uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Agnes Mucsi-Nagy

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.

GRADE / DEGREE

The School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Digital Fingerprinting for Sharing of Confidential Data

TITRE DE LA THÈSE / TITLE OF THESIS

Stan Matwin

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Jiying Zhao

P. Van Oorschot

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

DIGITAL FINGERPRINTING FOR SHARING OF CONFIDENTIAL DATA

Agnes Mucsi-Nagy

Thesis
submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of Master of Computer Science

2006

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-18448-6
Our file Notre référence
ISBN: 978-0-494-18448-6

Abstract

Currently, there are three approaches to limit disclosure of databases containing confidential data: 1) altering data before disclosure; 2) learning results without revealing data; and 3) watermarking to prove ownership over a database. This thesis describes a symmetric fingerprinting scheme that can be considered as a fourth approach or a complement to the three existing approaches.

The proposed scheme uses fingerprinting to identify the entity who received confidential data and consists of a fingerprint (mark uniquely identifying each recipient), encoder (fingerprint insertion algorithm), decoder (fingerprint extraction algorithm) and a detection algorithm (algorithm to detect fingerprint in data). The scheme distinguishes between trusted and distrusted recipients of data. Trusted recipients have a decoder to fully or partially restore data whereas distrusted recipients don't have a decoder.

Scheme requirements including algorithm performance, and attacks including collusion attacks, are analyzed. Proof of concept results are also provided.

Acknowledgements

My sincere thanks to my supervisor Dr. Stan Matwin, colleagues Svetlana Kiritchenko and William Elazmeh, and my parents, husband and four children. Thank you to all for your understanding and tremendous support.

Table of Contents

Table of Figures and Tables

1. Introduction

Privacy, one of the founding principles of democracy, has been around for a long time but has recently gained attention with the evolvement of the World Wide Web. Data can be rapidly circulated around, to both legitimate and illegitimate recipients, and it can suddenly become a value to the even most naive user.

Technology has helped both proponents and opponents of data disclosure, and definitely, data disclosure has both positive and negative effects. Every user appreciates fast, personalized service and to receive it, will unlikely object to data disclosure. On the other hand, users will be extremely dissatisfied if the same data is sold to a third party who will use the data for purposes other than those for which the data has been collected.

When data gets into the hands of unauthorized recipients, we are dealing with unauthorized data disclosure. Techniques introduced so far for limiting unauthorized data disclosure can be categorized as data altering techniques, techniques to learn information without revealing the data, and techniques to watermark data. Chapter 3 (Related Work) discusses these techniques in more detail.

This thesis explores the possibility of fingerprinting data prior to disclosure and this technique can be used on its own or as a complement to the existing techniques (contributions of this thesis are discussed in the next section). In fact, the proposed fingerprinting scheme is a refinement of the "adding tuples"[1] data altering technique as added tuples have the purpose of not only obscuring data but hiding information about the owner and recipient of the data in the form of a fingerprint.

Fingerprinting means that each distributed copy of the database has a unique hidden mark that identifies entities that receive and use the copies. It is as if by using the database, the user was leaving a "fingerprint" – mark uniquely associated with that particular recipient.

[1] In relational database context, the following notations are used in this work:
table = database
table row = record, tuple
table column = attribute, field

The importance of unauthorized disclosure prevention lies in the fact that control over a database is completely lost once the database has been disclosed (i.e. given to other users without authorization). However, if the database contains the unique fingerprint of a recipient, this recipient will be reluctant to illegally disclose the database since the embedded fingerprint could identify him. Fingerprinted data can still be used for privacy preserving data mining.

Fingerprinting schemes, like watermarking schemes, consist of three components:

1. Fingerprint;
2. Encoder (insertion algorithm); and
3. Decoder and comparator (verification, extraction and/or detection algorithm).

1.1. Contributions of this Thesis

1.1.1. Background

As explained in section 1 (Introduction), unauthorized disclosure happens when data is disclosed (divulged) to parties not allowed to see or read it (i.e. illegitimate recipients), who can use the data for purposes not necessarily consistent with original purposes (i.e. purposes data has originally been collected for). In other words, unauthorized disclosure happens when data is shared with illegitimate recipients intentionally or unintentionally.

Common to all techniques developed to date to limit unauthorized disclosure of data (data altering techniques, techniques to learn information without revealing the data, and techniques to watermark data) is the fact that they are not able to determine from the data who disclosed it. Tracing traitors encryption schemes (a data altering technique) can determine the entity who disclosed the data by identifying the source of the encryption key used by an unauthorized user but if decrypted data is found without the key, it is impossible to determine from this data who provided the key.

Data altering techniques modify data e.g. if A is the data owner, A would modify its data before disclosing it to B, C and D so that B, C and D cannot learn the original data from it. Note that modified data may still be useful for the purposes of B, C and D.

Learning techniques attempt to obtain desired data without disclosing it e.g. A would not disclose its full data to B, C and D, and B, C and D would learn new data from working with portions of data.

Watermarking techniques enable determining ownership from disclosed data e.g. if A is the owner of the data and if data somehow finds its way to B, C and D then watermarking techniques could identify from B's, C's or D's data that the real owner is A and not B, C or D. Also, if A legally disclosed its watermarked data to B, C and/or D and one of these three entities illegally disclose the same data to E, the watermark would identify A as the owner of the data and not the entity who disclosed it (B, C or D). This also means that if A wanted to disclose its data illegally, A would never want to insert its watermark in the data because he would never want to be identified. In other words, watermarking in this sense can only be used to prove ownership.

1.1.2. Contributions

The contributions of this thesis are threefold.

Firstly, this thesis proposes a fingerprinting scheme that can identify from the data itself the entity who illegally disclosed data. To use the above example, assume that A is the owner of the data and B, C and D are legitimate users of data. If B, C and/or D disclose data without authorization, the proposed fingerprinting technique could identify from the data the entity (B, C or D) who disclosed it as opposed to A, the real owner of the data. In the proposed scheme, A is the authority assigning fingerprints to users B, C and D, who don't know their fingerprints.

This is an important proposal from a privacy perspective. For example, if a person provides his personal information to A (let's say to his doctor) for a particular purpose (let's say to be used by researchers B, C and D) and this information is later used by E (let's say the person's car insurer) for a different purpose (let's say to increase the person's insurance fees because of a medical condition), the person will not trust A any more. However, if A had implemented this privacy proposal, A could have requested E's

data to be searched to find who (B, C and/or D) gave data to E. If the entity is identified, A will not trust him any more and could assure the person that this entity would never again receive A's data (sections 2.1 and 2.2 of this thesis discuss privacy, personal information and trust in more detail). More importantly, tracing traitors algorithms would not work in this situation since personal information is not accessible publicly and E would have to get data directly from B, C or D. If B, C or D would be willing to give data to E (perhaps for a monetary reward), they could give the decrypted data to E without revealing their keys so that they could not get caught. Also, if E were to steal data from B, C or D, he could do it either by stealing data only or stealing encrypted data together with the key, which of course, would require more effort.

Private and confidential data (section 2.1 provides explanation of private and confidential data) usually consist of categorical values[2]. Except for my paper ([38]), there have been no results published so far (see Related Work, chapter 3, of this thesis), in the area of embedding fingerprints in categorical values of a database such that existing data is not modified.

The second contribution of this thesis is addressing this gap. In the proposed fingerprinting scheme, fingerprints are inserted into the categorical values of new, fictitious records and these records are added to the existing database. Therefore, existing personal information (information about identifiable individuals) is not modified by the insertion of the fingerprint, thus satisfying the sixth privacy principle (Accuracy of Personal Information). Section 2.1 lists all ten privacy principles.

Thirdly, this thesis brings a novel contribution to the area of privacy in Canada because it proposes a partial solution to a privacy problem by using technical mechanisms and taking into account privacy legislation and practices in Canada, defines differences between a privacy and a security problem, between a privacy and a security proposal, and provides a list of existing privacy and security tools that can be used to conduct privacy and security analyses (sections 2.1 and 2.2 of this thesis).

[2] Categorical values = data consisting of character strings

1.1.3. Limitations

This fingerprinting scheme is symmetric as only the distributor of the database participates in the marking procedure, and both the distributor and the recipient have the same copy of the data. The distributor determines all fingerprints, unique for each recipient, and inserts the appropriate fingerprint in the database prior to its disclosure. Using the above example, A is the distributor (the owner of the data) and B, C and D are the recipients. Symmetric fingerprinting schemes have certain disadvantages (discussed in section 2.4 of the next chapter) and it would be interesting to see whether other types of fingerprinting schemes can be applied to limiting database disclosure.

The biggest challenge in fingerprinting is the collusion attack. In collusion attacks, recipients of the fingerprinted database compare their copies in order to detect and remove the fingerprint. Using the previous example, if distributor A inserts B's fingerprint in B's data, C's fingerprint in C's data and D's fingerprint in D's data, recipients B, C and D can compare their data to detect and remove their fingerprint.

It is important to note that Boneh and Shaw ([6]) provide a general fingerprinting solution (that can also be applied to this proposal) which is resistant to collusion attacks. In this thesis, the Boneh Shaw scheme is used only to determine the lower bound on the number of fingerprints to be inserted in order to defend against collusion attacks and is not an integral part of the proposed approach.

The Boneh Shaw formula determining the lower bound of the number of fingerprints to be inserted, however, is impractical in the case of a large number of recipients (for example, 50 recipients) due to the fact that the length of such a fingerprint (or the number of fictitious records to be inserted) is larger than the total number of characters in a database record (or the total number of records in a database).

On the other hand, if the fingerprinting scheme has nine recipients, it is sufficient to insert, as a lower bound, 6,261fictitious records (which is 6.26% of a very large database containing 100,000 records) to make the scheme resistant to collusion attacks. In this

case, the length of the fingerprints can be any feasible number. (Sections 4.5.6 and 4.6.6 provide more discussion on this.)

This attack, along with other possible attacks on the proposed fingerprinted scheme is discussed later in more detail.

Capacity, robustness, detectability and defence against horizontally partitioning attacks are interrelated and measures applied to achieve any of them could negatively influence the others. Besides collusion attacks, this interrelation is the most difficult problem of fingerprinting schemes, and it is also discussed later in more detail.

The idea of adding fictitious records to the database immediately triggers two important questions. Firstly, what is the number of fictitious records to be added in order to achieve accurate fingerprint detection and at the same time, to limit data alteration as much as possible? Secondly, is it possible to determine an optimum number of fingerprinted records to be inserted in the data such that the capacity, robustness and detectability properties are satisfied and at the same time, defence against horizontally partitioning attacks is achieved?

Since by embedding a fingerprint existing data is not modified, it is possible that the original data already contains the same character sequence as the fingerprint. Moreover, the original data can contain character sequences that constitute the fingerprint of other recipients. Proof of concept results show that the most important factor in detecting a fingerprint in a database is that a particular fingerprint appears more frequently in the data than other (not inserted but originally existing) fingerprints. This observation has led to the requirement that a database must be analyzed before the proposed fingerprinting scheme is applied, to determine the domain set from which the fingerprint characters are to be chosen. This problem and a proposed solution are discussed in sections 4.5.1 and 4.6.1.

Also, proof of concept results suggest that the more fingerprinted records the original database contains (i.e. the original database already contains different records with the fingerprints of recipients – see section 4.5.1 for these assumptions), the more robust the scheme is, detectability and capacity are higher and successful defence against horizontal partitioning attacks (see sections 3.1 and 4.4 for the description of horizontal partition) is more probable.

The answer to the above two questions is the same: the original database should include records containing the fingerprints of all recipients, in approximately same number of records for each recipient. These records should not be exactly the same records for any two recipients. The actual number of these records is not particularly important but it has to be emphasized that when some of these records are modified and inserted back in the database as fictitious records, the total number of fictitious records should not reduce the quality of resulting data. (section 6.1).

Since there is no requirement for a specific number of fictitious records, the number resulting from applying the Boneh Shaw formula can be used. For a large database of 100,000 records and for nine recipients, the lower bound of the number of fictitious records is therefore 6,261 (6.26%).

To properly position the contribution of this thesis with respect to practical feasibility, running time (process and database access) of the insertion, extraction and detection algorithms is estimated for nine recipients and 100,000 records to be between 20 (insertion algorithm, local access) and 495 seconds (detection algorithm, remote access). For more detail on these estimates see sections 4.5.5 and 4.6.5.

Detection may take up to 495 seconds (8 minutes 15 seconds) but it must be emphasized that detection is only performed if a request comes in to determine who divulged data.

Although running time is not negligible for any of these algorithms, it is not at all impractical. If an organization considers unauthorized data disclosure protection as an

important asset from a privacy or legal perspective, or simply, to gain trust of its customers, the benefits of implementing such a fingerprinting scheme would clearly outweigh the costs associated with increased running time.

1.2. Organization of this Thesis

In the next two chapters confidentiality, watermarking and fingerprinting are explored in more detail and related work is discussed. Chapter four introduces the proposed fingerprinting scheme and chapter five provides the proof of concept results. Chapters six, seven and eight discuss the scheme and results, identify areas for future research and close with conclusion.

2. Background

2.1. Confidential Data

Confidentiality has been defined by the International Standards Organization (ISO) as "ensuring that information is accessible only to those authorized to have access" and is one of the elements of information security. Other elements of information security are data integrity, authorization and authentication, non-repudiation, accuracy and availability.[3]

For the purposes of this thesis, confidential data includes any information that should be accessible only to those authorized to have access to it. Confidential data can range from secret, top-secret through financial to private information, but while protecting secret or financial information has always been essential, protecting private information has only recently gained attention with the introduction of the Internet.

To better understand the need for including private information under confidential information, this section attempts to clarify differences and similarities between privacy and security.

According to Webster's Dictionary, privacy is a "matter that is or should be removed from public view". However, in today's world, especially with the development of new technology, there are more privacy invasions than ever. Private information is used for market research, data mining, spamming, hacking or, more recently, identity theft and

---

3

Authentication – The assurance to an entity that another entity is who they claim to be.

Authorization – Concerned with what an identity is allowed to see and do.

Data integrity – The assurance to an entity that data has not been altered (intentionally or unintentionally) between "there" and "here", or between "then" and "now".

Confidentiality – The assurance to an entity that no one can read a particular piece of data except the receiver(s) explicitly intended.

Non-repudiation of origin – A user cannot falsely deny having originated a message or document.
Non-repudiation of receipt – A user cannot falsely deny having received a message or document.

Accuracy and availability – The assurance that data is accurate and available. (See reference[2])

phishing. As Canada's former Privacy Commissioner, Bruce Phillips, stated: "There is still more money to be made by invading privacy than by protecting it".

Figure 1 ([50]) demonstrates the difference between information security and information privacy. It can be observed that security and privacy do share some common elements. For example, for privacy protection it is necessary, although not sufficient, to ensure data accuracy and security safeguards. Also, openness in privacy can be considered as the equivalent of availability in security. These terms are explained in more detail later in this section.

Ensuring information security is not the same as privacy protection. Information security is explained in [50] as controls deployed by the organizations or their surrogates for the purposes of securely collecting and holding data. It applies to personal and non-personal data alike. On the contrary, privacy protection applies only to personally identifiable information and how the rights of an identifiable data subject - the person providing the information - are affected and enforced.

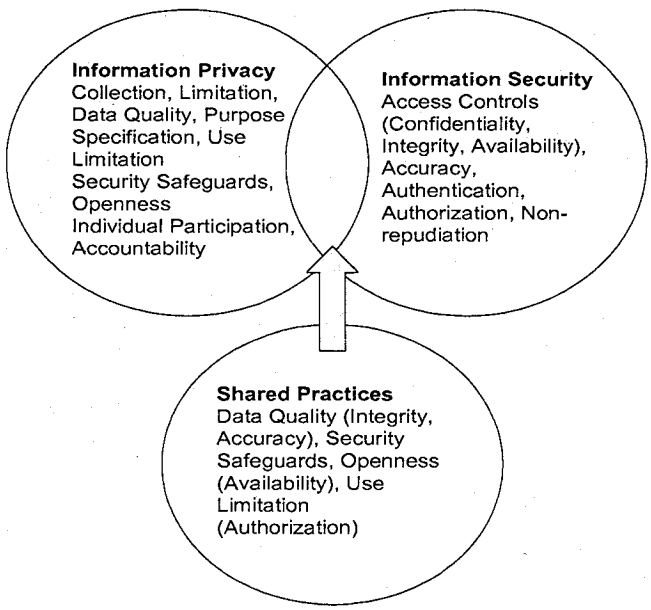

FIGURE 1: DIFFERENCES AND SIMILARITIES BETWEEN INFORMATION SECURITY AND PRIVACY (ONTARIO'S PRIVACY COMMISSIONER [50])

Although information security may provide adequate protection of secret or financial information, current practices of privacy protection, even if combined with information security, overlook the possibility of data matching and re-identification experiments that can restore previously removed identifiers and thus, re-identify individuals. Identifiers are defined as attributes (such as the social insurance number) that uniquely identify an individual while quasi-identifiers (such as birth date or postal code) identify an individual or a group of individuals but in combination uniquely identify an individual. For example, if two people have the same birth date but a different postal code, a combination of the birth date and postal code will uniquely identify both individuals.

Latanya Sweeney in her Ph.D. thesis [49] provides an example that illustrates the dangers of not respecting the difference between information security and privacy protection:

> *"As a state employee, William Weld, the former governor of Massachusetts, had his medical records included in a health insurance database of approximately 135,000 other records. Since the database was considered anonymous* (after removing the identifiers from the database), *its copy was sold to industry. Governor Weld lives in Cambridge, Massachusetts. By matching the database with the Cambridge Voter List, one could find only six people with the governor's birth date. From these 6 people, 3 were men and the governor was the only one in his 5-digit ZIP code".* (Sweeney [49])

The health insurance database in this example might have been safeguarded by controls in place to collect and hold data securely and by removing identifiers from the database prior to selling it to industry (information security). However, data thought to be anonymous was re-identified, demonstrating that data privacy was not adequately protected.

An argument can be made that privacy protection without the backing of privacy legislation is ineffective. This dilemma exists with all governments around the world, and as we study privacy law in various countries, we'll find completely different approaches

to privacy protection. In the U.S.A., for example, several privacy laws exist but the overall approach is mostly self-regulatory. On the other hand, the European Union has adopted aggressive privacy law. Canada's approach lies somewhere in between the two approaches described above with the Privacy Act for the public sector, the Personal Information Protection and Electronic Documents Act (PIPEDA [39]) for the private sector and several other legislation and Treasury Board policies such as the Access to Information Act ([1]), Privacy and Data Protection Policy ([43]) and the Privacy Impact Assessment (PIA) Policy ([44]).

In general, however, privacy legislation is only a deterrent, as it acts when a privacy breach has been committed. Legislation needs to be complemented by specific and practical tools that can prevent privacy breaches from happening.

Both the Privacy Act and PIPEDA define personal information as information about an identifiable individual that is recorded in any form. According to the Privacy Act ([42]), Personal information includes:

(a) information relating to the race, national or ethnic origin, colour, religion, age or marital status of the individual,
(b) information relating to the education or the medical, criminal or employment history of the individual or information relating to financial transactions in which the individual has been involved,
(c) any identifying number, symbol or other particular assigned to the individual,
(d) the address, fingerprints or blood type of the individual,
(e) the personal opinions or views of the individual except where they are about another individual or about a proposal for a grant, an award or a prize to be made to another individual by a government institution or a part of a government institution specified in the regulations,
(f) correspondence sent to a government institution by the individual that is implicitly or explicitly of a private or confidential nature, and replies to such correspondence that would reveal the contents of the original correspondence,
(g) the views or opinions of another individual about the individual,

(h) the views or opinions of another individual about a proposal for a grant, an award or a prize to be made to the individual by an institution or a part of an institution referred to in paragraph (e), but excluding the name of the other individual where it appears with the views or opinions of the other individual, and

(i) the name of the individual where it appears with other personal information relating to the individual or where the disclosure of the name itself would reveal information about the individual.

The PIPEDA and the PIA policy are based on the following ten privacy principles developed by the Canadian Standards Association ([9]):

1. Accountability

   This principle means complying with the 10 privacy principles, appointing an individual who is responsible for the compliance, protecting personal information, and developing privacy policies and practices.

2. Identifying Purposes

   Reasons for collecting personal information must be identified before or at the time of personal information collection.

3. Consent

   This principle means informing individuals of the purposes of collecting, using and disclosing their personal information and obtaining their consent before or at the time of collection.

4. Limiting Collection

   This principle means not collecting personal information indiscriminately and not deceiving and misleading individuals about the reasons for collecting personal information.

5. Limiting Use, Disclosure, and Retention

Personal information should be used and disclosed only for the purposes it was collected for and it should be retained only as long as it is necessary to satisfy the purposes.

6. Accuracy

   Personal information should be correct when making a decision about the individual.

7. Safeguards

   Personal information must be protected against loss, theft, unauthorized access, disclosure, copying, use or modification.

8. Openness

   Policies and practices for the management of personal information must be understandable and easily available.

9. Individual Access

   Individuals should have access to their personal information hold about them.

10. Challenging Compliance

    Simple and easily accessible complaint procedures must be in place.

Beside legal obligation, there is another important factor that could motivate organizations to apply privacy protection and that is gaining and keeping trust of their customers. Trust is defined by the Webster's Dictionary as a "confident reliance on the integrity, veracity, or justice of another; confidence, faith". Returning customers, whether it is an online or traditional brick and mortar business, represent a high percentage of the client base. These customers can easily turn to competition if they feel that their privacy has been invaded.

2.2. Privacy Problem

To position this work, several terms need to be clarified. As explained in the previous section, privacy and security do share some common elements such as safeguards, accuracy (integrity) or openness (availability) but they essentially differ as demonstrated by the example from [49].

A privacy problem is a problem involving privacy i.e. a problem that can be related to any one of the ten privacy principles such as the collection, use, disclosure or retention of personal information. On the other hand, a security problem is a problem involving security such as authorization, authentication or non-repudiation.

Therefore, a privacy proposal is a proposal put forward to fully/partially solve a privacy problem and a security proposal is a proposal put forward to fully/partially solve a security problem.

Privacy requirements are requirements defined by the ten privacy principles and security requirements are requirements to ensure the security of people, assets (including application security, data security, network security and software security) and national interests. Consequently, privacy attacks are attacks that attack privacy, as for example re-identification experiments, data mining in some instances, using personal information without the individual's consent or for purposes not identified at the time of data collection. Similarly, security attacks are attacks that attack the security of people, assets or a nation.

Privacy analysis is conducted to determine if a proposal (does not have to be a privacy proposal) meets privacy requirements. On the other hand, a security analysis is conducted to determine if a proposal (does not have to be a security proposal) meets security requirements. For example, if a proposal is to link two systems to facilitate easier, faster and more convenient data transfer, this proposal can be (and should be) analyzed both from a privacy and security perspective. If data matching (re-identification) will be made possible by the linkage of these two systems, a privacy analysis should determine what

needs to be done to meet privacy requirements. On the other hand, a security analysis should determine what needs to be done to meet security requirements. To assist privacy analysis, tools such as a Preliminary Privacy Impact Analysis (PPIA) or Privacy Impact Analysis (PIA) should be used. To assist security analysis, tools such as Statement of Sensitivity (SoS), Threat and Risk Assessment (TRA), Business Continuity Plan (BCP) should be used.

The privacy problem described in this work is projected on a generally accepted process of data flow in organizations. Within this context, data flow has three stages:

1) data collection,
2) use, and
3) disclosure/retention.

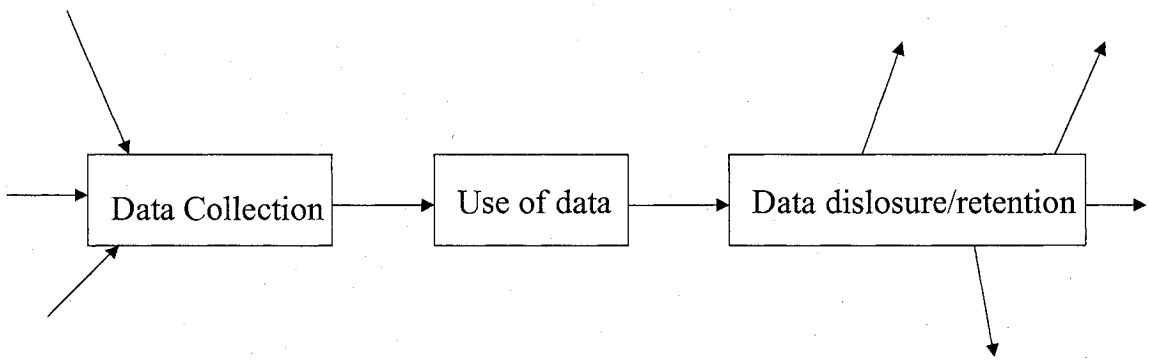

FIGURE 2: DATA FLOW

Consequently, privacy protection should be applied at three levels:

1. Personal information collection
   Privacy protection at this level should involve acquiring consent to collect personal information. This can be in a form of providing a signature or clicking on the Agree or Submit button.

2. Use of personal information.

Privacy protection at this level should involve adhering to privacy policies. In other words, owners of personal information should only use collected personal information for the purposes stated in their privacy policy.

TRUSTe ([51]), Platform for Privacy Preferences (P3P [40]), and IBM's Enterprise Privacy Authorization Language (EPAL [23]) have emerged as possible technical solutions to privacy protection at this level. The first two encourage organizations to display their privacy protection practices on their websites, while EPAL is a formal specification, which can be used to develop privacy policies for an enterprise.

Another attempt to privacy protection at this level has been proposed by S. Matwin and A. Felty [24]. They develop formal methods to produce a system in which the data holder proves that constraints on data use are respected.

3. Disclosure of personal information.
   If disclosure of personal information is identified as a purpose at the time of collection, disclosure is legal. However, problems may arise if collected personal information is disclosed without consent, without being identified as a purpose at the time of collection. Disclosure in this case can still happen legally as sometimes personal information needs to be disclosed because of unforeseen reasons such as to protect national interests (to prevent terrorist attacks, for example) or for statistical or research purposes.

   The third level is considered to be the most vulnerable level as once personal information is disclosed, the owner loses control over it.

   This thesis focuses exclusively on the third level, on a privacy problem that arises when personal information is disclosed. Nevertheless, to make it more general, this privacy problem is expanded beyond the disclosure of personal information only, and it includes (authorized and unauthorized) disclosure of confidential data in general.

The privacy proposal suggested in this thesis attempts to provide a partial solution to the privacy problem described above. It is a partial solution only as the proposed fingerprinting scheme may prevent some recipients to disclose data illegally but will not eliminate the problem entirely as there will always be people or organizations who will consider benefits (perhaps in a monetary form) gained from disclosing data illegally more substantial than detriments resulting from losing trust. If suggested for implementation, this proposal as any other proposal, should undergo a detailed privacy and security analysis using a PPIA, PIA, SoS, TRA and BCP.

2.3. Watermarking

Watermarking is an information hiding technique. Information hiding techniques are secret communication methods that conceal the message's very existence. Concealing the message's very existence does not mean that the message must be invisible. It only means that the message's existence is not known, and/or the message's location is not known and/or that the message appears visible only upon careful inspection.

Information hiding techniques have been around since the ancient times. Romans used to shave the hair off a slave's head, write a secret message on it and wait until the hair grew out again. Then, they sent the slave to their allies with the message hidden under the hair. Of course, the message on the slave's head was not invisible. However, if an enemy captured the slave, they would not know that the slave was carrying a hidden message. Even if they suspected that there was a hidden message, they would not know the location of the message and would be able to find the message only if they thoroughly examined the slave.

Cryptography is not an information hiding technique as it scrambles a message so it cannot be understood but the scrambled (unhidden) message can attract attention.

Figure 3, from [36], represents an excellent overview of information hiding techniques.

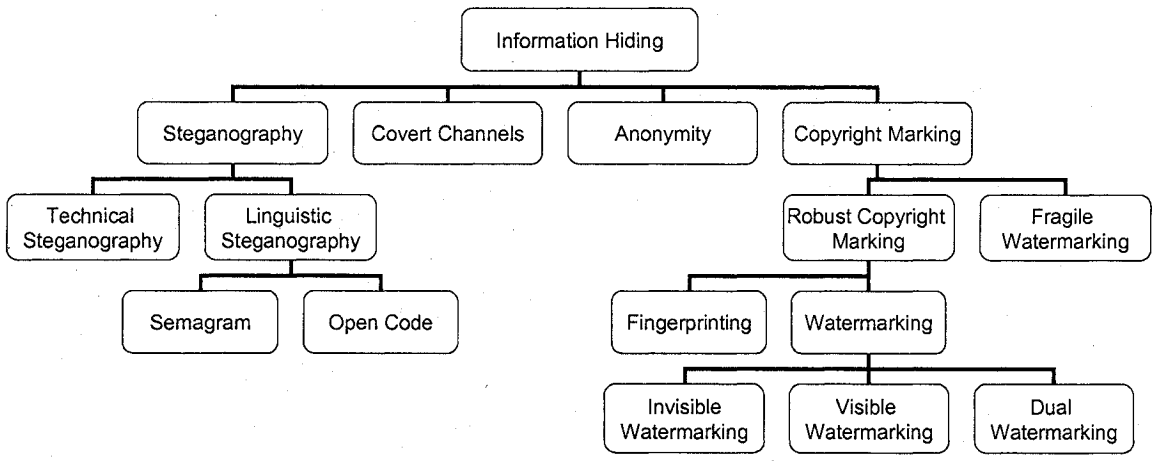

FIGURE 3: INFORMATION HIDING TECHNIQUES (MOHANTY [36])

Watermarking can identify the source, author, creator, owner, distributor or authorized recipient of the documents. A digital watermark is an identification code, permanently embedded into digital data, carrying information pertaining to copyright protection and data authentication.

Invisible watermarks cannot be easily recovered without an appropriate decoding mechanism. Visible watermarks appear visible on a careful inspection. Dual watermarks are a combination of a visible and invisible watermark. Fragile watermarks are used for authentication purposes as they are capable of detecting frequent changes of the watermarked content.

In general, any watermarking scheme (algorithm) consists of three parts:

1. The watermark.
2. The encoder (insertion algorithm).
3. The decoder and comparator (verification, extraction or detection algorithm).

Watermarking could be used for identifying the owner or distributor of confidential data in the case of illegal disclosure of confidential data. This would require the watermark to be embedded in each distributed copy of the database containing confidential data.

Traditional digital watermarking techniques of images ([7],[17],[26],[27],[36],[41],[45]) involve inserting the watermark by modifying the pixels of an image. Such modifications are not visible for the human eye. However, these techniques cannot be applied in the case of confidential data protection, unless confidential data involve a photograph or biometrics of an individual, as confidential data are mostly textual data; therefore, another technique is required.

Text watermarking algorithms ([8],[27]), on the other hand, are based on modifying text formatting properties, which again, cannot be used for watermarking databases containing confidential data.

Confidential data protection would also require watermarking of quasi-identifiers (that identify a group of people instead of an individual but in combination can uniquely identify an individual) since it must be ensured that even after removing identifiers (that uniquely identify an individual) from a database prior to its disclosure, the watermark remains in the database to identify the owner. That means, however, that while the owner is identifiable, entities that receive the copies are not. This can lead to unauthorized distribution of the database without the ability to identify the unauthorized recipients. Again, finding a more suitable technique is recommended.

2.4. Fingerprinting

In fingerprinting schemes, each distributed copy of the database has a unique, hidden mark that identifies not only the owner of the database but also the entity who receives that particular copy. This justifies the name of the technique – it is as if the recipient has "touched" the data after gaining access to it.

Fingerprinting schemes, like watermarking schemes, consist of three parts:

1. The fingerprint.
2. The encoder (insertion algorithm).
3. The decoder and comparator (verification, extraction or detection algorithm).

Fingerprinting can be symmetric, asymmetric and anonymous [25]. With symmetric fingerprinting, only the distributor participates in the marking procedure (both the distributor and the recipient have the same fingerprinted copy). A disadvantage of this scheme is that recipient A can be falsely accused of illegal distribution if the distributor provides another recipient B with the copy bearing A's mark.

In asymmetric fingerprinting, both the distributor and the recipient take part in the marking procedure (the distributor does not see the copy the recipient receives because the recipient inputs its own secret in the copy received from the distributor). However, this scheme lacks anonymity: the distributor knows the recipient's identity.

In anonymous fingerprinting, the system is based on having a trusted third party (central authority) who is the only entity knowing the recipient's real identity. Confidential data protection in this case involves a circle of database sharers, and a trusted central authority who assigns unique fingerprints to each recipient of the copy of the database. There is no distributor and circle members share the data in their possession by inserting their fingerprints in the data prior to disclosing it. Since the same data is circulated among circle members, the mark each circle member inserts must differ from marks of other circle members; hence this is a fingerprinting and not a watermarking scheme. The fingerprint is embedded in the quasi-identifiers of the database, again, to prevent the

destroying of the fingerprint by removing the identifiers from the database. This unique fingerprint then identifies the member in the case of illegal reselling (disclosure) of the database. The disadvantage of this approach could be the possibility of a malicious circle member discovering and using another circle member's fingerprint.

This thesis is based on the symmetric fingerprinting scheme (but in certain cases can be extended to satisfy asymmetric fingerprinting requirements as discussed in section 7.4 of chapter 7) as it involves database recipients and a database distributor who knows the identities and fingerprints of database recipients. Recipients do not know their fingerprints, only the distributor. If the distributor is a trusted party (central authority), there should be no malicious activities or false accusations with respect to the distributor.

The biggest challenge in fingerprinting is the collusion attack. In collusion attacks, recipients of the fingerprinted database compare their copies in order to detect and remove the fingerprint. This attack, along with other possible attacks on the proposed fingerprinted scheme are discussed later in more detail.

3. Related Work

3.1. Protection Against Unauthorized Disclosure

So far, there have been three approaches to limit disclosure of databases containing confidential data: 1) altering the data before disclosure, 2) learning results without revealing the data, and 3) watermarking databases.

The approach of altering data before disclosure has been around for a long time and it includes statistical disclosure control. Like other data holders, statistics offices use different methods to add noise to data when sharing it in order to protect its confidentiality. They have also recognized that drawing inference, arriving to new facts on the basis of other available information, can lead to disclosure of more data than originally intended. Drawing inference can occur in many ways, for example by deduction (logically drawing conclusions from facts), abduction (generating explanations from observations and relationships), induction (learning), link analysis, data matching, by performing multiple queries to a database ([19],[20],[22]), deriving more sensitive information from less sensitive information in multi-level databases ([48],[37],[28],[33]) etc. In fact, as Su and Ozsoyoglu show in [48], eliminating precise inference compromise due to functional dependencies and multi-valued dependencies is NP-complete. This makes sense because we can think of inference as an attack on data, and we know that knowing all attacks (on any matter in general) a priori is impossible as new attacks are devised regularly (see [35] for an example on software attacks).

Latanya Sweeney [49] summarized techniques available for data altering (techniques to obscure data):

- Anonymization - not releasing identifiers and quasi-identifiers that could identify an individual (name, SIN, ...)
- De-identification - not releasing identifiers (SIN, ...)
- Suppression - withholding values
- Encryption - values can be read only by privacy key holder
- Swapping - exchanging values associated with an attribute in two tuples
- Generalization - replacing a value with a more general, less specific alternative

- Substitution - replacing a value with another value in its equivalence class
- Outlier to medians - replacing unusual values with more common values
- Perturbation - making changes to values to maintain overall aggregate statistics
- Rounding - grouping values into ranges
- Adding noise - random incrementing or decrementing of values
- Sampling - restricting the number of tuples that will be released
- Adding tuples - dilutes the number of tuples containing real information
- Scrambling tuples - reordering of tuples
- Query restriction - allowing only certain queries, for example queries that will always result in more than one tuple
- Summaries - instead of exact data, summaries of data are provided

An example of generalization (replacing a value with a more general, less specific alternative) is shown in Figure 4 where values of Work ZIP are replaced by more general values (for example, 02138 with 021). A star indicates a digit of an unknown value.

	A1	A2	A3
	Home ZIP	Hospital ZIP	Work ZIP
t1	02138	02138	02138
t2	02138	02139	02138
t3	02138	02138	02141
t4	02138	02139	02139

	A1	A2	A3
	Home ZIP	Hospital ZIP	Work ZIP
t1	02138	02138	021**
t2	02138	02139	0213*
t3	02138	02138	021**
t4	02138	02139	0213*

FIGURE 4: EXAMPLE OF GENERALIZATION (SWEENEY[49])

Encryption schemes that enable tracing traitors have extensively been studied since 1994. These schemes were defined by [11] as cryptographic schemes that help trace the source of leaks when sensitive or proprietary data is made available to a large set of parties who are not supposed to access the data. They call this unauthorized access data piracy, non-authorized parties pirate users and authorized users who enable unauthorized access traitors. These schemes can be applied in the case of publicly accessible databases where keys to decrypt and access all or certain records can be obtained by paying a fee. If an

authorized user's key is discovered by a pirate user or if an authorized user makes its key available to a pirate user, we are talking about unauthorized disclosure since an unauthorized user can get access to the database by using the key.

Query restriction and Summaries are not really disclosure limitation techniques but rather special circumstances in which disclosure control is required - values are often suppressed so as not to reveal confidential data.

All these techniques have the advantage that a recipient of the data can be told what was done to the data in terms of protection.

Sweeney also observed that the less anonymization is applied, the more useful the data is. However, individuals are more identifiable, and thus, privacy is less respected. As we move towards more anonymization, individuals are less identifiable, privacy is more respected, but data is becoming less useful. She recommends an optimal solution, a tradeoff between privacy and quality, where quality is preserved but privacy is still protected. Figure 5 privacy versus data quality ([49]) demonstrates this observation.

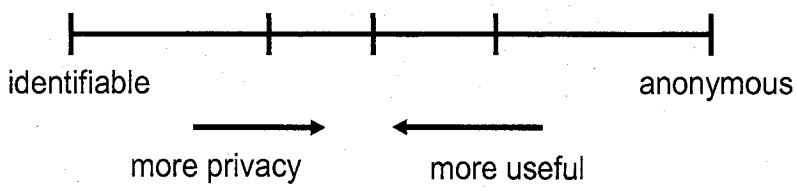

FIGURE 5: PRIVACY VERSUS DATA QUALITY (SWEENEY [49])

Sweeney introduces two algorithms for disclosure limitation. Her first algorithm, the Datafly, uses generalization and suppression, but tends to over-distort data. Her second algorithm, the k-similar algorithm[4] finds the optimal solution but has a combinatorial running time.

[4] While in the k-means clustering algorithm data are partitioned into k groups based on minimizing a distance between tuples, the k-similar algorithm partitions data into groups of size k or more, where a group will consist of k or more "nearest neighbour" tuples.

Perturbation is another way of altering data before disclosure. Rakesh Agrawal and Ramakrishnan Srikant [5], and Dakshi Agrawal and Charu C. Aggarwal [3] develop reconstruction procedures on perturbed data to accurately estimate distribution of original data values. They perturb data (make changes to values to maintain overall aggregate statistics: x + r instead of x and/or discretization of x) so values cannot be accurately estimated but distribution of values can be. They show that it is possible to do data mining on perturbed data as the accuracy of classifiers built with original data is close to the accuracy of classifiers built by using reconstructed distribution.

In 2003, Kargupta [30] showed that random value distortion techniques, as for example the method described above, preserve very little privacy as original data can be retrieved from the dataset distorted by adding values.

While privacy preserving data mining might work on numerical data, there is no solution yet to do privacy preserving data mining on categorical data. [4] proposes to perturb categorical values by retaining the value with probability p and choosing one of the other values at random with probability 1 - p (if more, $1 - p_2 - p_3 - \ldots - p_n$).

Clifton, Vaidya and Kantarcioğlu ([14],[13],[15]) approach privacy preserving data mining from a different angle. Instead of having a single site where privacy preserved data mining is performed, they suggest methods to enable data mining to be done across various sources, while respecting their privacy policies. These methods of learning new information from data without revealing the data are based on Secure Multiparty Computation (SMC) developed by Yao [55]. Figure 6 shows Secure Sum as an example of SMC.

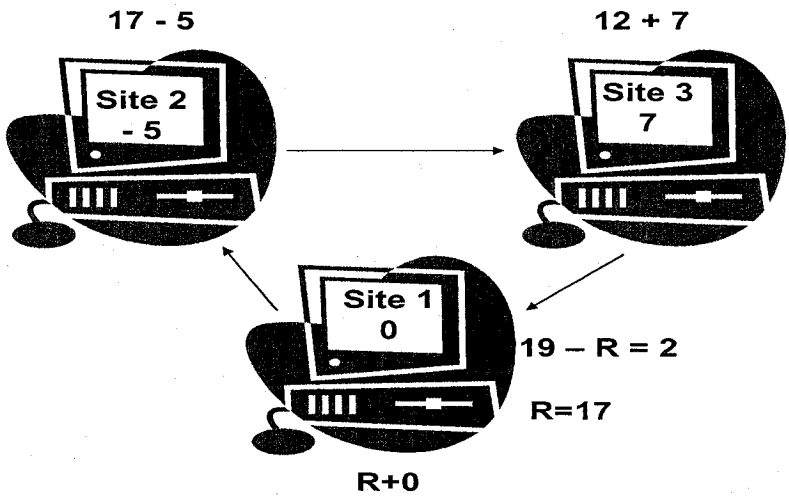

FIGURE 6: SECURE SUM AS A SIMPLE EXAMPLE OF SMC (CLIFTON [13])

Secure Sum in the example shown in Figure 6 has three participating sites (Note that Secure Sum as described in the example, cannot function for less than three sites). Each site has their own number, unknown to the other sites (Site 1 has 0, Site 2 has -5, and Site 3 has 7). The idea is to find the sum of these numbers without making public the numbers of each site (that's why it is called Secure Sum). Site 1 generates a random number, R=17, and then adds its number to R (17+0 = 17), which is then given to Site 2. Site 2 doesn't know R so it cannot determine the number of Site 1. Now, Site 2 adds its number to the number received from Site 1 (17+ (-5) = 12) and sends 12 to Site 3. Site 3 doesn't know R either so it cannot determine the numbers of Site 1 and Site 2. Site 3 adds its number to the number received from Site 2 (12+7 = 19) and passes it on to Site 1. Site 1 subtracts R from the number received from Site 3 (19-17 = 2) and gets the result 2, which is actually the sum of the three numbers (0+ (-5) +7 = 2). Obviously, Site 1 cannot find out from the number received from Site 3 the numbers of Site 2 and Site 3. The final sum (2) can be made public as no site can find out of it the numbers of the other two sites.

Techniques for learning results without revealing the data have been developed for association rules (horizontally [29] and vertically [52] partitioned data), classification [54], and clustering [53]. The idea is that despite of considerable communication among the parties involved, parties learn nothing but the results.

Horizontal partition means that each dataset is split across multiple sites, with each site having the same attributes but different records. Vertical partition means that each dataset is split across multiple sites, with each site having a different set of attributes but same records.

For horizontally partitioned data (different records, same attributes at each site), three or more noncolluding parties keep individual rules secret. It cannot be applied for less than three parties as a globally supported rule that is not supported at one site is surely supported by the other site. If there is collusion, each party divides its input into n parts and sends n-1 pieces to different sites.

Privacy preserving association rule mining in vertically partitioned data (same records, different attributes at each site) requires that the absence or presence of an attribute is represented as 0 or 1 (i.e. informing each site of the number of attributes in the database and also, indicating which attributes are absent or present in a particular partition).

3.2. Watermarking of Relational Databases

Although there is extensive literature in the area of watermarking in general, none of it can directly be applied for watermarking of databases (and for inserting fictitious records into a database) for several reasons as discussed below (and also in sections 1.1 and 2.3 of this thesis).

Digital watermarking of images involves modifying pixel intensities or transform coefficients of an image, and these techniques can be extended to watermarking video and audio in a straightforward manner ([45], [27], [7]). Digital watermarking of textual documents involves modifying text formatting (serifs or width) or layout (word and line spacing) properties ([8]). Software watermarking techniques may include storing the watermark directly into the data or code sections of a binary executable or (Java) class file, in the run-time structures of a program or in dynamic linked data structures ([16]).

Such techniques cannot be used for relational database watermarking as the characteristics of relational databases substantially differ from multimedia, formatted text or software ([4]):

- Multimedia objects consist of a large number of bits with considerable redundancy and thus, offering a large cover in which to hide the watermark. In a relational database, each record represents a separate object and the watermark should be spread over these separate objects.
- Contrary to multimedia, formatted text or software, there is no implied ordering between database records.
- Contrary to multimedia, formatted text or software, database records can be arbitrarily removed or substituted with records from other relations

Some requirements of general watermarking schemes and some general attacks on watermarks can, however, be applicable to watermarking of relational databases. These watermarking requirements and related attacks are summarized in Table 1 on the next page (colour codes indicate similar requirements/attacks but categorized differently by different works).

It can be observed that while some requirements and attacks are consistently subject of research in the area of watermarking (even though they might be named differently in different papers), other requirements and attacks appear only in few papers. This thesis attempts to identify requirements and attacks applicable to the scheme proposed in this thesis. These requirements and attacks are listed in sections 4.3 and 4.4, while their analysis is found in sections 4.5.5, 4.5.6, 4.6.5 and 4.6.6.

REQUIREMENTS [Source]		ATTACKS [Source]	
Robustness	The watermark should not degrade the quality of work.[41] The watermark should survive all attacks (and their combination) which do not degrade the work's perceived quality.[41] Robust to common distortions.[17]	**Robustness**	Diminish or remove the presence of a watermark. [41], [18] Removal attacks (includes the collusion attack): to remove the watermark. [26]
	Detecting the presence and/or value of the watermark should require knowledge of a secret.[41] Difficult to notice. [17]		Simple attacks: to manipulate the content without attempting to identify and isolate the watermark. [26]
	If multiple watermarks are inserted in a single object, they should not interfere with each other.[41] Allowing the co-existence of multiple watermarks. [17]		Inversion attacks: to detect the watermark and reverse the embedding process to remove it. [26]
	The watermark must be difficult (hopefully impossible) to remove. [26], [4]		Synchronization attacks: to change the object such that the watermark is not detectable (for example the mosaic attack where an image is chopped into small distinct sub-images). [26]
Tamper resistance	Resistant to removal attempts. [17]		Interpretation attacks (ambiguity attacks, deadlock attacks, confusion attacks etc.): to forge invalid or multiple interpretations from watermark evidence. [26]
	Scalable decoder [17]	**Additive attacks**	Adding another watermark to the watermarked relation and claiming ownership. [4]
Imperceptibility	The watermark should not be noticeable. [26]	**Attribute remapping**	Find an, at least partial, value-preserving mapping from the original attribute data domain to a new domain. [47]
	The watermark should not degrade the quality of the content [26] Modifications caused by the watermark should not reduce the usefulness of the database. [4]	**Invertibility attacks**	Reversing the embedding process after discovering the watermark to remove it. [4]

REQUIREMENTS [Source]		ATTACKS [Source]	
Detectability	The watermark should be detected with a high degree of reliability i.e. the probability of false positives (detecting the watermark when it is not present) and false negatives (not detecting the watermark when it is present) detection has to be extremely small. [26], [4]	**Presentation**	Modify the content such that the detector cannot detect the watermark. [41], [18]
Capacity	Number of watermarks that can be embedded without degrading the quality of the content and that can be extracted correctly. This requirement affects robustness, detectability and the uniqueness of the watermark. [26]	**Interpretation**	Devise a situation which prevents assertion of ownership. [41], [18]
Incremental updatability	The database should be updatable without destroying the watermark [4] Modification of the existing watermark and multiple watermarks. [26]	**Benign updates**	The watermarking should not be destroyed when the database is updated. [4]
Blindness	Watermark detection should neither require the knowledge of the original database nor the watermark [4]	**Bit attacks**	Attempts to destroy the watermark by updating some bits. [4]
Key-based system	The watermarking system should assume that the method used for inserting a watermark is public. Defense must lie only in the choice of the private key. [4]	**Randomization attacks**	Assigning random values to some number of bit positions (zero out attacks set values of some number of bit positions to zero, bit flipping attacks invert the values of some number of bit positions). [4]
		Rounding attacks	Rounding all the values of a numeric attribute or translating numeric values using units of measurement. [4]
		Subset attacks	Taking a subset of the records or attributes of a watermarked relation and hoping that the watermark is lost. [4]
			Subset addition: add records to the original dataset. [47]
			Subset alteration: alter a subset of records such that

REQUIREMENTS [Source]		ATTACKS [Source]	
			there is still value associated with the resulting set. [47]
			Subset resorting: sort records to make watermark detection/retrieval difficult. [47]
			Horizontal data partitioning: Select and use a subset of records of the original dataset. [47]
			Vertical data partitioning: select and use a subset of attributes of the original dataset. [47]
		Mix and match attacks	Creating a new relation by taking disjoint records from multiple relations containing similar information. [4]

TABLE 1: SUMMARY OF REQUIREMENTS AND ATTACKS (RELATED WORK)

The idea behind using watermarking for privacy protection is to use destination based watermarking where each distributed copy of the data gets a watermark that uniquely identifies the owner of the data.

Using watermarking for privacy protection can be considered as the third approach or a complement to the two existing approaches (altering data before disclosure and learning results without revealing the data). The idea is to insert a unique watermark in each distributed copy of the database or its subset to identify the owner of private information so it can be used to trace the owner in the case of illegal disclosure.

Agrawal and Kiernan [4] provide a solution for watermarking the numerical values of a database. The watermark insertion algorithm modifies the least significant bit of some numerical values. While it is a very intriguing solution, it cannot be directly applied to watermarking databases containing personal information as this information mostly consists of character strings. Social insurance numbers, postal codes and other numerical values cannot be watermarked as these values are exact and any modification to the least significant bit as proposed by Agrawal and Kiernan would render these data inaccurate. Also, as already mentioned, social insurance numbers and other identifiers are most probably removed from the database before disclosure (anonymization), thus watermarking these values would be useless.

On the other hand, watermarking of string values (i.e. quasi-identifiers such as medical diagnosis, gender, banking information etc.) could provide a more adequate solution. Sion [47] proposes a solution to watermarking string values based on the use of associations between categorical attributes. In this solution, watermark embedding modifies existing data, thus making data less useful.

This thesis proposes a scheme where mark embedding does not modify existing data. Since the mark to be embedded is a fingerprint (or as it is sometimes called, a destination based watermark) and not a watermark that is the same for all recipients, work related to fingerprinting databases is discussed before the actual review of the proposed scheme.

3.3. Fingerprinting of Relational Databases

Similarly to watermarking, extensive literature is available on fingerprinting research. One of the papers worth mentioning is Collusion-Secure Fingerprinting for Digital Data by Boneh and Shaw ([6]). They discuss methods for fingerprinting digital data in general (such as multimedia, text documents, software) and provide a general fingerprinting solution which is resistant to collusion attacks. A fingerprint is defined as a word of length L over an alphabet Σ of size s. Collusion attacks are presented as a coalition of c recipients ($c \leqslant n$ where n is the number of all "legal" recipients for whom a fingerprint has been determined by the distributor) who compare their fingerprinted copies in order to create a new copy that possibly does not contain the fingerprint of any of the c recipients. When this new copy is found, the distributor would like to detect a subset of the coalition who created it. Boneh and Shaw show that if the distributor prior to embedding the fingerprints randomly picks a permutation (that is being kept secret) from the full symmetric group of all permutations on l letters, and if that same permutation is used for all users, it is possible to create a function, the actual fingerprinting scheme (l,n), which 1) maps a user number $1 \leqslant u \leqslant n$ and the permutation to a fingerprint, and 2) is c-secure with ϵ error. c-secure with ϵ error means that the described fingerprinting scheme enables to capture a member of the coalition with probability at least $1-\epsilon$, for any $1 > \epsilon > 0$. Also, Boneh and Shaw provide the length of such a fingerprint as $O(c^4 \log n \log(1/\epsilon))$, and show that the lower bound for the fingerprint length is $\frac{1}{2}(c-3)\log(1/\epsilon)$. Despite the fact that the length of such a fingerprint is impractical as for example peer-to-peer services on the Internet could make c as large as 100 or more, this is a very important conclusion and has been a building block for many fingerprinting schemes.

Requirements for fingerprinting schemes are the same as requirements for watermarking schemes, with the exception of adding resistance to collusion attacks. Collusion attacks are not applicable to watermarking schemes (except for destination based watermarking) as in watermarking schemes all recipients receive the same watermarked copy.

An extension of watermarking for privacy protection is to use fingerprinting to identify not only the owner but the recipient of the database as well. Li, Swarup and Jajodia [32]

propose this idea but their solution provides fingerprinting of numerical values only. It also heavily depends on primary key attributes like Agrawal and Kiernan's work [4]. In 2003, Li, Swarup and Jajodia [31] suggest an improvement of their previous solution by constructing a virtual primary key instead of using the actual primary key. This work, however, inserts fingerprints in existing numerical values. Fingerprinting categorical values remains a challenge.

This thesis is based on my paper [38] and it explores the possibility of embedding fingerprints in categorical values of a database. In addition, in the proposed fingerprinting scheme, existing data is not modified. Fingerprints are inserted into the quasi-identifiers of categorical data and records containing the fingerprints are added to the existing database. This fingerprinting scheme can be used as a complement to the existing approaches, namely to altering data before disclosure and learning results without revealing the data. In fact, this fingerprinting scheme is a refinement of the "adding tuples" data altering technique (section 3.1) as added tuples have the purpose of not only obscuring data but hiding information about the owner and recipient of the data. Fingerprinted data can still be used for privacy preserving data mining.

4. Fingerprinting Scheme

4.1. Overview of the Scheme

As shown in [49], re-identification experiments (data matching) can restore identifiers (that uniquely identify an entity), which were removed by anonymization. Data matching means bringing data together from different sources and comparing it. Data matching is a special case of a collusion attack, which is explained in more detail in the next section.

For successful fingerprinting, the fingerprint should remain in the database after anonymization. Confidential data is divided into identifiers and quasi-identifiers, and the fingerprint is inserted into the quasi-identifiers. In this case, even if anonymization is applied, removing the identifiers will not remove the fingerprint as the fingerprint has never been inserted into the identifiers.

Therefore, this thesis proposes a fingerprinting scheme that creates fictitious records by inserting fingerprints into quasi-identifiers of data and adds these fictitious records to a database. At the same time, original data is not modified so information about identifiable individuals remains accurate. The purpose of this fingerprinting is to limit unauthorized disclosure i.e. to deter or dissuade legitimate data users from disclosing the data to illegitimate users who might use data for re-identification of individuals. If an individual or any other entity suspects that a user is an illegitimate user, data found with this user could be searched for fingerprints to determine who disclosed its data to the illegitimate user.

This fingerprinting scheme is composed of the:

- fingerprint;
- encoder (insertion algorithm);
- decoder (extraction algorithm); and
- detection algorithm.

A fingerprint is a mark that uniquely identifies each recipient of the database who receives an authorized copy directly from the distributor. As described in section 2.4, this

proposal assumes the existence of a trusted central authority called the distributor, who manages the fingerprinting process. Since the distributor discloses the database to only a finite number of recipients, it is possible to determine unique fingerprints. In the proposed scheme, a fingerprint is a unique sequence of characters. Each recipient has its own unique fingerprint and the distributor is the only entity in possession of all fingerprints.

4.1.1. Data Analysis Prior to Fingerprinting

The database should be analyzed to determine fingerprints by selecting variations of characters that appear in almost equal number of records (where the order of the characters is important, but the characters do not need to be adjacent in a record) of the database.

One way of conducting this analysis[5] could be as follows:

1. Create all variations of length n (n = 3,4,...,10) of the English alphabet characters
2. Count the number of records variations appear in in the database[6]
3. Select those m (m = 3,4,...,10) variations of the same length that appear the same (or almost the same) number of different records[7] in the database

In the worst case, the running time of step 2. of the algorithm is the same as the running time of the detection algoritm multiplied by a constant where this constant is the number of variations to be searched for or alternatively, the number of fingerprints to be determined.

An alternative view of fingerprints could be that they represent n-grams [34] (of not necessarily consecutive) characters appearing in the database.

[5] The proof of concept implementation in this proposal uses this algorithm.

[6] Note that step 2. does not mean that the number of records for each variation has to be counted. A frequency histogram can be done for all (upper or lower) characters and from this information it can be determined which characters are to be used for creating variations (for example, M or A may appear more frequently in the database whereas J or Z may appear less frequently so a good start may be using variations of these characters). Also, if a sufficient number of variations are found to be appearing almost the same number of times in the database, algorithm execution may be halted.

[7] These records must not be the same records for any pair of these variations. This is discussed in more detail in sections 4.5.1. and 4.6.1.

For example: if the following variations of length 5 appear equally (every such variation, let's say, in 8 records) in a database: {BADMS, BCIMS, CDEMM, CDEMS, CIMNS, CMNST, CNSTT, DAEMS, DAEMT, STAIM}, and these 8 records are not the same for any pair of these variations, these variations could be the fingerprints.

Therefore, the fingerprint list is defined as a set of letter variations of the same length that appear in almost the same number of records of a database, where records containing these variations are not the same records for any pair of variations. A fingerprint is an element of the fingerprint list.

The fingerprints and their length are kept secretly at the distributor's side. A fingerprint of a certain length can be determined randomly from this fingerprint list or by using a (known) hash function that transforms the recipient's name, registration number and the distributor's name into a fingerprint from the fingerprint list.

If fingerprints are determined randomly, then every newly selected fingerprint must be checked against all existing fingerprints to maintain uniqueness. This could happen when a new recipient who does not have an assigned fingerprint yet is added to the existing recipient list. A solution to this problem could be that each time a fingerprint is randomly selected from the available fingerprint list, this list is updated by removing the selected fingerprint from the fingerprint list.

If it is decided that a hash function will be used for determining recipients' fingerprints, then there are certain considerations to be taken into account. The hash function should be one-way and collision resistant, which, according to [21], means the following:

- Finding any input which transforms to any pre-specified output is computationally infeasible – so it is computationally infeasible to determine from a fingerprint who the recipient is;

- Finding any second input which has the same output as any specified input is computationally infeasible (weakly collision resistance) – so it is computationally infeasible to find another recipient with the same fingerprint;
- Collision resistance: finding two distinct inputs that transform to the same output is computationally infeasible – so it is computationally infeasible to find any two recipients with the same fingerprint.

The most popular hash functions are MD4, MD5 (the strengthened version of MD4, which is still not collision resistant), RIPEMD (also a strengthened version of MD4 and is not collision resistant), and Secure Hash Algorithm (SHA-1, which is known to be vulnerable to attacks).

Incremental back up of the database is assumed as well as the time stamping of the requested dataset. Given the dynamic nature of databases i.e. that they are constantly modified, records are added, modified and deleted, it is necessary to analyze the data time to time (the frequency of data analysis depends on the nature of data but as a very general rule, data can be analyzed every six months to two years[8]) and if variations that appear almost equally in the database change, the actual fingerprints have to be changed, too. Thus, it is important that the distributor keeps the list of all fingerprints (past and present) for all recipients. Since datasets are time stamped, the distributor will know which set of fingerprints to be searched for when there is a request to determine the recipient who illegally disclosed data.

4.1.2. Fingerprinting Process

Figure 7 illustrates the fingerprinting process. Upon request, the distributor embeds the fingerprint in the copy of the database using the insertion algorithm (encoding) and transfers the database to the recipient who requested it. In some instances (discussed in more detail later in this chapter), the received copy is decoded using the extraction algorithm (decoding).

[8] Note that creating variations needs to be done only once.

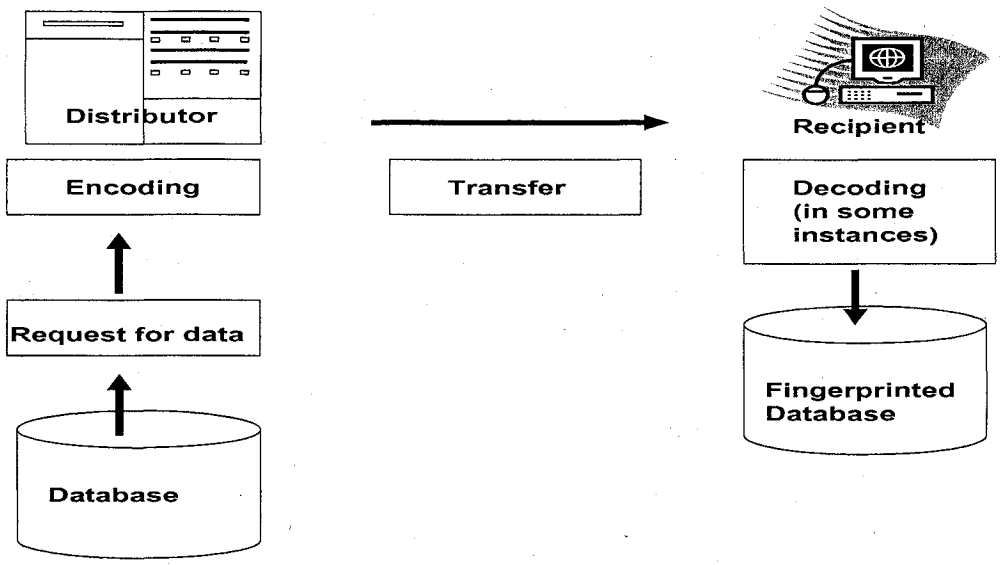

FIGURE 7: FINGERPRINTING PROCESS

The proposed fingerprinting scheme does not modify existing data. Instead, fingerprints are inserted into the quasi-identifiers of categorical data and records containing the fingerprints (fictitious records) are added to the existing database.

When requested by a recipient, the distributor embeds the appropriate unique fingerprint (encoding) in the quasi-identifiers of the copy of the database and discloses it to the recipient who asked for the copy. For the purposes of this fingerprinting scheme, only one database is kept and it does not contain any fictitious records. Each requested copy is fingerprinted when a request is made and is sent off to the recipient who requested it.

Data requested by a recipient is encoded by the distributor prior to transfer. Encoding is done using the insertion algorithm (see sections 4.5.2 and 4.6.2) and it is in fact inserting the fingerprint in the quasi-identifiers of the database. Since existing data is not modified, the fingerprint is inserted into one or more fictitious records created by populating record fields from a pre-determined domain set. When inserting the fingerprint into a fictitious record, the characters of the fingerprint are inserted in the same order as they appear in

the fingerprint, but they do not have to be adjacent to each other in the record. Then, fictitious records containing the fingerprint are inserted in the database.

Decoding means applying the fingerprint extraction algorithm. The purpose of the fingerprint extraction algorithm is to, in certain cases, extract the fingerprint by removing almost all fictitious records from the data (see sections 4.2 and 4.5.3).

The purpose of the detection algorithm (see sections 4.5.4 and 4.6.4), on the other hand, is to identify the recipient who performed an unauthorized disclosure of the database. The detection algorithm is applied when a request comes in to determine the recipient of a particular dataset. Since only the distributor knows the fingerprints of its recipients, it must search the dataset for all fingerprints to determine, which recipient disclosed the data. The search is conducted in a sequential order, record by record for each fingerprint. Each record is searched for every fingerprint. If a character of a particular fingerprint is found, the search continues for the next character of that fingerprint in the same record. If not all the characters of a particular fingerprint can be found in a record, the search continues for the next fingerprint in that record. Detection is time consuming as it depends on the length and number of fingerprints, number of records in the dataset and the length of the records in the dataset.

Since the domain set, its cardinality and the length of the fingerprints are not known, for deducing the fingerprint (whether it was determined randomly or by using a hash function), someone would have to apply the detection algorithm to search for all variations that appear almost equally in a database and determine which one of them appears most frequently in their copy of the database (data analysis, section 4.1.1). This attack, along with other attacks, is discussed in more detail in 4.4., 4.5.6. and 4.6.6.

As explained in section 2.4, this scheme can be classified as a symmetric fingerprinting scheme (but it can also be extended to satisfy asymmetric fingerprinting requirements for certain elements of the scheme – section 7.4) because it defines the distributor as the entity who knows the real identity of the database recipients and who determines the

fingerprint of each recipient. If the distributor is a trusted party (registered with a central authority), malicious activities or false accusations that can happen with symmetric fingerprinting schemes can be avoided.

4.2. Trusted and Distrusted Recipients

The proposed scheme is a two-tier scheme as we distinguish between trusted (tier 1) and distrusted (tier 2) circles of recipients. This distinction, determined by the distributor, is not necessary but has a benefit of demonstrating various situations when the distributor wishes to distribute "cleaner" or "less clean" data. Under "cleaner" or "less clean" data we mean less or more noise caused by the fingerprints.

Members of the distrusted circle are defined as recipients (and are determined by the distributor) who should never get pure data because there is a risk that they could share the data with third parties. Such discrimination has nothing to do with social values and is frequently present in real life: for example, determining the security clearance of employees of an organization or approving authorized access to certain systems or data in an organization.

Members of the trusted circle are defined as recipients who could receive Almost Clean Data (ACD). The emphasis is on "almost" as these recipients will also have their fingerprints inserted in the data but in a less frequent manner. Members of the trusted circle are also determined by the distributor. They have full (read and write) or limited (read but no write) access to ACD.

While the fingerprinting scheme for the trusted circle of database sharers contains a decoder, the fingerprinting scheme for the distrusted circle does not. The decoder extracts all fictitious records containing the fingerprint from the database except one, and the purpose of the decoder, therefore, is to safeguard the quality of data for trusted recipients. The remaining record is always the same record (for example, the third fictitious record) for the same recipient and it serves to deter the trusted circle member from disclosing the data. A trusted circle member is trusted because the probability he would disclose the

data illegally is extremely small (let's say less than 0.01). So if he knows that his data is fingerprinted, this probability is even more reduced. Not because the trusted circle member could not try to disclose data or attack the scheme but because he would not want to compromise his trusted status and future access to ACD.

The circle of distrusted recipients includes recipients who are also receiving the data but because there is a high risk that these recipients could share the data with entities that do not belong to the trusted or distrusted circle of database sharers, data must be fully fingerprinted. Therefore, the distrusted circle doesn't have a decoder. In other words, it doesn't have the opportunity to use the extraction algorithm. Members of the distrusted circle have only limited (read but no write) access to fully fingerprinted data (not ACD).

Although there is no secret about the decoder (extracting algorithm), distrusted circle members could not benefit from it unless they know which records to extract from the database. The distributor applies the encoder (insertion algorithm) for trusted circle members when the recipient is a trusted member and the encoder for distrusted circle members when the recipient is a distrusted member. The main difference between the two encoders is that all fictitious records containing the trusted member's fingerprint, except one, will be marked prior to transfer and no fictitious records containing the distrusted member's fingerprint will be marked prior to transfer. So if the distrusted circle member does not know which records are fictitious in the received database, its decoder will not be able to remove them.

Possible criteria to determine trust include:

- Type of the purpose the recipient uses the confidential data for (primary versus secondary purpose)

- The recipient's relation to the Distributor: dependent (headquarters, offices, regions, laboratories, departments, branches...) versus independent (separate entity)

- Security reasons (full versus limited access)

The first criterion builds directly on the privacy principles discussed in chapter 2 (Background) of this work. Purposes for which personal information is collected must be identified before or at the time of collection. These purposes can be directly related to the use of personal information (for example determining eligibility for a program or service) and then these purposes can be characterized as primary purposes, or they can be indirectly related to the use of personal information (for example, statistical purposes), in which case they can be characterized as secondary purposes.

Example 1:

A social housing corporation collects confidential data (name, age, birth date, salary, banking information) from its tenants to determine payable rent (identified primary purpose). However, on the form to be completed by the tenant, there is a statement that informs the tenant that the confidential data collected may be used for statistical analysis (identified secondary purpose). If the statistical analysis is performed by a separate entity who does not belong to the social housing corporation, this entity could be identified as a distrusted recipient since it is independent from the corporation and it will use the personal information for a secondary purpose.

Example 2:

Another example is SEARS, the multi-channel retailer. SEARS Canada's privacy policy, posted on http://www.sears.ca/e/info/hwuyinfo.htm states that:

- SEARS collects confidential data (name, mailing address, e-mail address, telephone number, date of birth, credit history, transaction history) from its customers to identify the customer, establish and maintain a relationship with the customer, provide ongoing service and protection against error and fraud, and comply with legal requirements (identified primary purpose)
- "Data we collect with personal identifiers removed, so that it is impossible to determine the identity of the person to whom the information relates, is not considered Personal Information"
- "We will not sell your Personal Information to any organization, for any purpose"

In other words, SEARS will sell its data but without personal identifiers, which means that re-identification (using quasi-identifiers for identification) and data matching procedures are not taken into account.

Also, there is a statement in the privacy policy that informs the customer that confidential data collected may be used for creating statistics about SEARS' business (identified secondary purpose).

This fingerprinting scheme could be applied in the SEARS case as follows: all entities that belong to SEARS are considered members of the trusted circle of database sharers (have full or limited access to ACD); any other entity who is receiving data from SEARS (even with personal identifiers removed) is a member of the distrusted circle of database sharers (has no access to ACD and has limited access to fully fingerprinted data). Data received by distrusted circle members is definitely less useful than data received by trusted circle members as it is fully fingerprinted data but it is important to make this distinction in order to safeguard against unauthorized data disclosure.

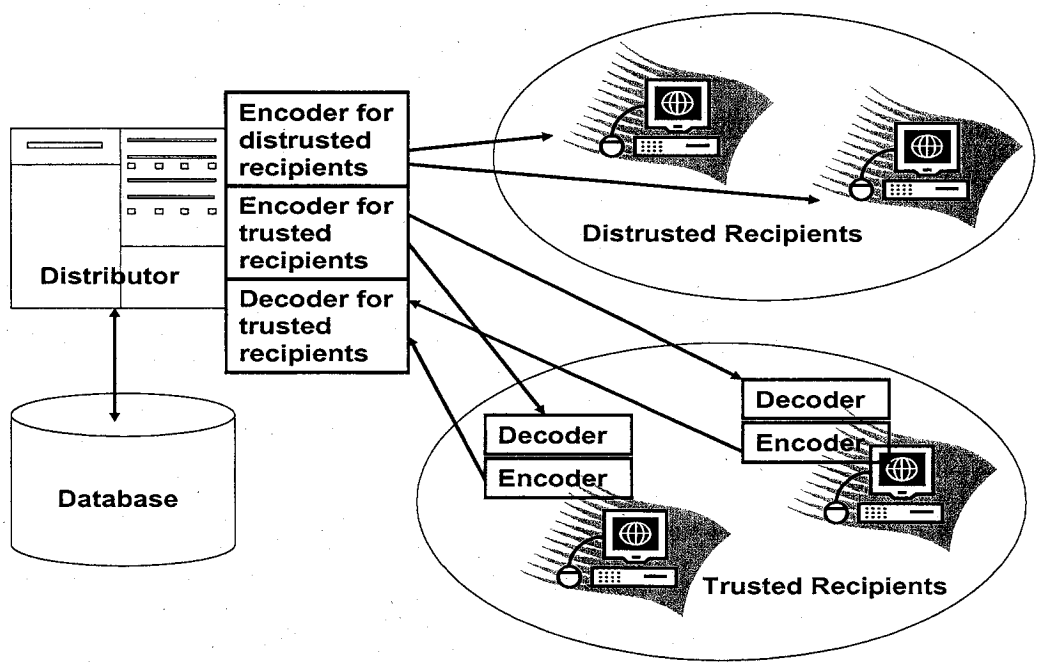

FIGURE 8: PROPOSED FINGERPRINTING SCHEME

As Figure 8 demonstrates, data requested by a recipient (member of the trusted or distrusted circle) is encoded by the distributor prior to transfer. Encoding is done using the insertion algorithm. All members of the trusted circle have a decoder so they can access ACD whereas members of the distrusted circle get only the fully fingerprinted copy of the requested data. Members of the trusted circle may modify the requested data and send it back to the distributor. This data is encoded again at the trusted recipient prior to transfer and the distributor also has a decoder to extract the fingerprint before storing the data in the database.

Attacks on the fingerprint can occur any time between the encoder and decoder and in these cases data is fully fingerprinted. However, if a trusted recipient discloses data to a third party after decoding it, ownership can again be determined as the data is ACD meaning that it still contains fingerprinted records. A detailed analysis of this fingerprinting scheme is presented in section 4.5. of this chapter.

Members of the distrusted circle of database sharers do not have a decoder and an encoder, which means that they receive the fully fingerprinted dataset. They have only limited (read but not write) access to data so they cannot modify received data to send it back to the distributor. In other words, distrusted circle members always receive the data "as is", they do not have physical control of the database and do not have the right to access the original database. A detailed analysis of this fingerprinting scheme is presented in section 4.6. of this chapter.

4.3. Requirements

The proposed fingerprinting scheme is discussed in more detail in sections 4.5. and 4.6. of this chapter. Both the tier-1 and tier-2 schemes are analyzed (sections 4.5.5 and 4.6.5) to determine whether they satisfy the requirements specified herein. As explained in the previous section, the tier 1 scheme involves members of the trusted circle and the tier 2 scheme involves members of the distrusted circle. Trusted circle members could receive

ACD, whereas distrusted circle members should never get pure data because there is a risk that they could share the data with third parties.

The following requirements have been determined by adapting requirements discussed in chapter 3 (Related Work) of this thesis to the proposed fingerprinting scheme:

Usefulness/Imperceptibility

Data has to be useful at all times and the encoding and decoding algorithms must be executable in reasonable running time. Imperceptibility means that modifications caused by the fingerprint do not reduce the usefulness of the data.

Robustness

The fingerprinting scheme should be robust. In other words, it must be difficult (hopefully impossible) to destroy or overwrite the fingerprint i.e. the fingerprinting scheme has to be resistant to attacks that attempt to destroy or overwrite the fingerprint. Robustness attacks (along with other attacks) are listed and discussed in more detail in section 4.4.

Detectability

An authority has to be able to determine from the data who the owner is. This is called the high detection reliability and it means that the probability of detecting the fingerprint in the data when there is no fingerprint (false positive or type I error) and the probability of a missed detection of the fingerprint when there is in fact a fingerprint in the data (false negative or type II error) are extremely small (Guo [26]). Since lowering the probability of false negatives increases the probability of false positives, a balance must be chosen between risks of false negatives or false positives.

Since in the proposed fingerprinting scheme fictitious records containing the fingerprint are inserted in the database, and the original database already contains the fingerprints of all recipients (trusted or distrusted), this requirement translates into the following: to achieve high detection reliability, the fingerprint of the recipient who actually disclosed the data must appear more frequently in the data than the

fingerprints of other recipients[9]. If the fingerprint of a recipient who did not disclose the data appears most frequently in the data, this recipient will be accused of disclosing the data and this is false positive. To reduce the number of false positives i.e. to have an extremely small probability of false positives, fictitious records to be inserted should ideally include the fingerprint of the recipient only and not the fingerprints of other recipients. If the fingerprint of a recipient who did disclose the data does not appear most frequently in the data, this recipient will not be accused of disclosing the data and this is false negative. However, if this recipient is not accused, another recipient will be accused whose fingerprint appears most frequently in the data (false positive). In other words, for every false positive there is a false negative and vice versa. Therefore, to reduce the number of false negatives, i.e. to have an extremely small probability of false negatives, fictitious records to be inserted should ideally include the fingerprint of the recipient only and not the fingerprints of other recipients. This is exactly the same requirement as the requirement to reduce the number of false positives.

Incremental Updatability

Updating of records is possible without modifying the fingerprint. This property is called incremental updatability.

Blindness

The fingerprinting scheme has to be blind, which means that for detecting the fingerprint there is no need to access the original data.

Capacity

As Guo ([26]) defined (see section 3.2, Related Work: Watermarking), capacity is the number of watermarks that can be embedded without degrading the quality of the content and that can be extracted correctly. This requirement affects robustness, detectability and the uniqueness of the watermark.

[9] Fingerprints are chosen from existing n-grams in the original database: see sections 4.1.1 and 4.5.1

In the proposed fingerprinting scheme, capacity is the number of fictitious records that can be added to the database without degrading its quality (i.e. not increasing its size such that it becomes impractical) and the length of fingerprints that can be detected correctly by the detection algorithm.

Although the fingerprint can always be detected for a recipient (the detection algorithm is deterministic), the fingerprints of other recipients can also be always detected. To determine the recipient who disclosed the data, this particular recipient's fingerprint must appear more frequently in the database than the fingerprints of other recipients. If more fictitious records containing the fingerprint of a particular recipient are inserted into the database, the higher the probability is to detect the fingerprint of that recipient. But, at the same time, the more fictitious records are inserted, the size of the database increases.

4.4. Attacks

The attacks below have been determined by adapting attacks discussed in chapter 3 (Related Work) of this thesis to the proposed fingerprinting scheme and by adding other attacks that might be applicable to the proposed fingerprinting scheme. These attacks are analyzed in this section, section 4.5.6 or section 4.6.6, depending on whether additional information is needed on the proposed fingerprinting scheme. Attacks that require more detail on the proposed Tier 1 or Tier 2 scheme, will be discussed in 4.5.6 or 4.6.6, respectively.

1. **Robustness attacks - attempts to destroy or remove the fingerprint:**

Fingerprint deduction attack (inversion or invertibility attack)

As noted in section 4.1, for deducing the fingerprint (whether it was determined randomly or by using a hash function), someone would have to apply the detection algorithm to search for all variations that appear almost equally in a database and determine which one of them appears most frequently in their copy of the database (see 4.1.1 Data Analysis Prior to Fingerprinting). The difference between the

distributor performing the detection algorithm and an attacker performing the detection algorithm lies in the fact that the distributor knows which fingerprint(s) to search for, whereas the attacker has to guess the fingerprint by analyzing the data. Since the domain set fingerprint characters are drawn from, the cardinality of the domain set and the length of the fingerprints are not known, the attacker would have to:

a. Determine all variations of "reasonable" length of all characters that appear almost equally in a database.
   Assume that the attacker chooses the English alphabet as the domain set of the fingerprints (26 letters) and determines all variations of "reasonable" length to be variations of length 3,4,5,6,7,8,9,10,11,12,13,14,15. That means that the attacker has to generate all possible variations ($26^3+26^4+\ldots+26^{15}$) e.g. AAA, AAB,...,AAZ,...,ZZZ,AAAA,...,ZZZZZZZZZZZZZZY,ZZZZZZZZZZZZZZZ and choose those (as for example, it is reasonable to expect that 15 Z's will not appear in one record of any database, but ZZZ might well appear) that need to be searched for in the database.
b. Search for all of these variations in the database to determine which one of them appears most frequently in the database.
   Since the attacker possesses only the fingerprinted copy of the database and since from the fingerprinted copy of the database it cannot be determined which variations appeared equally in the original database, the attacker's best bet is to search for those variations that appear most frequently in the database.
c. Assume that the most frequently appearing variation is the fingerprint.
d. Remove or modify all records that contain this variation.
   Beside fictitious records containing the fingerprint, the original database also contains records with the fingerprint (see section 4.5.1). If the attacker does not have access to the original database (it is assumed that the original database is protected from intruders – analyzing this type of security is outside the scope of this work), or to other copies of the

database (to produce a collusion attack – discussed later in this section and in sections 4.5.6 and 4.6.6), s/he can only guess the difference between a fictitious and an original record. Removing original records from the database or keeping fictitious records in the database will render data less useful.

Benign attacks (Presentation attacks)

Benign attacks are attacks where the fingerprint is accidentally destroyed. Presentation attacks are attacks when the content is modified such that the detector cannot detect the fingerprint. In both cases, though, the attacks should render the data useless.

Mix and match attacks

In mix and match attacks, different datasets from the same source can be compared to discover fingerprinted records.

Collusion attacks

When recipients compare/share their data (in other words, when mix and match attacks involve more than one recipient), we call it a collusion attack (data matching). This attack is the most challenging attack for fingerprinting schemes.

2. Interpretation attack (Additive attack)

An interpretation attack means that the attacker devises a situation in which the assertion of ownership is prevented. This could happen by inserting another entity's fingerprint in the data. However, for this attack, the attacker should know another entity's fingerprint and should produce fictitious records containing this fingerprint. In addition, to make this attack really powerful, the attacker should also remove the fictitious record(s) containing his own fingerprint.

3. Synchronization attacks (Subset attacks)

Vertical data mining attacks

Vertical partition [52] occurs when each dataset is split across multiple sites, with each site having a different set of attributes. The relations at the individual sites must be joined to get the relation to be mined.

Therefore, one way to survive a vertically partitioning attack, or any attack that involves only a few attributes instead of the whole attribute set, is that the fingerprint is inserted in attributes as well as records. The database would then include fictitious attribute(s) in addition to fictitious records that contain the fingerprint.

Horizontal data mining attacks

Horizontal partition [29] means that each dataset is split across multiple sites, with each site having the same attributes but different records.

Therefore, one way to survive a horizontally partitioning attack, or any attack that involves only a few records instead of the whole set of records, is that the fingerprint is inserted in more than one record, and in such a way, that fingerprinted records are spread around in the database. This strategy, however, challenges the capacity property of fingerprinting schemes. In other words, the more fingerprinted records are embedded in the data, the less useful the data becomes.

Capacity, robustness, detectability and defence against horizontally partitioning attacks are interrelated, and measures applied to achieve any of them could negatively influence the others. Beside collusion attacks, this interrelation is the most difficult problem to address in this work. More precisely, is it possible to determine an optimum number of fingerprinted records to be inserted in the data such that the capacity, robustness and detectability properties are satisfied and at the same time, defence against horizontally partitioning attacks is achieved? Test results of the Proof of Concept implementation, as shown in chapter 5. of this thesis, and further analysis in 4.5.6 and 4.6.6 suggest some conclusions. However, to support these conclusions, more tests, especially on real data, are required in the future.

4.5. Tier 1: Trusted Circle of Database Sharers

Members of the trusted circle are defined as recipients who could receive Almost Clean Data (ACD). They are determined by the distributor and have full (read and write) or limited (read but no write) access to ACD.

This fingerprinting scheme consists of the:

- Fingerprint
- Encoder (insertion algorithm)
- Decoder (extraction algorithm)
- Detector (detection algorithm)

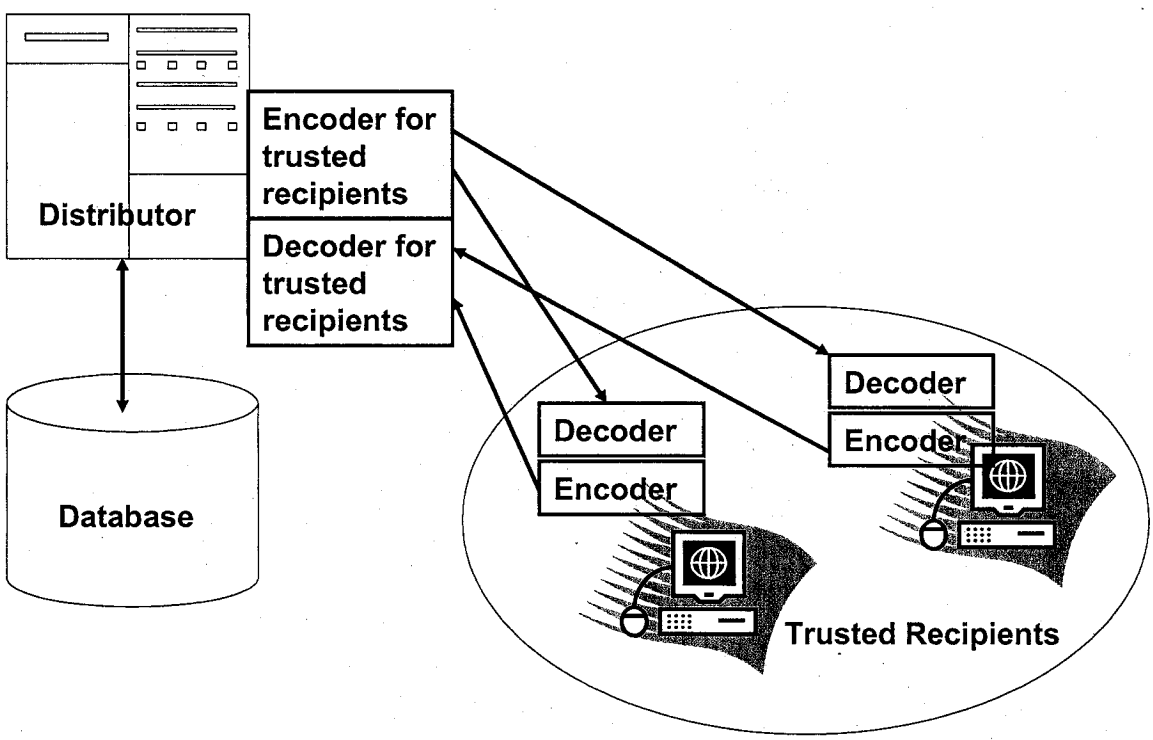

FIGURE 9: FINGERPRINTING SCHEME – TIER 1

The distributor distributes the database to a finite number of recipients. Therefore, it can determine finite number of fingerprints that uniquely identify recipients.

Sections 4.5.2 and 4.5.3. explain the rationale behind having encoders and decoders on Trusted Recipients' side.

The following notations are used:

TC – trusted circle of database sharers;

M – size of TC; i.e. the total number of TC members;

$m_1, m_2, \ldots, m_M$ – TC members;

M is therefore the number of unique fingerprints as a unique fingerprint is assigned to each TC member;

$f_1, f_2, \ldots, f_M$ – fingerprints of TC members, where f_1 is the unique fingerprint of m_1, f_2 is the unique fingerprint of m_2, …, f_M is the unique fingerprint of m_M;

L is the length of each fingerprint; each fingerprint has the same length.

4.5.1. Fingerprint

Prior to determining fingerprints, the domain set from which fingerprint characters are drawn has to be determined. This should be done by analyzing the database, e.g. by selecting variations of characters that appear in almost equal number of records (where the order of the characters is important, but the characters do not need to be adjacent in a record) of the database (see section 4.1.1).

If n is the number of characters comprising the fingerprints and L is the length of the fingerprint, there are n^L unique fingerprints in total. For example, if n = 10 and L = 5, it is possible to determine 100,000 unique fingerprints, which is more than enough for a distributor as it is very certain that the TC will not contain 100,000 members.

For example, the first character of the fingerprint can be from the character set of {B,D,M,N,S,T}, the second character of the fingerprint can be any character from {A,B,C,D,E,I,L,M,N,R}, and the third (last) character of the fingerprint can be any character from {C,M}. In this case, the total number of unique three-character long

fingerprints generated by the distributor is: 6x10x2 = 120. These fingerprints are: BAC,BAM,BBC,BBM,BCC,BCM,...,TRM. Therefore, the domain set of all possible fingerprints is: D = {BAC, BAM, BBC, BBM, BCC, BCM, ..., TRM}.

After the domain set is determined, fingerprints can be assigned randomly to recipients or a one way (and possibly collision resistant) hash function can be used which uniquely maps each recipient's name, number and the distributor's name onto D.

Assumptions

By analyzing the database, it is possible to determine the necessary number of characters (length of the fingerprint) and the characters themselves (the fingerprint) in such a way that the same set of characters already appears in the same order (not necessarily one after another) in approximately τ records of the pure data.

The second assumption we need to make is that the selected fingerprints not overlap. In other words, it is not the same τ records that contain all the fingerprints. Again, it is not difficult to achieve given the large number of possibilities to select from i.e. the total number of unique fingerprints is n^L (if necessary, the length of the fingerprint and the characters of the fingerprint can be modified).

To summarize:

Assumption I. Assigned fingerprints are present in $\tau \pm \epsilon$ pure data records, where ϵ is preferably 0 or a very small integer (e.g. 1, 2 ...).

Assumption II. Fingerprints assigned to different users are not contained in exactly the same records of the original database.

The insertion algorithm (encoder) takes those records that already contain the fingerprint, modifies them in such a way that they resemble the original data but contain different values, and inserts these modified records back in the database. In general, this is possible to do for databases containing categorical values where values can be expanded on,

replaced with similar values or deleted. If the database contains medical information such as symptoms or diagnosis, values can be modified by, for example, expanding on explanations, or replacing words by their synonyms or Latin equivalent. Or, if the database contains retail information such as the buying habits of customers, commodities purchased by a customer can be replaced by other commodities or expanded by adding more commodities. Note, that data can also be deleted during modification, for example, words that don't include the characters of the fingerprint. Explanations for medical diagnosis, or the list of commodities can also be chosen from a domain set determined from the distribution of the data in the database. In addition, identifiers such as the SIN and some quasi-identifiers such as the name or birth date of a person can be replaced by similar but different data. An example of changing a birth date would be to add or subtract one year from the actual year of birth, and select a month and a day in a month. An example of changing a person's name would be to replace the name with a name from a domain set of names (let's say, the domain set would include the 200 most popular female and male names in the last 50 years in North America). An example of changing the postal code is to select a postal code of all applicable postal codes in a certain area. If for example, the database containing retail information is for a store located in Toronto, Greater Toronto Area postal codes would be in the domain set. Also, this information could be selected from a domain set determined from the data distribution in the database.

The modified database will actually contain $2\tau \pm \epsilon$ fingerprinted records and because of Assumption II, fingerprints of other recipients will be present in definitely less than $2\tau \pm \epsilon$ fingerprinted records.

The distributor always has access to the original (pure) database, and inserts the fingerprint of a particular recipient in the database once that recipient requested the data. The fingerprinted copy of the database is then sent to the recipient. If another recipient requests the data, the distributor will again take the original data and insert the fingerprint of this recipient in the original data.

If a TC member sends modified data back to the distributor, data will contain fictitious records. The decoder on the distributor's side will, however, remove all marked fictitious records, except the one that was not marked when the distributor sent the data to the TC member. Since this unmarked record is always the same for the same recipient (as discussed in 4.2), and since the distributor was the entity who inserted it, the distributor is able to find and remove this record from the data.

It can be concluded that (unless the distributor is malicious – we assume that the distributor is a trusted central authority or was selected by a trusted central authority – symmetric fingerprinting schemes, section 4.1) the distributor's encoder will never insert a second fingerprint in the unmarked record which is always the same for the same recipient.

The only important factor for detecting the fingerprint in a database is that the fingerprint of the recipient which illegally disclosed the database appears more frequently than the fingerprint of other recipients (Proof of Concept results). Therefore, there is no exact requirement for τ in terms of its value or range it should fall into. Yet, to achieve resistance against collusion attacks (discussed in 4.5.6 and 4.6.6), there is a desirable value for τ (see section 4.5.6).

4.5.2. Encoder

The proposed fingerprinting scheme for the TC contains two different types of insertion (encoder) algorithms:

- Encoder on the distributor's side for TC members;
- Encoder on the TC members' side

Encoder on the distributor's side for TC members

The steps of the insertion algorithm are as follows:

1. Select $\tau \pm \epsilon$ records that already contain the fingerprint.

2. Modify the copies of chosen records by replacing identifiers and quasi-identifiers with fictitious values chosen from pre-determined domain sets. Domain sets are created using the distribution of the data in the database.

3. Insert modified copies in the database. The database will contain $2\tau \pm \epsilon$ fingerprinted records from which τ records are fictitious records.

4. Steps 2. and 3. can be repeated several times. The only requirement is that each modified copy contains the fingerprint and that the identifiers and quasi-identifiers differ in each copy. With each repetition, the number of fingerprinted records will increase by $\tau \pm \epsilon$.

5. A separate column can be added or the primary key column can be modified to contain "y" or "n" depending on whether the record contains the fingerprint or not.

6. The modified database is encrypted with the TC member's public key.

The last two steps of this insertion algorithm are not indispensable. They only serve to ensure that members of the TC get ACD in a fast way. Once the member of the TC receives the data, its decoder decrypts the data and removes marked fictitious records. If the fictitious records were not marked, the decoder of the TC member would have to apply the detection algorithm to find and remove fictitious records in order for the TC member to get ACD and this would take more time then just removing marked records. If fictitious records were marked but the data were not encrypted, and if data were captured by an intruder during transfer from the encoder to the decoder, the intruder could easily find out which records were fictitious.

For TC members to get ACD in a fast way could also be achieved by inserting less than τ fictitious records (for example, inserting only 1 fictitious record) in the data. In that case, however, if data with only one fictitious record gets into the hands of an intruder between the encoder and decoder, the intruder would get ACD (for this not to happen, unmarked data can still be encrypted). The above algorithm is intended to serve as a general algorithm where τ can be any number and steps 4. and 5. are optional. See more discussion on this under Encoder on TC Members' Side later in this section.

To ensure that members of the TC leave their fingerprint on the data, one fictitious record is left unmarked (step 5. of the algorithm) so the decoder cannot remove it. The index of this record is selected randomly the first time from the list of fictitious records for a TC member and then the record is kept along with TC member information in the member database along with fingerprints. The total number of marked records is therefore $(\alpha\tau - 1) \pm \epsilon$, where α is the number of repetitions (step 4. of the algorithm); original records containing the fingerprint are not marked. This fingerprinted record has the preventive purpose of deterring TC members from disclosing pure data. That unmarked record identifies the TC member.

Pseudocode:

/*Original is the database to be fingerprinted*/
/*Mock is a database where fictitious values are chosen from*/
/*theRecord is a record from Original*/
/*theFingerprint is the fingerprint of the TC member*/
/*Data is the data to be sent to the TC member*/
/*This algorithm takes an input the Original and produces the Data, using Mock*/

/*the following code is repeated for all records in Original*/
$\tau = 0$
/* τ is the number of records containing the fingerprint*/
theRecord = select all records one by one from Original

```
        if the fingerprint is found in theRecord
                increase τ
                insert theRecord in Data
                modify theRecord
                insert the modified theRecord in Data
        end
        else
                add "n" to theRecord
                insert theRecord in Data
        end
end
encrypt Data with the TC Member's public key
found function (returns a boolean)
        result = false
        i,j = 0
        while (i < length of the fingerprint AND  j < length of the record)
                if (theFingerprint.charAt(i) == theRecord.charAt(j))
                        i++
                j++
        end
        if (i == length of the fingerprint)
                        result = true
        return result

modify function (returns a modified record)
        select a record from Mock
        if theRecord is setRecord /*the one fictitious record not marked*/
                add "n" to theRecord
                addIdentifiers to theRecord
        end
        else
```

add "y" to theRecord /*marked fictitious records*/

addIdentifiers to theRecord

end

return the modified theRecord

addIdentifiers function (returns identifiers)

/*if a fingerprint character is not found between two commas of theRecord, select between 1) deleting everything between the two commas, and 2) replacing everything between the two commas with a general identifier from Mock*/

/*select among 1) not adding anything, 2) adding before identifier a general identifier from Mock, 3) adding after identifier a general identifier from Mock, and 4) adding before and after identifier a general identifier from Mock*/

Encoder on the TC members' side

Having an encoder and decoder on the TC members' side is optional. TC members can always receive and send back ACD to the distributor without having a decoder to remove fictitious records and having an encoder to add fictitious records removed by the decoder. However, the main purpose of the proposed fingerprinting scheme is to identify from the data itself the entity who illegally disclosed data and that is easier to do from fully fingerprinted data than ACD (detectability). On the other side, TC members should have ACD to be able to work with more useful data (usefulness). To meet both these requirements (detectability and usefulness), the compromise is to have both a decoder and encoder on the TC members' side to ensure that ACD is accessible only by the TC member and data on its way to a TC member and on its way to the distributor is fully fingerprinted.

Members of the TC have full (read and write) or limited (read but no write) access to ACD. They receive encoded data from the distributor and their decoder removes all but one fictitious record. In other words, the decoder removes $(\alpha\tau - 1) \pm \epsilon$ records. If trusted members have limited access to ACD (pure data with one fictitious record), they read and use the ACD.

If they have full access to ACD, they can modify ACD and after they complete their work with the ACD, their encoder applies the following insertion algorithm:

1. Reattach all fictitious records removed by the decoder. The total number of reattached records is $(\alpha\tau - 1) \pm \epsilon$.

2. Update the added column or the primary key column: if there was an added column when the data was received, reattach it and mark reattached fictitious records, or modify the primary key column by embedding the marks identifying reattached fictitious records.

3. Encrypt data with the distributor's public key

4. Send the encrypted and fingerprinted dataset back to the distributor. The total number of fingerprinted records sent back is $(\alpha+1)\tau \pm \epsilon$, from which $\tau \pm \epsilon$ records represent original records where the fingerprint originally exist.

Pseudocode

```
/*Data is data to be sent to the Distributor*/
/*Marked records are stored in Storage*/
/*Database is the TC member's data*/

theFictitiousRecord = select all records one by one from Storage
theRecord = select all records one by one from Database
        add "n" to theRecord
        insert theRecord in Data
        insert theFictitiousRecord in Data
encrypt Data with the Distributor's publicKey
```

4.5.3. Decoder

The proposed fingerprinting scheme for the TC contains two different types of extraction (decoder) algorithms:

- Decoder on the distributor's side for the members of the TC;

- Decoder on the TC members' side.

Decoder on the distributor's side for the members of the TC

The decoder (extraction algorithm) decrypts data with the distributor's private key, deletes marked fictitious records, restores the primary key column to its original state if it was modified or deletes the added column if there was any, removes the unmarked fictitious record and stores the data in the database.
Steps of the Extraction algorithm:

1. Decrypt data with the distributor's private key (reverse of step 3. of the encoder on the TC members' side)

2. Delete marked fictitious records

3. If the primary key column was modified, restore it to its original state or if there is an added column, remove it

4. Remove the unmarked fictitious record (always the same for the same recipient)

Pseudocode

/*Data is data sent by the TC Member*/

/*Database is the Distributor's data*/

decrypt Data with the Distributor's privateKey

theRecord = select all records one by one from Data

```
while (theRecord.next())
        if theRecord is not marked with "y"
                remove "n" and insert record into Database
        else
        if theRecord is setRecord /*the one fictitious record not marked*/
                remove "n" and insert record into Database
end
```

Decoder on the trusted recipient's side

The decoder (extraction algorithm) decrypts data with the TC member's private key and removes all marked records (one fictitious record is not marked so it is not removed). It also restores the primary key column to its original state if it was modified or deletes the added column if there was any, and uses the data. In the case of members with full access to ACD, the fictitious records are stored separately so that the encoder on the TC member's side can reattach them before sending the modified data back to the distributor. Steps of the Extraction algorithm:

1. Decrypt data with the TC member' private key (reverse of step 6. of the encoder on the distributor's side for TC members)

2. Remove all marked fictitious records (and store them if the TC member has full access to ACD)

3. If the primary key column was modified, restore it to its original state or if there is an added column, remove it

Pseudocode

```
/*Data is data sent by the Distributor*/
/*Marked records are stored in Storage*/
/*Database is the TC member's data*/
```

```
decrypt Data with the TC Member's private key
theRecord = select all records one by one from Data
while (theRecord.next())
        if theRecord is not marked with "y"
                remove "n" and insert record into Database
        else
                insert theRecord in Storage
end
```

TC members with limited access use decoded data but not modify it and therefore, they do not need to send it back to the distributor.

Since the added column or the modified primary key column is not marked for one of the fictitious records, the decoder extracts all records containing the fingerprint from the database, except one. The remaining one fingerprinted record serves to deter the TC member from disclosing the data.

So when a trusted member removes the fingerprint based on knowing which records are fictitious, one fictitious record with the fingerprint remains in the dataset. The trusted member does not know which record it is (only the distributor knows this), and this record identifies the trusted recipient.

That way:

- data is minimally perturbed, one additional fictitious record in a large dataset does not make much difference;
- dishonest trusted recipients cannot remove the fingerprint as they do not know where it is;
- when the pure data is found outside, it can be discovered by an audit in connection with the distributor (i.e. knowing the fingerprint) where the fingerprint is, and therefore who has given it outside the circle of trust.

4.5.4. Detection

The detection algorithm is applied when a request comes in to determine the recipient of a particular dataset. Since only the distributor knows the fingerprints of the members, it must search the dataset for all fingerprints to determine which member disclosed the data. The search is conducted in a sequential order, record by record for each fingerprint. If a character of a particular fingerprint is found, the search continues for the next character of that fingerprint in the same record. If not all the characters of a particular fingerprint can be found in a record, the search continues for the same fingerprint in the next record. The number of occurrence of each fingerprint in the dataset is recorded. The owner of the most frequently occurring fingerprint is accused of unauthorized disclosure.

Pseudocode

```
/*FingerprintList contains fingerprints of all members and is generated by the distributor*/
    /*the following code is repeated for all fingerprints in FingerprintList*/
    theFingerprint = select all fingerprints one by one from FingerprintList
    while (theFingerprint.next())
        total = detect(theFingerprint)
    end

detect function (returns an integer)
    /*DataforDetection is the dataset to be searched for fingerprints*/
    /*theRecord is the current record from DataforDetection*/
    /*the following code is repeated for all records in DataforDetection*/
    theRecord = select all records one by one from DataforDetection
    while (theRecords.next())
        if found(theFingerprint, theRecord)
            number++
    end
    return number
```

```
found function (returns a boolean)
        result = false
        i,j = 0
        while (i < length of the fingerprint AND  j < length of the record)
                if (theFingerprint.charAt(i) == theRecord.charAt(j))
                        i++
                j++
        end
        if (i == length of the fingerprint)
                        result = true
        return result
```

Detection time depends on the number of fingerprints, number of records in the dataset and the length of the records in the dataset. If N is the number of records in the database, $r_1,r_2,r_3,\ldots,r_N$ are the lengths of $record_1$, $record_2$, $record_3,\ldots$, and $record_N$ respectively and M is the number of unique fingerprints, the detection algorithm will have to perform not more than $M \times (r_1+r_2+\ldots r_N)$ comparisons in every case, including the worst case scenario when no fingerprints are found in the dataset. Of course, the latter will never happen if Assumption I (Assigned fingerprints are present in $\tau \pm \epsilon$ pure data records) holds. It should also be noted that in the TC case, N will be the number of original records plus one fictitious record.

If we determine r as the maximum of $r_1,r_2,r_3,\ldots,r_N$, then the running time of the detection algorithm will never be more than $M \times r \times N$.

If, for example, the dataset contains 100,000 records, none of these records contain more than 500 characters and there are 50 recipients (the number of fingerprints), the detection algorithm will perform not more than 25×10^8 comparisons. That means that if a computer can perform 10 computations every microsecond, the detection algorithm would take 25×10^7 μs, i.e. $25 \times 10^7 \times 10^{-6} = 250$ seconds. If there 9 recipients, the detection algorithm would take 45 seconds. It must be emphasized though that detection

is only performed if a request comes in. As future improvement, however, the detection algorithm could be optimized.

4.5.5. Analysis: Requirements

The analysis of the tier 1 fingerprinting scheme is done only for the case when fingerprinted data is decoded at the trusted circle members' side. The reason for this is that encoded data has the same format (except for the modified primary key column or the added column) as data for the tier 2 fingerprinting scheme and it will be analyzed in the next section of this chapter.

It must be noted, however, that the presence of the modified primary key column or the added column in data does pose an additional threat to the scheme when an attack occurs between the encoder and decoder. If the attacker knows the TC member's or the distributor's private key, it could decrypt data and find out which records are marked. By analyzing marked records, the attacker might identify and delete similar records in the database. Although in this case the attacker may render data less useful, the real defence against such an attack should be providing adequate security to ensure that data transfer between the distributor and TC members is safeguarded and private keys are kept securely. This is, however, out of the scope of this work as it falls under security and system vulnerabilities.

Analysis of whether the tier 1 fingerprinting scheme satisfies the requirements defined in section 4.3 of this chapter:

Usefulness/Imperceptibility

Data has to be useful at all times and the encoding and decoding algorithms must be executable in reasonable running time. Imperceptibility means that modifications caused by the fingerprint do not reduce the usefulness of the data.

Members of the trusted circle have access to ACD, where only one record is fictitious. Since the decoder on the trusted circle members' side removes all but one

fictitious record, data is ACD and almost as useful as pure data. This modification is so minor that it does not reduce the usefulness of the data.

The running time of the detection algorithm is M x r x N where M is the number of unique fingerprints (number of trusted and distrusted circle members), r is the length of the longest record in the database and N is the number of (original and 1 fictitious) records in the database.

The encoder on the distributor's side also requires searching for records that contain the fingerprint, but only through original data and for only one fingerprint. Therefore, its running time is r x (N - 1). If, using the same example from section 4.5.4, the database contains 100,000 records and none of these records contain more than 500 characters, the insertion algorithm will perform not more than 49,999,500 comparisons. That means that if a computer can perform 10 computations every microsecond, the insertion algorithm would take 4,999,950 μs, i.e. 4.99 seconds. However, it should be noted that in practice for database modifications (as is the case with trusted members) not the whole database is needed only datasets that respond to queries and these datasets will definitely have less records than the whole database in itself.

Nevertheless, it would be interesting to see, as part of future research, how the insertion algorithm can be optimized.

The two extraction algorithms (decoder on the distributor's side, decoder on the TC member's side) require searches to the database only to determine which fictitious records are marked. That means that instead of searching all records, only the first column or primary key column needs to be searched. The running time of these two algorithms is $(N - 1 + \alpha\tau) \pm \epsilon$, where α is the number of repetitions. If $\tau = 10{,}000$ and $\alpha = 5$, and using the example above, the extraction algorithms will have to perform approximately $100{,}000 - 1 + 50{,}000 = 149{,}999$ comparisons which could take $14{,}999\mu$s i.e. 0.0149 seconds.

The second insertion algorithm (encoder on the TC member's side) does not involve any comparisons, just reattaching the marked records removed by the decoder on the TC member's side and thus, it requires even less time than the extraction algorithms.

It should be noted that the above estimates (performances of the insertion, extraction and detection algorithms) do not include time to access the database, which time can vary depending on the proximity of the database to the application. An IBM Redpaper ([10]) reports findings on a performance test conducted on local and remote database access, and concludes that it is a good practice to co-locate the application and the database because it is more efficient and eliminates network traffic. Since the workload in the described performance test involved issuing the SELECT SQL statement (the same statement my algorithms use to access the databases) one million times in a loop, the results reported in the IBM paper can be used to estimate database access time for my algorithms.[10] These results, based on LAN configuration, are as follows:

- It took 2 minutes 27 seconds to access the data locally (one million times in a loop, including database processing);
- It took 10 minutes 15 seconds to access data remotely (one million times in a loop, including database processing) – the database was located about one mile network distance from the application;
- Network latency was the major contributing factor for this substantial difference;
- The duration of each remote round trip was approximately 500 microseconds.

Using these results to estimate database access time, we can conclude that to execute the insertion, extraction and detection algorithms described here for a database of 100,000 records, 50 and 9 recipients, the following times would be needed:

[10] The proof of concept implementation (chapter 5) used one computer for all databases, the server and 10 clients, and the database was small (221 records) so the running time of the application and database access may not be a good indicator for estimating running and access time of the application and database access in general.

	PROCESS	ACCESS		TOTAL	
	Process (seconds)	Local (seconds)	Remote (seconds)	Local (seconds)	Remote (seconds)
Insertion	5	15	50	20	55
Extraction (Distributor)	0.0149	15	50	15.0149	50.0149
Extraction (Recipient)	0.0149	15	50	15.0149	50.0149
Detection (9 recipients)	45	135	450	180	495

TABLE 2: RUNNING TIME OF ALGORITHMS (TC MEMBERS)

As we can see, detection may take up to 495 seconds (8 minutes 15 seconds) but it must be emphasized that detection is only performed if a request comes in to determine who divulged data.

Although running time is not negligible for any of these algorithms, it is not at all impractical. If an organization considers unauthorized data disclosure protection as an important asset from a privacy or legal perspective, or simply, to gain trust of its customers, the benefits of implementing such a fingerprinting scheme would clearly outweigh the costs associated with increased running time.

Robustness

The fingerprinting scheme should be robust. In other words, it must be difficult (hopefully impossible) to destroy or overwrite the fingerprint i.e. the fingerprinting scheme has to be resistant to attacks that attempt to destroy or overwrite the fingerprint.

Attempts to destroy the fingerprint are definitely making the ACD less useful in this fingerprinting scheme. Unless the location of the only one fictitious record that is left in the data after the decoding is known, the probability of finding this record is very small (1/N). The larger the database (the more records the database has), the lower the probability is to locate the fictitious record.

The fingerprint deduction attack (a robustness attack) has already been discussed in section 4.4 (Analysis). As step 4. of this attack demonstrates, even if the fingerprint is guessed right, without knowing the exact location of the one remaining fictitious record, one would need to remove or modify all $\tau \pm \epsilon$ original records plus the one fictitious record to destroy the fingerprint, which would definitely render the data less useful.

Other robustness attacks are discussed in section 4.5.6.

Detectability

An authority has to be able to determine from the fingerprint who the owner is. This is called the high detection reliability and it means that the probability of detecting the fingerprint in the data when there is no fingerprint and the probability of a missed detection of the fingerprint when there is in fact a fingerprint in the data are extremely small.[26]

In this scheme, only the distributor is in possession of the fingerprints. If a request comes in to determine who the owner of a particular dataset is, the distributor applies the detection algorithm. It checks the fingerprints of every member and counts how many times each fingerprint appears in the dataset. The owner of the fingerprint with the highest count is most probably the member who disclosed the data.

Only one fictitious fingerprinted record exists in the ACD and if other records in the dataset are modified (due to the dynamic nature of databases) such that the number of original records containing the fingerprint of other members increase by chance, these members will be falsely accused by the detection algorithm as they will have more fingerprinted records than the member who actually disclosed the data.

For example if there are N records in the database, if there are two members and if the fingerprints of both members appear 10 times in the database, the added fictitious record will identify which member disclosed the database. However, if the database

is modified such that it now contains N+3 records and one of the added records accidentally contains the fingerprint of member 2, then member 2 will have its fingerprint appear 11 times in the database while the fingerprint of member 1 will appear only 10 times. In this case, if the data is requested by member 1, the added fictitious record will increase the number of fingerprinted records of member 1 to 11 while the number of fingerprinted records for member 2 will remain 11. Alternatively, if the data is requested by member 2, the number of fingerprinted records of member 2 will increase to 12 while the number of fingerprinted records of member 1 will remain 10. Therefore, the output of the detection algorithm will never be member 1 even if member 1 disclosed the data.

This problem can be solved by introducing longer fingerprints so the probability of increasing the number of records that contain the fingerprints of other trusted circle members caused by data modification becomes much lower.

Longer fingerprints yield higher detection reliability. However, caution must be exercised with regards to the meaning of the fingerprinted values of quasi-identifiers, and with regards to the fact that longer fingerprints increase detection time. One cannot insert a really long fingerprint in categorical values without making categorical values useless. It is very important to keep fingerprinted categorical values logical so they do not stand out of the data.

Incremental Updatability

Incremental updatability means that updating of records is possible without modifying the fingerprint.

Only those members of the trusted circle can update records that have full access to the ACD. ACD contains only one fictitious record so fingerprint modification when updating the records can occur with a very low probability (1/N).

Blindness

The fingerprinting scheme has to be blind, which means that for detecting the fingerprint there is no need to access the original data.

This fingerprinting scheme is blind because for detecting the fingerprint only the fingerprint is required.

Capacity

In the proposed fingerprinting scheme, capacity is the number of fictitious records that can be added to the database without degrading its quality (i.e. not increasing its size such that it becomes impractical, especially for performing the detection and insertion algorithms) and the number and length of fingerprints that can be detected correctly by the detection algorithm (the detectability requirement suggests longer fingerprints). The more fictitious records containing the fingerprint are inserted in the data, the higher the probability is to detect the fingerprint. But, at the same time, the more fictitious records are inserted, the size of the database increases.

In the Tier 1 scheme, only one fictitious record exists in the database. Although Proof of Concept results suggest that this might be sufficient for detection, more tests, especially on real data, are required to determine whether this is always the case. More tests would also show what the required length of the fingerprint should be to increase detectability.

4.5.6. Analysis: Attacks

1. **Robustness attacks - attempts to destroy or remove the fingerprint:**

   Fingerprint deduction attack (inversion or invertibility attack)

   Already discussed in section 4.4.

   Benign attacks (Presentation attacks)

   Benign attacks are attacks where the fingerprint is accidentally destroyed. Presentation attacks are attacks when the content is modified such that the detector cannot detect the fingerprint. In both cases, though, the attacks should render the data useless.

   Since there are $(\tau+1) \pm \epsilon$ fingerprinted records in the ACD, from which only one record is fictitious, accidentally destroying the fictitious record out of all records containing the fingerprint can occur with a probability of $1/(\tau+1)$. So, the more original records contain the fingerprint, the lower the probability is to accidentally destroy the fictitious record containing the fingerprint (capacity property).

   If the fingerprint is destroyed accidentally, a subset of the original $\tau \pm \epsilon$ records might also be destroyed, which means less useful data.

   Mix and match attacks

   In mix and match attacks, different datasets from the same source can be compared to discover fingerprinted records.

   In this attack, a member of the trusted circle would request data over and over to find out, which records are fictitious. It is important, therefore, that the same fictitious record is left always unmarked by the encoder so it looks like it is part of the data (as already noted in section 4.1).

   Since mix and match attacks are a special case of collusion attacks, see the general analysis provided under collusion attacks below.

Collusion attacks

When recipients compare/share their data (in other words, when mix and match attacks involve more than one recipient), we call it a collusion attack (data matching). This attack is the most challenging attack for fingerprinting schemes.

As described in section 3.3, Boneh and Shaw ([6]) in Collusion-Secure Fingerprinting for Digital Data show that if the distributor prior to embedding the fingerprints randomly picks a permutation (that is being kept secret) from the full symmetric group of all permutations on p letters, and if that same permutation is used for all users, it is possible to create a function, the actual fingerprinting scheme (1,n), which 1) maps a user number $1 \leqslant u \leqslant n$ and the permutation to a fingerprint, and 2) is c-secure with ϵ error. c-secure with ϵ error means that the described fingerprinting scheme enables to capture a member of the coalition with probability at least 1- ϵ, for any $1 > \epsilon > 0$. Also, they provide the length of such a fingerprint as $O(c^4 \log n \log (1/\epsilon))$.

Therefore, if we want to generate collusion free fingerprints, we would have to have fingerprints of length $O(n^4 \log n \log (1/\epsilon))$ which in the case of 50 members would yield fingerprints of length approximately 10,618,563 characters. This is of course, impractical (although it would also satisfy the detectability requirement) since we have to take into account the length of records in the database.

We can also apply Boneh and Shaw's result on the number of fingerprinted records instead of the fingerprint itself. This application would make sense as in the proposed fingerprinting scheme we use not one fingerprint but many fingerprints embedded in many records where most of them appear already in the original database. In addition, in the case of a collusion attack, recipients of the database would compare their data in attempt to identify and remove records that do not appear in the intersection of all their records by assuming that these records are actually the fictitious records.

So, to apply this result to the proposed fingerprinting scheme, we could have a secret permutation of p fictitious records used for all members and $(n^4 \log n \log (1/\epsilon) - p)$ unique fictitious records containing only the fingerprint of a particular member.

In the case of n = 50 members, each member would have approximately $n^4 \log n \log (1/\epsilon) = 10{,}618{,}563$ fictitious records containing its fingerprint, from which p records would be the same and include the fingerprints of all members, and the rest would include the fingerprint of a particular member. This solution would be collusion secure but would not yield ACD. In fact, if the total number of records would be 100,000, the addition of over 10 million fictitious records would definitely render data useless.

However, as Figure 10 demonstrates, in the case of $n = 10$ members, each member would have approximately 10,000 (10% of 100,000) fictitious records for $\epsilon = 0.1$; 3,010 (3%) for $\epsilon = 0.5$; and 458 (0.4%) for $\epsilon = 0.9$. In the case of $n = 9$ members, the number of fictitious records required is even smaller: 6,261 (6.26% of 100,000) for $\epsilon = 0.1$; 1,885 (1.8%) for $\epsilon = 0.5$; and 286 (0.2%) for $\epsilon = 0.9$. In these cases, data would not be useless since in the worst case, fictitious records make up 10% of all data. Of course, ACD is not achieved but the organization could make a decision regarding how much trust should it give to its trusted members: it can be assumed that trusted members will not be involved in collusion attacks, so resistance to collusion attacks would not be needed; if, however, trusted members are not fully trusted, they will become distrusted and therefore, could not receive ACD.

If the secret permutation of p records forms part of the original records (containing the fingerprints of all members) and the other $(n^4 \log n \log (1/\epsilon) - p)$ original records containing the fingerprint of a particular member are disclosed to only that particular member, we could achieve ACD but would not be able to make the complete database available to members. Withholding data would therefore enable collusion secure fingerprinting.

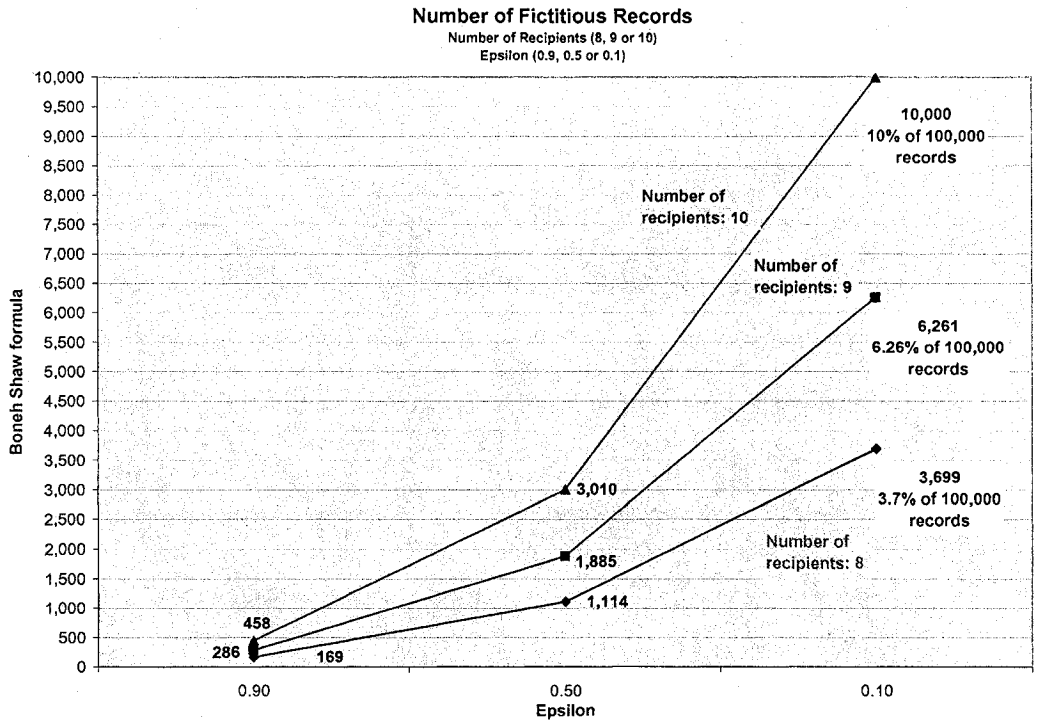

FIGURE 10: NUMBER OF FICTITIOUS RECORDS

Alternative ways of reducing but not eliminating collusion attacks would be to never disclose the same data to members, or apply other disclosure limitation techniques such as perturbation prior to fingerprinting the data.

It is therefore clear that a trade-off is required between resistance to robustness attacks and capacity and usefulness. If resistance against collusion attacks is required, then the approach proposed here should be used with a limited number of recipients (e.g. to 10 recipients for a database of size 100,000 records).

2. Interpretation attack (Additive attack)

An interpretation attack means that the attacker devises a situation in which the assertion of ownership is prevented. This could happen by inserting another entity's fingerprint in the data. However, for this attack, the attacker should know another entity's fingerprint and should produce fictitious records containing this fingerprint.

To learn someone else's fingerprint, the attacker either must break the security around the distributor and access the list of fingerprints (as noted previously, system security is assumed throughout this work), or must deduce someone else's fingerprint from an available database. This attack is similar to the fingerprint deduction attack discussed in section 4.4 but is much more difficult as someone else's fingerprint is present in a less frequent manner in the database than the entity's own fingerprint. For this attack to realize, the attacker would again have to generate all possible variations of "reasonable length" (let's say variations of length 3,4,...,15) using, for example, the English alphabet (26 letters) and search for them in the database using the detection algorithm.

If N is the number of records in the database, $r_1,r_2,r_3,\ldots,r_N$ are the lengths of $record_1$, $record_2$, $record_3,\ldots$, and $record_N$ respectively, and M is the number of unique fingerprints of TC and DC members, the detection algorithm will have to perform $M \times (r_1+r_2+\ldots r_N)$ searches in every case. Since the attacker does not know which variations to search for, he will have to search for all variations i.e. for $M = (26^3+26^4+\ldots+26^{15})$. If N = 100,000 and $r_1 = r_2 = r_3 =\ldots= r_N= 500$, it would take the attacker 85×10^{27} comparisons (compared to the 25×10^8 comparisons needed by the detection algorithm) to determine the frequency of each variation in the database. Contrary to the detection algorithm, determining the frequency of each variation in a fingerprint deduction and interpretation attack will not mean the end of the process since the attacker does not know which variations are actually fingerprints.

The attacker could assume that variations appearing approximately in equal numbers in the database are someone else's fingerprint and try to create and add fictitious records containing these assumed fingerprints.

In addition, to make this attack really powerful, the attacker should also remove the fictitious record(s) containing his own fingerprint which again should be unknown for the attacker if security has not been breached.

3. Synchronization attacks (Subset attacks)

Vertical data mining attacks

Vertical partition [52] occurs when each dataset is split across multiple sites, with each site having a different set of attributes. The relations at the individual sites must be joined to get the relation to be mined.

Therefore, one way to survive a vertically partitioning attack, or any attack that involves only a few attributes instead of the whole attribute set, is that the fingerprint is inserted in attributes as well as records. The database would then include fictitious attribute(s) in addition to fictitious records that contain the fingerprint.

To enable fast decoding, an additional row can be inserted to mark fictitious attributes the same way as the additional column is inserted (or the primary key column is modified) to indicate fictitious records. Decoders would remove these fictitious attributes.

Horizontal data mining attacks

Horizontal partition [29] means that each dataset is split across multiple sites, with each site having the same attributes but different records.

Therefore, one way to survive a horizontally partitioning attack, or any attack that involves only a few records instead of the whole set of records, is that the fingerprint is inserted in more than one record, and in such a way, that fingerprinted records are spread around in the database. This strategy, however, challenges the capacity property of fingerprinting schemes. In other words, the more fingerprinted records are embedded in the data, the less useful the data becomes.

In the proposed tier 1 scheme, there are $(\tau+1) \pm \epsilon$ fingerprinted records in the ACD. It depends on the original distribution of these records, whether a subset of the ACD is resistant to the horizontally partitioning attack.

4.6. Tier 2: Distrusted Circle of Database Sharers

The previous section discussed the TC of database sharers. The members of this circle are determined by the distributor, based on some criteria for trustworthiness.

In some applications (for example, the SEARS Canada case as described in section 4.1 of this chapter), it might be useful to have a distrusted circle where members of the circle would receive data that is always fully fingerprinted. This data can still be useful but it contains more fingerprinted records than data disclosed to TC members.

Members of the distrusted circle of database sharers do not have a decoder and an encoder, which means that they receive the fully fingerprinted data and are not able to extract the fingerprint. They have only limited (read but not write) access to the data so they cannot send data back to the distributor.

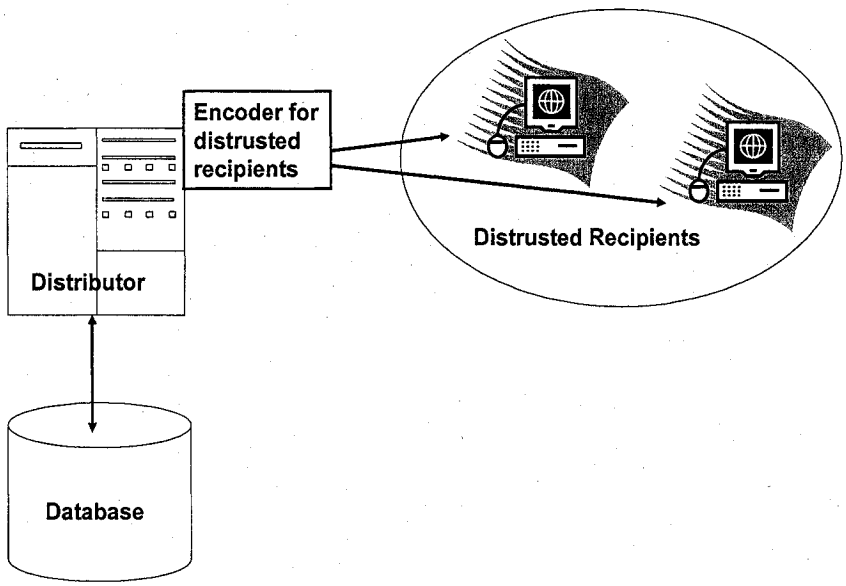

FIGURE 11: FINGERPRINTING SCHEME – TIER 2

It can be observed in the above figure that after data has been encoded, it is sent to distrusted circle members and that attacks could occur any time on the encoded data. Therefore, the analysis of this scheme includes analysis for both unauthorized sharing of data by distrusted circle members and unauthorized access to encoded data in general.

Given that we cannot distinguish between unauthorized sharing of data by distrusted circle members and unauthorized access to encoded data in general; that data is always fingerprinted after it leaves the encoder on the Distributor's side; and that data contains the fingerprint of a trusted or distrusted circle member who requested the data, four possible attacks could happen:

1. The trusted or distrusted circle member who requested the data is the entity who illegally disclosed the data. Performed detection determines that the trusted or distrusted circle member's fingerprint is the most frequently present fingerprint in the dataset and he is rightly accused of illegal data disclosure.

2. An outside entity attacked the scheme and stole the fingerprinted data. If the attack involved changing/replacing the existing fingerprint, then it is discussed under benign, invertibility or other attacks. If there was just data theft, then the information security aspect of the scheme has been breached, which is out of the scope of this work (see also section 4.5.5). By all means, security has to be always present to guard against data theft, whether it is the fingerprinted data itself or information about the fingerprints.

3. A trusted or distrusted circle member discovered the fingerprint of another entity and is inserting this fingerprint to mask his own entity and be able to illegally disclose data. This attack is discussed under the invertibility attack.

4. The distributor is malicious and wants a trusted or distrusted circle member to be falsely accused of illegal disclosure. This is a disadvantage of this symmetric scheme. Future work should include research on anonymous fingerprinting of databases.

There is a finite number of distrusted circle members. Thus, the distributor can determine a finite number of fingerprints that uniquely identify distrusted recipients. Fingerprint generation is exactly the same as in the case of trusted circle database sharers. The same two assumptions exist:

Assumption I. Assigned fingerprints are present in $\tau \pm \epsilon$ pure data records.

Assumption II. Fingerprints assigned to different users are not contained in exactly the same records of the original database.

This fingerprinting scheme contains only one insertion algorithm, the encoder on the distributor's side.

The following notations are used:

DC – distrusted circle of database sharers;

D – size of DC; i.e. the total number of DC members;

$d_1, d_2, \ldots, d_D$ – DC members;

D is therefore the number of unique fingerprints as a unique fingerprint is assigned to each DC member;

$g_1, g_2, \ldots, g_D$ – fingerprints of DC members, where g_1 is the unique fingerprint of d_1, g_2 is the unique fingerprint of d_2, …, g_D is the unique fingerprint of d_D; and

$g_i \neq f_j$ for $\forall i \in \{1,\ldots,M\} \wedge \forall j \in \{1,\ldots,D\}$

L is the length of each fingerprint; every fingerprint has the same length.

4.6.1. Fingerprint

The fingerprints of DC members are created exactly the same way as fingerprints for TC members. It must be emphasized though that fingerprints are always unique for all TC and DC members.

4.6.2. Encoder

The difference between this insertion algorithm (encoder on the distributor's side) and the encoder on the distributor's side for TC members is that this algorithm does not include any record marking nor encryption. This algorithm is comprised of the following steps:

1. Select $\tau \pm \epsilon$ records that already contain the fingerprint.

2. Modify the copies of chosen records in such a way that existing identifiers and quasi-identifiers are replaced by fictitious values chosen from a pre-determined domain set. The domain set is created using the distribution of the data in the database.

3. Insert modified copies in the database. The database will contain $2\tau \pm \epsilon$ fingerprinted records.

4. Steps 2. and 3. can be repeated several times. The only requirements are that each copy contains the fingerprint and that the identifiers and quasi-identifiers differ in each copy. With each repetition, the number of fingerprinted records will increase by $\tau \pm \epsilon$.

After the insertion algorithm is applied, the total number of fingerprinted records will be $(\alpha+1)\tau \pm \epsilon$, where α is the number of repetitions (step 4. of the algorithm).

This algorithm differs from the encoder on the distributor's side for the TC of database sharers as it doesn't contain any encryption. Encryption is not needed because there is no decoder on the recipients' side. Fingerprinted records are not marked and they all form part of the database.

Pseudocode:

```
/*Original is the database to be fingerprinted*/
/*Mock is a database where fictitious values are chosen from*/
/*theRecord is a record from Original*/
/*theFingerprint is the fingerprint of the DC member*/
/*Data is the data to be sent to the DC member*/

/*the following code is repeated for all records in Original*/
τ = 0
/* τ is the number of records containing the fingerprint*/
theRecord = select all records one by one from Original
```

```
        if the fingerprint is found in theRecord
                increase τ
                insert theRecord in Data
                modify theRecord
                insert the modified theRecord in Data
        end
        else
                add "n" to theRecord
                insert theRecord in Data
        end
end
found function (returns a boolean)
        result = false
        i,j = 0
        while (i < length of the fingerprint AND  j < length of the record)
                if (theFingerprint.charAt(i) == theRecord.charAt(j))
                        i++
                j++
        end
        if (i == length of the fingerprint)
                        result = true
        return result

modify function (returns a modified record)
        select a record from Mock
        addIdentifiers to theRecord
return the modified theRecord

addIdentifiers function (returns identifiers)
```

/*if a fingerprint character is not found between two commas of theRecord, select between 1) deleting everything between the two commas, and 2) replacing everything between the two commas with a general identifier from Mock*/
/*select among 1) not adding anything, 2) adding before identifiers a general identifier from Mock, 3) adding after identifiers a general identifier from Mock, and 4) adding before and after identifiers a general identifier from Mock*/

4.6.3. Decoder

This fingerprinting scheme does not contain any extraction algorithm.

4.6.4. Detection

The detection algorithm is exactly the same as the detection algorithm for the tier 1 scheme (section 4.5.4).

Detection time depends on the number of fingerprints, number of records in the dataset and the length of the records in the dataset. If N is the number of records in the database, $r_1,r_2,r_3,\ldots,r_N$ are the lengths of $record_1$, $record_2$, $record_3,\ldots$, and $record_N$ respectively, and M is the number of unique fingerprints of TC and DC members, the detection algorithm will have to perform $M \times (r_1+r_2+\ldots r_N)$ searches in every case, including the worst scenario when none of the fingerprints is found in the dataset. Of course, the latter will never happen if Assumption I (Assigned fingerprints are present in $\tau \pm \epsilon$ pure data records) holds.

If r is the maximum of $r_1,r_2,r_3,\ldots,r_N$, then the running time of the detection algorithm will never be more than $M \times r \times N$.

If, for example, the dataset contains 100,000 records, none of these records contain more than 500 characters and there are 50 recipients (the number of fingerprints), the detection algorithm will perform not more than 25×10^8 comparisons. That means that if a computer can perform 10 computations every microsecond, the detection algorithm would take 25×10^7 μs, i.e. $25 \times 10^7 \times 10^{-6} = 250$ seconds. If there 9 recipients, the detection algorithm would take 45 seconds. It must be emphasized though that detection

is only performed if a request comes in. As future improvement, however, the detection algorithm could be optimized.

4.6.5. Analysis

The analysis of this scheme includes analysis for both unauthorized sharing of data by distrusted circle members and unauthorized access to data in general.

Analysis of whether the tier 2 fingerprinting scheme satisfies the requirements defined in section 4.3 of this chapter:

Usefulness/Imperceptibility

Data has to be useful at all times and the encoding and decoding algorithms must be executable in reasonable running time. Imperceptibility means that modifications caused by the fingerprint do not reduce the usefulness of the data.

Since the fully fingerprinted database contains $(\alpha+1)\tau \pm \epsilon$ fingerprinted records, from which $\tau \pm \epsilon$ are original records (Assumption I), this fingerprinted data is less useful than the fingerprinted data in the tier 1 scheme. However, this is acceptable as recipients of this fingerprinted data are not trusted. Also, with the decrease of α, the usefulness of the data increases. The distributor can decide which level of "trustworthiness" it assigns to its distrusted circle, and this decision very much depends on the system, environment and circumstances.

"Trustworthiness" could be for example determined as follows:

(a) If there is high risk that a distrusted circle member could share the data, "trustworthiness" could be low, with $\alpha \geq 5$;
(b) If there is a medium risk that a distrusted circle member could share the data, "trustworthiness" could be medium, with $2 < \alpha < 5$;
(c) If there is a low risk that a distrusted circle member could share the data, "trustworthiness" could be high, with $0.5 < \alpha \leq 2$;

Note that if $\alpha < 1$, only a subset of the original $\tau \pm \epsilon$ records containing the fingerprint is modified and inserted into the database.

The running time of the detection algorithm is M x r x N where M is the number of unique fingerprints (number of trusted and distrusted circle members), r is the length of the longest record in the database and N is the number of (original and fictitious) records in the database.

The encoder on the distributor's side also requires searching for records that contain the fingerprint, but only through original data and for only one fingerprint. Therefore, its running time is r x (N - ($\alpha\tau \pm \epsilon$)) where $\alpha\tau \pm \epsilon$ is the number of fictitious records. If the database contains 150,000 records and none of these records contain more than 500 characters, τ = 10,000 and α = 5, the insertion algorithm will perform approximately 50,000,000 comparisons. That means that if a computer can perform 10 computations every microsecond, the insertion algorithm would take 5,000,000 μs, i.e. 5 seconds. It would be interesting to see, as part of future research, how the insertion algorithm can be optimized.

There are no other insertion or extraction algorithms for the tier 2 scheme.

Using results described in section 4.5.5 to estimate database access time, we can conclude that to execute the insertion and detection algorithms described here for a database of 100,000 records, 50 and 9 recipients, the following time would be needed:

	PROCESS	ACCESS		TOTAL	
	Process (seconds)	Local (seconds)	Remote (seconds)	Local (seconds)	Remote (seconds)
Insertion	5	15	50	20	55
Detection (9 recipients)	45	135	450	180	495

TABLE 3: RUNNING TIME OF ALGORITHMS (DC MEMBERS)

As we can see, detection may take up to 495 seconds (8 minutes 15 seconds) but it must be emphasized that detection is only performed if a request comes in to determine who divulged data.

Although running time is not negligible for any of these algorithms, it is not at all impractical. If an organization considers unauthorized data disclosure protection as an important asset from a privacy or legal perspective, or simply, to gain trust of its customers, the benefits of implementing such a fingerprinting scheme would clearly outweigh the costs associated with increased running time.

Robustness

The fingerprinting scheme should be robust. In other words, it must be difficult (hopefully impossible) to destroy or overwrite the fingerprint i.e. the fingerprinting scheme has to be resistant to attacks that attempt to destroy or overwrite the fingerprint.

Unless the locations of the fictitious records are known, the probability of finding these records is small $((\alpha\tau \pm \epsilon)$ /N where N is the number of records in the database). The larger the database (the more records the database has) and the smaller the α, the lower the probability is to locate the fictitious records.

The fingerprint deduction attack has already been discussed in section 4.4 (Analysis). As step 4. of this attack demonstrates, even if the fingerprint is guessed right, without knowing the exact locations of the fictitious records, one would have to remove or modify all $\tau \pm \epsilon$ original records plus all the $\alpha\tau \pm \epsilon$ fictitious records to destroy the fingerprint, which would definitely render the data less useful. The greater the α, the higher the number of fictitious records is, and the more records have to be removed or modified to destroy the fingerprint.

Although this scheme is less robust than the tier 1 scheme, it is still robust for the right value of α. As discussed under Usefulness/Imperceptibility, "trustworthiness" (α) depends on the system, environment and circumstances.

Other robustness attacks are discussed in section 4.5.6

Detectability

In this scheme, only the distributor is in possession of the fingerprints. If a request comes in to determine who the owner of a particular dataset is, the distributor applies the detection algorithm. It checks the fingerprints of every recipient (members of both the trusted and distrusted circle) and counts how many times each fingerprint appears in the dataset. The owner of the fingerprint with the highest count is most probably the member who disclosed the data.

Since this scheme is less robust than the tier 1 scheme, the probability of detecting the fingerprint is higher in this scheme than the probability of detecting the fingerprint in the tier 1 scheme. However, at the same time, it is easier for the detection algorithm to determine which recipient has it's fingerprint appearing the most frequently in the database, thus indicating with high probability who disclosed the data.

This conclusion is exactly the conclusion we wanted. Since one of the criteria determining distrusted and trusted circle members is that the likelihood of DC members disclosing confidential data is greater than that of TC members, it is anticipated that detection will be more frequently required for data that has been disclosed by DC members. Frequent requirements will be perfectly supported by the higher detection reliability requirement of the tier 2 scheme.

Incremental Updatability

This property is not applicable for the tier 2 scheme because DC members are not allowed to modify the data.

The distributor has access to the pure data and can modify it. The encoder only inserts the fingerprints after the data is requested. Therefore, the distributor can always update records without modifying the fingerprints.

Blindness

The fingerprinting scheme has to be blind, which means that for detecting the fingerprint there is no need to access the original data.

This fingerprinting scheme is blind because for detecting the fingerprint only the fingerprint is required.

Capacity

This fingerprinting scheme is less robust than the tier 1 scheme, and the quality of the data is lower than the quality of the data in the tier 1 scheme. However, the probability of detecting the fingerprint is higher in this scheme than in the tier 1 scheme.

4.6.6. Attacks

1. **Robustness attacks - attempts to destroy or remove the fingerprint:**

   Fingerprint deduction attack (inversion or invertibility attack)

   Already discussed in section 4.4.

   Benign attacks (Presentation attacks)

   Benign attacks are attacks where the fingerprint is accidentally destroyed.

   Presentation attacks are attacks when the content is modified such that the detector cannot detect the fingerprint. In both cases, though, the attacks should render the data useless.

   Since there are $(\alpha+1)\tau \pm \epsilon$ fingerprinted records in the data, from which $\alpha\tau \pm \epsilon$ records are fictitious, accidentally destroying the fictitious record out of all records

containing the fingerprint can occur with a probability of $\alpha/(\alpha+1) = 1/(1+1/\alpha)$. So, for smaller α the probability to destroy the fictitious records is lower (capacity property).

If the fingerprint is destroyed accidentally, a subset of the original $\tau \pm \epsilon$ records might also be destroyed, which means less useful data.

Mix and match attacks

In mix and match attacks, different datasets from the same source can be compared to discover fingerprinted records.

In this attack, one member of the DC would request the same data over and over to find out, which records are fictitious. It is important, therefore, that the encoder always inserts the same fictitious records so they look like they are part of the data (as already noted in section 4.1).

Since mix and match attacks are a special case of collusion attacks, see the general analysis provided under collusion attacks below.

Collusion attacks

When recipients compare/share their data (in other words, when mix and match attacks involve more than one recipient), we call it a collusion attack (data matching). This attack is the most challenging attack for fingerprinting schemes.

As described in section 3.3, Boneh and Shaw ([6]) in Collusion-Secure Fingerprinting for Digital Data show that if the distributor prior to embedding the fingerprints randomly picks a permutation (that is being kept secret) from the full symmetric group of all permutations on p letters, and if that same permutation is used for all users, it is possible to create a function, the actual fingerprinting scheme (l,n), which 1) maps a user number $1 \leqslant u \leqslant n$ and the permutation to a fingerprint, and 2) is c-secure with ϵ error. c-secure with ϵ error means that the described fingerprinting scheme enables to capture a member of the coalition with probability at least $1-\epsilon$, for

any $1 > \epsilon > 0$. Also, they provide the length of such a fingerprint as $O(c^4 \log n \log (1/\epsilon))$.

Therefore, if we want to generate collusion free fingerprints, we would have to have fingerprints of length $O(n^4 \log n \log (1/\epsilon))$ which in the case of 50 members would yield fingerprints of length approximately 10,618,563 characters. This is of course, impractical (although it would also satisfy the detectability requirement) since we have to take into account the length of records in the database which can be only 500 characters.

We can also apply Boneh and Shaw's result on the number of fingerprinted records instead of the fingerprint itself. This application would make sense as in the proposed fingerprinting scheme we use not one fingerprint but many fingerprints embedded in many records where most of them appear already in the original database. In addition, in the case of a collusion attack, recipients of the database would compare their data in attempt to identify and remove records that do not appear in the intersection of all their records by assuming that these records are actually the fictitious records.
Figure 10 in section 4.5.6 is also applicable to DC members as it illustrates a general situation.

If the secret permutation of p records forms part of the original records (containing the fingerprints of all members) and the other $(n^4 \log n \log (1/\epsilon) - p)$ original records containing the fingerprint of a particular member are disclosed to only that particular member, we could achieve more useful data but would not be able to make the complete database available to members. Withholding data would therefore enable collusion secure fingerprinting. For 50 members we would still have to have over 10 million records in the database but no fictitious records would be required.

Alternative ways of reducing but not eliminating collusion attacks would be to never disclose the same data to members, or apply other disclosure limitation techniques such as perturbation prior to fingerprinting the data.

It is therefore clear that a trade-off is required between resistance to robustness attacks and capacity and usefulness.

2. Interpretation attack (Additive attack)

An interpretation attack means that the attacker devises a situation in which the assertion of ownership is prevented. This could happen by inserting another entity's fingerprint in the data. However, for this attack, the attacker should know another entity's fingerprint and should produce fictitious records containing this fingerprint.

To learn someone else's fingerprint, the attacker either must break the security around the distributor and access the list of fingerprints (as noted previously, system security is assumed throughout this work), or must deduce someone else's fingerprint from an available database. This attack is similar to the fingerprint deduction attack discussed in section 4.4 but is much more difficult as someone else's fingerprint is present in a less frequent manner in the database than the entity's own fingerprint. For this attack to realize, the attacker would again have to generate all possible variations of "reasonable length" using, for example, the English alphabet and search for them in the database. Then, the attacker could assume that variations appearing approximately in equal numbers in the database are someone else's fingerprint and try to create and add fictitious records containing these assumed fingerprints.

In addition, to make this attack really powerful, the attacker should also remove the fictitious record(s) containing his own fingerprint.

This attack has been discussed in more detail in section 4.5.6.

3. Synchronization attacks (Subset attacks)

Vertical data mining attacks

Vertical partition [52] occurs when each dataset is split across multiple sites, with each site having a different set of attributes. The relations at the individual sites must

be joined to get the relation to be mined. Therefore, to survive a vertically partitioning attack, or any attack that involves only a few attributes instead of the whole attribute set, the fingerprint should be inserted in attributes as well as records. In other words, the database should include fictitious attribute(s) in addition to fictitious records that contain the fingerprint.

Therefore, one way to survive a vertically partitioning attack, or any attack that involves only a few attributes instead of the whole attribute set, is that the fingerprint is inserted in attributes as well as records. The database would then include fictitious attribute(s) in addition to fictitious records that contain the fingerprint.

Horizontal data mining attacks

Horizontal partition [29] means that each dataset is split across multiple sites, with each site having the same attributes but different records. Therefore, to survive a horizontally partitioning attack, or any attack that involves only a few records instead of the whole set of records, the fingerprint should be inserted in more than one record, and in such a way, that fingerprinted records are spread around in the database. This strategy, however, challenges the capacity property of fingerprinting schemes. In other words, the more fingerprinted records are embedded in the data, the less useful the data becomes.

Therefore, one way to survive a horizontally partitioning attack, or any attack that involves only a few records instead of the whole set of records, is that the fingerprint is inserted in more than one record, and in such a way, that fingerprinted records are spread around in the database.

In the proposed tier 2 scheme, there are $(\alpha+1)\tau \pm \epsilon$ fingerprinted records in the data. Fictitious records should be inserted into the database in a distributed way so that most (if not all) subsets of the data are resistant to the horizontally partitioning attack.

5. Implementation

The proposed two-tier fingerprinting scheme has been implemented to obtain proof of concept results in order to demonstrate that it can contribute to unauthorized disclosure prevention of confidential data. As proof of concept results show, fingerprint insertion and detection in a database is possible.

5.1. Set-up

The proof of concept implementation includes a database, an encoder, a decoder and a detection algorithm. The database (Patient Data) has been created in Microsoft Access 1997 and contains 221 synthetic records with confidential data (patient name, health card number, gender, birth date and symptoms). In this implementation, gender, birth date and symptoms are considered quasi-identifiers. Although patient name can also be a quasi-identifier as different persons can have the same name, it is not considered for fingerprint insertion as it is assumed that only anonymized data is disclosed to distrusted circle members. Anonymization in this implementation means removing patient name and health card number.

This assumption is fully supported by applicable legislation and relevant policies, which state that personal information must never be disclosed without consent. Patient name, accompanied with birth date and symptoms, definitely identifies an individual, and is personal information. Hospitals might in some cases inform patients upon admission that their data could be used for secondary purposes, but this possibility does not add any significance to this work. The fingerprint insertion algorithm can insert fingerprints in any attribute, and it is up to a specific application to determine which attributes to consider.

The encoder, decoder and detection algorithms are all written in Java™ 2 SDK, Standard Edition 1.4.2-04. The Java Database Connectivity Package (JDBC) is used to connect the Java code to the MS Access Driver in the 32 bit Open Database Connectivity (ODBC) Data Sources. Interaction with the database is done by using the database language, the Structured Query Language (SQL).

The set-up includes the distributor, which is the only entity accessing the database, the TC members (six members in total) and the DC members (four members in total).

The TC members are assumed to be the six different subspecialty units of Civic Hospital, more specifically in vascular (CH1), thoracic (CH2), neurosurgical (CH3), obstetrical (CH4), acute (CH5) and chronic pain (CH6) management. Since they all belong to Civic Hospital, they are all trusted database sharers.

The DC members for this proof of concept implementation are Researcher 1, Researcher 2, Statistician 1 and Statistician 2. For the sake of an example, Researcher 1 is the University of Ottawa, Researcher 2 is EKOS Research Associates Inc., Statistician 1 is the InKine Pharmaceutical Company Inc., and Statistician 2 is the Canadian Institute for Health Information (CIHI).

The distributor creates a separate database (User Data) containing information on database sharers (Name, Identifier, User Name, Password, Trustworthiness, and the Public Key in the case of TC members). This database is used for authenticating clients when they initiate communication with the distributor's server. When the client enters his user name and password, the server is able to determine whether the client is a trusted or distrusted circle member, or an unknown entity. The public keys of trusted circle members are kept in the User Data to enable encryption of the fingerprinted data before sending them to the trusted circle members as described in the encoder algorithm in chapter 4.3 of this work.

The distributor also creates a database (Mock Patients) that contains the domain set. This domain set contains fictitious data that is used for substituting identifiers and quasi-identifiers of a record containing the fingerprint.

This proof of concept implementation shows that by carefully analyzing the database it is possible to determine the necessary number of characters (length of the fingerprint) and the characters themselves (the fingerprint itself) in such a way that the same set of

characters already appears in the same order (not necessarily one after each other) in 5 records (τ = 5 has been arbitrarily chosen for this proof of concept) of the quasi-identifiers of pure data.

The first character is chosen from {S,T}[11], the second, third and fourth characters are chosen from {A,B,C,D,E,I,L,M,N,R}, and the fifth character is chosen from {A,B,C,D,E,I,L,M,N,R,O,P,S,T,G,K,U,F,X,Y,H,J,V,W}. These sets have been determined by analyzing the 221 records of the Patients database. More specifically, characters S and T appear more frequently in the data than other characters so one of them is always part of the fingerprint. To limit the number of all possible fingerprints, the size of the second set has been limited to 10 and the first ten characters from the English alphabet except F,G,H,J,K,O,P,Q have been chosen as elements of this set. F,G,H,J,K,O,P,U,V,W,X,Y are the least frequent characters in the data but they are still included in the third set. In fact, variations of length 5 that end with X, J, Y, and W are the least common of all variations so the actual fingerprints are chosen from these variations to limit the number of original records, that contain the fingerprint, to 5. Characters Q and Z appear only seldom in the data so they are not considered at all.

With this approach, 48,000 distinct fingerprints can be created (2x10x10x10x24). Most of the time, the distributor would not want to distribute its database to 48,000 recipients; therefore, it is possible to pick those fingerprints that are present in 5 records. However, to limit the number of records (that contain the fingerprint) to 5, the content of the last set is reduced to contain {X,J,Y,W}. This way, the total number of distinct fingerprints is reduced to 8,000 (2x10x10x10x4). Assignment of fingerprints is a one time event, when TC and DC members are determined. Of course, as discussed in section 4.1, replacing fingerprints may happen given the dynamic nature of databases. Databases are modified on a constant basis and if at one point the two assumptions (see section 4.5.1) do not hold any more, the distributor should assign new fingerprints to TC and DC members as necessary. New fingerprints should be distinct from not only each other but from

[11] A polynomial of degree 6 is used to map members' name and number into a fingerprint.

previously assigned fingerprints as well. Therefore, the distributor has to keep all past and present fingerprints to enable eventual detection.

So when a recipient is assigned a fingerprint, it is safe to assume that this fingerprint is already present in 5 records, which in our case, is less than 2.3% of the pure data. The second assumption we need to make is that the selected fingerprints not overlap so it is not the same τ records that contain all the fingerprints. In other words, the cardinality of the intersection of all records that contain the fingerprints is always less than 5, and the cardinality of the union of all records that contain the fingerprints is always more than 5. Again, this is not difficult to achieve, given the large number of possibilities to select from (if necessary, the length of the fingerprint and the characters of the fingerprint can be modified).

After a recipient is authenticated, the distributor applies the insertion algorithm (encoder) on the data (for more details on the insertion algorithm, see sections 4.5.2 and 4.6.2) and sends the fingerprinted data to the recipient. Distrusted recipients are not able to modify received data, only trusted recipients. Trusted recipients use their decoder (see section 4.5.3 for more details on the extraction algorithm) to remove fingerprinted records except one record that remains in the data as an identifier of that recipient. Also, one record from the original dataset for trusted recipients and two records from the original dataset for distrusted recipients are removed to make collusion attacks more difficult.

The detection algorithm (see sections 4.5.4 and 4.6.4 for more details on the detection algorithm) is applied on data that might contain fingerprinted records. All fingerprints are searched for in that dataset and the detection algorithm displays the number of fingerprinted records for each recipient.
The system properties of the machine used for the proof of concept implementation have been:

Intel Pentium
4 CPU 1.60 GHz
261,424 KB RAM

The next section provides more detail on the experiments performed using the above set-up, and the last section of this chapter reports and explains results.

5.2. Experiments

Experiments involving the set-up (generation of fingerprints, insertion, extraction and detection algorithms) described in section 5.1, are explained in more detail through the following two examples.

Example 1: TC member

Let's assume that the authenticated client is Civic Hospital 5 (the acute pain subspecialty unit of Civic Hospital). The distributor knows that the fingerprint of this client is **TLMRX** and finds it in the following records of the original database:

ID|Name|Health Card Number|Gender|Birth Date|Symptoms
117|Lynette Sauve|4602 440 300 CP|female|1957-08-26 00:00:00| EATING DISORDER, LOW POTASSIUM, HYPOKALEMIA, IRREGULAR HEART BEAT, FREQUENT URINATION, THIRST – EXCESSIVE, CHANGES IN THE EXTRACELLULAR POTASSIUM LEVEL
169|Anthony Dufour|7953 382 064 CI|male|1951-04-24 00:00:00| NAUSEA, VOMITING, LOSS OF APPETITE, BLOODY DIARRHEA, FEVER - RECENT, STOMACH PAIN, GASTROINTESTINAL ANTHRAX
208|Jennifer Brown|3853 569 103 OS|female|1990-07-11 00:00:00| TONSILLITIS, SWOLLEN GLANDS ON BOTH SIDES OF THE NECK, HIGH TEMPERATURE, DIFFICULTY IN SWALLOWING, THIRST - EXCESSIVE
209|Neville Johnston|3462 976 577 OL|male|1992-06-10 00:00:00| STRESS, BED-WETTING (ENURESIS), SLOW DEVELOPMENT OF BLADDER CONTROL, FRIGHTENING EXPERIENCE
211|Marissa Wesley|3583 394 182 OW|female|1998-08-03 00:00:00| SPOTS, MILD FEVER, GENERAL MALAISE, FLAT RED RASH BEGAN ON THE SCALP, FACE AND BACK, ITCHY TINY CLEAR BLISTERS, FRESH RED SPOTS, MILD

STOMACH ACHE, CHICKEN POX

TABLE 4: FINGERPRINT IN ORIGINAL RECORDS (TC MEMBER)

The copies of these records are modified in such a way that existing identifiers and quasi-identifiers are replaced by fictitious values chosen from the domain set.

Mark|Name|Postal Code|Health Card Number|Gender|Birth Date|Symptoms
y|Lisa MacKay|K2B 5F4|6391 951 048 GW|female|1958-08-26 00:00:00|EATING DISORDER, LOW POTASSIUM, CHILLS, IRREGULAR HEART BEAT, THIRST – EXCESSIVE, STOMACH PAIN, ANXIETY
y|Justin Cote|J0V 7G1|5625 612 841 FP|male|1954-06-24 00:00:00| VOMITING, STRESS, LOSS OF APPETITE, STOMACH PAIN, GASTROINTESTINAL ANTHRAX, SEVERE STOMACH ACHE
n|Karen Gauthier|K2A 9M4|2052 851 832 GQ|female|1993-09-13 00:00:00|TONSILLITIS, HIGH TEMPERATURE, FEVER, THROAT ACHE, THIRST - EXCESSIVE
y|Nicholas Campbell|K8F 9N6|7341 892 923 DT|male|1992-06-12 00:00:00|STRESS, SLOW DEVELOPMENT OF BLADDER CONTROL, LOW SELF CONFIDENCE, FRIGHTENING EXPERIENCE
y|Donna Abbott|K4F 9N7|9351 835 901 HY|female|1998-08-04 00:00:00|SPOTS, MILD FEVER, GENERAL MALAISE, LOSS OF APPETITE, FLAT RED RASH BEGAN ON THE SCALP, CHICKEN POX

TABLE 5: MODIFIED RECORDS (TC MEMBER)

The attributes in Table 5 are the direct implication of the insertion algorithm (Section 4.3. Tier 1: Trusted Circle of Database Sharers) and property requirements for the Tier 1 scheme. Specifically, attribute Mark is an added column with values "y" or "n", depending on whether the record contains the fingerprint or not. All records marked with "y" are removed by the decoder on the trusted circle member's side. In this example, the third record is marked "n" although it contains the fingerprint. This row is not removed by the decoder because it is not marked "y" and thus, it serves to identify the trusted

circle member. Postal Code is a fictitious attribute, added to protect against vertical data mining attacks, which are also discussed in section 4.3. The Postal Code column is also marked and the decoder will remove it. The marked column that contains the "y"s and "n"s, and the marked row that contains the Postal Code attribute are encrypted with the trusted circle member's public key.

The modified copies of the records containing the fingerprint are inserted into the database in a place that is always same for same data. Now, the database contains 5 additional records and one additional attribute. Also, one record from the database is removed by the encoder as discussed under Collusion attacks in section 4.3. In total, 225 records are encrypted with the public key of the TC member (in this case, Civic Hospital 5) and transferred to the decoder on the TC member's side. The decoder (extraction algorithm) decrypts the data with the trusted circle member's private key, removes the 4 records marked with "y" and fictitious attribute Postal Code. At the end, the TC member receives 221 records, from which one record is fictitious, one original record is removed and from which 6 records contain the fingerprint (5 original records and one fictitious record).

Example 1: DC member

Let's assume that the authenticated client is Researcher 1 (University of Ottawa). The distributor knows that the fingerprint of this client is **SIRMJ** and finds it in the following records of the original database:

ID|Name|Health Card Number|Gender|Birth Date|Symptoms
173|Margaret White|9044 823 122 SO|female|1926-03-03 00:00:00| DYSARTHIA, SLURRED SPEECH, SPEECH DISORDER, PARKINSON'S DISEASE, TREMOR, STIFF MUSCLES AND JOINTS
202|Jack Kovacs|2446 621 592 UC|male|1940-04-14 00:00:00| OSTEOPOROSIS, BROKEN BONE, SUDDEN SEVERE BACK PAIN, COLLAPSED VERTEBRAE, LACK OF ACTIVITY, WEAKNESS - MAJOR FACTOR

212|Theresa Plavins|2321 590 104 IS|female|1937-09-25 00:00:00| SEVERE PAIN, RHEUMATOID ARTHRITIS BEFORE, FEVER, FEELING FLUSHED AND HOT, THE JOINT IS IMPOSSIBLE TO MOVE, SWEATING EXCESSIVE, THE JOINT IS VERY HOT
218|John Allard|2353 823 023 UX|male|1956-08-28 00:00:00| PAIN IN WRIST GETS STEADILY WORSE AS THE DAY GOES ON, JOINT IS MOVED - CRACKLING SENSATION, AT WORK PAIN COMES BACK, REPETITIVE STRAIN INJURY - TENDONITIS, TENOSYNOVITIS
221|Robert Ta|7234 394 295 UV|male|1975-05-01 00:00:00| PROTRUDED LOOP OF BOWEL, FEELING IMPULSE FROM COUGH, INGUINAL HERNIA BECAME LARGE AND CUT OFF BLOOD SUPPLY, WEAKNESS IN THE ABDOMINAL WALL, SWELLING AT THE JOIN BETWEEN THE THIGH AND ABDOMEN

TABLE 6: FINGERPRINT IN ORIGINAL RECORDS (DC MEMBER)

The copies of these records are modified in such a way that existing identifiers and quasi-identifiers are replaced by fictitious values chosen from the domain set.

Name|Postal Code|Health Card Number|Gender|Birth Date|Symptoms
Amanda Johnson|K2B 5C3|8109 923 102 IX|female|1928-03-05 00:00:00|DYSARTHIA, SLURRED SPEECH, DIFFICULTY SPEAKING, TREMOR, STIFF MUSCLES AND JOINTS
Brian Periard|J9A 8H3|4274 921 035 SA|male|1944-06-16 00:00:00|OSTEOPOROSIS, BROKEN BONE, WEAKNESS - MAJOR FACTOR, DIFFICULTY WALKING
Stephanie Lewis|K2B 6V3|7351 934 712 RX|female|1941-11-26 00:00:00|SEVERE PAIN, RHEUMATOID ARTHRITIS BEFORE, WEAKNESS, THE JOINT IS IMPOSSIBLE TO MOVE
Scott Meyer|K2G 9B3|9231 853 723 GA|male|1957-08-28 00:00:00|PAIN IN WRIST GETS STEADILY WORSE AS THE DAY GOES ON, JOINT IS MOVED - CRACKLING SENSATION, MILD FEVER, WEAKNESS, REPETITIVE STRAIN INJURY - TENDONITIS
Joshua Foley|K4S 5T2|2555 717 834 DE|male|1977-06-03 00:00:00|FEELING

IMPULSE FROM COUGH, INGUINAL HERNIA BECAME LARGE AND CUT OFF BLOOD SUPPLY, SWELLING AT THE JOIN BETWEEN THE THIGH AND ABDOMEN, GENERAL DISCOMFORT

TABLE 7: MODIFIED RECORDS (DC MEMBER)

Distrusted circle members do not have decoders; they receive the fully fingerprinted data. In this example, the fully fingerprinted data contains 5 fictitious records and these records are not marked. The fictitious attribute Postal Code exists in this example, since protection against vertical data mining attacks applies here as well. Protection against collusion attacks is attempted to be achieved by removing two records from the original database (Section 4.4, Tier 2: Distrusted Circle of Database Sharers, provides more explanation). Therefore, the database sent to Researcher 1 in this example contains 224 records in total, from which 5 records are fictitious, 2 original records are removed, 10 records contain the fingerprint and the fictitious attribute Postal Code is present.

Detection

The detection algorithm is applied on a dataset that might contain fingerprinted records. All fingerprints are searched for in that dataset and the detection algorithm displays the number of fingerprinted records for each recipient. The next section in this chapter provides examples when data is found outside the group of recipients (trusted or distrusted) and the task is to detect who divulged it.

5.3. Results

The following examples demonstrate the case when data is found outside the group of recipients (trusted or distrusted) and the task is to detect who divulged it.

Example 1 shows results when the detection algorithm is applied on data that was sent to trusted circle member 2 (containing the fingerprint of Civic Hospital 2), and example 2 shows results when the detection algorithm is applied on data that was sent to distrusted recipient 4 (containing the fingerprint of Statistician 2).

Example 1:

Recipient	Trusted	Number of records containing the fingerprint of the recipient
Civic Hospital 2	yes	6
Researcher 1	no	5
Researcher 2	no	5
Statistician 1	no	5
Statistician 2	no	5
Civic Hospital 1	yes	5
Civic Hospital 3	yes	5
Civic Hospital 4	yes	5
Civic Hospital 5	yes	5
Civic Hospital 6	yes	5

TABLE 8: RESULTS OF THE DETECTION ALGORITHM (TC)

(Data includes the fingerprint of a trusted circle member)

Example 2:

Recipient	Trusted	Number of records containing the fingerprint of the recipient
Statistician 2	no	10
Civic Hospital 5	yes	8
Researcher 2	no	8
Statistician 1	no	6
Civic Hospital 4	yes	6
Civic Hospital 6	yes	5
Researcher 1	no	5
Civic Hospital 3	yes	5
Civic Hospital 1	yes	5
Civic Hospital 2	yes	4

TABLE 9: RESULTS OF THE DETECTION ALGORITHM (DC)

(Data includes the fingerprint of a distrusted circle member)

These results show that the most important fact in detecting a particular fingerprint in the database is that this fingerprint should appear more frequently than other fingerprints in the data.

In example 1, Civic Hospital 2 received 5 additional (fingerprinted) records but 4 were removed by its decoder. The fingerprint of Civic Hospital 2 occurred (and was detected) most frequently.

In example 2, Statistician 2 received 5 additional records and they were not removed since distrusted recipients do not have decoders. However, some of these 5 additional records contained the fingerprints of other recipients as well, that is why Civic Hospital 5 and Researcher 2 have 8 fingerprinted records in the database. The same applies to Statistician 1 and Civic Hospital 4 (each has 6 fingerprinted records). Civic Hospital 2 has only 4 fingerprinted records in the database because the fifth record that contained its fingerprint was removed in attempt to defend against collusion attacks.

The purpose of this proof of concept implementation has been to demonstrate that fingerprinting of categorical data without modifying existing data is possible, and also is fingerprint detection. Future work is required to implement the scheme with real data and to test its resistance against attacks.

6. Discussion

The fingerprinting scheme proposed in this thesis suggests several interesting points for discussion.

6.1. Number of Fictitious Records

First of all, since existing data is not modified, and since fictitious records containing the fingerprint are added to the database, one question could be whether it is important to determine the optimum number of fictitious records required to achieve accurate fingerprint detection and at the same time, to limit data alteration as much as possible.

Proof of concept implementation results show that the fingerprint can be selected such that it already appears in a certain number of records in the original database so the actual number of fictitious records can be low. These results also show that the most important fact in detecting a particular fingerprint in the database is that this fingerprint appears more frequently than fingerprints of other recipients in the data. It can be therefore concluded that for detection the number of fictitious records is not important. Instead, analysis of the database (as explained in section 4.1.) is required prior to fingerprinting to determine, which characters of the categorical data are best suitable for fingerprint creation so that the number of original records containing the fingerprints is approximately equal for all recipients but at the same time, these original records are different for each recipient. This is a very important step because unique fingerprints are created only once and they should sustain all future changes in the database and also, future changes in recipient lists. This step may sound difficult but given that fingerprints can be of any length, characters can be any characters, and that database attributes usually have special characteristics, it is achievable. As an example, in the proof of concept implementation it was determined that characters S and T appeared more frequently in the Symptoms attribute than any other character and that characters J and X appeared only seldom.

Another question with regards to the proposed scheme is whether it is possible to determine an optimum number of fingerprinted records to be inserted in the data such that

the robustness, detectability and capability properties are satisfied and at the same time, defence against horizontally partitioning attacks is achieved.

Robustness means that it must be difficult (hopefully impossible) to destroy or overwrite the fingerprint. Detectability ensures that an authority is able to determine the owner from the fingerprint. Capacity is the ratio of usefulness and detectability i.e. the more fictitious records are inserted, the higher the detectability is but the data is less useful and vice versa. Capacity has impact on robustness as the higher the detectability, the larger is the possibility to destroy fingerprinted records but at the same time more records can survive an attack to destroy them.

Horizontal partition means that each dataset is split across multiple sites, with each site having the same attributes but different records. Therefore, to survive a horizontally partitioning attack, or any attack that involves only a few records instead of the whole set of records, the fingerprint should be inserted in more than one record, and in such a way that fingerprinted records are spread around in the database.

Since the proposed fingerprinting scheme is created such that some original records already contain the fingerprint, attempts to destroy the fingerprint definitely render data less useful, satisfying by this the robustness property. Detectability is achieved if the fingerprint of a recipient appears more frequently than fingerprints of other recipients in the data. This depends on whether sufficient number of original records contain the fingerprint so the decoder can only copy some of these records. This way, the usefulness of records is not compromised and the capacity of the overall scheme is high. Also, if many original records contain the fingerprint, horizontal partitioning attacks are very much defendable.

Therefore, the answer to the second question is exactly the same as the answer to the first question. More precisely, prior to assigning fingerprint, the database has to be examined to determine the character set from which the fingerprint will be chosen, and also, to determine the length of the fingerprint. The more fingerprinted records the original

database contains (while Assumptions I and II are satisfied), the more robust the scheme is, detectability and capacity are higher and successful defence against horizontal partitioning attacks is more probable.

6.2. Collusion Attacks

Collusion attacks (data matching) are special cases of mix and match attacks, where different datasets from the same source can be compared by different recipients to discover fingerprinted records.

The answer to the first two questions in section 6.1. cannot be applied for these attacks as malicious recipients can still compare their copies and find fictitious records. As discussed under sections 4.5.6 and 4.6.6, a fingerprinting scheme can be resistant to collusion attacks and data can still be useful if the number of recipients is reduced. For example, if a database has 100,000 records, resistance to collusion attacks can be achieved with 10 users as the number of fictitious records to be added to the database would be 10,000. If this is considered high, more useful data can be achieved with 9 recipients, where the number of fictitious records to be added would be 6,261.

Another solution this work proposes is that besides adding fictitious records to the database, some original records are removed to make it more difficult for malicious recipients to determine, which records come from the original database.

Alternative ways of reducing but not eliminating collusion attacks would be to never disclose the same data to members, or apply other disclosure limitation techniques such as perturbation prior to fingerprinting the data.

6.3. Two-tiered Approach

The last discussion point is the two-tiered approach taken in this thesis. This approach, of course, is not necessary. The assumption is that data is sometimes disclosed to recipients who should not be fully trusted. Disclosure in these instances can happen out of carelessness or because the entity who disclosed the database took precautionary measures such as data anonymization and was not aware of data matching or re-identification. Beside this, there is one more advantage to the two-tiered approach. Specifically, trusted recipients can have a decoder, which removes fictitious records on their side and thus making data more useful.

Attacks on the data can happen before and after encoding, and after decoding. Attacks before encoding are out of the scope of this work as encoders and the database are part of the distributor and should be safeguarded. If the attacks happen after decoding, it means that disclosure happened at a fully trusted member's side. As explained already, decoders remove all but one fictitious record to guard against these attacks.

If the attacks happen after encoding, the data is encoded with a trusted or distrusted member's fingerprint. In this case, there are four possibilities:

1. The trusted or distrusted circle member who requested the data is the entity who illegally disclosed the data. Performing detection determines that the trusted or distrusted circle member's fingerprint is the most frequently present fingerprint in the dataset and this member is rightly accused of illegal data disclosure.

2. An outside entity attacked the scheme and stole the fingerprinted data. If the attack involved changing/replacing the existing fingerprint, then it is a benign or invertibility attack. If there was just data theft, then the information security aspect of the scheme has been breached, which is out of the scope of this work. By all means, security has to be always present to guard against data theft, whether it is the fingerprinted data itself or information about the fingerprints.

3. Another trusted or distrusted circle member knows the fingerprint of the trusted or distrusted circle member and is inserting this fingerprint to mask his own entity and be able to illegally disclose data. This attack is the invertibility attack.

4. The distributor is malicious and wants the trusted or distrusted circle member to be falsely accused of illegal disclosure. This is a disadvantage of this symmetric scheme. Future work should include research on anonymous fingerprinting of databases.

7. Future Work

7.1. Real Data

The first and foremost task to be definitely done in the future is testing on real data. The proof of concept implementation served to show that the proposed scheme is implementable.

Real data as opposed to synthetic data is needed to see whether it is indeed possible to determine the character set from which the fingerprints are to be chosen. Also, testing on real data will enable exploring in more depth the scheme's vulnerability to attacks. And thirdly, experiments on real data will provide more insight on robustness, performance and effectiveness.

7.2. Resistance to Attacks

Although this work is not a security proposal, it would still be desirable to develop a "threat model" and test the implemented scheme's resistance to it. It has already been noted in sections 4.5.6 and 4.6.6 that resistance against collusion attacks is possible if the number of recipients is reduced, or alternatively, if original records are withheld when a request for data is made. Nevertheless, future work should include privacy and security analyses of the proposed fingerprinting scheme.

7.3. Detection Algorithm

The detection algorithm is applied when a request comes in to determine the recipient of a particular dataset. Since only the distributor knows the fingerprints of its recipients, it must search the dataset for all fingerprints to determine, which recipient disclosed the data. The search is conducted in a sequential order, record by record for each fingerprint. Each record is searched for every fingerprint. If a character of a particular fingerprint is found, the search continues for the next character of that fingerprint in the same record. If not all the characters of a particular fingerprint can be found in a record, the search continues for the next fingerprint in that record. Detection depends on the length and

number of fingerprints, number of records in the dataset and the length of the records in the dataset. As part of future work, the detection algorithm could be optimized.

7.4. Anonymous scheme

The fingerprinting scheme proposed in this work is symmetric as only the distributor of the database participates in the marking procedure. The distributor determines all fingerprints, unique for each recipient, and inserts the appropriate fingerprint in the database prior to sending it to the recipient. This scheme is symmetric as both the distributor and the recipient have the same fingerprinted copy. A disadvantage of this scheme is that recipient A can be falsely accused of illegal distribution if the distributor provides another recipient B with a copy bearing A's mark.

In asymmetric fingerprinting, both the distributor and the recipient take part in the marking procedure (the distributor does not see the copy the recipient receives because the recipient inputs its own secret in the copy received from the distributor). The TC fingerprinting scheme as proposed in this thesis can easily be extended to satisfy asymmetric fingerprinting requirements. Specifically, decoders of TC members could, upon receiving the fingerprinted dataset from the distributor, insert a secret mark (known only by that particular TC member) into the data. Obviously, this step cannot be applied in the case of DC members as they do not have a decoder.

This fingerprinting scheme still lacks anonymity and it would be interesting to see whether the anonymous fingerprinting scheme can be applied to limiting database disclosure.

In anonymous fingerprinting, the system is based on having a trusted third party (central authority) who is the only entity who knows the recipient's real identity. Confidential data protection in anonymous fingerprinting scheme involves a circle of database sharers, and a trusted central authority who assigns unique fingerprints to each recipient of the copy of the database. There is no distributor and circle members share the data in their possession by inserting their fingerprints in the data prior to disclosing it. The fingerprint

is embedded in the quasi-identifiers of the database, again, to prevent the destroying of the fingerprint by removing the identifiers from the database. This unique fingerprint then identifies the member in the case of illegal reselling (disclosure) of the database.

7.5. Trustworthiness

Another improvement to the proposed scheme could be to somehow analyze the behaviour of the recipients to dynamically determine their trustworthiness. This idea is not exactly relevant to this work but might be a topic of future research.

8. Conclusion

The proposed fingerprinting scheme is an important proposal from a privacy perspective since it can identify from the data itself the entity who illegally disclosed data.

This thesis is based on my paper [38] and it explores the possibility of embedding fingerprints in categorical values of a database. In addition, in the proposed fingerprinting scheme, existing data is not modified. Fingerprints are inserted into the quasi-identifiers of categorical data and records containing the fingerprints are added to the existing database. This fingerprinting scheme can be used as a complement to the existing approaches, namely to altering data before disclosure and learning results without revealing the data. In fact, this fingerprinting scheme is a refinement of the adding tuples data altering technique as added tuples have the purpose of not only obscuring data but hiding information about the owner and recipient of the data. Fingerprinted data can still be used for privacy preserving data mining.

Also, this thesis brings a novel contribution to the area of privacy in Canada, especially because it proposes a solution to a privacy problem by using technical mechanisms and taking into account privacy legislation and practices in Canada, defines differences between a privacy and a security problem, between a privacy and a security proposal, and provides a list of existing privacy and security tools that can be used to conduct privacy and security analyses.

Proof of concept implementation results show that a fingerprint can be selected such that it already appears in a certain number of records in the original database so the actual number of fictitious records can be low. These results also show that the most important fact in detecting a particular fingerprint in the database is that this fingerprint appears more frequently than fingerprints of other recipients in the data. It can be therefore concluded that determining the number of fictitious records is not important. Instead, analysis of the database is required prior to fingerprinting to determine, which characters of the categorical data are best suitable for fingerprint creation. The goal is to make the number of original records containing the fingerprints almost equal for all recipients, but

at the same time, ensure that a subset of these original records is different for each recipient.

Collusion attacks are the most difficult problem fingerprinting schemes face. To achieve a collusion secure fingerprinting scheme, including the proposal of this thesis, the number of recipients should be no more than 10 if there are 100,000 original records in the database. Another solution this work proposes is that besides adding fictitious records to the database, some original records are removed to make it more difficult for malicious recipients to determine, which records come from the original database. Alternative ways of reducing but not eliminating collusion attacks would be to never disclose the same data to members, or apply other disclosure limitation techniques such as perturbation prior to fingerprinting the data.

The two-tiered approach taken in this work is not necessary but has certain advantages when compared to a one-tiered approach (all recipients trusted or all recipients distrusted). Data could be illegally disclosed out of maliciousness, carelessness or because the recipient is uninformed (takes precautionary measures such as data anonymization but is not aware of data matching or re-identification). This definitely justifies having a distrusted circle. On the other hand, having a trusted circle enables members to get Almost Clean Data (ACD) in a faster way.

This fingerprinting scheme is symmetric as only the distributor of the database participates in the marking procedure. The distributor determines all fingerprints, unique for each recipient, and inserts the appropriate fingerprint in the database prior to its disclosure. Also, the trusted circle fingerprinting scheme proposed in this work can be extended to satisfy asymmetric fingerprinting requirements. However, the proposed fingerprinting scheme lacks anonymity and it would be interesting to see whether the anonymous fingerprinting scheme can be applied to limiting database disclosure.

Glossary

ACD – Almost Clean Data; contains a limited number $((\tau + 1) \pm \varepsilon)$ fingerprinted records where $\tau \pm \varepsilon$ fingerprinted records belong to the original dataset

BCP – Business Continuity Plan

CIHI – Canadian Institute for Health Information

DC – Distrusted circle of database sharers

EPAL – Enterprise Privacy Authorization Language

Fully fingerprinted data – Contrary to ACD, fully fingerprinted data contains $(a + 1)\ \tau \pm \varepsilon$ records where $\tau \pm \varepsilon$ fingerprinted records belong to the original dataset

ISO – International Standards Organization

JDBC – Java Database Connectivity Package

ODBC – Open Database Connectivity

P3P – Platform for Privacy Preferences

PIA – Privacy Impact Assessment

PPIA – Preliminary Privacy Impact Assessment

PIPEDA – Personal Information Protection and Electronic Documents Act

SIN – Social Insurance Number

SMC – Secure Multiparty Computation

SoS – Statement of Sensitivity

SQL – Structured Query Language

TC – Trusted circle of database sharers

TRA – Threat and Risk Assessment

References

1. Access to Information Act (R.S. 1985, c. A-1), unofficial version is available on the Department of Justice Canada Web site at http://laws.justice.gc.ca/en/a-1/8.html
2. Adams C., Lloyd S.: Understanding Public-Key Infrastructure. Concepts, Standards, and Deployment Considerations. Macmillan Technical Publishing, 1999
3. Agrawal D., Aggarwal C.: On the Design and Quantification of Privacy Preserving Data Mining Algorithms. In proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Santa Barbara, California, USA, ACM, pages 247-255, May 2001
4. Agrawal R., Kiernan J.: Watermarking Relational Databases. In Proceedings of the 28th International Conference on Very Large Databases. Hong Kong, China, 2002
5. Agrawal R., Srikant R.: Privacy-Preserving Data Mining. In Proceedings of the ACM SIGMOD Conference on Management of Data. Dallas, TX, USA, pages 439-450, 2000
6. Boneh D., Shaw J.: Collusion-Secure Fingerprinting for Digital Data. In Advances in Cryptology - CRYPTO 95, pages 452--465, 1995
7. Boney L., Tewfik A.H., Hamdy K.N.: Digital watermarks for audio signals. International Conference on Multimedia Computing and Systems. Hiroshima, Japan, pages 473-480, 1996.
8. Brassil J.T., Low S., Maxemchuk N.F.: Copyright protection for the electronic distribution of text documents. Proceedings of IEEE, Volume 87, Number 7, pages 1181-1196, 1999
9. Canadian Standards Association, available at http://www.csa.ca/Default.asp?language=english
10. Chao V., Cruz L., Lei N.: Local vs Remote Database Access – A Performance Test, IBM Redpaper, 29 December 2005, available at http://www.redbooks.ibm.com/abstracts/redp4113.html?Open
11. Chor B., Fiat A. and Naor M.: Tracing traitors, in "Advances in Cryptology – Crypto'94", Lecture Notes in Computer Science 839 (1994), 480–491.

12. Clifton C., Marks D.: Security and Privacy Implications of Data Mining. In Proceedings of the 1996 ACM SIGMOD Workshop on Data Mining and Knowledge Discovery. Montréal, Canada, pages 15-19, May 1996
13. Clifton C., Kantarcioğlu M., Vaidya J., Lin X., Zhu M.Y.: Tools for Privacy Preserving Distributed Data Mining. SIGKDD Explorations, Volume 4, Issue 2, pages 28-34, January 2003
14. Clifton C., Kantarcioğlu M., Vaidya J.: Defining Privacy for Data Mining. In National Science Foundation Workshop on Next Generation Data Mining, Kargupta H., Joshi A., Sivakumar K., Eds., Baltimore, MD, pages 126-133, November 2002
15. Clifton C., Vaidya J. : Privacy preserving data mining. In Advances in Cryptology – CRYPTO 2000, pages 36-54. Springer-Verlag, August 2000
16. Collberg C., Thomborson C.: On the Limits of Software Watermarking. Technical report #164, Department of Computer Science, The University of Auckland, August 1998
17. Cox I. J. and Miller M. L.: A review of watermarking and the importance of perceptual modeling. Book title: Proceedings of Electronic Imaging '97. In Rogowitz and Pappas. 37 ISBN 0-8194-2427-7, ISSN 0277-786X
18. Craver S., Yeo B.L., Yeung M.: Technical trials and legal tribulations. Communications of the ACM, Volume 41, Number 7, pages: 44-54, July 1998
19. Denning D.: Cryptography and Data Security (Chapter 6: Inference Controls), Addison-Wesley, 1982
20. Denning D., Denning P. and Schwartz M.: The Tracker: A threat to statistical database security. ACM Transactions on Database Systems, 4(1) pages 76-96, March 1979
21. Dobbertin H., Bosselaers A. and Preneel B., RIPEMD160: A strengthened version of RIPEMD, Fast Software Encryption, LNCS1039, Springer-Verlag, 1996, pages 71-82.
22. Duncan G. and Mukherjee S.: Microdata disclosure limitation in statistical databases: query size and random sample query control. In Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, May 20-22, Oakland, California, 1991

23. Enterprise Privacy Authorization Language (IBM), available at http://www.zurich.ibm.com/pri/projects/epal.html
24. Felty A., Matwin S.: Privacy-Oriented Data Mining by Proof Checking. University of Ottawa, available at http://www.site.uottawa.ca/~stan/contents/research.html
25. Fernández-Muñoz M., Soriano-Ibañez M., Domingo-Ferrer J., Sebé-Feixas F.: Fingerprinting Schemes for the Protection of Multimedia Distribution Rights. Upgrade, The European Online Magazine for the IT Professional, Volume III, Number 6, December 2002, pages 36-40
26. Guo H.: Digital Image Watermarking for Ownership Verification. Ph.D. Thesis, University of Ottawa, 2003
27. Hartung F., Kutter M: Multimedia watermarking techniques. Proceedings of the IEEE, Volume 87, Number 7, pages 1079-1107, July 1999
28. Hinke T.: Inference aggregation detection in database management systems. In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 96-107, Oakland, 1988
29. Kantarcioğlu M., Clifton C.: Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. In the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pages 24-31, June 2002
30. Kargupta H., Datta S., Wang Q., Sivakumar K.: Random Data Perturbation Techniques and Privacy Preserving Data Mining. Extended version of the Best Paper Award winning paper IEEE 2003 International Conference on Data Mining
31. Li Y., Swarup V., Jajodia S.: Constructing a Virtual Primary Key for Fingerprinting Relational Data. DRM'03, October 27, 2003, Washington, DC, USA, 2003 ACM 1-58113-786-9/03/0010, pages 133-141
32. Li Y., Swarup V., Jajodia S.: Fingerprinting Relational Databases. Technical Report, Center for Secure Information Systems, George Mason University, Fairfax, VA, May 2003
33. Lodhi H., Saunders C., Shawe-Taylor J., Cristianini N.: Text Classification Using String Kernels Journal of Machine Learning Research 2 (2002) pages 419-444
34. Lunt T.: Aggregation and Inference: Facts and fallacies. In Proceedings of the IEEE Symposium on Security and Privacy, pages 102-109, Oakland, California, 1989

35. Main A., van Oorschot P.: Software Protection and Application Security: Understanding the Battleground. State of the Art and Evolution of Computer Security and Industrial Cryptography, June 2003, Heverlee, Belgium, Springer-Verlag LNCS
36. Mohanty S. P.: Digital Watermarking: A Tutorial Review, available at http://www.csee.usf.edu/~smohanty/research/Reports/WMSurvey1999Mohanty.pdf
37. Morgenstern M.: Security and Inference in multilevel database and knowledge based systems. Proceedings of the ACM SIGMOD Conference, pages 357-373, 1987
38. Mucsi-Nagy A., Matwin S.: Digital Fingerprinting for Sharing of Confidential Data. International Workshop on Privacy and Security Issues in Data Mining in Conjunction with the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, September 20, 2004, pages 11-26
39. Personal Information Protection and Electronic Documents Act (2000, c. 5), unofficial version is available on the Department of Justice Canada Web site at http://laws.justice.gc.ca/en/P-8.6/
40. Platform for Privacy Preferences (P3P), an industry standard developed by the World Wide Consortorium to provide a simple, automated way for users to gain more control over the use of personal information on Web sites they visit. Web site: http://www.w3.org/P3P/
41. Petitcolas F., Anderson R. J., and Kuhn M. G.: Information Hiding: A Survey. Proceedings to IEEE special issue on Protection of Multimedia Content , 1999
42. Privacy Act (R.S. 1985, c. P-21), unofficial version is available on the Department of Justice Canada Web site at http://laws.justice.gc.ca/en/P-21/index.html
43. Privacy and Data Protection Policy, available on the Treasury Board of Secretariat Web site at: http://www.tbs-sct.gc.ca/pubs_pol/gospubs/TBM_128/CHAP1_1-1_e.asp
44. Privacy Impact Assessment Policy, available on the Treasury Board of Canada Secretariat Web site at http://www.tbs-sct.gc.ca/pubs_pol/ciopubs/pia-pefr/paip-pefr_e.asp
45. Ruanaidh J.J.K., Dowling W.J., Boland F.M.: Watermarking digital images for copyrights protection. IEEE Proceedings on Vision, Signal and Image Processing, 143(4), pages 250-256, 1996

46. Sion R., Atallah M., Prabhakar S.: Rights Protection for Relational Data. In proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pages 98-109, 2003
47. Sion R.: Proving Ownership over Categorical Data. In Proceedings of the IEEE International Conference on Data Engineering IEEE ICDE, Boston, 2004
48. Su T., Ozsoyoglu G.: Controlling FD and MVD inference in multilevel relational database systems. IEEE Transactions on Knowledge and Data Engineering, 3:474-485, 1991
49. Sweeney L.: Computational Disclosure Control: A Primer on Data Privacy Protection. Ph.D. Thesis, Massachusetts Institute of Technology, 2001
50. The Information and Privacy Commissioner/Ontario and Deloitte & Touche: The Security-Privacy Paradox: Issues, Misconceptions and Strategies. Joint Report, 2003, available at http://www.ipc.on.ca/scripts/index_.asp?action=31&P_ID=14447&N_ID=1&PT_ID=11351&U_ID=0
51. TRUSTe® - an independent, nonprofit organization dedicated to enabling individuals and organizations to establish trusting relationships based on respect for personal identity and information in the evolving networked world, Web site: http://www.truste.org/
52. Vaidya J., Clifton C.: Privacy Preserving Association Rule Mining in Vertically Partitioned Data. The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, ACM, pages 639-644, July 2002
53. Vaidya J., Clifton C.: Privacy preserving k-means clustering over vertically partitioned data. SIGKDD, Washington, DC, USA. ACM August 2003
54. Yang Z., Zhong S., Wright R.: Privacy-Preserving Classification of Customer Data without Loss of Accuracy. Proceedings of the Fifth SIAM International Conference on Data Mining, Newport Beach, CA, April 21-23, 2005
55. Yao A.C.: How to generate and exchange secrets. In Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pages 162-167. IEEE Computer Society, 1986.