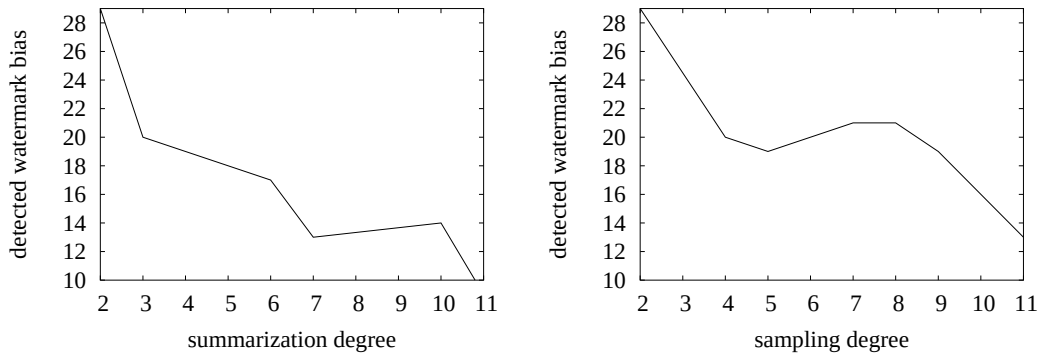


Figure 5.11. Discrete Streaming Data: (a) Watermark survival to summarization. An increasing summarization degree results in a decreasing detected watermark bias. (b) Watermark survival to sampling. A bias of 10 ensures a true-positive probability of 99.999%. (real data)

5.6.3 Segmentation. Combinations

In Section 5.5 we theoretically assessed the ability of our scheme to survive segmentation (A3), by answering the question: what is the minimum size of a stream segment from which we are able to recover the watermark? In Figure 5.12 (a) we analyze the impact of actual recovered segment size on the detected watermark bias. From a segment of only 2000 stream values we can detect a watermark bias of 10, corresponding to a very convincing low false positive rate of roughly 0.001. In Figure 5.12 (b) we outline the impact of a *combined* transformation (sampling and summarization) on the watermark embedding. Because of the nature of both transformations and of the resilience featured in each case, the combination seems to be survived equally well. For example, a 25% sampling, followed by a 25% summarization process still yields a watermark bias of up to 20, corresponding to a low false-positive rate of "one in a million".

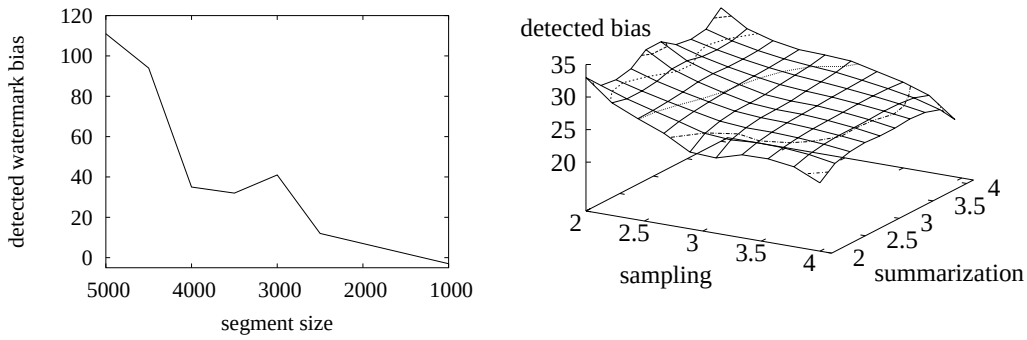


Figure 5.12. Discrete Streaming Data: (a) Watermark survival to segmentation. (b) Watermark survival to combined sampling and summarization. (real data)

5.6.4 Overhead and Impact on Data Quality

The proposed watermarking solution is highly adaptive to both speed and space constraints. By far the most computationally intensive operation is the one-bit encoding operation which alters the characteristic subset data to conform to the bit encoding convention defined in Section 5.4.5. At the expense of embedding resilience, this operation can be sped up significantly by both pruning of the search space or, more importantly, deployment of a computation-reducing technique as described in Section 5.4.5. Depending on the actual stream rate, these speed-ups can be gradually deployed to be able to keep up with the incoming data. Additionally, the average amount of computation to be performed per window-load of data is defined also by the actual fraction of extremes "selected" to be bit-carriers. This fraction is determined by $\frac{b(wm)}{\phi}$. If the incoming data rate is too high, ϕ can be increased to reduce the workload.

While our solution is naturally designed for stream processing it is of importance to assess this ability also in practice. We performed experiments aimed at evaluating the introduced watermarking computation overhead. Unless specified otherwise, we used the multi-hash encoding discussed in Section 5.4.5 and parameters set such

that the resulting watermark survives 100% any combined sampling and summarization up to a degree of 6. First, we compared the computing times required by the watermarking process with the times spent in a simple read and copy model in which each stream item is read and copied to an output port (with fixed writing time-cost). We obtained consistent value classes clearly identifying each of the separate encoding methods presented. It became clear that, as expected, the majority of time is spent in the actual bit encoding convention routine (and not as much in the labeling module). Not surprising, the encoding convention introduced in Section 5.3.2 performed fastest with an average of only 5.7% increase in processing times per stream item. The poorest performer was the more complex multi-hash routine in Section 5.4.5 with an average increase of over 1000%, as expected decreasing almost perfectly exponential with the decrease of guaranteed resilience (see Figure 5.13 (a)).

There are two lessons to be learned here. First, different encodings should be used for different scenarios with associated value models. For example for a temperature stream with a likely average reading rate of under 1Hz, deploying the multi-hash encoding routine for high resilience would be best suited whereas in a very fast streaming scenario the encoding in Section 5.3.2 would perform much better. Additionally, subject to future research is the issue of better pruning algorithms as discussed in Section 5.4.5.

We also performed experiments evaluating the impact of our encoding on data quality. More specifically we analyzed the alterations incurred by the mean and standard deviation of the stream data. For the above parameter settings, over a large number (12000+) of runs over the real (and synthetic) data sets, the value of the mean of the watermarked stream varied less than a mere 0.21% average from the original. The alteration to the standard deviation also maintained itself nicely within 0.27% of the original data. There exists a tunable trade-off between attack/transformation resilience and the incurred alterations. A lower level of resilience would definitely require less modifications to the data and have a lower impact on global statistics. In Figure 5.13 (b) we show how decreasing the number of consid-

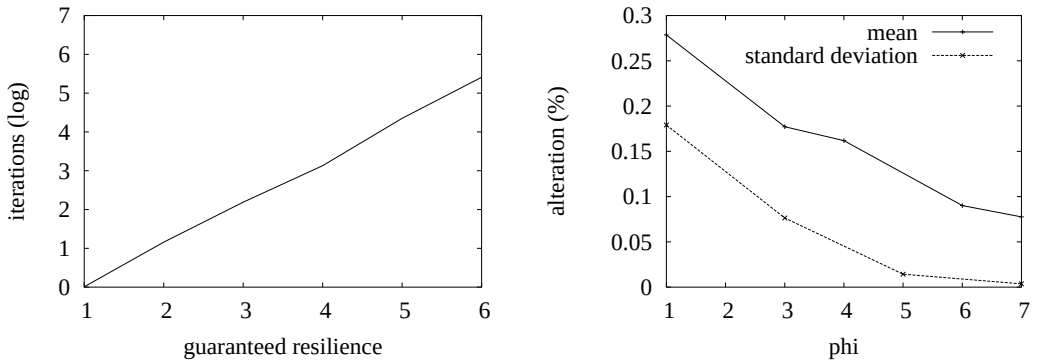


Figure 5.13. Discrete Streaming Data: (a) Computation overhead (iterations) in multi-hash encoding increases with increasing guaranteed resilience (e.g., sampling degree) levels (logarithmic scale). (b) Decreasing the number of considered bit-encoding extremes (increasing ϕ) decreases the impact on mean and standard deviation in the watermarked data.

ered bit-encoding major extremes decreases the impact on the average and standard deviation in the result.

Due to the random nature (with respect to the stream data values) of the encoding specifics we expected a virtually zero impact on such statistics over the longer term. While we observed a certain convergence to zero, it had not as fast a pace as expected; we were actually not able to actually reach the zero-impact point. We suspect this is due to a bias introduced by the MD5 hash implementation used in our proof of concept, although the complex nature of the multi-hash embedding used (see Section 5.4.5) might also hold some of the answers. We are further investigating this.

5.7 Conclusions

In this chapter we introduced the issue of rights protection for sensor streams. We proposed a watermarking solution, based on novel ideas such as on-the-fly labeling and watermark encoding, resilient to important domain-specific transforms. We implemented a proof of concept of the proposed solution and evaluated it experi-

mentally on real data. The method proves to be extremely resilient to all considered transforms, including sampling, summarization, random alterations and combined transforms. Published research results of this work include [12].

In upcoming research we propose to analyze streams of categorical data, to investigate other aggregates (instead of averages) in the summarization process (e.g., min, max, most likely value) and to experiment with alternative bit-encodings.

6 SEMI-STRUCTURED AGGREGATES

While in the previous chapters we discussed very specific data types, here we introduce an algorithm for watermarking abstract structured aggregates of multiple types of content that can be usually represented as graphs and are characterized by value lying both in the structure *and* in the individual nodes (e.g., XML documents, complex Web Content, workflow and planning descriptions).

6.1 Introduction

In this chapter we discuss the watermarking of abstract structured aggregates of multiple types of content, such as multi-type/media documents. These *semi-structures* can be usually represented as graphs and are characterized by value lying both in the structure *and* in the individual nodes. Example instances include XML documents, complex web content, workflow and planning descriptions, etc. We propose a scheme for watermarking abstract semi-structures and discuss its resilience with respect to attacks. While content specific watermarking deals with the issue of protecting the value in the structure's nodes, protecting the value pertaining to the structure itself is a new, distinct challenge. Nodes in semi-structures are value-carrying, thus a watermarking algorithm could make use of their encoding capacity by using traditional watermarking. For example if a node contains an image then image watermarking algorithms can be deployed for that node to encode parts of the global watermark. But, given the intrinsic value attached to it, the graph that "glues" these nodes together is in itself a central element of the watermarking process we propose. We show how our approach makes use of these two value facets, structural and node-content.

In fact, good resilience can be achieved even if fairly weak watermarking algorithms are used in the individual nodes – using the aggregation graph as well as the individual nodes has an "amplification" effect on the overall resilience.

Our solution is based on a canonical labeling algorithm that self-adjusts to the specifics of the content. Labeling is tolerant to a significant number of graph attacks ("surgeries") and relies on a complex "training" phase at watermarking time in which it reaches an optimal stability point with respect to these attacks. We perform attack experiments on the introduced algorithms under different conditions.

The chapter is structured as follows. Section 6.2 is dedicated to a more in depth presentation of generic issues associated with watermarking in the framework of semi-structures. We analyze associated challenges and discuss attacks. Section 6.3 introduces important building blocks and concepts for the presented semi-structure watermarking algorithm, such as *tolerant canonical labeling* and *tolerant content summaries*. It presents and analyzes the main algorithm. Section 6.4.2 discusses experimental results. The **wmx.*** package is introduced. Section 6.5 concludes.

6.2 Challenges

When dealing with graphs in general and semi-structures in particular, we are faced with the issue of uniquely identifying and referencing nodes [1]. In graph theory, this is summarized under the term *canonical labeling* [56] [57] [58] [59] and no solution has been provided with a high enough degree of generality.

Thus, before deploying any specific mark encoding techniques we have to ensure a resilient labeling scheme, able to survive minor modifications and attacks on the actual graph structure. We show how content specific watermarking techniques (for node content watermarking) coupled with a technique of content summarization provide a resilient labeling scheme, suited for our watermarking purposes. The value-carrying nodes are solving the labeling issue in quite a surprising manner.

[1] Especially if required to maintain consistency *before* and *after* attacks (e.g., possible structural changes).

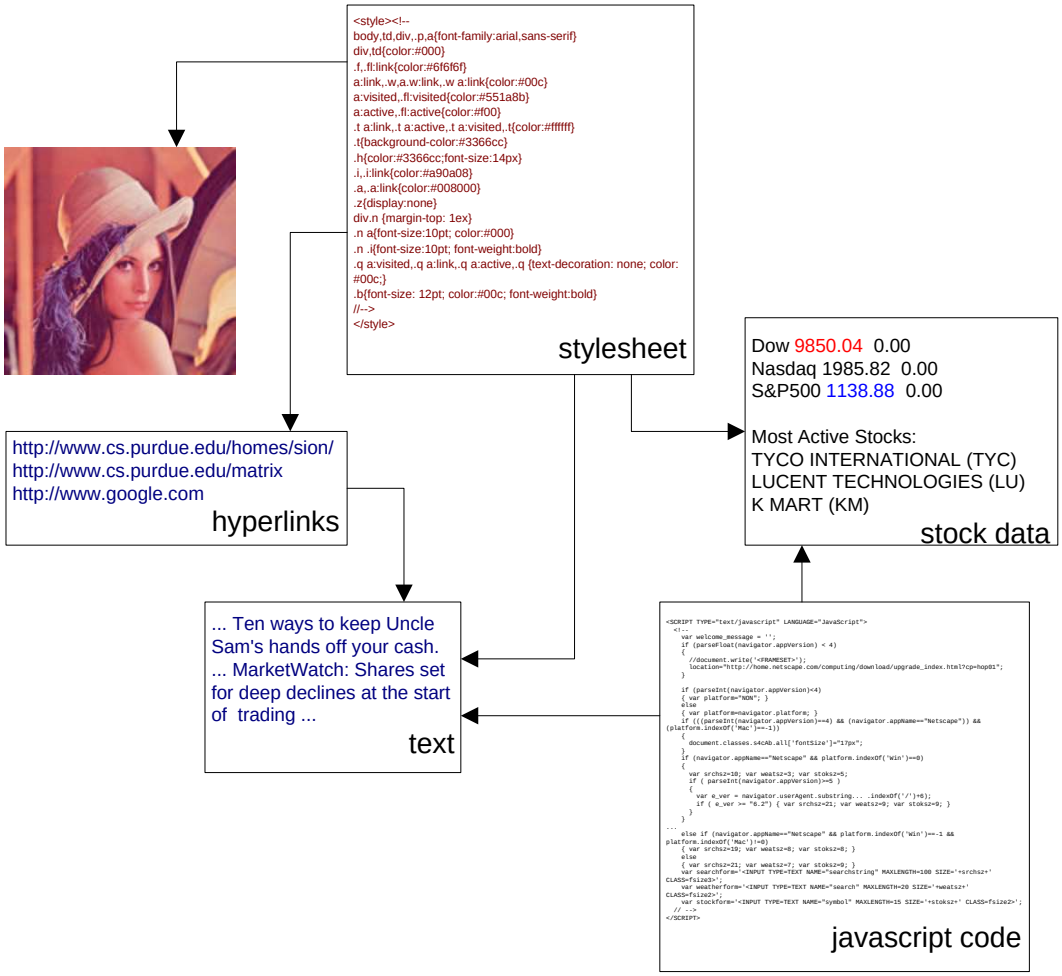


Figure 6.1. Semi-structured Aggregates: A webpage as a semi-structure.

6.2.1 The Adversary

Given a certain value carrying watermarked semi-structure several attack options present themselves, including: elimination of value-"insignificant" nodes (A1), elimination of inter node relations (A2), value preserving graph partitioning into independent usable partitions (A3), modification of node content, within usability vicinity (A4), addition of value insignificant nodes aimed at destroying ulterior labeling attempts (A5). One has to keep in mind the ultimate goal of any attack, namely eliminating the watermark property, while preserving most of the attached value, within usability limits [2].

In order to prevent success for A5, we propose a preliminary step of **value pruning** in which all value-insignificant nodes are marked as to-be-ignored in the ulterior watermarking steps. Another approach deploys structural changes to bring the semi-structure to the limits of the usability space (see Chapter 7), increasing its fragility to further modifications and thus the failure likelihood of any ulterior attempts to attack by adding nodes. A4 mandates the ability of the labeling scheme to depend as little as possible on node content or to provide for a mechanism of detecting altered-content nodes at extraction time. Another possibility of defending against A4 would be to actually alter the main considered nodes toward their allowed fragility limit, such that any further un-knowledgeable changes will fail to provide a usable result. Attack A3 is one of the most powerful challenges. In order to survive it, meaning that the watermark has to be preserved (maybe in a weaker form) also in the resulting graph's partitions, the watermarking scheme has to consider some form of hierarchical embedding in such a way as to "touch" most of the potential partitions in the graph. The issue becomes more complex if the usability domains of all possible graph partitions are unknown, making it difficult to envision the attacker's "cut". Fortunately, in many cases the number of available partitioning schemes that make

[2]Collusion attacks are not discussed in this chapter as they are relevant when fingerprinting is deployed. Although we envision extensions of this work for fingerprinting, we are not considering these here.

sense and the associated usability domains are limited. Cases A1 and A2 make it necessary to devise a node labeling scheme that tolerates node and edge elimination while preserving most of the other nodes' labels. This is a must because of the necessity to reference nodes at extraction time. Even if there would exist a working traditional canonical graph labeling algorithm it would need to be heavily modified in order to provide for edge and node removal tolerance. We used the term "heavily" to outline the fact that canonical labeling has always been linked to proofs of graph isomorphism, whereas in this case the trend is aimed exactly toward the opposite, namely preserving node labels in the context of admittedly slight graph changes.

6.3 A Solution

6.3.1 Tolerant Canonical Labeling

The node labeling scheme is at the heart of watermarking semi-structures. The ability to identify and reference nodes within the to-be-watermarked structure is of paramount importance and the labeling scheme has to take into account the specifics of the case, in particular the requirement to be able to "recognize" all relevant nodes in an attacked version of the graph, based on labels issued on the original one.

Although canonical labeling for graphs was known for a long time to be a hard problem of graph theory, specific algorithms have been developed for some cases . In particular, reasonable solutions have been proposed for tree canonical labeling and apparently, many semi-structure watermarking applications (e.g., HTML) would fit the assumption of tree structuring. One can partition existing value-carrying semi-structures into a set of tree-shapes and remaining structural elements. Watermarking only those partitions might provide enough power and reduce the problem to tree shapes. Unfortunately the requirement of being able to label nodes consistently before and especially *after* attacks, renders useless existing tree canonical labeling algorithms due to their high fragility to any structural changes (e.g., attacks).

Fortunately, the dual nature of semi-structures enables a novel approach to labeling, the main idea being the use of a combination of structural and node content information. On the one hand, content is combined in computing a node's label by using a special "tolerant" summary (i.e., a function of the content with specific properties, see Section 6.3.2) of its content. The assumption here is that content changes are small and that we are able to construct a function of the node content that will basically degrade gracefully with minor alterations to its input. On the other hand some node topology information is necessarily involved in the relative position of the node versus its neighbors and the entire graph. One simple solution that comes to mind is to use the neighbors' labels, which does capture the position of the current node in relationship to its neighbors, and through the entire labeling scheme, applied recursively, to the graph as a whole. Thus the primitive labeling algorithm can be summarized by the following iterative formula:

$$l(node) = \alpha * l(node) + \gamma * \sum_{nb \in neighbors(node)} l(nb)$$

Note: α determines the "weight" of the node content in the labeling scheme. If essential content changes are unlikely in an attack, α is to be increased so as to provide labeling stability. γ provides control over being able to more specifically localize the node with respect to the neighbors and also to the entire graph. If structural changes are highly unlikely in the course of an attack an increased γ provides for stability [3].

The algorithm starts with the initial labels as being the keyed tolerant content summary values $SUMMARY(key, content(node), key)$ (see Section 6.3.2).

Step One. The first step performs a number of iterations i over the formula above (this number being kept as part of the watermark detection key and used later on in re-labeling the attacked graph), until the necessary labeling provisions are met. At this stage we are mainly concerned with a minimal number of identical labels [4].

[3] It might be interesting to note the fact that if γ is 0, this labeling scheme converges to a simple intuitive content-based addressing scheme.

[4] A number of iterations at least equal to the diameter of the graph are necessary in order to localize a given node with respect to the entire graph. But this is sometimes not desired nor required. The

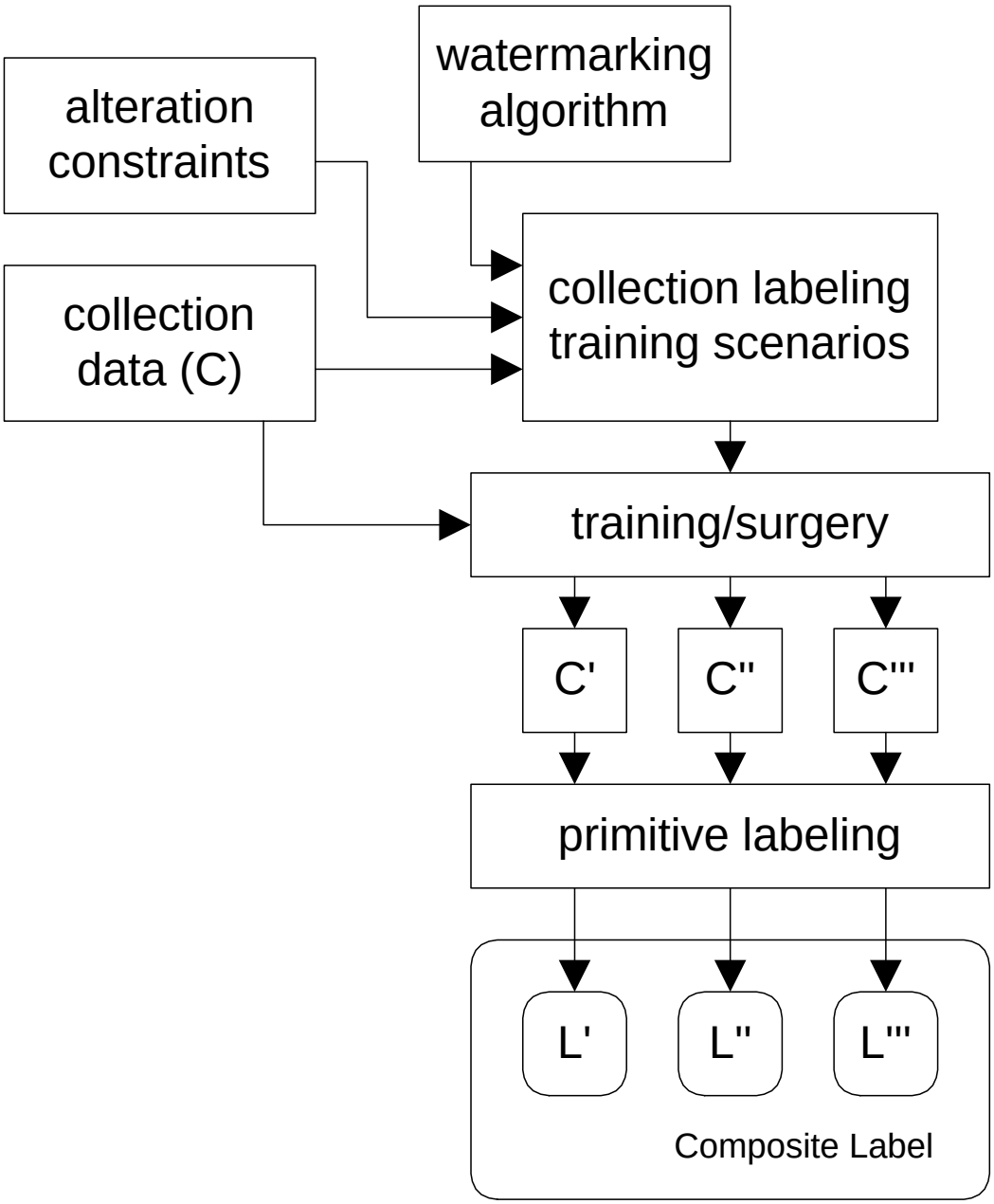


Figure 6.2. Semi-structured Aggregates: Tolerant Canonical Labeling. Composite Labels are a result of successive training sessions.

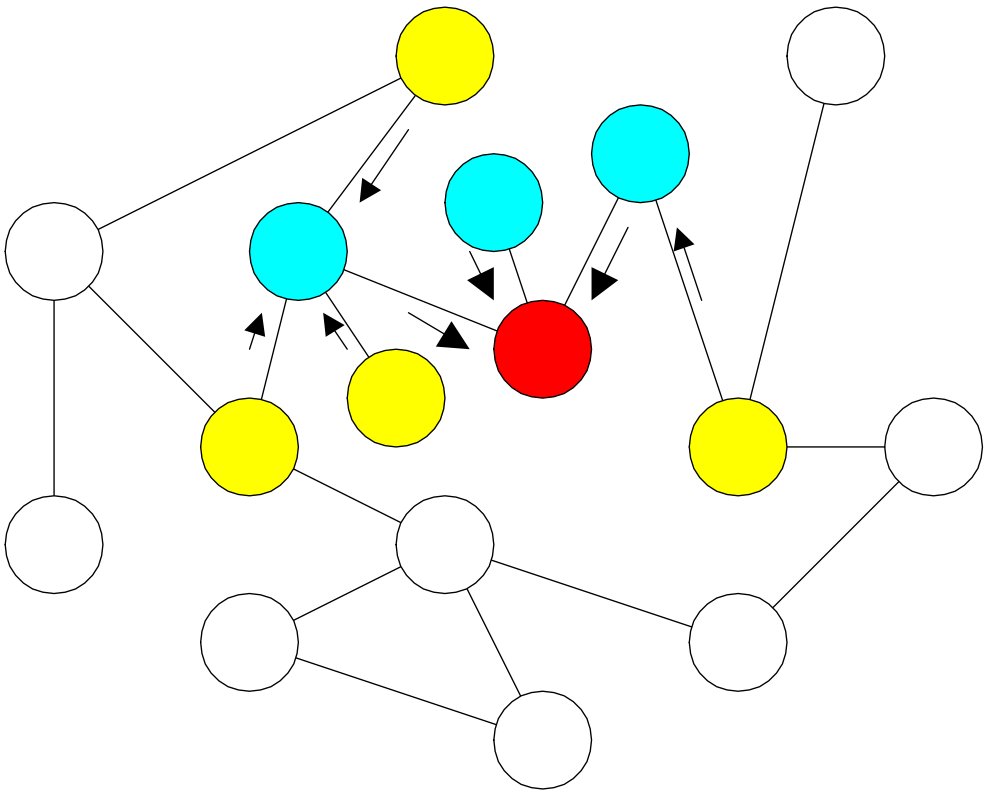


Figure 6.3. Semi-structured Aggregates: A combination of propagated structural and node content information determines a node label.

Step Two. In order to provide resilience to a certain number of graph modifications ("surgery"), the next step is to artificially degrade the graph and re-perform step one again.

Intuitively (for experimental results see Section 6.4.2), removing and/or adding nodes and relations to the graph will result in changes in the initial labeling performed on an un-modified graph. Control over those changes is enabled by specifying the α and γ values. Experiments show that, given a graph, for certain α and γ value bounds, labeling becomes controllable.

In each step, all the resulting labels and the corresponding iteration number, α and γ values are kept. This enables later on, computing of the optimal values. This indeed leads to non-trivial storage requirements and a $O(n^2)$ complexity. But this only happens once in the lifetime of the watermarked object, that is, at graph labeling time. Subject to future exploration is how to deploy dynamic programming and dependency graphs to reduce storage and computation requirements.

The result of step two, for each node, is a range of values for the corresponding label, depending also on the three main control factors (step-one iteration number, α, γ). The actual label of the node will be defined by the lower and upper bounds of the resulting labeling range. This basically ensures that, when labeling the attacked/modified version of the graph (i.e., by performing step one of this same algorithm later on, in court), the resulting labels will fall within the corresponding node's label interval with a high likelihood. For a given set of surgeries, performing the labeling algorithm in the space of (α, γ, i) results in a "bounded space of labeling points" (see Figure 6.4).

The next challenge is to identify an optimum in this "space", given a certain ability to compare two particular "points". Remember that a "point" corresponds to a labeled graph as a set of interval-labels for the graph's nodes, given the particular (α, γ, i) coordinates. Our initial comparison formula for two different graph interval-

ability to set the number of performed iterations and make it part of the recovery key is another point of control over the labeling scheme.

label sets aims at capturing optimality in terms of both minimal number of label overlaps within each set as well as minimal potential for future overlap. If the two considered "points" are the actual interval-label sets $A = \{(a_{11}, a_{12}), ..., (a_{n1}, a_{n2})\}$ and $B = \{(b_{11}, b_{12}), ..., (b_{n1}, b_{n2})\}$ (i.e., (a_{i1}, a_{i2}) is the label-interval corresponding to node i in the graph labeling A) then the comparison formula is

$compare_1(A, B)$ = overlaps(B) × avg_overlap(B) - overlaps(A) × avg_overlap(A)

$compare_2(A, B)$ = closest_inter_label_size(A) - closest_inter_label_size(B)

$compare(A, B) = compare_1(A, B) + compare_2(A, B)$

where $overlaps(X)$ is the number of overlapping interval-labels in labeling X, $avg\_\ overlap(X)$ the average interval size of the overlapping portions and $closest\_\ inter\_\ label\_\ size(X)$ the size of the interval between the closest two interval-labels in X. Intuitively, $compare_1()$ models and compares the current optimality of both labellings and $compare_2()$ captures the potential for future overlap (i.e., because having very "close" interval-labels hints to possible issues in labeling the graph in an attacked version of it). Now let us consider what happens if labeling intervals overlap ("colliding composite labels")?

- If nodes are in virtually identical/indistinguishable positions and with similar content then this is normal. The nodes are marked as such and treated identical throughout the watermarking process.

- If nodes differ in content but positions are similar, or content is close but positions are different, then variations in α, γ and the content summary key are performed in such a way as to differentiate labels

- If nodes differ in both content and position, changing also the iteration number in step one is required.

- If everything has been done and label intervals are still overlapping we can simply "melt" the labels together and treats the nodes as in case 1 (i.e., identical).

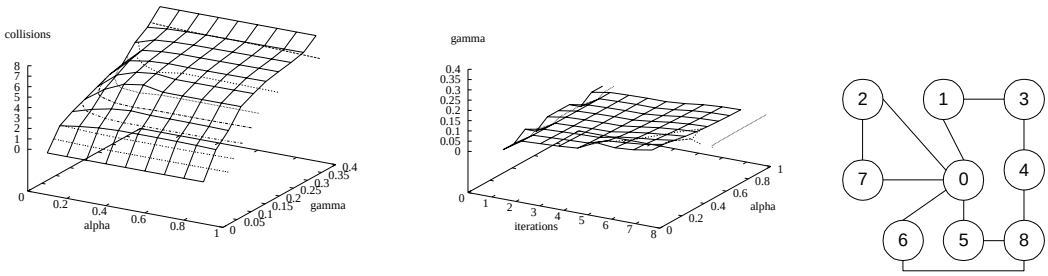


Figure 6.4. Semi-structured Aggregates: (a) The surface defining the composite label collisions appearing after 4 stages of training (i.e., $i = 4$) with a random generated set of surgeries applied to the graph. It is to be noted that lower γ values seem to yield a lower number of composite label collisions but in turn results in a lower resistance to structural attacks (i.e., as labeling will not be as resilient to graph surgeries). (b) The zero-collision (for composite labels) surface in the (iterations,alpha,gamma) space corresponding to the same set of surgeries. Its existence proves the ability to label resiliently (to the considered surgeries) without colliding resulting composite labels. Computed using the **wmx.*** package. (c) The considered graph.

In summary, the labeling process (i) collects all relevant labeling data over a number of iterations in which all of (α, γ, numbers of step-one iterations (i), content summary key and number of performed surgeries) are varied, and then (ii) decides upon a certain point in this space (defined by α, γ, i, content summary key and number of performed surgeries) which minimizes the number of overlapping label intervals and the potential for future overlaps (in case of attacks). By adapting to the given structure (i.e., by adjusting α, γ, etc), the labeling algorithm allows for control over the required trade-offs between label resilience and tolerated changes.

Note: Special consideration needs to be offered to the case of an attack modifying all existing nodes' content in a similar fashion. Alteration to the labeling scheme can be prevented in this case by introducing an additional final step of globally normalizing the labels (label intervals).

label(graph G)
  foreach node n **do** $label(n) = SUMMARY(key, n)$
  for ($\alpha = 0.1$; $\alpha < 0.9$; $\alpha = \alpha + 0.1$)
    for ($\gamma = 0.1$; $\gamma < 0.9$; $\gamma = \gamma + 0.1$)
      foreach artificial graph "surgery" (i.e., expected attacks) **do**
        perform surgery (remove node/relation(s))
        for ($iteration = 1$; $iteration < diameter(G)$; $iteration++$)
          foreach node n **do**
            $label(n) = \alpha \times label(n) + \gamma \times \sum_{neighbors(n)} label(nb)$
        foreach node n **do store** $label(n)$
      foreach node n **do store** $clabel(n) = [min(label(n)), max(label(n))]$
  choose (α, γ) minimizing the number of overlapping label intervals

Figure 6.5. Semi-structured Aggregates: Labeling Algorithm.

6.3.2 Tolerant Content Summaries

Finding an appropriate (set of) content summary function(s) that satisfy the requirements above is not trivial and strongly tied to the type of semi-structure node content and its associated transforms and envisioned attacks. The main requirements of the content summary functions considered are *the ability to be at the same time quite content specific while also degrading gracefully with minor changes in the content*. The idea is to capture and quantify certain global specific properties of the content that are still preserved in the watermarking/attack process. Research by Ari Juels et. al. [60] investigate a related notion, "fuzzy commitment".

In our implementation we used a simple content summary, a linear combination of the high order bits of the node content. The assumption here was that high order bits are reasonably stable to changes and attacks. Other applications require

different consideration. In the case of JPEG content for example, frequency domain transforms like the DCT could be considered. The tolerant summary of a JPEG file would be a combination of the most significant bits of its significant DCT coefficients etc. In the multimedia framework, feature extraction algorithms (e.g., property histograms) could be also explored as means to provide tolerant content summaries.

6.3.3 Algorithm

The main idea behind our algorithm is to use the structural resilience of the labeling scheme while leveraging content-specific one-bit watermarking methods for each node. In other words, each node in the semi-structure is considered to be a potential recipient of a one-bit watermark (using a traditional content-type specific marking method), while the actual instances of these encodings are going to be determined in a secret fashion by the node labels.

Let $clabels()$ be the composite labeling intervals as computed above (see Section 6.3.1). Depending on the size of the intervals in $clabels()$, choose b as the maximal number of most significant bits that can be considered in the numbers of every interval such that $\forall(x,y)_j \in clabels(), msb(x,b) = msb(y,b)$. In other words we aim to discover an interval-specific invariant. For each node j and corresponding interval $(x,y)_j$ by notation, let $msb_j = msb(x,b)$.

Let k be a seed to a b-bit random number generator RND and $k_1, ..., k_n$ the first n b-bit random numbers produced by RND after a secret-size initial warm-up run. We say that node j is "fit" for encoding iff $(msb_j \oplus k_j) \bmod e = 0$, where e is a adjustable encoding parameter determining the percentage of considered nodes. In other words, a node is considered "fit" if its label satisfies a certain secret criteria. On the one hand this ensures the secrecy and resilience of our method, on the other hand, it effectively "modulates" the watermark encoding process according to the actual graph structure. This naturally provides a witness and rights "protector" for the structure itself.

embed(G, wm, k, e)

$clabels() =$ **label**(graph)

$b = \{max(z) | \forall (x,y)_j \in clabels(), msb(x,z) = msb(y,z)\}$

initialize $RND(k)$

$i = 0$

sort $clabels()$

foreach $(x,y)_k \in clabels()$ **do**

$k_j = RND()$

if $((msb(x,b) \oplus k_j) \bmod e = 0)$ **then**

content_wm_node(k,$(wm_i \oplus lsb(k_j,1))$,k_j)

$i = i + 1$

Figure 6.6. Semi-structured Aggregates: Watermark Embedding Algorithm

Each node considered fit is then watermarked with the one-bit watermark defined by the XOR between the least significant bit of its corresponding k_j and $wm_i, i \in (0, |wm|)$ the corresponding watermark bit. Because of the e factor we have an average guaranteed bandwidth of $\frac{n}{e}$. In case the watermark length $|wm|$ is less than $\frac{n}{e}$, we can choose for example, the watermark bit ($\frac{n}{e}$ mod $|wm|$), effectively deploying a majority voting scheme etc. The 1-bit watermark embedding uses traditional (node) content-watermarking techniques. Because these do not constitute the main contribution of this research, in our abstract watermarking suite we considered a simple place-holder, in which each node contains a large integer value. A "1" watermark bit is considered to be present when the value is odd, a "0" otherwise.

Note: The key used in the content-watermarking technique can be the same k_j or any other agreed upon secret. There might be some benefit associated with using the *same* single key for all nodes as this could defeat inter-node collusion attacks (in which the same node content is watermarked with different keys). It is also

assumed that the content watermarking method deployed is respecting the maximum allowable distortion bounds associated with the given content. In particular, these node content-specific constraints are not impacting structural consistency. In other words, slight modifications to the actual node content (e.g., JPEG images or natural language text) do not alter global structural consistency constraints. This is subject to further research.

In the decoding phase, the *clabels*() set is re-computed. The result should be identical (in case no alterations occurred) or fairly close (because of the inherent labeling tolerance to alterations). We know $k_1, ..., k_n$, the secret node-selection keys and b. Based on these values and the composite labels, the algorithm performs node-selection and identifies a majority (or all in the case of some graph alterations occurring) of the initial nodes that were watermarked. Content-specific watermark detection is then applied to each node to retrieve each watermark bit.

---

detect(G, k, e, b)
    $clabels() =$ **label**(graph)
    initialize $RND(k)$
    $i = 0$
    sort $clabels()$
    foreach $(x, y)_k \in clabels()$ **do**
        $k_j = RND()$
        if $((msb(x, b) \oplus k_j) \bmod e = 0)$ **then**
            $wm_i =$ **content_det_node**(k,k_j)
            $i = i + 1$

Figure 6.7. Semi-structured Aggregates: Watermark Detection Algorithm

In order to perform error correction, if enough bandwidth is available (e.g., n is large enough), the algorithm embeds multiple copies of the watermark (or any other

error correction encoding). Upon detection, majority voting is deployed to increase the likelihood of accurate detection.

6.3.4 Discussion

What happens if we cannot discover a "nice enough" b? That is, what happens if different label intervals in $clabels()$ are behaving so "wildly" apart that b is going to be really small. In other words, what if there exists a node whose composite label interval has its endpoints very very far away such that the MSB common bits are just a few, or even none.

We would argue that this is a highly unlikely scenario and experiments confirm it. But if it is indeed the case then we have several options, one of which is simply ignoring the label(s) that are having far-away endpoints. Another option would be to introduce an initial normalizing step in which all the labels are normalized with respect to a common "average" value (e.g., means of means).

In order to fight false-positive claims in court we ask: What is the probability of a given watermark of length m to be detected in a random graph of size n. The assumption is of course that $m < \frac{n}{e}$. It is easy to prove that this probability is $(\frac{1}{2})^m$. In case multiple embeddings are used (e.g., majority voting) and all available bits are utilized, this probability decreases even more to $(\frac{1}{2})^{\frac{n}{e}}$. For example, in the case of a structure with 60 nodes and with $e = 3$, this probability reads *one in a million*, reasonably low.

In the absence of additional information, Mallory, faced with the issue of destroying the watermark while preserving the value of the data, has only one alternative available, namely a random attack. Two sub-types of attacks present themselves as outlined in Section 6.2.1: structural and node-content altering.

Structural attacks are handled by the tolerant nature of the labeling scheme and an experimental analysis is presented in Section 6.4.2. Here we are concerned with the node-content alteration attacks. We ask: what is the probability of success

of such an attack? In other words, if an attacker starts to randomly alter a total number of a nodes and succeeds in each case to flip the embedded watermark bit with a success rate p, what is the probability of success of altering at least $r, r < a$ watermark bits in the result, $P(r, a)$? It can be shown that

$$P(r, a) = \sum_{i=r}^{a} \binom{a}{i} \times p^a \times (1-p)^{a-i}$$

Now, remember that only every e-th node is watermarked, thus the attacker effectively attacks only an average of $\frac{a}{e}$ nodes actually watermarked. If $r > \frac{a}{e}$ then $P(r, a) = 0$. In the case of $r < \frac{a}{e}$ we have we have the corrected version

$$P(r, a) = \sum_{i=r}^{(\frac{a}{e})} \binom{(\frac{a}{e})}{i} \times p^{(\frac{a}{e})} \times (1-p)^{(\frac{a}{e})-i}$$

If $r = 4$, $p = 10\%$, $a = 20$ (33% of the nodes are altered by the attacker !) and $e = 4$, we have $P(4, 20) \approx 55 \times 10^{-6}$, again a reasonable figure, reading *fifty-five in a million*. Space constraints do not allow for a more in-depth analysis.

6.4 Implementation and Experiments

6.4.1 The wmx.* Package

wmx.* is our java software test-bed package for watermarking abstract semi-structures. We developed and implemented the algorithms presented and experimented with various semi-structured shapes. The package allows for dynamic generation and storing of graphs, graph surgeries and attacks, as well as for run-time customizable labeling and watermarking parameters (e.g., α, γ, *iterations*, *collision_bounds*). In the experiments, most of the nodes were defined as allowing for specific node content watermarking that encodes one bit per node.

Given the low probability of attack and false-positives discussed above in Section 6.3.4, one thing we believe needs to be analyzed in more detail is the actual feasibility and resilience of the labeling method. Given its importance to the overall algorithm, we implemented a test suite that allows experiments on abstract, dynamically

redefine-able structures composed of a customizable number of nodes with associated random generated (or predefined) content. We then extended the package to allow for watermarking of abstract define-able semi-structures. We performed experiments on structures with varying number of nodes and levels of connectedness. The computations were conducted on a 500Mhz PC with 128MB RAM running Linux. Code was written in Java.

6.4.2 Experiments

One of our main concern was labeling collisions, i.e., composite label sets $clabels()$ in which multiple labeling intervals are overlapping. These appear as a result of the training surgery phase, in which modifications are performed to the graph to produce the new label set. It is bad news as it creates potential ambiguity in the detection process. Surprisingly, in most cases, by adjusting the labeling training parameters $\alpha, \gamma, iterations$ we could obtain points that did feature zero collisions. In Figure 6.8 we show the zero-collision surfaces (in the α, γ space, with 3 training iterations) for two simple structures.

The considered set of training surgeries (i.e., the set of surgeries performed on the original graph before each individual labeling iteration) was randomly computer-generated from a set of global surgeries and included peripheral node removals, edge additions and removals. (To be noted that this is consistent with the assumptions made in Section 6.2.1 when discussing attack A5).

In Figure 6.9 we show the watermark behavior in the case of a random artificially generated structure with 32 nodes and 64 edges. The embedded watermark is 8 bits long. The labeling scheme was trained for 3 surgeries, also $e = 3$ (average bandwidth available is thus 10.6 bits, enough for the 8 bit watermark). It can be seen how composite labeling training results in highly resilient labels. As the number of attack surgeries increases, the watermark degrades slightly. The results are averaged over 10 runs on the same graph with different random attacks. When 8 attack surgeries

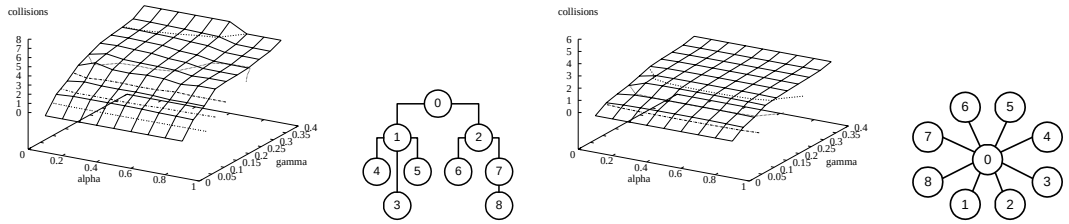


Figure 6.8. Semi-structured Aggregates: Surfaces defining the composite label collisions appearing after 3 stages of training with a random generated set of surgeries. (a) Tree shaped graph. Much of the web content online is tree-shaped. Again, note that lower γ values seem to yield a lower number of composite label collisions. (b) Star shaped graph. Note the smoother shape and the lower collision bounds, compared to (a). The same nodes were used, differently interconnected. Computed using the **wmx.*** package.

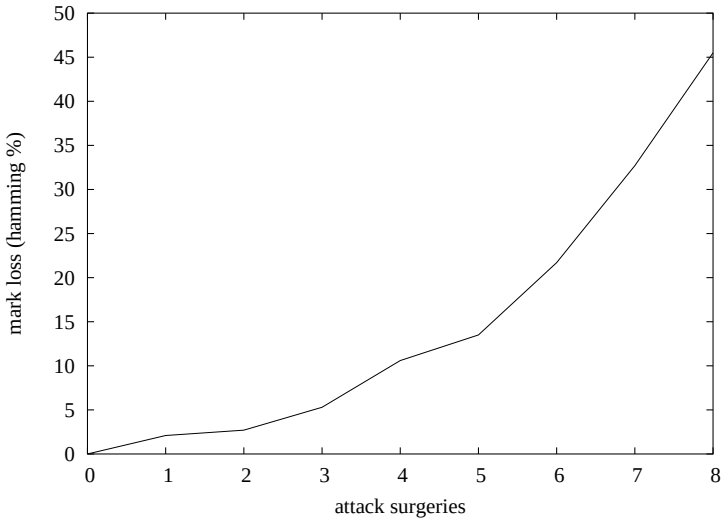


Figure 6.9. Semi-structured Aggregates: Averaged watermark loss over 10 runs of an 8 bit watermark embedded into an arbitrary 32 node graph with 64 edges. Surgery attacks are applied randomly (node removals 60%, link addition 20%, link removal 20%). The labeling scheme was trained for 3 surgeries.

are applied to the graph we can still recover 60-65% of the watermark. One has to consider also the fact that an attacker is bound not to modify the structure too much as it will eventually distort.

6.5 Conclusions

We introduced an algorithm for rights protection watermarking of semi-structured content. More specifically we are concerned with protecting the value inherent in the structure itself. Various new challenges are associated with this new domain. Benefiting from the dual nature of semi-structures, our algorithm makes use of both the available node content as well as of the value-carrying structure, through the processes of canonical labeling, node content summarization and content-specific mark encoding. The idea behind content-specific mark encoding is to use traditional known watermarking techniques, in encoding parts of the watermark in the node content. Providing a canonical labeling scheme, "trained" to tolerance for a set of graph modifications is essential in being able to later-on identify nodes selected in the 1-bit node mark content-specific encoding process. Our algorithm does not require the original un-watermarked object in order to perform mark detection. Published research results of this work include [13].

Further work is required in improving content summarization and tolerant labeling. Different application domains will require specific approaches. An alternative idea would be using bandwidth available in the specifications of inter-node relations.

7 MODEL OF WATERMARKING (PART TWO)

In Chapter 2 we introduced a model defining main constructs for our watermarking framework. After designing a set of specific watermarking solutions for different data types and aggregates, we are aiming to understand some of the more generic challenges and limitations of watermarking. Does it indeed live up to its expectations? Are there generic limits to it? Can those be assessed and quantified for a broad class of applications? These are some of the issues we are looking at in this chapter.

7.1 First Principle of Watermarking

Intuitively, the more one modifies a Work with the purpose of encoding a watermark, the higher the potential of that watermark to be resilient to attacks. This is so, because now Mallory will need to perform a proportionally higher amount of work. However this potential can only materialize if there is a direct semantic and quantitative link between the amount of modifications performed in watermarking and the work that Mallory has to perform to defeat the watermark. In the absence of any external aids (e.g., legal means), this link is enforceable by mark encodings that actually alter the value of the data, thus necessarily forcing any related attack to also operate in this domain and potentially alter the value of the data.

Then, as mentioned before (Chapter 2), this becomes (metaphorically) a game between the watermarker and Mallory. In this game, the watermarker and Mallory play against each other within subtle trade-off rules aimed at keeping the quality of the result within acceptable bounds. This is why from the watermarker's perspective it would be nice to yield the "best bang for the buck", i.e., for minimal watermarking modifications to yield a maximum resilience to attacks (see Section 7.2).

Now, if the encoding is based in a non-valued part of the Work, it can be simply removed without altering the resulting value. Similarly, given a set of dimensions of value of the Work (usability domains), if the watermarking encoding does not alter *all* of them, then Mallory can simply "extract" and illicitly profit from that particular value. Thus we define the *first principle of watermarking*.

> Given a Work and a set of associated dimensions of value (usability domains), a sound watermark algorithm results in an encoding that necessarily alters the Work in all of these domains.

What then happens is that we, in effect, tie Mallory's hands behind the back and force him into the only viable attack scenario which is also altering the Work. This enables the semantic link discussed above and starts the game.

7.2 Challenge of Watermarking

From the watermarker's perspective it would be nice to yield the "best bang for the buck", i.e., for minimal watermarking modifications to yield a maximum resilience to attacks. Thus, one challenge of watermarking derives from the main trade-off to be found here, namely between the requirement to preserve data usability and the desire to provide an encoding as resilient as possible to malicious transformations and attacks.

Given a maximum allowable bound for difference in usability Δu_{max} and a maximum upper bound on the false positive probability ϵ_{max}, we define *the challenge of watermarking* as the ability to find the most powerful marking algorithm $a \in \mathbb{A}_{\mathbb{D}}$ for a given key $k \in \mathbb{K}$ that still works within the given usability bounds, that is,

$$\Delta u(d', d) < \Delta u_{max}$$

and

$$P(w(x, k) = 1 | w(d', k) = 1) < \epsilon_w < \epsilon_{max}$$

for all x inside of d's usability vicinity of radius Δu_{max}, where d' is defined by $a(d,k) = (d', w)$.

In other words, the main concern in watermarking lies with keeping the required usability level of the object unchanged or close to its original value, while still featuring enough power. A sound solution will try to determine the main usability domains for a particular to-be-watermarked object and then preserve usability in those domains.

7.3 Limits

This section discusses inherent vulnerabilities of digital watermarking that affect its mainstream purpose of rights protection. We ask: how resistant is watermarking to un-informed attacks? We identify an inherent trade-off between two important properties of watermarking algorithms: *being "convincing enough" in court* while at the same time *surviving a set of attacks*, for a broad class of watermarking algorithms. We show that there exist inherent limitations in protecting rights over digital Works. In the attempt to become as convincing as possible (e.g., in a court of law, low rate of false positives), watermarking applications become more fragile to attacks aimed at removing the watermark while preserving the value of the Work. They are thus necessarily characterized by a significant (e.g., in some cases 35%+) non-zero probability of being successfully attacked without any knowledge about their algorithmic details. We quantify this vulnerability for a class of algorithms and show how a minimizing "sweet spot" can be found. We then derive a set of recommendations for watermarking algorithm design.

7.3.1 Introduction

Up to this point, a common consensus has been implicitly assumed with respect to watermarking, namely that it indeed lives up to its claimed features. As previously outlined, the main desiderata and features of watermarking include: it should

not degrade the perceived quality of the marked Work; the ability to detect the presence/content of a watermark should require the knowledge of a secret (key); different watermarks in the same Work should not interfere with each other; collusion attacks should not be possible; *the watermark in a certain Work should survive any value-preserving transformation on that particular Work.*

But how well do algorithms and their associated implementations conform to these ideals? With respect to the fingerprinting application of watermarking, we know now that arbitrary large collusion attacks cannot be defeated against [15]. Moreover, while most watermarking algorithms prove to be safe against a considered set of value-preserving transformations (e.g., JPEG compression) they certainly fail with respect to many others. This shortcoming can be directly traced back to the relativity of the "value" and "quality" concepts.

Several, mostly experimental efforts explored the ability to analyze and quantify the "goodness" of watermarking applications, resulting in various watermark benchmarking "suites" such as StirMark [61] CheckMark [62] OptiMark [63] mainly for multimedia (i.e., images). Additional research [16] [17] [18] aimed at analyzing concepts such as available bandwidth in the broader area of information hiding from a signal-processing, information-theoretic perspective, focusing mainly on various multimedia techniques.

For example, in Chapter 3, watermarking relational data is often subject to a very restrictive set of data quality assessments, making it often difficult to embed even a single bit watermark. This narrow embedding bandwidth raises major concerns with respect to theoretical watermarking vulnerability limits. After all, it seems that if the changes allowed to be performed are small and very application-specific, an attacker would be more likely to undo them or simply alter the Work enough to destroy the watermark while preserving sufficient value.

Thus one particular question becomes of interest, namely: *Are there theoretically assessable bounds on watermark vulnerability with respect to an arbitrary watermarking method?* In other words, what is the inherent safety/vulnerability of a generic

(i.e., with a minimum amount of assumptions, without considering implementation particularities) watermarking algorithm? An answer to this question might afterward derive real-life recommendations for fine-tuning actual algorithms to increase their marking resilience.

In this section we: **(i)** identify and analyze inherent limitations of watermarking, including the trade-off between two properties: *being enough 'convincing" in court* while at the same time *surviving a set of attacks*. This trade-off derives naturally from the inverse proportional nature of their relationship. In the attempt to become as court convincing as possible, a watermarking application becomes more fragile to attacks aimed at removing the watermark, while preserving the value of the Work. It becomes thus necessary characterized by a significant non-zero probability of being successfully attacked. **(ii)** define a quantifiable measure of generic watermarking safety, namely watermark *vulnerability*. **(iii)** outline an optimality principle (quantified and proved for a broad class of algorithms) that postulates the minimization of watermark vulnerability in specific data points.

7.3.2 A Sample Watermarking Algorithm

The main purpose of this section is not the discussion of a certain watermarking algorithm but rather of general bounds that could be discovered over large classes of algorithms. While, in order to validate and quantify any potential results, it is desirable to be able to refer to a certain algorithm, care needs to be taken in making sure that it features just enough specifics to be usable for the purpose, while at the same time being representative for a majority of watermarking applications.

Thus, let us define a generic marking algorithm, featuring properties likely to be found in many existing ones. The starting point is our work on numeric relational data in Chapter 3. While the implementation and design details are complex, the main behavior characteristics of the encoding, we believe, is quite representative of

many watermarking applications. Here we present a simplified view of that work, that should allow the use of this as an example without sacrificing any generality.

Let us consider the input Work (i.e., relational data) being a set of n numbers $(s_i)_{i\in(1,n)}$. The output of the watermarking algorithm is another set of numbers $(v_i)_{i\in(1,n)}$, the watermarked version. The usability space is n-dimensional in this case. It suffices to know that watermark encoding occurs by slight modifications in the number set (altering some secret distribution characteristics) [1]. For more embedding algorithm details see e.g., Chapter 3. $(v_i)_{i\in(1,n)}$ is an altered version of $(s_i)_{i\in(1,n)}$ and, in a sound watermarking application, the performed alterations are bounded by semantic value constraints. In other words, the result (watermarked data) has to still be of (acceptable) value, with respect to the value metric for the considered application.

One potential quality metric here can be defined in terms of the euclidean distance between the original Work and it's watermarked version, i.e., as the normalized sum of the square roots of all the performed (small) changes:

$$MSE = \frac{\sum (s_i - v_i)^2}{n} \tag{7.1}$$

The quality of the result is said to be satisfactory if this distance is below a certain allowed upper bound Δu_{max}, in other words, if the resulting watermarked Work is not "too far away" from the original. In the associated n-dimensional usability space this quality assessment defines a vicinity U_{max} of the initial data set, in the shape of a sphere of radius Δu_{max}.

For simplicity, let's assume for now that U_{wm} (the zone in which we find Works that exhibit the watermark property) is continuous and also sphere-shaped (any euclidean-type of distance would result in this). We are relaxing this assumption later on. A 2-dimensional illustration of this is given in Figure 2.1 (a).

[1]The generality of this is warranted by the fact that many other existing algorithms [2] are proceeding similarly, by mapping the input Works (e.g., images) to a quantifiable domain (e.g., DCT coefficients, Least Significant Bit spaces) that is then to be "modulated" (i.e., altered) according to the watermark signal.

While, intuitively, considering other more complex encoding details and different quality metrics might result in changes in the quantitative results, arguably, qualitative aspects are likely to be very similar.

7.3.3 Analysis

Given a certain Work O (whose rights belong to Alice) and its watermarked version O', lets put ourselves in the position of Mallory attempting to attack O' and yield an un-watermarked Work O'', still usable with respect to O.

There exists a court-authority to which both Mallory and Alice would like to prove their rights ownership to. Lets assume this court mandates a certain maximal value on ϵ_w (see Section 2.1), for example 3% [2] , value which Mallory knows (usually public). There also exists an associated maximal allowable distortion bound, Δu_{max} (the radius of U_{max}), guaranteed on the resulting watermarked Work with respect to the original. This value is probably also public as it is included in the watermarked Work specification, delivered by Alice.

Mallory knows O' and it is only reasonable to assume that it is also aware of the quality metric (e.g., MSE, see Section 7.3.2) associated with the data domain of the Work in question. What Mallory doesn't know is the original Work O as well as its maximal usability vicinity (U_{max}).

Getting back to playing Mallory, what are our options in attacking? Given the knowledge of O', it is the only natural starting point of our attack. It is also safe to assume that Alice encoded a watermark that does not exceed the ϵ_w false positive rate (modeled by $U_{max} \cap U_{wm}$), thus

$$||U_{max} \cap U_{wm}|| = \epsilon_w ||U_{max}||$$

Given this limited amount of available knowledge, the remaining thing to do is to randomly define a distance R_a from O' and "attack" by picking a point inside the resulting R_a radius vicinity of O', U_{attack}, see Figure 7.1.

[2] The court is convinced only above a 97% true-pozitive rate.

Note: Because we know ϵ_w, we can estimate the minimal area of U_{wm} as being at least $\epsilon_w||U_{max}||$. This is why when choosing R_a we have to make sure that the resulting U_{attack} vicinity shape is larger than U_{wm}. Thus we have at least

$$R_a > \sqrt{\epsilon_w}\Delta u_{max}$$

Otherwise, this, and the fact that U_{attack} is also centered in O' lead immediately to the conclusion that each point inside of U_{attack} corresponds to a watermarked Work, thus no attack success is achieved. Thus, we can improve our success probability even more by picking an "attack point" only within $U_{attack} \setminus U_{\epsilon_w}$, where U_{ϵ_w} is a vicinity of O' of radius ϵ_w. For space reasons this provision is not included in this analysis. An extension is discussed in Section 7.3.4.

Also because Δu_{max} is assumed to be public, Mallory can infer that there is no point in choosing an U_{attack} larger than required to just include U_{max} (all points outside of U_{max} are irrelevant anyway). Thus at the limit, $R_a < 2\Delta u_{max}$. In conclusion, we have

$$\frac{R_a}{\Delta u_{max}} \in (\sqrt{\epsilon_w}, 2) \tag{7.2}$$

Given the above, a successful attack is one which yields results in the area $U_{sa} = U_{max} \cap (U_a \setminus U_{wm})$. The probability of this becomes then

$$P_{sa} = \frac{||U_{sa}||}{||U_a||} = \frac{||U_{a2} \setminus U_{wm1}||}{||U_a||} = \frac{||U_{a2}|| - ||U_{wm1}||}{||U_a||} \tag{7.3}$$

We know that

$$\frac{||U_{wm} \cap U_{max}||}{||U_{max}||} = \frac{||U_{wm1}||}{||U_{max}||} = \epsilon_w$$

$$||U_{max}|| = \pi\Delta u_{max}^2$$

and

$$||U_a|| = \pi R_a^2$$

thus (7.3) becomes

$$P_{sa} = \frac{||U_{a2}|| - \epsilon_w||U_{max}||}{||U_a||} = \frac{||U_{a2}|| - \epsilon_w\pi\Delta u_{max}^2}{\pi R_a^2} \tag{7.4}$$

Convince-ability Trade-off: Equation (7.4) outlines the direct relationship between the probability of a successful attack and ϵ_w.

> The smaller the ϵ_w value is (i.e., the more "convincing" in court, from a false-positive rate point of view), the higher the probability of success of an attack.

Because the usability shapes are "circular" (see Section 7.3.2), the radius of U_{max} is Δu_{max} and the radius of U_a is R_a. $||U_{a2}||$ becomes

$$||U_{a2}|| = \Delta u_{max}^2 cos^{-1}(\frac{d^2 + \Delta u_{max}^2 - R_a^2}{2d\Delta u_{max}}) + R_a^2 cos^{-1}(\frac{d^2 - \Delta u_{max}^2 + R_a^2}{2dR_a}) - \frac{1}{2}\sqrt{(-d + R_a + \Delta u_{max})(d + R_a - \Delta u_{max})(d - R_a + \Delta u_{max})(d + R_a + \Delta u_{max})} \quad (7.5)$$

where $d = d(O, O')$ is the distance between O and it's watermarked version (O') [3] Because O' has to be within maximum allowable usability vicinity of the original, it is necessary that $d \leq \Delta u_{max}$. It can be seen that $||U_{a2}||$ decreases with the increase of d. It reaches minimum when d is maximal, namely when $d = \Delta u_{max}$, in other words, when the watermarking algorithm produces a resulting watermarked Work O' (asymptotically) on the boundary of U_{max}.

Optimality Principle: From (7.4) and (7.5) we derive that (in our class)

> the vulnerability of a watermarking scheme is minimized when it yields watermarked results on the boundary of the maximum allowable usability vicinity of the original un-watermarked Works.

Recommendation: The optimality principle postulates (and we prove it for this example) the naturally occurring minimization of the successful attack probability on the usability vicinity boundary. This yields a recommendation to make the

[3] For simplicity, this formula is valid for values of $R_a < \Delta u_{max}$, see [64]. The reasoning for $R_a \in (\Delta u_{max}, 2\Delta u_{max})$ is identical.

watermarking algorithms by design conform to this principle. Choosing this design in our relational database watermarking software increased its resilience to random ϵ-attacks (see Chapter 3) by as much as 15%.

From now on let us assume that we are going to conform to this principle, in other words $d = \Delta u_{max}$, in which case we have the revised version of (7.5)

$$||U_{a2}|| = d^2 \cos^{-1}(1 - \frac{R_a^2}{2d^2}) + R_a^2 \cos^{-1}(\frac{R_a}{2d}) - \frac{1}{2} R_a \sqrt{(2d - R_a)(2d + R_a)}$$

and (7.4) becomes

$$P_{sa} = \frac{d^2 \cos^{-1}(1 - \frac{R_a^2}{2d^2}) + R_a^2 \cos^{-1}(\frac{R_a}{2d}) - \frac{1}{2} R_a \sqrt{4d^2 - R_a^2} - \epsilon_w \pi d^2}{\pi R_a^2}$$

For a typical value of $\epsilon_w = 0.03$ (e.g., the court is convinced with a 3% false-pozitive rate) Figure 7.2 (a) depicts P_{sa} as a function of the $R_a/\Delta u_{max}$ ratio. It can be seen that there exists a clear maximum vulnerability spot (around $R_a/\Delta u_{max} = 0.4$) in which the probability of success of an attack exceeds $30 - 35\%$! This result is not dependent on the technicalities of the watermarking method. It becomes so much more compelling as it occurs for *any* marking algorithm (using similar Works quality metrics, see Section 7.3.4) and any input. In this same figure we depicted the evolution of P_{sa} for $\epsilon_w = 0.01$. This more court-convincing setting (lower false-positive rate) results in even higher attack success rates (up to 40% and above).

From Mallory's perspective this is good news. It turns out that it *is* possible to defeat watermarking algorithms with a surprisingly high success rate, without any additional (insider's) knowledge [4]. This is the case even if these algorithms conform to the optimality principle outlined above. There seems to exist a "sweet spot" (characterized by $\frac{R_a}{\Delta u_{max}} \in (0.4, 0.6)$) in which P_{sa} is maximized. Mallory could make use of this by fine-tuning its attacks. This is confirmed in real data experiments (see Chapter 3) in which random attacks were more likely to succeed within this range.

[4]This is an inherent limitation of watermarking as a concept. From the smart attacker's perspective, any additional knowledge can only improve on this probability.

7.3.4 Discussion

In the previous sections we analyzed a special class of watermarking algorithms considered to be simple and illustrative enough yet within the space and complexity constraints of the current scope. This class is defined by a set of assumptions. The quantitative details of our attack analysis are developed for a particular data quality metric, mean squared error. This resulted in the usability vicinity for the watermarking algorithm considered to be continuous and of a spheric shape. How restrictive are these assumptions? Can the results be applied to broader classes of algorithms and how? In the following we are discussing these and other issues and propose extensions.

High dimensional usability spaces. In the above we considered a finite dimension of the usability space. In particular, the quantitative analysis was performed on 2D shapes. While many of real life applications feature only a finite dimensionality we can't help to wonder what happens (e.g., to U_{sa}) when the number of dimensions of the usability space grows. To the extreme, what happens when we go to infinity? While a full fledged analysis is out of the current scope, the intuitive feel is that in that case U_{sa} goes to zero. On the other hand, arguably, the U_{sa}/U_{attack} ratio has to be preserved. Thus it is unclear what the behavior would be in that case. We propose to explore this in future research.

Shape. The assumption of a particular quality metric (mean squared error) determines directly the shapes of the analyzed Work's usability vicinities. How does this affect the generality of our analysis? With respect to continuous shapes, while quantities may vary, the behavior of any watermarking algorithm is arguably ruled qualitatively by the *convince-ability trade-off*. This is an immediate result of the generality of (7.3) which does not depend on shape.

The validity of the *optimality principle* is intuitive for quality metrics resulting in convex usability vicinities. As the watermarked results are closer to the boundary of U_{max}, the probability of success of an uninformed attack decreases (see Figure 7.1).

This is not obvious in the non-convex case. Figure 7.3 (b) illustrates a case where it does not work. Both O'_1 and O'_2 are (asymptotically) on the boundary of U_{max} but while O'_1 seems to minimize it's associated P_{sa}, O'_2 offers a highly attackable starting point for an uninformed Mallory[5]. A possible solution could be the construction of a "safe" subset of the shape's boundary. In this case, a watermarking algorithm should start by determining a subset of U_{max} in which watermarked Works result in a minimized P_{sa}. This is subject to further exploration.

Sparsity. How does our analysis suffer if the considered usability vicinities (e.g., U_{max}, U_{wm}) were to be sparse, "scattered" throughout U_{data}? Sparsity in this domain is directly related to the function defining the maximal allowable change to the initial Work. While most known applications (e.g., wavelets, DCT for media) feature continuous usability vicinities [6], considering the case of sparsity becomes interesting as it might offer more generality. Sparsity is often associated with higher level semantic constraints. One scenario, image watermarking within a health care framework might present various legal and medical restrictions. For example, encoding a mark in a composite heart-disease image might be required not to result in alterations to the actual region containing the heart itself.

Given the sparse nature of the vicinity shapes (see Figure 7.3 (a)), it all boils down to the assumption made about the amount of knowledge available to Mallory. If Mallory is aware of the details of the actual cause of sparsity (i.e., function of allowable change), then it can deploy a virtually identical attack (with the one considered in our initial analysis) targeted at each sub-part composing the sparse distribution. In this case our analysis is identical.

The scenario becomes different if this knowledge is not available to Mallory. In this case the only viable option is to randomly attack within the known data domain. In this case the attack success probability has to suffer at least by a factor of U_{max}/U_{data} (assuming uniform distribution of the sparse fragments of U_{max} through-

[5]Because its attack vicinity ($U_{attack2}$) "intersects" comparably much of U_{max} as any interior point does. As discussed in the case of the optimality principle, interior points are by default non-optimal.

[6]Encoding ultimately occurs by altering a set of numbers within MSE types of constraints.

out U_{data}). Apparently this is good news for Alice. It seems that if the distortion constraints are of a higher semantic nature (i.e., resulting in sparse vicinities), successful watermark embedding [7] is less vulnerable (by a factor of U_{max}/U_{data}). This scenario requires more attention.

Partial Sparsity. A particular case occurs when only some of the vicinities are sparse, for example U_{wm}. This is equivalent to saying that the false positive upper bound, ϵ_w is "spread" throughout U_{max}. Then (see Figure 7.1) in (7.3) $||U_{wm1}||$ is to be replaced by $\epsilon_w U_{max}\frac{U_{a2}}{U_{max}} = \epsilon_w U_{a2}$ (assuming uniform distribution of U_{wm1} across U_{max}) and we yield

$$P_{sa} = (1 - \epsilon_w)\frac{||U_{a2}||}{||U_a||} \tag{7.6}$$

with a similar qualitative behavior to (7.3), thus our analysis holds.

Increasing P_{sa}. In the case of a continuous U_{wm}, Mallory can increase P_{sa} even more as follows. Knowing Δu_{max} and

$$||U_{wm1}|| = \epsilon_w \pi \Delta u_{max}^2$$

can immediately result in the computation of the radius R_{wm} of (the entire) U_{wm} with a formula similar to (7.5). Once R_{wm} is known, Mallory can increase P_{sa} by choosing a point only within $U_{attack} \setminus U_{wm}$ which is now clearly specified. This increase to P_{sa} in (7.4) could be quantitatively significant, in our example, by a factor of

$$\frac{R_a^2}{R_a^2 + R_{wm}^2}$$

7.4 Discussion

7.4.1 Oracle Attacks

But couldn't Mallory simply try out different watermarking keys with the purpose of removing a potential watermark in a given Work? While this might appear to be

[7] Although probably less likely to succeed given these very constraints often hard to accommodate. For example most watermarking methods in the frequency domain [3] can not directly handle a constraint such as the health-care example above.

a valid concern at a first glance, a deeper insight will immediately reveal the fact that, in order to actually "know" that a certain key does or doesn't "work", Mallory needs to "ask" somebody that does know. This would be usually Alice (the actual rights holder, see Figure 1.2) or any party enabled to deliver such a response. Let's call this party an *oracle* and Mallory's attempts an *oracle attack*. An oracle could be used by Mallory to effectively "explore" usability spaces possibly "plotting" them out fully, thus defeating any other watermark.

Fortunately, oracle attacks are not of concern in the general case simply because such oracles do not exist usually. The incentive of Alice or anyone else being able to perform watermark detection (thus usually possessing the original key – in a symmetric watermarking system), in doing so at the request of arbitrary third parties is (arguably) zero.

While the existence of an oracle is not (arguably) of concern, Mallory could however attend court sessions in which certain Works are in dispute. Because of the nature of most court-proceedings, the watermark detection key is going to be usually subject to public disclosure (or limited to the disputing parties, but then Mallory could be one of them). This would enable Mallory to (in effect) build a limited oracle attack machine by collecting disclosed copies of (Work,watermark) pairs. While we believe that the limited number of such disclosures does not generally enable effective oracle attacks (with respect to the usually large key space), nevertheless, we believe that oracle attacks are interesting to explore, for example under a limited oracle model (e.g., limited number of allowed inquiries). Alternately a zero-knowledge method for mark detection could be devised. This is subject for future research.

Another related public-disclosure court-related issue is the fact that, in the case of symmetric watermarking methods, revealing the detection key is equivalent to revealing the original (un-watermarked) Work. The court cost-model and default trial claims should be devised so as to consider this and provide strong counter-incentives for Mallory. The damages that Mallory should pay to Alice should cover also this associated public disclosure of the original Work.

7.4.2 Persuasiveness and Watermark Length. Distance Metrics

In this section we are trying to answer the following question: *what happens if the recovered watermark is disturbed with respect to the expected one?* In other words, how convincing is a proof based on a detected watermark that does not exactly match the original? As the main goal of such a solution is to provide a method for proving ownership of data in a court of law this question becomes of particular concern.

We start the search for an answer by a reminder that the essence and resilience power of a certain watermark as a rights protection witness (its ability to convince in court) can be naturally modeled by the "improbability" of its occurrence in a random piece of data (see Chapter 2). The probability of a random occurrence of the watermark property (false positives) needs to be bounded and sufficiently small. Nevertheless, because watermarking deploys Information Hiding and historically derives from it, the common practice is to associate the (essentially boolean) watermark property with a string of bits with some attached meaning, e.g., "(c) walmart". Then the Work to be protected is said to be watermarked if it somehow "hides" the given string of bits. In other words, the boolean watermark property now becomes the predicate "Work contains mark string". The challenge of a good watermarking algorithm is to make optimal use of the data in such a way as to maximize the ability to answer the "contains mark string" question at detection time, beyond doubt (i.e., beyond a certain threshold of likelihood, e.g., 97%) while retaining its value.

Given a certain amount of encoding bandwidth (e.g., considering a particular encoding method), at one extreme, the mark string can simply be 1 bit long, in which case it directly corresponds to the boolean translation of the yes/no answer to the detection question. In the other extreme, the mark string could be as long as the actual available bandwidth. In the general case, the watermark is probably going to be of a size equal to a fraction of the available bandwidth.

In all of these encodings, the main resilience of the particular method is not related to the length of the mark string but rather to the resistance to attacks

(of the ability to answer the question beyond doubt) and the probability of false positives, directly deriving from the underlying maximal bandwidth. As also outlined in Section 2.3, this is one of the important differences that distinguish Watermarking from pure Steganography. In Watermarking, what matters is the available underlying encoding bandwidth and its optimal utilization in inducing a highly-unlikely data property (rights witness) resistant to alterations and attacks. The payload (mark bit) size and content plays a little (if any) role in rights protection. The underlying bandwidth is the element that directly defines the available encoding entropy.

For example, packing/hiding x y-sized bit strings into a $z = xy$ sized underlying band (e.g., z "fit" tuples) is ultimately equivalent from a convince-ability point of view with packing $2x$ $\frac{y}{2}$-sized strings in the same z sized band. Intuitively, using a mark string longer than one bit does not affect the inherent mark resilience (probability of a false-positive and resilience to random changes) but rather only the ability to recover all the mark bits correctly, as each bits embedding is going to be on average weaker (less underlying "support" bits, before error correction). But, because the mark string is longer, the probability that it appears randomly (even damaged) decreases with its length. In summary, the resulting false-positive probability is the same and derives from the number of underlying available bits, before error correction, an often clearly quantifiable measure.

For example, as also discussed in Section 4.4.4 the likelihood of a certain string of size s to appear in a random Work is 2^{-s}. Thus if the original underlying watermark bandwidth was (approximately) $\frac{N}{e}$ (see Section 4.3.3) and, at detection time it turns out that Mallory succeeded in distorting a number of l bits in the result (before error correction), this likelihood drops down according to the number of still "matching" bits and it becomes $2^{-(\frac{N}{e}-l)}$. The convince-ability of the embedding is then $1 - 2^{-(\frac{N}{e}-l)}$. If $N = 6000$, $e = 60$ and Mallory succeeds in eliminating 50% of the bits (i.e., $l = 50$, highly unlikely, see Section 4.4.4), this is still a very convincing $1-2^{-50}$, very close to 1.

With respect to the mark string associated with a given watermark, the question arises: *how do we compare two strings in this domain in a quantifiable way?* More specifically, how do we measure watermark distortion? Because of the direct dependence between the number of correct recovered watermark bits and the actual encoding convince-ability outlined above, we chose to use the hamming distance as a natural metric for mark loss [8]. Thus, if w is the "original" mark string and w' a "distorted" version of it, then the mark distortion is defined to be the normalized Hamming distance $\frac{Hamming(w,w')}{|w|}$, between the two strings. Using a normalized value ensures the ability to compare distortion values between different applications (i.e., with different mark string sizes).

7.4.3 Note on Collusions

In this dissertation Watermarking was defined as a method deploying Information Hiding for the purpose or Rights Assessment for Digital Works. A related mechanism that could benefit from some of the tools developed here, is fingerprinting, i.e., deploying Information Hiding for the purpose of tracing license agreement violators.

Fingerprinting works by deploying different watermarks for each sold copy of a certain digital Work, each mark identifying the legal license purchaser of the corresponding Work. If a certain Work containing a watermark pointing to party A is then found in possession of a different party B (or in the public domain) it can be inferred that A did provide B with a copy of the Work. This may or may not violate A's use license for the Work.

A specific challenge associated with fingerprinting is an attack by multiple malicious parties "colluding" by e.g., purchasing (different) copies of the same Work (i.e., fingerprinted with different watermarks) and then "combining" them into one copy that doesn't contain a valid watermark anymore.

[8] Because the actual dependency is exponential, this also models a somewhat logarithmic scale of the "loss of convince-ability".

While some of the tools and solutions developed in this dissertation could be used for fingerprinting, this potential was not addressed in detail here. This is also why collusion attacks have not been specifically discussed as, for rights assessment only, it is (arguably) reasonable to assume the existence of a single watermark identifying the rights owner throughout the lifetime of the digital Work [9].

While arbitrary large collusions (many Mallory's) cannot be defeated against[10] if one can bound the number of possible sold copies (and thus the maximum collusion size) a special coding scheme was designed to defeat against collusions [15]. These results can be directly applied in our framework. It would be interesting for future research to explore this into more details.

7.5 Conclusions

In this chapter we explored limitations of watermarking. We asked: does it indeed live up to its expectations? Are there generic limits to it? We then quantified those limits for a broad class of applications and discovered a set of associated bounding principles. Published results of this work include [6].

[9] If the rights-owner changes (e.g., from A to B) an external sale-document can be signed by A and handed to B specifying the "sale of rights" and the actual watermark can remain un-changed in the sold Works.

[10] At the extreme, each Mallory's copy contains at least one bit of the original Work. Then a collusion equal to the bit-size of the Work can simply concatenate these "correct" bits together to obtain an original un-watermarked Work.

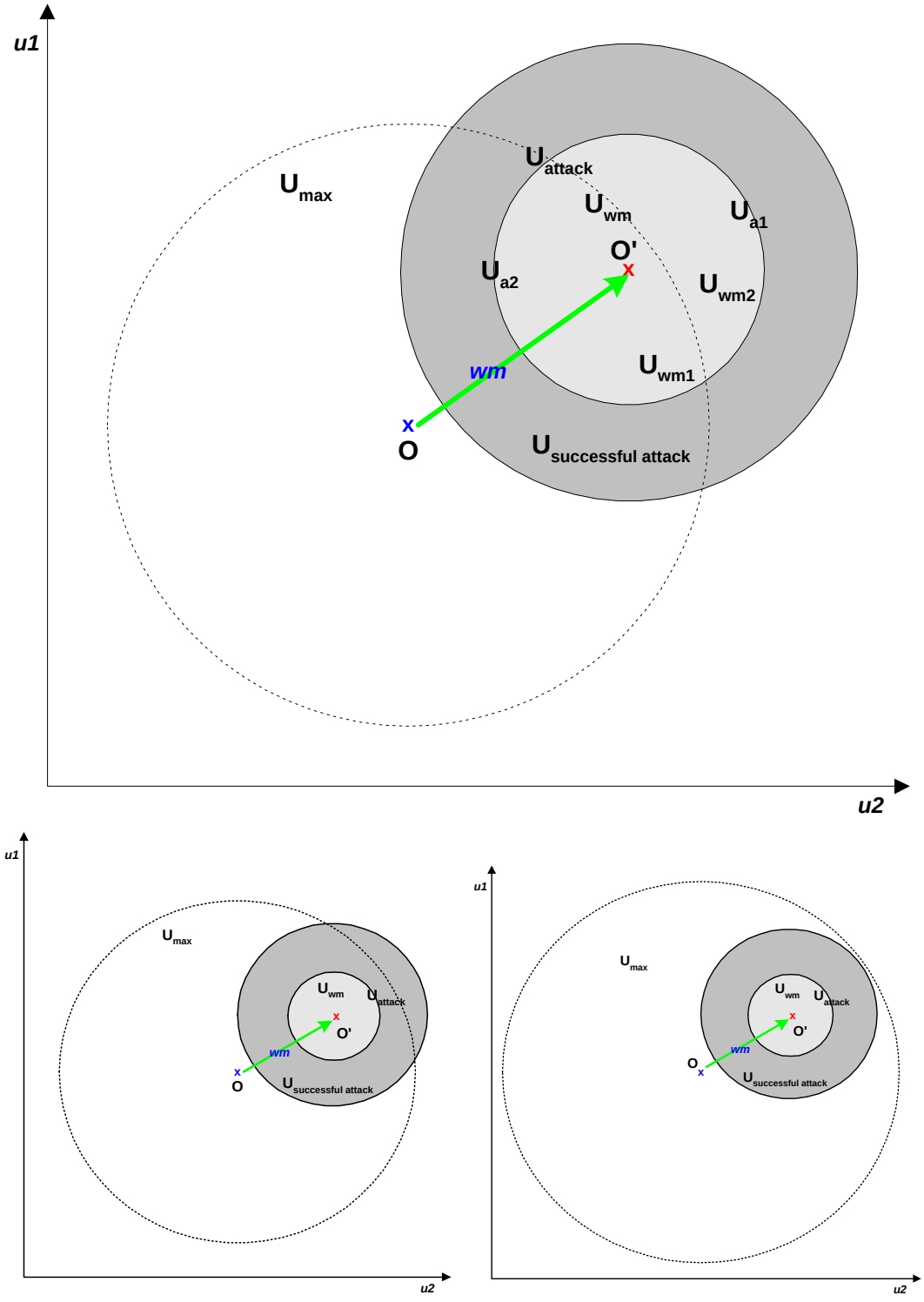


Figure 7.1. Model of Watermarking: Mallory attacks (different variations).

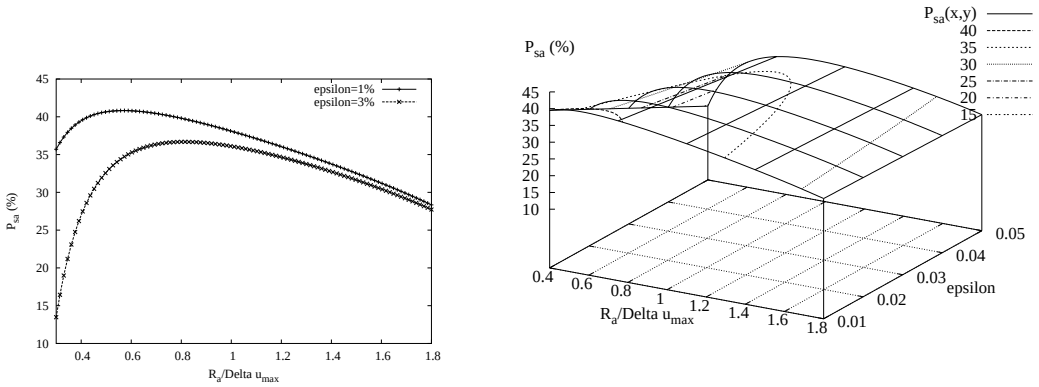


Figure 7.2. Model of Watermarking: (a) No matter how sophisticated the watermarking method, there exists a random attack with a success probability (e.g. of 35% and above, shown here for 2-dimensional usability spaces, see Section 7.3.4 for a discussion on high dimensional spaces). It can be seen that a lower ϵ_w value (more convincing in court) yields an even higher upper bound on attack success probability (2D cut through (b)). (b) The 3D evolution of P_{sa} with varying ϵ_w and $R_a/\Delta u_{max}$.

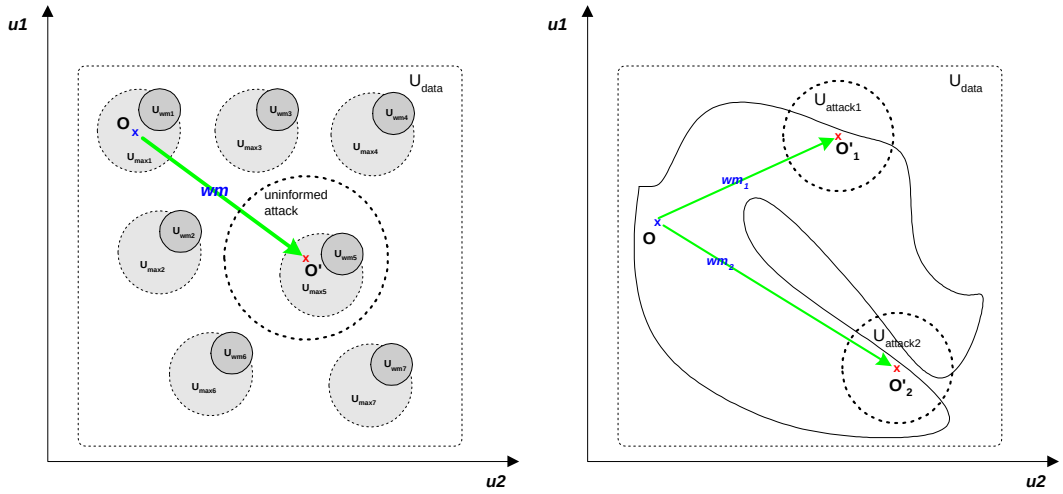


Figure 7.3. Model of Watermarking: (a) Sparse maximum allowable usability vicinity, $U_{max} = \cup(U_{max_i})$, (b) A concave U_{max} does not respect optimality.

8 THE FUTURE

In this dissertation we defended our thesis that Information Hiding can be successfully deployed as a tool for Rights Assessment for discrete digital Works. We defined and explored a foundational model and used it to discover associated principles and challenges of watermarking. We proposed and analyzed solutions for resilient Information Hiding for different discrete data types, including numeric and categorical in a relational framework, streaming sensors, and semi-structured aggregates.

This work yielded real-world impact results, including a industry-level software package for relational data watermarking and an associated pending patent application. We believe it to be of significant additional potential, for example in the deployment in an industry standard database management system.

A multitude of associated research avenues present themselves, including: a deeper understanding of limits of watermarking for a broader class of algorithms, the ability to defeat additive watermark attacks, an exploration of oracle attacks, zero-knowledge watermarking, deploying the solutions introduced here for fingerprinting, a solution for categorical sensor streams, handling scenarios at the intersection of numeric and categorical data types in a relational framework.

Additionally, of particular interest for future research exploration, we envision cross-domain applications of Information Hiding in distributed environments such as sensor networks, with applications ranging from resilient content annotation to runtime authentication and data integrity proofs.

As increasing amounts of valuable discrete information is transferred through and processed within distributed inter-connected environments, the technological ability to assert and prove associated rights (and possibly propagate integrity proofs) becomes essential. Our work is to constitute a step toward an unified rights protection framework for arbitrary digital Works.

LIST OF REFERENCES

[1] Christian Collberg and Clark Thomborson. Software watermarking: Models and dynamic embeddings. In *Principles of Programming Languages*, San Antonio, TX, January 1999.

[2] I. Cox, J. Bloom, and M. Miller. Digital watermarking. In *Digital Watermarking*. Morgan Kaufmann, 2001.

[3] S. Katzenbeisser and F. Petitcolas (editors). Information hiding techniques for steganography and digital watermarking. Artech House, 2001.

[4] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Information hiding – a survey. *Proceedings of the IEEE*, 87(7):1062–1078, July 1999. Special issue on protection of multimedia content.

[5] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Power: Metrics for evaluating watermarking algorithms. In *Proceedings of IEEE ITCC 2002*. IEEE Computer Society Press, 2002.

[6] Radu Sion and Mikhail Atallah. Attacking digital watermarks. In *Proceedings of the Symposium on Electronic Imaging SPIE*, 2004.

[7] Radu Sion. wmdb.*: A suite for database watermarking (demo). In *Proceedings of the IEEE International Conference on Data Engineering ICDE*, 2004.

[8] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Relational data rights protection through watermarking. *IEEE Transactions on Knowledge and Data Engineering TKDE*, 16(6), June 2004.

[9] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *Proceedings of ACM SIGMOD*, 2003.

[10] Radu Sion. Proving ownership over categorical data. In *Proceedings of the IEEE International Conference on Data Engineering ICDE*, 2004.

[11] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Ownership proofs for categorical data. *IEEE Transactions on Knowledge and Data Engineering TKDE*, (to appear, subject to revisions), 2004.

[12] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Watermarking streams. In *Proceedings of VLDB*, 2004.

[13] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. On watermarking abstract semi-structures. In *Proceedings of the International Workshop on Digital Watermarking IWDW, Lecture Notes in Computer Science*, 2003.

[14] Joachim Eggers and Bernd Girod. Informed watermarking. In *Informed Watermarking*. Kluwer Academic Publishers, 2002.

[15] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *Lecture Notes in Computer Science*, 963:452–464, 1995.

[16] B. Chen and G. Wornell. An information–theoretic approach to the design of robust digital watermarking systems. In *Proceedings Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Phoenix, AZ*, 1999.

[17] P. Moulin and J. O'Sullivan. Information-theoretic analysis of information hiding, citeseer.nj.nec.com/article/moulin99informationtheoretic.html, 1999.

[18] Pierre Moulin, M. K. Mihcak, and Gen-Iu (Alan) Lin. An information–theoretic model for image watermarking and data hiding. Online at `citeseer.nj.nec.com/387798.html`, 2000.

[19] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Attacks on copyright marking systems. In David Aucsmith, editor, *Information Hiding: Second International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 218–238, Portland, 1998. Springer-Verlag.

[20] Bruce Schneier. Applied Cryptography: Protocols, Algorithms and Source Code in C. John Wiley and Sons, 1996.

[21] E.F. Codd. A Relational Model of Data for Large Shared Data Banks. In *Communications of the ACM*, volume 13, pages 377–387, 1970.

[22] Elisa Bertino, M. Braun, Silvana Castano, Elena Ferrari, and Marco Mesiti. Author-X: A Java-Based System for XML Data Protection. In *Proceedings of the IFIP Workshop on Database Security*, pages 15–26, 2000.

[23] Elisa Bertino, Sushil Jajodia, and Pierangela Samarati. A flexible authorization mechanism for relational data management systems. *ACM Transactions on Information Systems*, 17(2), 1999.

[24] Chris Clifton and Don Marks. Security and privacy implications of data mining. In *Workshop on Data Mining and Knowledge Discovery*, pages 15–19, Montreal, Canada, 1996. University of British Columbia Department of Computer Science.

[25] J. Hale, J. Threet, and S. Shenoi. A framework for high assurance security of distributed objects, 1997.

[26] E. Hildebrandt and G. Saake. User Authentication in Multidatabase Systems. In R. R. Wagner, editor, *Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, August 26–28, 1998, Vienna, Austria*, pages 281–286, Los Alamitos, CA, 1998. IEEE Computer Society Press.

[27] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy. Oakland, CA*, pages 31–42, 1997.

[28] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *SIGMOD*, 1997.

[29] Li, Feigenbaum, and Grosof. A logic-based knowledge representation for authorization with delegation. In *PCSFW: Proceedings of the 12th Computer Security Foundations Workshop*, 1999.

[30] M. Nyanchama and S. L. Osborn. Access rights administration in role-based security systems. In *Proceedings of the IFIP Workshop on Database Security*, pages 37–56, 1994.

[31] Sylvia L. Osborn. Database security integration using role-based access control. In *Proceedings of the IFIP Workshop on Database Security*, pages 245–258, 2000.

[32] David Rasikan, Sang H. Son, and Ravi Mukkamala. Supporting security requirements in multilevel real-time databases, citeseer.nj.nec.com/david95supporting.html, 1995.

[33] Ravi S. Sandhu. On five definitions of data integrity. In *Proceedings of the IFIP Workshop on Database Security*, pages 257–267, 1993.

[34] M. J. Atallah and S. S. Wagstaff, Jr. Watermarking with quadratic residues. In *Proceedings of IS-T/SPIE Conf. on Security and Watermarking of Multimedia Contents, SPIE Vol. 3657, pp. 283–288.*, 1999.

[35] J. Kiernan and R. Agrawal. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Databases VLDB*, 2002.

[36] D. Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. In *Proceedings Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2003.

[37] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. On watermarking numeric sets. In *Proceedings of IWDW 2002, Lecture Notes in Computer Science, CERIAS TR 2001-60*. Springer-Verlag, 2002.

[38] Sheldon Ross. A first course in probability. Prentice Hall, 2001.

[39] J. Eggers, J.Su, and B.Girod. Public key watermarking by eigenvectors of linear transforms. In *EUSIPCO*, Tampere, Finland, September 2000.

[40] J. Stern and J.-P. Tillich. Automatic detection of a watermarked document using a private key. In Ira S. Moskowitz, editor, *4th Int. Work. on Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, page electronic version, Pittsburgh, PA, April 2001. Springer.

[41] L. de C.T. Gomes, M. Mboup, M. Bonnet, and N. Moreau. Cyclostationnarity-based audio watermarking with private and public hidden data. In *A.E.S.*, September 2000.

[42] F. Hartung and B. Girod. Fast public-key watermarking of compressed video. In *Proceedings IEEE Int. Conf. on Image Processing*, October 1997.

[43] T. Furon and P. Duhamel. Audio asymmetric watermarking technique. In *Proceedings of Int. Conf. on Audio, Speech and Signal Processing*, Istanbul, Turkey, June 2000. IEEE.

[44] T. Furon and P. Duhamel. Robustness of an asymmetric technique. In *Proceedings of Int. Conf. on Image Processing*, Vancouver, Canada, September 2000. IEEE.

[45] T. Furon, I. Venturini, and P. Duhamel. Unified approach of asymmetric watermarking schemes. In P.W. Wong and E. Delp, editors, *Security and Watermarking of Multimedia Contents III*, San Jose, California, 2001. SPIE.

[46] B. Babcock, S. Babu, M. Datar, and Motwani R. Models and issues in data stream systems. In *Proceedings ACM Symp. on Principles of Database Systems (PODS)*, pages 1–16, 2002.

[47] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, N. Stonebraker, M.and Tatbul, and S. Zdonik. Monitoring streams – a new class of data management applications. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2002.

[48] S. Chandrasekaran and M. J. Franklin. Streaming queries over streaming data. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 203–214, 2002.

[49] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 635–644, 2002.

[50] J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *Proceedings of ICDE*, 2003.

[51] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse nearest neighbor aggregates over streams. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2002.

[52] B. Chen and G. W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4), 2001.

[53] D. Kirovski and H.S. Malvar. Spread-spectrum watermarking of audio signals. *IEEE Transactions on Signal Processing*, 51(4), 2003.

[54] M. D. Swanson, B. Zhu, and A. H. Tewfik. Audio watermarking and data embedding – current state of the art, challenges and future directions. In J. Dittmann, P. Wohlmacher, P. Horster, and R. Steinmetz, editors, *Multimedia and Security Workshop at ACM Multimedia*, volume 41 of *GMD*, Bristol, United Kingdom, September 1998. ACM.

[55] NASA. The hawaii university infrared telescope facility. Online at `http://irtfweb.ifa.hawaii.edu`, 2004.

[56] L. Babai and L. Kucera. Canonical labeling of graphs in linear average time. In *Proceedings 20th IEEE Symposium on Foundations of Computer Science, 39-46.*, 1979.

[57] L. Babai and E. Luks. Canonical labeling of graphs. In *Fifteenth Annual ACM Symposium on Theory of Computing, pages 171–183. ACM*, 1983.

[58] Faulon J. Automorphism partitioning and canonical labeling can be solved in polynomial time for molecular graphs. In *J. Chem. Inf. Comput. Sci. 38, 1998, 432-444.*, 1998.

[59] Ludek Kucera. Canonical labeling of regular graphs in linear average time. In *IEEE Symposium on Foundations of Computer Science*, pages 271–279, 1987.

[60] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM Conference on Computer and Communications Security*, pages 28–36, 1999.

[61] Stirmark. Online at `http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark`, 1998.

[62] Checkmark. Online at `http://watermarking.unige.ch/Checkmark/index.html`, 1998.

[63] Optimark. Online at `http://poseidon.csd.auth.gr/optimark`, 1998.

[64] Editor Eric Weisstein. Online world of mathematics. Online at `http://mathworld.wolfram.com`.

A APPENDIX

Appendix A: Excerpt from the Wal-Mart Sales Database Schema

```
(842,556,378 tuples)
Table Univ_Class_Tables.Item_Scan
  Visit_Nbr Integer Not Null
, Store_Nbr Smallint Not Null
, Item_Nbr Integer Not Null
, Item_Quantity Decimal(9,2) Not Null
, Total_Scan_Amount Decimal(9,2) Not Null
, transaction_Date Date Format 'YYYYMMDD' Not Null
, Unit_Cost_Amount Decimal(9,4) Not Null
, Unit_Retail_Amount Decimal(9,2) Not Null
, Tax_Collect_Code Char(1) Not Null Compress '1'
     Primary Index (Visit_Nbr);

(835,760,698 tuples)
Table Univ_Class_Tables.Visit_Scan_lookup
  Store_nbr           SMALLINT  NOT NULL,
, Transaction_date    DATE Format 'YYYYMMDD'NOT NULL,
, Item_Nbr          INTEGER   NOT NULL,
, Visit_nbr           INTEGER   NOT NULL
Primary Index (Store_Nbr, Transaction_Date, Item_Nbr);
```

VITA

Radu Sion was born sometime in 1975. He received his computer science B.Sc. (1998) and M.Sc. (1999) degrees from the "Politehnica" University of Bucharest. He remembers: punch-cards, the times when 600bps modems where hot, when 16KBytes of RAM was *huge*, when Minix fit on a few floppies, when Mosaic came out, when Java had no packages and came on tapes with lots of CRC errors.

In eighth grade Radu managed to stop most of the trains in Romania by playing a newly discovered exciting game called "shutdown". He played the game at the (only) green VT100 console of the main railway (PDP-11 based) control center where he was taken by his nice grown-up friend in a moment of weakness. His friend was fired. Radu had to wait almost 4 years to play that game again. The resulting adrenalin was arguably *the* determining factor in his going for computer science instead of particle physics in College. Thoughts of unifying quantum mechanics with a peculiar concept of Zen relativity still haunt him, mostly in the allergy season.

In the summer of 1999 Radu crossed the Atlantic. Once on the left side of it, he worked hard during most of the year and for fun and cash during the summers, including IBM in 2000 and NEC Research Labs in 2001 and 2003. Radu also interviewed repeatedly for Microsoft but managed never to work there. At the time of this writing, he spends most of his afternoons and some of his mornings at the IBM Almaden Research Center in San Jose, California, in the middle of a protected natural habitat, surrounded by amazingly smart people (mostly during lunch) and cows (rest of the day).

Radu is currently interested in inter-connected entities that access data and need to do so with assurances of security, privacy, and functionality, preferably efficient. His research lies at the intersection of security, databases and distributed systems. Applications include: authentication, rights protection and integrity proofs in sensor networks, secure storage in peer to peer and ad-hoc environments, data privacy and bounds on illicit inference over multiple data sources, security in computation/data grids, detection of intrusions by access profiling for on-line web portals.

Radu has been an invited speaker and guest lecturer at several institutions. He has been reviewing for numerous journals and conferences including ACM TODS, IEEE TKDE and IEEE S&P. In 2004, Radu received the CERIAS Diamond Award for his Ph.D. dissertation work. In the Spring of 2005, Radu is joining the Computer Sciences Department of SUNY at Stony Brook, with the declared goal of both teaching youngsters about computing and fathering many good academic children.