Collusion Resistant Inference Control for Cadastral Databases

Khalil Firas Al, Alban Gabillon, Patrick Capolsini

Collusion Resistant Inference Control for Cadastral Databases

Firas Al Khalil, Alban Gabillon and Patrick Capolsini

Université de la Polynésie française
BP 6570 Faa'a Aéroport
{firas.khalil,alban.gabillon,patrick.capolsini}@upf.pf

Abstract. In this paper we present a novel inference control technique, based on graphs, to control the number of accessible parcels in a cadastral database. Different levels of collusion resistance are introduced as part of the approach. The dynamic aspect of the cadastral application, caused by mutation operations, is handled. We propose a scheme for gradually resetting the inference graph allowing continuous access to the data.

Keywords: Security, Inference Control, Database, Collusion

1 Introduction

In the context of collaboration with the French Polynesian computer science service, specifically with the GIS (Geographic Information System) department, we were expected to analyze their requirements in terms of security and propose solutions to secure access to their databases and applications.

Int his work we study the cadastral application used to manage French Polynesian real estate: ownership management, land boundaries operations, and other legal processes. The application is accessed through a mapping interface by employees of the service (to deliver official statements about lands to the public upon personal request), geographers, civil law notaries, among other employees from different services. The service intends to make all cadastral information available to the public. Basically, the security policy shall be the following: 1) Users are permitted to see parcel boundaries. 2) Users are permitted to know the owner's name of any random parcel.

However, the security policy includes also the following two prohibitions restricting the scope of the second permission: Pr_1**:** Users are prohibited to know the complete list of owners in a confined geographic region; Pr_2**:** Users are prohibited to know the complete list of parcels owned by a single legal entity.

In other words, the owner's name of any random parcel is considered as an unclassified data whereas the complete list of owners in a confined geographic region and the complete list of parcels owned by a single legal entity are both considered as classified data.

In the online cadastral application, users can see region maps through a mapping interface showing parcel boundaries. Users can select any parcel to learn its owner's name. However, as a basic restriction mechanism to enforce prohibitions Pr_1 and Pr_2, it is not possible to issue queries selecting multiple parcels filtered by the owner's name or by region. Users can only query parcels one by one.

The inference problem in databases occurs when sensitive information can be disclosed from non-sensitive data and meta-data[14]. In the context of our cadastral database, users can infer information regarded as sensitive (and therefore violate either Pr_1 or Pr_2) by repeatedly querying (either manually or by using a bot) non-sensitive data. Both Pr_1 and Pr_2 are similar to the phone book problem as presented by Lunt[16] where a user has the right to access some phone numbers but not the entire phone book; she classifies these problems as quantity-based aggregation problems.

In this paper, we propose a dynamic inference control approach based on user query history that evolves with the evolution of the database through time. We also provide different levels of collusion resistance over the complete database or a subset of the database. We show how to use the same approach (with minor modifications) to enforce both Pr_1 and Pr_2.

This paper is organized as follows: In Section 2 we present the legislative context behind the security policy. In particular we investigate the reasons motivating prohibitions Pr_1 and Pr_2. Section 3 introduces our approach for Pr_1, step by step, starting from the basic model and scheme, ending collusion resistance. Afterwards, we discuss necessary modifications that should be introduced on the approach in order to enforce Pr_2 in Section 4. Section 5 provides solutions for operations specific to the cadastral application, and the general issue of resetting the scheme over time. Related work is presented in Section 6. Finally, we conclude this paper in Section 7.

2 Legislative context

We investigated the state of online cadastral applications, and in this section we will give a couple of examples from different countries reflecting the legal point of view on the publication of parcel ownership information. Afterwards, we explain the French point of view on the subject and the case for French Polynesia motivating this work.

For instance, access to the Spanish[6] cadaster is provided through a mapping interface built with Google Maps . Parcel ownership information is considered sensitive and it is not available to the public. Land owners form a different level of users (more privileged than the public) and they are granted access to all information related to their own properties if they provide a valid X509 certificate associated with their national electronic ID Similarly, the Belgian cadaster is available online for the publicwhere ownership information is considered sensitive, thus prohibited. Authenticated users, using their national electronic ID, can access land ownership through another website.

In Croatia, parcel ownership information is public. Users can access the online websitewhere they can submit a query on any parcel and get a list of information related to the parcel, including land ownership. Queries are submitted by selecting the desired department, office and parcel ID or deed ID. Users are required to solve a CAPTCHA before query submission. Similarly, the state of Montana considers land ownership as public information and they provide the cadaster for online browsing through a mapping interface.

In France, the cadaster is available through a mapping interface,however no information is available to the public (except for land boundaries). This prohibition is due to the CNIL recommendation[1] (La Commission Nationale de l'Informatique et des Libertés, an independent administrative authority whose mission is to ensure that IT is at the service of citizens and does not undermine human identity, rights, private life or individual and public liberties) where it is stated "the diffusion of any identifying information (directly or indirectly) on interactive terminals or public websites entails the risk of using this information for other purposes, including commercial, without the concerned people's consent". The Cada[2] (La Commission d'accès aux documents administratifs, an independent administrative authority responsible for ensuring freedom of access to administrative documents) indicates that "punctual demands" of cadastral excerpts are allowed. Furthermore, cadastral excerpts may contain the name of land owners, but no other identifying information such as the national ID or the address. The frequency of demands and the number of parcels requested should be analyzed to insure that these demands do not infringe the principle of free communication of cadastral documents. However there is no clear definition of "punctual demands" and it is subject to various interpretations, therefore the Cada recommends a restrictive interpretation of the term.

French Polynesia is an overseas country of France, where the recommendations of the CNIL and Cada are applicable. Currently, the "punctuality" of demands issued by citizens is insured by employees of the French Polynesian real estate service when they are physically present at their offices. The work presented in this paper is a requirement of the French Polynesian computer science service expressing their interpretation of the recommendations of both CNIL and Cada in order to provide the same service offered by the real estate service through the internet: a user should have access to the ownership information of any parcel, at random, but s/he is not allowed to exploit the service for commercial (or social, ...) ends. This interpretation is the foundation of prohibitions Pr_1 and Pr_2 presented in Section 1.

3 Enforcing Pr_1

3.1 Definitions

In this section we show how to solve Pr_1 presented in Section 1. First, we need to introduce the following definition:

Definition 1. *[Parcel] The smallest geographical unit considered. A geo-referenced polygon, defining the surface and the boundaries of a piece of land owned by one or many legal entities.*

Let P be the set of parcels in the cadastral database and O the set of owners; o_p denotes the set of owners of a parcel p; $\delta(.,.)$ is a function that returns the minimal euclidean distance between 2 parcels. We create an inference graph $G(V,E)$: $G(V) = P$, while $G(E)$ is defined as follows: two parcels are neighbors in the inference graph if they touch each other, or if they are separated by a maximal distance τ . Formally $G(E) = \{(p,q) : p,q \in G(V), p \neq q, 0 \leq \delta(p,q) \leq \tau\}$ for a given $\tau \in \mathbb{Z}$. We could select $\tau = 0$, i.e. only parcels touching each other, but parcels which are separated by thin boundaries, like rivers or roads, require a value of τ greater than 0 to be considered as neighbors. Figure 1a shows an inference graph for Pr_1 representing part of the cadastral database, where parcel 1 touches $\{2,3,8\}$, parcel 4 touches 5, parcel 7 touches $\{3,5,6,8\}$, etc.

$G(V,E)$ is an undirected unlabeled graph. $N(.)$ and $dist(.,.)$ denote the graph's neighborhood and distance functions respectively.

Definition 2. *[Zone] The zone of a parcel* $\boldsymbol{p}$*, namely* $zone_p$*, is the network formed by* $\boldsymbol{p}$ *itself and all the neighbors* $N(p)$ *of* $\boldsymbol{p}$*.*

Notice that every parcel belongs to its proper zone and to every zone formed by every neighboring parcel.

Section 3.2 introduces the basic model and scheme. Then we define collusion in Section 3.3, and we build upon the basic scheme to support different levels of collusion resistance in Sections 3.4 and 3.5. We discuss the problem of inference from user's a priori knowledge in Section 3.6. And finally we investigate whether users can derive sensitive data from a denial of access in Section 3.7.

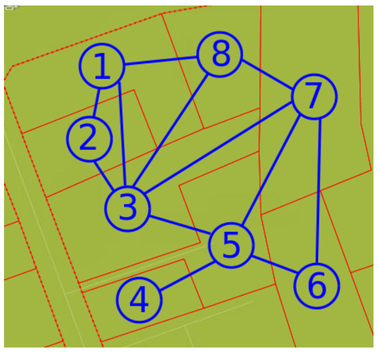
3.2 Model and Scheme

Pr_1 clearly states that "users are prohibited to know the complete list of owners in a confined geographic ***region***". The definition of a region is problematic: what is the minimum and maximum number of parcels that could define a region? How can we define limits between two neighboring regions?

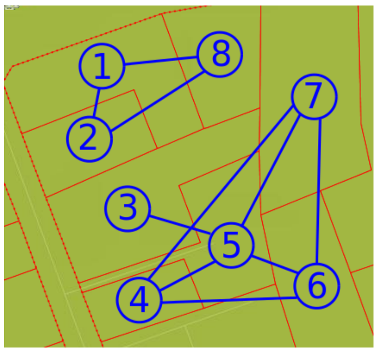
We opt to use the narrowest definition of a region, namely a zone as per Definition 2. Preventing users from knowing the complete list of owners in any zone implies that they cannot learn the complete list of owners in any region of any size and location.

We consider that "isolated" parcels, i.e. parcels that do not have neighbors in the range τ , do not fall within the scope of Pr_1: access to these parcels is granted automatically.

Preventing the full disclosure of any zone is simply preventing the disclosure of the full network. For instance, assume that p has 5 neighbors. The maximal number of parcels that could be disclosed from the zone of p is 5 ($|zone_p| - 1$). We could be more restrictive by preventing the full disclosure of ($|zone_p| - \alpha$), where α is a positive integer, but we will consider $\alpha = 1$ for the sake of simplicity. At the heart of the zone is the parcel, therefore the vertex. Every vertex v holds 3 pieces of information: 1) $\mathbf{id_v}$: a unique token. 2) $\mathbf{sig_v}$: the signature of v (set of acceptable tokens). It is the set of $\mathbf{id}$s of all parcels in $zone_v$. 3) $\mathbf{d_v}[][]$: a map associating users (key) with the set of consumed tokens by each one of them



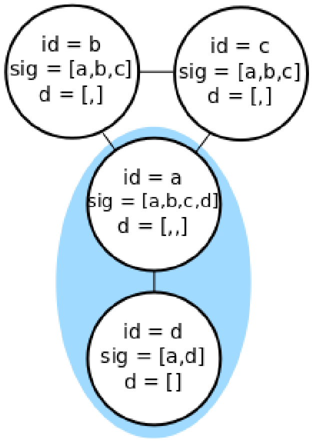
(a) A graph for Pr_1



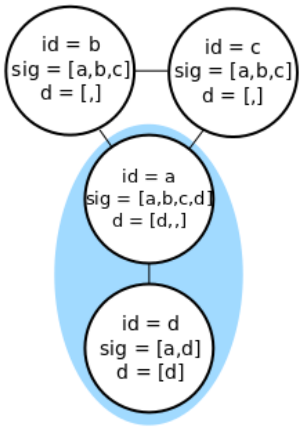
(b) A graph for Pr_2

Fig. 1: Different graphs for the same part of a cadastral database.

(value), initially empty. The set of consumed tokens in d_v is restricted to values from sig_v.



(a) Empty graph



(b) After a successful access to the vertex with $id = d$.

Fig. 2: A graph showing for every node its id, signature and the set of consumed tokens. Highlighted vertices form the zone of the vertex with $id = d$.

Figure 2a shows an example of an inference graph: the highlighted area represents the zone of the vertex with $id = d$.

Querying a vertex v is allowed if the **PreQuery** condition (line 2) of Algorithm 1 is satisfied. In fact, access to vertex v is allowed if it has been queried (line 8). If it has not been queried before, access is granted if all vertices in $zone_v$ can accept the token id_v(line 10). If the **PreQuery** condition is satisfied, id_v is propagated to all parcels in $zone_v$ (line 4) and the set of owner o_v is returned (line 5).

To illustrate the approach, let us consider the graph of Figure 2a. A user wants to access information contained in the vertex $id = d$. The **PreQuery** condition is satisfied: $id_d \notin d_d$, $id_d \notin d_a$, and $(|d_a| = 0) < (|sig_a| - 1 = 3)$. Therefore, access is granted. Figure 2b shows the state of the same graph after query execution: $(d_a \mathrel{+}= id_d) = \{d\}$; similarly,$(d_d \mathrel{+}= id_d) = \{d\}$. If the same user wants to access information contained in vertex a, s/he will be blocked for violating the **PreQuery** condition: $(|d_d| = 1) > (|sig_d| - 1 = 0)$, whereas access to other vertices is still allowed.

Note that, after querying parcel d, the user can maximally query $\{b, c, d\}$. The set of available parcels for any user depends on his querying behavior: other possible query combinations for the inference graph presented in Figure 2a are $\{a, c\}$ and $\{a, b\}$.

Algorithm 1 for Pr_1

```
 1: function Query(A parcel $v$, A user $u$)
 2:     if PreQuery($v$, $u$) then
 3:         for all $z \in zone_v$ do
 4:             $d_z[u] \mathrel{+}= id_v$
 5:         return $o_v$
 6:     return $\emptyset$
 7: function PreQuery(A parcel $v$, A user $u$)
 8:     if $id_v \in d_v[u]$ then return true                ▷ Access granted
 9:     for all $z \in zone_v$ do
10:         if $|d_z[u]| \geq |sig_z| - 1$ then return false   ▷ Access denied
11:     return true                                          ▷ Access granted
```

A zone is maximized if the PreQuery condition cannot be satisfied for that zone. A graph is saturated when the PreQuery condition cannot be satisfied by any unqueried zone. A graph is maximally saturated if the set of queried parcels is maximal (highest cardinality). It is possible to find multiple maximal combinations for a given graph. e.g. in Figure 2a: – the zone of parcel d is maximized if d or a is queried; – the graph is saturated if $\{a, c\}$, $\{a, b\}$, or $\{b, c, d\}$ are queried; – the graph is maximally saturated if the set $\{b, c, d\}$ is queried.

Our model as presented in sections 3.1 and 3.2 is basically on how to prevent a user from violating prohibition Pr_1, by repeatedly querying the database. However, there are other potential inference channels that might help the user to violate Pr_1. First, users can collude to break Pr_1. This issue is discussed in Sections 3.3, 3.4 and 3.5. Second, users can use some external knowledge they possess on the cadaster. This issue is discussed in Section 3.6. Finally, users can derive some information from a denial of access. This issue is discussed in Section 3.7.

3.3 Collusion Resistance

The Merriam-Webster online dictionary defines collusion as a “secret agreement or cooperation especially for an illegal or deceitful purpose”. In our context, the illegal or deceitful purpose is to access a given parcel and all of its neighbors (thus the zone of that parcel). Therefore a collusion happens when x users secretly agree or cooperate to access a given zone.

Definition 3. *[Zone-level collusion] We say that x users collude to compromise a zone of a parcel p if, and only if, the union of consumed tokens by those x users cover the signature of p.*

However, devising a scheme preventing x users from colluding on a single zone will prevent all subsequent users, possibly legitimate, from accessing the parcel.

This level of collusion resistance does not differentiate between accidental and intentional collusions, therefore we refine our definition of collusion resistance in the upcoming sections.

3.4 (x, y)-Collusion Resistance

Algorithm 2 for Pr_1

```
1: function QUERY(A parcel $v$, A user $u$)
2:     new[][] = empty map                                   ▷ Same type as $c$
3:     if PreQuery($v$, $u$, $new$) then
4:         for all $z \in zone_v$ do
5:             $d_z[u] \mathrel{+}= id_v$
6:         AddNewCollusions($c$,$new$)
7:         return $o_v$
8:     return $\emptyset$
9: function PREQUERY(A parcel $v$, A user $u$, A map $new$[][])
10:    if $id_v \in d_v[u]$ then return true                  ▷ Access granted
11:    for all $z \in zone_v$ do
12:        if $|d_z[u]| \geq |sig_z| - 1$ then return false   ▷ Access denied
13:    new = NewCollusions($zone_v$, $u$)    ▷ Returns a map of old and new potential
       collusions relative to user $u$ on all parcels of $zone_v$
14:    for all $n \in new$ do
15:        if $|n.value| \geq y - 1$ then    ▷ n.value returns the set of signatures that the
       set of users in n.key are colluding on
16:            return false                                   ▷ Access denied
17:    return true                                            ▷ Access granted
```

Definition 4. *[MultiZone-level collusion] We say that x users collude to compromise y zones if there is y zone-level collusions between those x users on all y zones. A scheme is (x, y)-collusion resistant if it forbids x or less users to collude on y or more zones.*

For instance, a scheme is (4,7)-collusion resistant if it forbids 2, 3, or 4 users from colluding on 7 parcels or more; 5 or more colluding users will not be detected.

In order to implement (x, y)-collusion resistance in our scheme, we record zone-level collusions with a map c associating a combination of $2, 3, ..., x$ users to a combination of 1, 2, ..., $y - 1$ signatures.

We **modify** Algorithm 1 to support (x, y)-collusion resistance and we get Algorithm 2. Querying a vertex v is allowed if the PreQuery condition (line 3) of Algorithm 2 is satisfied. In fact, access to vertex v is allowed if it has been queried (line 10), and it is denied if there is no vertex in $zone_v$ that can accept the token id_v(line 12). Access is also denied if querying the designated parcel causes additional collusions that, if added to the existing ones, can exceed the $y - 1$ threshold (lines 13 to 16). If the PreQuery condition is satisfied, id_v is propagated to all parcels of $zone_v$ (line 5), new collusions (if there is any) are registered (line 6), and the set of owner o_v is returned (line 7).

Note that if $x = 1$, $(1, y)$-collusion resistance will limit every user on y parcels in the whole graph. If $x > 1$ and $y = 1$, $(x, 1)$-collusion resistance will perform a zone-level collusion resistance.

3.5 (x, y, z)-Collusion Resistance

Definition 5. *[z-Zone] Is defined as a subset of the database where the distance dist between any two parcels belonging to the subset is lower than z (where dist is the graph distance function). Formally, a set V' of parcels belong to a z-Zone if, and only if,*

$$v_m, v_n \in V', v_m \neq v_n, 1 \leq dist(v_m, v_n) \leq z \tag{1}$$

The only difference between (x, y)-collusion resistance and (x, y, z)-collusion resistance is the following: (x, y)-collusion resistance considers that y collusions can be anywhere in the database, whereas (x, y, z)-collusion resistance considers that y collusions are all within the same subset of the database, namely in the same z-Zone subset.

Collusions should be tracked on geographical subsets, i.e. the number of allowed collusions between x users on a given z-Zone, where any 2 parcels belonging to that z-Zone are separated by $z - 1$ parcels at most, should be less than y.

Definition 6. *[z-Zone-level collusion] We say that x users collude to compromise y zones belonging to a z-Zone if there is y zone-level collusions between those x users on all y zones. A scheme is (x, y, z)-collusion resistant if it is (x, y)-collusion resistant in that z-Zone.*

The algorithm for (x, y, z)-collusion resistance is the same as the one presented in Algorithm 2, with the following single exception: the NewCollusions method will return a map of old collusions relative to user u falling in the z-Zone of v only, thus removing signatures of parcels that do not belong to the z-Zone from the value field of each entry, in addition to new potential collusions on all parcels of $zone_v$.

It is obvious that the probability of detecting false collusions with (x, y, z)-collusion resistance is lower than the probability of detecting false collusions with (x, y)-collusion resistance, which itself is lower than the probability of detecting false collusions with zone-level collusion resistance only.

Disclosing the values of x, y and z does not compromise the security of the approach. In fact, an attacker might be able to infer the values of x, y, and z by means of trial and error using a known reference set of parcels. For instance, a user might know the owners of parcels in his neighborhood, village, or belonging to a member of his family. By issuing queries on known parcels, this user can derive the values of x, y and z.

3.6 Inference from external knowledge

Most of the people using the cadastral application do have some external knowledge. Very often, they know the owner's name of some parcels from their neighborhood or their village. Because of this external knowledge, users can break Pr_1 without being detected by the inference control mechanisms. Dealing with external knowledge is theoretically impossible since it is simply impossible to know what a given user knows. However, the security administrator can roughly estimate the average level of users' external knowledge. This estimation can help him to choose the proper α value discussed in Section 3.2. An $\alpha = 1$ means the users are assumed to have no external knowledge whereas an $\alpha > 1$ means the users are assumed to have some external knowledge.

3.7 Inference from denial of access

In the framework of multilevel database, Sandhu and Jajodia[19] underlined the fact that a denial of access provides the user with the information that the data s/he is trying to access is highly classified. In the context of our application, if a user is denied an access then s/he can conclude that s/he is about to break prohibition Pr_1.

If the user is trying to break Pr_1 then s/he actually does not learn much from the denial of access. From the parcels s/he has accessed before the denial of access, s/he can simply verify that s/he has queried at least one complete zone. If it is not the case, i.e. s/he has not queried a complete zone, then s/he can only deduce that other users have been querying the same region and that the system has detected a potential collusion attack.

4 Enforcing Pr_2

The basic idea for enforcing Pr_2 is to use the scheme we developed for Pr_1 in the previous section by only modifying the graph definition as follows:

We consider an ownership notion of neighborhood. Two parcels are considered neighbors in the inference graph if they belong to the same owner. In such a graph, vertices belonging to the same owner are all interconnected, forming a complete graph. Formally, $G(E) = \{(p, q) : p, q \in G(V), p \neq q, o_p \cap o_q \neq \emptyset\}$. For instance, Figure 1b shows an inference graph for Pr_2 representing part of the cadastral database (the same part as in Figure 1a), where parcels $\{1, 2, 8\}$ are owned by Joe, $\{4, 5, 6, 7\}$ are owned by Elissa, and $\{3, 5\}$ are owned by Lucy. Notice that parcel 5 has two owners, namely Elissa and Lucy.

It is clear that a zone, as presented in Definition 2, depends only on the inference graph: for Pr_1, the zone of a parcel $\mathbf{p}$ is $\mathbf{p}$ and the set of parcels touching, or located at a given distance from $\mathbf{p}$; for Pr_2, the zone of a parcel $\mathbf{p}$ is $\mathbf{p}$ and the set of parcels owned by the same legal entity.

However, applying the scheme developed for Pr_1, as is, on the resulting graph does not work for the following reasons: – Within the framework of Pr_2, a user with some external knowledge can deduce from a denial of access that s/he has found a land belonging to the target entity. This issue is discussed in Section 4.1; – The definition of z-Zone from Section 3.5 is a geographical one best suited for Pr_1, which is a geographical prohibition. A new definition for Pr_2, which is rather a social prohibition than geographical, is presented in Section 4.2;

Note that we consider that “isolated” parcels, i.e. a parcel belonging to a single owner who himself does not own other parcels, do not fall within the scope of Pr_2: access to these parcels is granted automatically.

4.1 Inference channel from denial of access and external knowledge

First of all, we should notice that in order to be successful, an attacker trying to break Pr_2 should already know the approximate location of all the target entity's parcels. Without this external knowledge, the attacker would need to randomly select parcels from the entire database which is of course infeasible.

Nonetheless, we consider it as a probable attack and we shall address it: let us assume that Bob already knows the approximate location of all Alice's parcels. We also assume that after several queries, Bob has identified several parcels belonging to Alice. If Bob is denied access to an additional parcel then he can reasonably deduce that this parcel belongs to Alice. Returning “access denied” can even be seen as worse than returning Alice's name since it informs Bob that he has found the last parcel in Alice's list of parcels.

One possible solution to prevent Bob from deducing that he has found the last parcel in Alice's parcels list is to increase the α value as discussed in Section 3.2. In that case, Bob would be denied access to Alice's parcels before finding the last parcel. However, there is no solution to prevent Bob from deducing from a denial of access that he has found a parcel belonging to Alice.

Another possible solution would be to return a cover story instead of denying access. A cover story is a lie introduced in the database in order to hide the existence of a sensitive data [7]. Cover stories have mainly been used in the framework of military multilevel databases [10]. In our cadastral application using a cover story would mean returning a fake owner for a given parcel. This solution is of course unacceptable for an official online public cadastral application where answers to query have a legal value and have therefore to be trusted.

We propose another solution: we deny access to the remaining parcel and all its geographical neighbors. In the same example of Alice and Bob, we deny access for the remaining parcel, namely **p**, and all parcels of its geographical zone. This way, we increase the confusion for Bob, thus lowering his confidence in the inference by denial of access from 1/1 (the case where only the remaining parcel is blocked) to $1/n$, where n is the number of geographical neighbors of **p**. This confidence can even be lowered by increasing the number of blocked parcels by including 2^{nd} degree neighbors, i.e. the geographical neighbors of neighbors of **p**.

Algorithm 3 for Pr_2

```
1: function QUERY(A parcel $v$, A user $u$)                          ▷ Same type as $c$
2:     if PreQuery($v$, $u$) then
3:         for all $z \in zone_v$ do
4:             $d_z[u] \mathrel{+}= id_v$
5:         if $|d_v[u]| \geq |sig_v| - 1$ then
6:             r = GetParcelByID($sig_v - d_v[u]$)    ▷ Returns the remaining unqueried parcel
7:             for all $n \in neighbors_r$ do
8:                 $d_n[u] += di_r$ ▷ Blocking token of the ramining parcel assigned to all geographical neighbors
9:         return $o_v$
10:    return $\emptyset$
11: function PREQUERY(A parcel $v$, A user $u$)
12:    if $d_v[u]$ contains a $di$ then return false                   ▷ Access denied
13:    if $id_v \in d_v[u]$ then return true                           ▷ Access granted
14:    for all $z \in zone_v$ do
15:        if $|d_z[u]| \geq |sig_z| - 1$ then return false             ▷ Access denied
16:    return true                                                     ▷ Access granted
```

Considering this, we present a modified version of the model, of Algorithm 1 and of its collusion resistant version, namely Algorithm 2: the only difference is in the way we treat the last parcel in a zone (since we chose to limit access to $|zone_p|-1$ parcels of any zone, otherwise it would be the last α parcels). 1) In the model, every vertex v should be associated with a "blocking token" or "blocking id", denoted di_v. This token acts as a boolean flag: its presence indicates a denial of access and can only exist in the set of geographical neighbors of v, namely *neighbors*$_v$. We opted to use this method instead of actually putting a boolean flag in the model for 1 reason: to keep the resetting scheme of Section 5.2 simpler. 2) The modification of Algorithm 1 is reflected in Algorithm 3: in the PreQuery condition, if a vertex contains a "blocking token" access is immediately denied (line 12). When the limit $|sig - 1|$ is reached, the geographical neighbors of the remaining parcel are filled with the "blocking token" of the remaining parcel (lines 5 to 8; the remaining parcel is already blocked, i.e if the user wants to query the remaining parcel s/he will fail for not satisfying the PreQuery condition). 3) The modification of Algorithm 2 is reflected in Algorithm 4: in the PreQuery condition, if a vertex contains a "blocking token" access is immediately denied (line 14). When the limit $|sig - 1|$ is reached, the geographical neighbors of the remaining parcel are filled with the "blocking token" of the remaining parcel (lines 6 to 9; the remaining parcel is already blocked for the same reason presented earlier in point 2.).

4.2 (x,y,z)-Collusion Resistance

Algorithm 4 for Pr_2

```
 1: function QUERY(A parcel $v$, A user $u$)
 2:     new[][] = empty map                                   ▷ Same type as $c$
 3:     if PreQuery($v$, $u$, $new$) then
 4:         for all $z \in zone_v$ do
 5:             $d_z[u]$ += $id_v$
 6:         if $|d_v[u]| \geq |sig_v| - 1$ then
 7:             r = GetParcelByID($sig_v - d_v[u]$)    ▷ Returns the remaining unqueried
    parcel
 8:             for all $n \in neighbors_r$ do
 9:                 $d_n[u] += di_r$ ▷ Blocking token of the ramining parcel assigned to all
    geographical neighbors
10:         AddNewCollusions($c$,$new$)
11:         return $o_v$
12:     return $\emptyset$
13: function PREQUERY(A parcel $v$, A user $u$, A map $new$[][])
14:     if $d_v[u]$ contains a $di$ then return false          ▷ Access denied
15:     if $id_v \in d_v[u]$ then return true                   ▷ Access granted
16:     for all $z \in zone_v$ do
17:         if $|d_z[u]| \geq |sig_z| - 1$ then return false     ▷ Access denied
18:     new = NewCollusions($zone_v$, $u$)     ▷ Returns a map of old and new potential
    collusions relative to user $u$ on all parcels of $zone_v$
19:     for all $n \in new$ do
20:         if $|n.value| \geq y - 1$ then    ▷ n.value returns the set of signatures that the
    set of users in n.key are colluding on
21:             return false                                   ▷ Access denied
22:     return true                                            ▷ Access granted
```

For this collusion resistance level, we define a distance function $dist_{social}$ as follows:

$$dist_{social} : V^2 \rightarrow \mathbb{Z} \ . \tag{2}$$

$dist_{social}$ returns the smallest social distance between the owners of 2 parcels according to some social relationship (friend, friend of friend, father, grand-child, etc). This distance function is essential to the definition of a z-Zone in Pr_2: Definition 5 is substituted by Definition 7.

Definition 7. *[z-Zone] Is defined as a subset of the database where the distance $dist_{social}$ between any two parcels belonging to the subset is lower than z . Formally, a set V' of parcels belong to a z-Zone if, and only if,*

$$v_m, v_n \in V', v_m \neq v_n, 1 \leq dist_{social}(v_m, v_n) \leq z \tag{3}$$

In Pr_2 collusions should be tracked on social subsets, i.e. the number of allowed collusions between x users on a given z-Zone, where the owners of any 2 parcels belonging to that z-Zone are separated by z social relationships at most, should be less than y.

The algorithm for (x, y, z)-collusion resistance in Pr_2 is the same as the one presented in Algorithm 4, with the following single exception: the NewCollusions method will return a map of old collusions relative to user u falling in the z-Zone of v only, thus removing signatures of parcels that do not belong to the z-Zone from the value field of each entry, in addition to new potential collusions on all parcels of $zone_v$.

5 Life-cycle

So far, we have considered a static database, where a graph is built from parcel data. Afterwards, the graph is initialized with tokens, access to the parcels is handled according to described schemes, with the desired collusion resistance level. But in fact, the cadastral application is highly dynamic, and our approach as presented in Section 3 falls short in addressing issues raised by the the following: 1) Mutation operations (i.e. buy, sell, merge and split), discussed in Section 5.1, and 2) Resetting the scheme for continuous access, discussed in Section 5.2.

5.1 Mutations

Four cadastral operations (called mutations) are performed daily on the database: – Buy and Sell: a parcel's ownership is transferred from its original owner to a new person, affecting the topology of the graph in Pr_2 only; – Merge and Split: two or more parcels are merged (split) into a single parcel (multiple parcels), affecting the topology of the graph in Pr_1 and Pr_2.

For any operation, existing vertices (and edges, if needed) should be removed and replaced by new ones (with new ids): the ownership information is to be protected and the result of any mutation operations changes this information.

Therefore, new vertices that are produced by mutation operations should be considered as new and unqueried. These operations affect the signature of designated parcel(s) and all neighbors, hence its (their) zone and zones of all neighbors.

We developed a passive strategy to remove all traces of old parcels and their access tokens that applies for both Pr_1 and Pr_2 (except for steps related to dis which renders them specific to Pr_2): 1) initially, the querying history of every new parcel is empty, i.e. d is empty for all users; 2) old ids are removed from old neighbors, i.e. removed from the signature sig and the query history d of neighbors of new parcels; old dis of old parcels are removed from the query history of their neighbors; 3) signatures of new parcels' neighbors are modified, i.e. the id of a new parcel is added to the signature of all its neighbors; 4) new parcels inherit access tokens from their neighbors if they were queried, i.e. d of new parcels contains the id of every queried neighbor; new parcels inherit blocking tokens issued by their neighbors; 5) finally, every signature containing an old id is removed from every entry in the collusion map c .

The details of these strategies are omitted due to space limitations. One would be tempted to define an active strategy where new parcels inherit the query state of their parents (parcels they are replacing). While it could be reasonable for buy/sell operations, merge and split operations turn out to be problematic. For instance, if a user queried 1 parcel out of $V' = \{v_1, v_2, v_3\}$ and V' is merged into a new parcel v_4, what should be the state of v_4? Is it fair to consider it as queried? If yes, zones of non-queried parcels from V' and their neighbors would be affected and may add fake collusions to the account of the user. Similarly, if v_4 is split into V', what should be the state of V'. Is it fair to consider all new parcels as queried? If yes, then fake collusions from V' and its neighbors could be added to the account of the user.

5.2 Gradually Resetting the scheme

A simple analysis of the presented scheme shows that, given enough time, the inference graph becomes saturated, restricting existing users on a set of parcels defined by their own querying behavior. Access rights of new users are influenced by previous users' behavior.

To eliminate this issue, we propose a periodical soft resetting strategy. First of all, we add (to our model) a global timer, ticking every given unit of time. We also define a map e to manage the expiry date of access tokens and collusions. e associates a tuple (u, id) with a non-negative integer. Initially, all entries of e are set to 0.

Timer ticks are separated by a time-span t: the minimum time a user should wait before s/he is granted access to a full zone (given that none of its parcels was the subject of any mutation). On every timer tick, a tuple (u, id) holds one of the following values: **0 :** The parcel has never been queried; **1 :** The parcel has been queried, but its zone, and the zones of all of its neighbors has never been maximized; >**1 :** The parcel has been queried, and at least one of the zones that id belongs to, has been maximized by u.

If the value is 1, (i) queries executed by the user are legitimate (so far) if there is no collusion on id, or (ii) the user has been waiting for at least t unit of times to access the complete zone. If the value is > 1, the user is a probable attacker. A value higher than 2 suggest a persistence from the user to gain full access to a zone of a parcel, therefore s/he should wait for longer time-spans before re-gaining access to the complete zone.

Algorithm 5 for Pr_1

```
 1: function QUERY(A parcel $v$, A user $u$)
 2:     new[][] = empty map                                  ▷ Same type as $c$
 3:     if PreQuery($v$, $u$, $new$) then
 4:         for all $z \in zone_v$ do
 5:             $d_z[u]$ += $id_v$
 6:         AddNewCollusions($c$,$new$)
 7:         e($u$, $id_v$) = 1
 8:         return $o_v$
 9:     else
10:         if $new$ is not empty then
11:             e($u$, $id_v$) += 1
12:     return $\emptyset$
13: procedure ONTIMERTICK
14:     for all $entry \in e$ do
15:         if $entry.value = 1$ then
16:             v = GetParcelByID($entry.key$[1])      ▷ entry.key[1] returns the second
    item in the key tuple, namely parcel $id$
17:             $d_v[entry.key[0]]$ -= $id_v$
18:             RemoveCollusions($entry.key$[0], $sig_v$)      ▷ entry.key[0] returns the first
    item in the key tuple, namely the user
19:         if $entry.value > 0$ then entry.value -= 1
```

Algorithm 5 shows a modified version of the Query function from Algorithm 2 for Pr_1 adding support for the resetting scheme (the PreQuery function does not change therefore it is not included in this algorithm). If the PreQuery condition for user u on parcel v is met, $e(u, id_v)$ is set to one, marking it for removal after a single timer tick (line 7). If the PreQuery condition was not met because of a collusion attempt (line 10), the user is penalized by postponing the reset of this particular parcel another tick (line 11).

In addition to the modification of the Query function, Algorithm 5 introduces a procedure, OnTimerTick, that is scheduled for execution on every t units of time. This procedure routinely releases access tokens (line 17) and collusion entries (line 18) that already expired, then decreases the expiration date of access tokens (line 19).

Algorithm 6 for Pr_2

```
1:  function QUERY(A parcel $v$, A user $u$)
2:      new[][] = empty map                                  ▷ Same type as $c$
3:      if PreQuery($v$, $u$, $new$) then
4:          for all $z \in zone_v$ do
5:              $d_z[u]$ += $id_v$
6:          if $|d_v[u]| \geq |sig_v| - 1$ then
7:              r = GetParcelByID($sig_v - d_v[u]$)      ▷ Returns the remaining unqueried parcel
8:              for all $n \in neighbors_r$ do
9:                  $d_n[u] += di_r$
10:             e($u$, $di_r$) = 1
11:         AddNewCollusions($c$,$new$)
12:         e($u$, $id_v$) = 1
13:         return $o_v$
14:     else
15:         if $d_v[u]$ contains a $di$ then
16:             $blocking$ = GetAllDis($d_v[u]$)
17:             for all $b \in blocking$ do
18:                 e($u$, $b$) += 1
19:         if $new$ is not empty then
20:             e($u$, $id_v$) += 1
21:     return $\emptyset$
22: procedure ONTIMERTICK
23:     for all $entry \in e$ do
24:         if $entry.value = 1$ then
25:             v = GetParcelByIDorDI($entry.key[1]$)
26:             if $d_v[entry.key[0]]$ is an id then
27:                 $d_v[entry.key[0]]$ -= $id_v$
28:             else
29:                 $d_v[entry.key[0]]$ -= $di_v$
30:             RemoveCollusions($entry.key[0]$, $sig_v$)
31:         if $entry.value > 0$ then entry.value -= 1
```

For Pr_2, 3 additional modifications should be applied: 1) the map e should accept **id**s and **di**s (introduced in Section 4.1). 2) Algorithm 5 is modified and this modification is reflected in Algorithm 6. When the limit $|sig - 1|$ is reached, all geographical neighbors of the remaining parcel are filled with the "blocking token" of the remaining parcel (lines 6 to 9; the remaining parcel is already blocked, i.e if the user wants to query the remaining parcel s/he will fail for not satisfying the PreQuery condition), where they are tracked for removal according to the resetting scheme (line 10). If the PreQuery condition was not met because the user is blocked by one or many blocking tokens (line 15), the user is penalized by postponing the reset of already blocked parcels to another tick (lines 16 to 18). If it was not met because of a collusion attempt (line 19), the user is penalized by postponing the reset of this particular parcel to another tick (line 20). 3) OnTimerTick, that is scheduled for execution on every t unit of times. This procedure releases routinely access tokens (line 27) and collusion entries (line 30) that already expired, then decreases the expiration date of access tokens (line 31).

The choice of t is very important: a small value renders the complete inference control, proposed in this paper, useless; a high value renders the resetting scheme, described in this section, useless too. The value should take into consideration the "normal" behavior of a user when accessing zones, derived from a study of user access patterns on the cadastral database.

6 Related work

The problem of inference and inference control has been heavily studied in the literature [12]. In the domain of relational databases [14,27], Delugach and Hinke [9] developed a system that takes the database schema and a knowledge source as input, then informs database administrators about potential inference channels. Their approach is based on conceptual graphs for knowledge representation. Cuppens and Gabillon [7,8] proposed a method based on coverstories (lies) for closing the inference channels caused by the integrity constraints of a multilevel database. Chen and Chu [5] created a semantic inference model based on data, schema and semantic information which initiated a semantic inference graph to detect inferences while executing queries. Tolas, Farkas, and Eastman [24] extended their previous work [13] on inference control in the presence of database updates, to guarantee confidentiality and maximize availability; a problem that we tackle in Section 5.1. Katos, Vrakas, and Katsaros [15] proposed an approach to reduce inference control to access control, where they consider the probabilistic correlation between attributes in the inference channel. They consider inference control in stochastic channels, while we consider them in deterministic channels.

Concerning data publishing, Yang and Li [26,28] worked on the inference problem in XML documents, showing how users can use common knowledge in conjunction with partially published documents to infer sensitive data. Staddon, Golle and Zimmy [21] showed how data from partial documents, when used with a source of knowledge like the web, can be used to infer hidden information.

Inference is also an issue in micro-data *publishing* (privacy preserving data publishing, or PPDP), where Sweeney [22] shows that 87% of the population in the U.S. had reported characteristics that likely made them unique based only on 3 quasi-identifiers {5-digit ZIP, gender, date of birth}. Therefore removing directly identifying attributes (e.g. name or SSN) from the micro-data before publishing is not enough. Techniques such as $k-anonymity$ [23], $l-diversity$ [17] and *anatomy* [25] were developed to prevent these types of inference, but they target a problem different from ours: these techniques look for the disassociation of data owners and their data, while we want to publish this association as long as it does not violate the given constraints (Pr_1 or Pr_2).

While PPDP focuses on anonymizing datasets before publishing them for later statistical use (by means of generalization, suppression, etc.), privacy preserving data *mining* (or PPDM) does not transform original data before publishing. In fact, in PPDM, data holders provide a querying interface for interested parties so that they can run mining queries on the original data. Data holders must ensure that the result of such queries do not violate the privacy of data subjects. The main technique used is ϵ-differential privacy[11], that shares a lot of similarities with our approach: limiting (and knowing beforehand) the types of queries permitted to be run on the original data and ensuring collusion resistance[18]. However, the problem that ϵ-differential privacy addresses is different from ours: the goal is to use data for statistical purposes, where personal identifying information is not accessible (like PPDP). In addition, ϵ-differential privacy is usually achieved by adding noise to the resulting queries (unacceptable for our problem).

Another close area of research is controlled query evaluation (CQE) [3,4]. In CQE, user's a priori knowledge is taken into account with the history of submitted queries in order to perform inference control. Refusal and lying are employed as means of restriction and perturbation respectively to protect the confidentiality of classified information. CQE cannot identify colluding users.

The closest approach to ours is the one presented by Staddon [20]. In her paper, Staddon presented a dynamic inference control scheme that does not depend on user query history. This implies fast processing time and ensures a crowd-control property: a strong collusion resistance property that not only prevents c collaborating users (where c is the the degree of collusion-resistance) from issuing complementary queries to complete an inference channel. This property also guarantees that "if a large number of users have queried all but one of the objects in an inference channel, then no one will be able to query the remaining object regardless of the level of collusion resistance provided by the scheme". Initially, we tried to adopt this approach to our problem but we faced two issues unsolved in the original paper: (i) objects shared by different channels and (ii) resetting the scheme. Nonetheless, we managed to solve the case of channels sharing multiple objects, but resetting the scheme for continuous inference control turned out to be a major and nontrivial challenge. Furthermore, we failed to justify the crowd-control property that may be suitable in the case of Staddon, but undesirable in ours.

7 Conclusion

In this paper, we presented an approach for inference control in cadastral databases, based on user query history, with different levels of collusion resistance. We then discussed the effects of database updates specific to the cadastral application, namely buy/sell and merge/split operations, and how to deal with these operations in the overall scheme. A gradual re-initialization strategy to allow continuous access to the database is also proposed.

We do realize that achieving inference control efficiently, as proposed, requires an extensive analysis on the target database, and a fine tuning of all the parameters presented in this paper, i.e. x, y and z for (x, y, z)-collusion resistance, and the timespan t for gradual resetting. Tuning these parameters should be done according to the application's requirements: a trade-off between confidentiality of cadastral data and their availability.

This approach can be applied to other types of inference control (geographic or not), by simply modifying the definition of neighborhood (edges) from our definition, to an application-specific one. It is worth noting that we have developed a prototype to simulate inference graphs using eclipse RCP and its zest framework. Sources can be found on Bitbucket (https://bitbucket.org/fearus/rattack/). We are currently researching efficient methods to implement this approach on a real cadastral database. We are also preparing a user access pattern study for the gradual resetting scheme (see Section 5.2).

Acknowledgments The authors wish to thank the following people for fruitful discussions on some legal aspects applying to the cadastral application and for providing them access to a sample cadastral database. – Tania Berthou, Directrice des Affaires Foncières de la Polynésie Française; – Bertrand Malet, Chef de la Division du Cadastre, Direction des Affaires Foncières; – Jean-Louis Garry, Directeur du Service Informatique de la Polynésie Française; – Emmanuel Bouniot, Chef de la Cellule Systèmes d'Informations Géographique du Service Informatique de la Polynésie Française.

References

1. Les guides de la cnil. les collectivités locales. (2009), http://www.cnil.fr/fileadmin/documents/Guides_pratiques/CNIL_Guide_CollLocales.pdf
2. Fiscalité locale et cadastre. (accessed June 2013), http://www.cada.fr/fiscalite-locale-et-cadastre,6090.html
3. Biskup, J., Bonatti, P.: Controlled query evaluation for known policies by combining lying and refusal. Annals of Mathematics and Artificial Intelligence 40(1), 37–62 (2004)
4. Biskup, J., Tadros, C.: Inference-proof view update transactions with minimal refusals. Data Privacy Management and Autonomous Spontaneus Security pp. 104–121 (2012)
5. Chen, Y., Chu, W.: Protection of database security via collaborative inference detection. Intelligence and Security Informatics pp. 275–303 (2008)
6. Conejo, C., Velasco, A.: Cadastral web services in spain (accessed June 2013), http://www.ec-gis.org/Workshops/13ec-gis/presentations/4_sdi_implementaion_I_Conejo_1.pdf
7. Cuppens, F., Gabillon, A.: Logical foundations of multilevel databases. Data & Knowledge Engineering 29(3), 259–291 (1999)
8. Cuppens, F., Gabillon, A.: Cover story management. Data & Knowledge Engineering 37(2), 177–201 (2001)
9. Delugach, H., Hinke, T.: Wizard: A database inference analysis and detection system. Knowledge and Data Engineering, IEEE Transactions on 8(1), 56–66 (1996)

10. Denning, D.E., Lunt, T.F., Schell, R.R., Shockley, W.R., Heckman, M.: The seaview security model. In: Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on. pp. 218–233. IEEE (1988)
11. Dwork, C.: Differential privacy: A survey of results. In: Theory and Applications of Models of Computation, pp. 1–19. Springer (2008)
12. Farkas, C., Jajodia, S.: The inference problem: a survey. ACM SIGKDD Explorations Newsletter 4(2), 6–11 (2002)
13. Farkas, C., Toland, T.S., Eastman, C.M.: The inference problem and updates in relational databases. In: Proceedings of the fifteenth annual working conference on Database and application security. pp. 181–194. Das'01, Kluwer Academic Publishers, Norwell, MA, USA (2002)
14. Jajodia, S., Meadows, C.: Inference problems in multilevel secure database management systems. Information Security: An integrated collection of essays 1, 570–584 (1995)
15. Katos, V., Vrakas, D., Katsaros, P.: A framework for access control with inference constraints. In: Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual. pp. 289–297. IEEE (2011)
16. Lunt, T.F.: Aggregation and inference: Facts and fallacies. In: Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on. pp. 102–109. IEEE (1989)
17. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1), 3 (2007)
18. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on. pp. 94–103. IEEE (2007)
19. Sandhu, R.S., Jajodia, S.: Polyinstantiation for cover stories. In: Computer Security-ESORICS 92, pp. 305–328. Springer (1992)
20. Staddon, J.: Dynamic inference control. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. pp. 94–100. ACM (2003)
21. Staddon, J., Golle, P., Zimny, B.: Web-based inference detection. In: Proceedings of 16th USENIX Security Symposium. pp. 71–86 (2007)
22. Sweeney, L.: Simple demographics often identify people uniquely. Health (San Francisco) pp. 1–34 (2000)
23. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(05), 557–570 (2002)
24. Toland, T., Farkas, C., Eastman, C.: The inference problem: Maintaining maximal availability in the presence of database updates. Computers & Security 29(1), 88–103 (2010)
25. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: Proceedings of the 32nd international conference on Very large data bases. pp. 139–150. VLDB Endowment (2006)
26. Yang, X., Li, C.: Secure xml publishing without information leakage in the presence of data inference. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. pp. 96–107. VLDB Endowment (2004)
27. Yip, R., Levitt, E.: Data level inference detection in database systems. In: Computer Security Foundations Workshop, 1998. Proceedings. 11th IEEE. pp. 179–189. IEEE (1998)
28. Yixiang, D., Tao, P., Minghua, J.: Secure multiple xml documents publishing without information leakage. In: Convergence Information Technology, 2007. International Conference on. pp. 2114–2119. IEEE (2007)