Dynamic Watermarking-Based Integrity Protection of Homomorphically Encrypted Databases – Application to Outsourced Genetic Data

David Niyitegeka[1(✉)], Gouenou Coatrieux[1,3], Reda Bellafqira[1], Emmanuelle Genin[2], and Javier Franco-Contreras[3]

[1] IMT Atlantique, Unité INSERM UMR 1101 LaTIM, Brest, France
david.niyitegeka@imt-atlantique.fr
[2] Unité INSERM UMR 1078, Brest, France
[3] WaToo, Brest, France

Abstract. In this paper, we propose a dynamic database crypto-watermarking scheme that enables a cloud service provider (CSP) to verify the integrity of encrypted databases outsourced by different users. This scheme takes advantage of the semantic security property most homomorphic cryptosystems have, so as to embed a watermark into encrypted data without altering users' data. The incorrect detection of the watermark, not only informs the CSP the database has been illegally modified but also indicates which data have been altered. In addition, the proposed scheme is dynamic in the sense the watermarking and integrity verification processes can be conducted along the database lifecycle, i.e. even when the database owner updates his or her data (i.e. addition, suppression or modification of database elements). Experimental results carried out with the Paillier cryptosystem on a genetic database demonstrate that our method can efficiently detect different illegal data tamper with a high location precision.

Keywords: Confidentiality · Data outsourcing · Database watermarking · Genetic data · Homomorphic encryption · Integrity

1 Introduction

Nowadays, cloud computing allows data owners to flexibly store and process large amounts of data remotely, without a need to purchase and maintain their own infrastructure. Despite these benefits, such a data outsourcing induces critical security issues especially in terms of data confidentiality and integrity. Indeed, users lose the control over the data they outsource. Among available security mechanisms, encryption ensures the confidentiality of data. In cloud environment, homomorphic encryption has recently gained in interest due to the fact that it allows performing linear operations (e.g. "+", "×") onto encrypted data with the guarantee that the decrypted result equals the one carried out onto unencrypted data [1]. Information can thus be processed without accessing it in a clear form. In this work, we are interested in giving to the

C. D. Yoo et al. (Eds.): IWDW 2018, LNCS 11378, pp. 151–166, 2019.
https://doi.org/10.1007/978-3-030-11389-6_12

cloud service provider the capacity to verify the integrity of databases outsourced homomorphically encrypted by their owners with the help of watermarking, under the constraint that users can update their data (i.e. modify, suppress or add data).

Different tools can be used so as to verify the integrity of a database such as: digital signatures (DS) [2], message authentication code (MAC) [3] or, more recently, watermarking. DS and MAC are common solutions exploited in database management systems (DBMS). However, they introduce additional pieces of information into the database. On the contrary, watermarking relies on the imperceptible insertion of a message into the data by modifying them based on the principle of controlled distortion. As defined, watermarking leaves access to the data while maintaining them protected by the message. Depending on the relationship between the message and the host data, one can ensure various security services like integrity control, in particular.

The first database watermarking method was introduced by Agrawal *et al.* [4]. As most database watermarking schemes, this one focuses on copyright protection or traitor tracing applications where the message corresponds to the buyer or user identifier [5, 6]. Embedding is conducted in a robust way so as to be able to retrieve the identifier even if the watermarked database has been modified. Some watermarking methods have been especially designed in order to verify the integrity of databases. Contrarily to the previous schemes, these ones embed a "fragile" watermark that will not survive any database modifications [7–11]. Sometimes, such schemes provide the capability to identify which database elements have been altered [9]. These methods are either distortion-free or reversible. Distortion-free methods do not modify the values of the database elements. The database is watermarked: by introducing new data, like some "virtual" attributes where the watermark is dissimulated [8] or by modulating the organization of the database elements (i.e. tuples or attributes [7]). Regarding reversible methods, they ensure it is possible to invert the insertion process and to remove the watermark distortion restoring thus the original attribute values of the database. They are well adapted for verifying the integrity. In particular, one can insert a digital signature of the database itself. At the verification stage, the digital signature is extracted and compared with the one computed on the restored database. Such an approach has been proposed either for numerical data [10] or categorical data [11], applying reversible histogram shifting or difference expansion modulations. It is important to notice that the above methods have several limitations. All of them work on static databases i.e. on databases that are not updated. Moreover, they consider tuple additions, suppressions or modifications as non-authorized modifications. Distortion-free methods can localize modifications but without a really good precision (i.e. tuple level at best). On their side, reversible methods only indicate whether a database has been modified. There is thus a need for a watermarking scheme capable to protect database integrity in a dynamic way with also good localization performance while being also compliant with data encryption. Several crypto-watermarking methods, i.e. solutions that combine encryption and watermarking, have been proposed. Most of them focus on multimedia data (e.g. image, video) in order to ensure at the same time data confidentiality and copyright protection in their distribution [12] or watermarking-based integrity and authenticity services from decrypted/encrypted data [13]. Crypto-watermarking schemes can be differentiated depending on whether the embedded message is available in the clear domain, in the encryption domain, or in both domains [14]. To the best of our knowledge, the method

proposed by Xiang et al. [15] is the first that combines watermarking and encryption for the protection of databases. It is based on Order Preserving Encryption (OPE), whose encryption function preserves numerical ordering of the plain-texts, and Discrete Cosine Transformation (DCT). To embed the watermark, the encrypted database is divided into groups, and for each group, DCT coefficients (i.e. DC and AC coefficients) are calculated. AC coefficients are used to generate the watermark bits which are then embedded into the DC coefficients by using quantization index modulation (QIM) [16]. After that, the encrypted and watermarked database can be obtained after executing inverse DCT operations. At the verification stage, integrity of the database can be verified by matching the hash value of AC coefficients and the extracted watermark information from DC coefficients. If this method allows verifying the integrity of an encrypted database, it does not consider update operations. Beyond, it relies on OPE which has several security limitations due to some of its deterministic properties [17].

In this paper, we propose a watermarking method that allows a Cloud Service Provider (CSP) to verify the integrity of homomorphically encrypted databases that are outsourced, handled or updated by their owners remotely. The objective is to detect and localize non-authorized database modifications. To do so, we take advantage of the semantic security property of some homomorphic cryptosystems so as to embed a watermark, a binary message, into encrypted data without altering users' data. As we will see, being available from the hash of subset of attribute values, this message, if not detected properly, not only informs CSP that the database has been modified but also indicates which data have been altered. Contrarily to all the above schemes, the proposed solution is dynamic in the sense the watermarking and integrity verification processes can be conducted along the database lifecycle.

The rest of this paper is organized as follows. In Sect. 2, we come back on some homomorphic encryption preliminaries as well as on the database outsourcing scenario we consider. Section 3 provides the details of the proposed solution while experimental results and performance and security analysis are given in Sect. 4. Conclusions and some perspectives are drawn in Sect. 5.

2 Homomorphic Encryption Preliminaries and Data Outsourcing Scenario

2.1 Homomorphic Encryption: Paillier Cryptosystem

In this work, we opted for the well-known asymmetric Paillier cryptosystem because of its additive homomorphic properties and its simplicity of use [18]. Its principles are as follows. Let p and q be two large prime numbers, the user public key is given by $K_p = pq$. Let $Z^*_{K_p^2}$ denotes the set of integers in $Z_{K_p^2} = \left\{0, 1, .., K_p^2 - 1\right\}$ that have multiplicative inverses modulo K_p^2, and select $g \in Z^*_{K_p^2}$ such that $gcd\left(L\left(g^{K_s} \, mod \, K_p^2\right), K_p\right) = 1$, where: $gcd(.)$ is the greatest common divisor function, $L(s) = (s-1)/K_p$ and $K_s = lcm(p-1, q-1)$ defines the user private key with $lcm(.)$ the least common multiple function. The cipher-text of the clear message $m \in Z_{K_p}$ is derived as

$$c = E[m, r] = g^m r^{K_p} \, mod \, K_p^2 \tag{1}$$

where $E[.]$ is the encryption function and $r \in Z_{K_p}^*$ is a random integer that ensures the Paillier cryptosystem satisfies the so-called "semantic security". More clearly, the same plain-text has different cipher-texts depending on the value of r. The decryption of the cipher-text c is based on the decryption function $D[.]$ such that

$$m = D[c, K_s] = L(c^{K_s} \, mod \, K_p^2)/L\left(g^{K_s} \, mod \, K_p^2\right) mod \, K_p \tag{2}$$

This cryptosystem has additive homomorphic properties. Considering two plain-texts m_1 and m_2, we have

$$E[m_1, r_1] * E[m_2, r_2] = E[m_1 + m_2, r_1 r_2] \tag{3}$$

$$E[m_1, r_1]^{m_2} = E\left[m_1 m_2, r_1^{m_2}\right] \tag{4}$$

As we will see in Sect. 3, both semantic security and additive homomorphic properties of the Paillier cryptosystem will be of importance in our scheme.

2.2 Data Outsourcing Scenario and Database Model

The scenario we consider is given in Fig. 1, where a data owner securely outsources his database into the cloud after independently homomorphically encrypting its elements. By doing so, the owner can ask the cloud service provider (CSP) to process his data while preserving their confidentiality. Herein, the CSP honestly stores and processes encrypted data uploaded based on the owners' requests (processing, updating data). The CSP is not malicious and will not try to alter owners' data. At least, it can be curious, aiming at inferring user data. These privacy issues are however out of the scope of this work where we focus on the verification by CSP of the integrity of the encrypted data under his responsibility. Notice that CSP that is authorized to store users' data with the help of sub-contracted service providers that can be malicious.

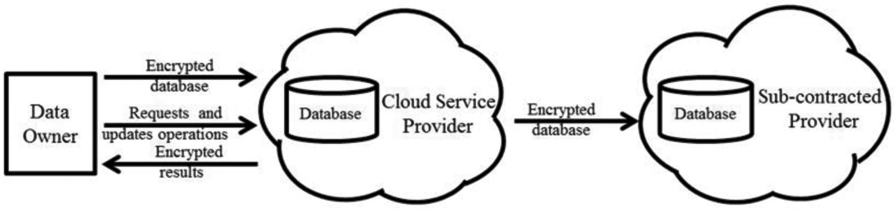

Fig. 1. The considered data outsourcing scenario

In the sequel, we consider the relational database model. A database DB is composed of a finite set of tables $\{T_i\}_{i=1,\ldots,N}$. From here on and for sake of simplicity, we use a database constituted of one single table of u tuples $\{t_i\}_{i=1,\ldots,u}$, where each tuple has m attributes $\{A_1, A_2, \ldots, A_m\}$. The attribute A_j takes its values within an attribute domain and $t_i.A_j$ refers to the value of the j^{th} attribute of the i^{th} tuple. In a database, each tuple is uniquely identified by either one attribute or a set of attributes which is called primary key, noted $t_i.PK$. The encrypted version DB_e of DB is obtained by independently encrypting the values $\{t_i.A_j\}_{i=1,\ldots,u,j=1,\ldots,m}$ using Eq. (1)

$$E\left[t_i.A_j, r_{ij}\right] = g^{t_i.A_j} r_{ij}^{K_p} \, mod \, K_p^2 \tag{5}$$

where K_p is the public key of the database owner and $r_{ij} \in Z_{K_p}^*$ is a random integer.

The objective we pursue in this work is to allow the Cloud Service Provider to protect DB_e in terms of integrity using watermarking under the constraint not altering the owners' data and that data can be updated during time. To do so, and as we will see, we will take advantage of the semantic security property of homomorphic encryption. It is important to notice that all modifications conducted at the request of one data owner, i.e. deletion, addition or modification of tuples or attributes, are authorized. Modifications resulting from system errors (e.g. transmission or storage errors) or from malicious actions, by an intruder for instance, should be detected.

3 Watermarking of Homomorphically Encrypted Database

In this section, we first present our watermarking method for protecting the integrity of homomorphically encrypted databases in the case of "static" databases, before extending it to "dynamic" databases, i.e. when data are remotely updated by their owners.

3.1 Watermarking of Static Homomorphically Encrypted Database

The general architecture and principles of our system are illustrated in Fig. 2. It is based on two main processes: database protection and database integrity verification. The protection process (see Fig. 2a), takes as input an encrypted database DB_e in order to embed a message M that will be available in the encrypted domain. This process stands on three steps: a preprocessing step the purpose of which is to secretly reorganize the database DB_e into DB_e^r based on the secret watermarking key K_w, followed by the insertion of M in DB_e^r to produce DB_e^{wr}, and the back reorganization of DB_e^{wr} into the watermarked encrypted database DB_e^w. The verification process works in a similar way (see Fig. 2b). Considering a protected database $\widehat{DB_e^w}$, based on the secret watermarking key K_w, it first secretly reorganizes $\widehat{DB_e^w}$, elements; the message $\widehat{M}$ is extracted and compared to the message M. Any differences between these two messages will: *(i)* alert the CSP of the database integrity loss; *(ii)* identify which attribute values have been altered. We detail these different steps in the sequel.

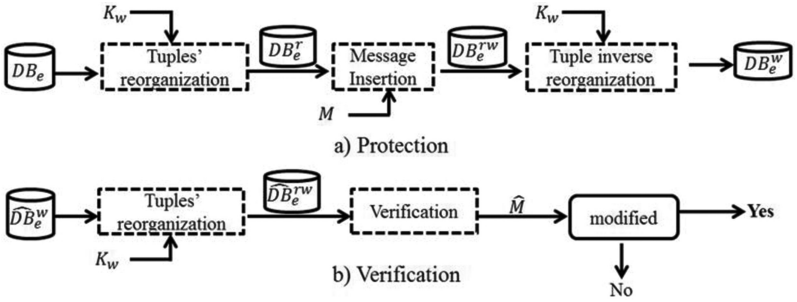

Fig. 2. General architecture of the proposed system. M, $\widehat{M}$ and K_w are the embedded message, extracted message and the secret watermarking key, respectively.

Data Protection. This process is constituted of three main steps:

Preprocessing – Secret Database Reorganization. The purpose of this step is to ensure that a non-authorized user cannot access to M. It basically consists in secretly reorganizing the database DB_e into the database DB_e^r based on the secret watermarking key K_w. In this work, we reorganize the database tuples in the ascending order of the cryptographic hash values: $hash(t_i) = hash(K_w||E[t_i.PK, r_{iPK}])$, where: '||' represents the concatenation operator, $t_i.PK$ is the primary key of the tuple t_i. and *hash* is the cryptographic Secure Hash Algorithm-2(SHA-2). The security of this procedure thus relies on the one of SHA-2 and, in particular, on its collision and diffusion properties [19], as well on the knowledge of the watermarking key K_w.

Message Embedding for Integrity Control. The basic idea of this process stands in the embedding of one bit of the message M into the hash value of a subset of encrypted attribute values of the database. Assuming the database is constituted of k subsets, M is thus a sequence of k uniformly distributed bits ($M = \{b_l\}_{l=1..k}$, $b_l \in \{0, 1\}$) secretly generated based on the watermarking key K_w. The integrity of the database will be verified by checking the presence of M into the subset hash values. Working with subsets provides the capacity to identify which parts or attributes of the database have been altered. This insertion step relies on two sub-steps:

i. *Database partitioning in subsets* – As illustrated in Fig. 3, the secretly reorganized encrypted database DB_e^r is partitioned into k overlapping 'subsets' $\{B_l\}_{l=1...k}$ of 3×3 elements.
ii. *Insertion of one bit b_l of M in an attribute subset B_l* – B_l is watermarked into B_l^w such that $b_l = hash\left(B_l^w\right)_v = s_v$, where s_v represents the v^{th} bit of the cryptographic S hash of B_l^w, *i.e.* $S = hash\left(B_l^w\right)$. The choice of the value of v depends on the watermarking key K_w. Based on the fact it is not possible to predict the SHA-2 output for a given input, we use an iterative procedure so as to watermark B_l into B_l^w. The center element of a subset (e.g. $E\left[t_i.A_j, r_{ij}\right]$ of the subset B_l in Fig. 3) is modified taking advantage of the homomorphic and semantic properties of the Paillier cryptosystem as follows

$B_l^w = B_l$;
While $b_l \neq hash(B_l^w)_v$
$\alpha = rand(.)$;
$E[t_i.A_j, r_{ij}\alpha] = E[t_i.A_j, r_{ij}]E[0, \alpha]$; % $E[t_i.A_j, r_{ij}]$ center element of B_l^w
End

where *rand(.)* is a uniform random function in $Z_{K_p}^*$. Due to the "strength" of SHA-2, there is half a chance to get the correct value of s_v at each iteration.

Back Reorganization of the Watermarked Database DB_e^{rw}. Once all subsets watermarked, the database DB_e^{wr} is reorganized back so as to give access to the encrypted watermarked database DB_e^w.

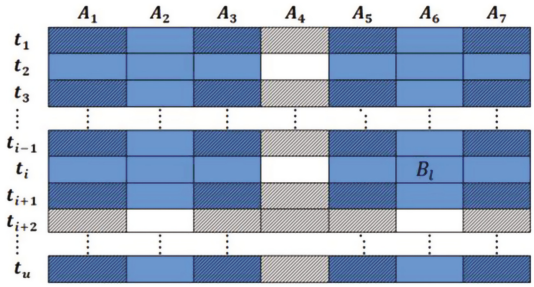

Fig. 3. Partitioning of DB_e^r into subsets. Blue areas represent subsets in the database and hatched areas represent the intersection between blue subsets and grey subsets. Note that standalone attribute values are regrouped into independent subsets. (Color figure online)

Message Extraction and Database Integrity Verification. The integrity verification of a protected database works in a similar way as the database protection. Let us consider a suspicious database $\widehat{DB_e^w}$. Based on the secret watermarking key K_w, $\widehat{DB_e^w}$ is secretly reorganized into $\widehat{DB_e^{rw}}$ and partitioned into subsets. The cryptographic hash value of each subset is computed and the bits of the message $\widehat{M}$ are extracted from these hashes. Any differences in between $\widehat{M}$ and the a priori known or original embedded message M will indicate the database has been altered. It is also possible to identify/localize which subsets have been modified. As we will see in the experimental section, this protection allows detecting different attacks like tuple suppression, tuple addition or modification of encrypted attribute value.

3.2 Watermarking of Updatable Homomorphically Encrypted Database

In this scenario, the user is allowed to remotely update the database. Database tuples can be added, suppressed or modified. Being requested by an authorized owner, such an update should not be at the origin of an alarm. We want to detect unauthorized modifications like addition, suppression or modification of tuples by an intruder, for instance.

Rather than re-watermarking the whole database using the previous scheme, we propose a dynamic watermarking method which allows protecting the database integrity on the fly with the capability to localize data modifications as before. To do so, our scheme takes advantage of a journal table J_t that contains some pieces of information such as the historical details of all added or suppressed tuples. Beyond, the protection and verification processes of this scheme are similar to those depicted in Sect. 3.1.

To give an idea about how our solution works, let us consider an already protected database DB_e^w along its journal J_t. As shown in Table 1, one record of J_t is associated to one tuple of DB_e^w. Its components correspond to: the tuple identifier (e.g. the encrypted primary key $E[t_i.PK, r_{iPK}]$), the action applied to this tuple: addition (A) or suppression (S); and the binary message (m_i) that has been embedded into the tuple. J_t is organized according to the chronological order database elements have been updated. As we will see in the rest of this section, this organization will be used for verifying the integrity of the database. It is important to notice that the elements of J_t should only be known from the CSP. To do so, the CSP encrypts J_t record elements (see Table 1) and permutes records using a permutation algorithm (PA) parameterized by a secret permutation key K_π. PA is used in order to hide the chronological order of J_t records. We will come back on the security of this journal in Sect. 4.3.

Table 1. A sample view of the journal table J_t.

Added(A)/Suppressed(S)	Identifier (Id_i)	Embedded message (m_i)
$E[A, r_{1a}]$	$E[Id_1, r_1]$	$E[m_1, r_{m_1}]$
$E[A, r_{2a}]$	$E[Id_2, r_2]$	$E[m_2, r_{m_2}]$
$E[S, r_{3s}]$	$E[Id_3, r_3]$	$E[m_3, r_{m_3}]$

In the following, we go into the details of our scheme. For the sake of simplicity, we first present how it works when new tuples are added, before presenting its principles when considering tuple suppression and authorized attribute value modification.

Protection on the Fly When Adding One New Tuple. To illustrate this process, let us consider an encrypted watermarked database DB_e^w constituted of only two tuples as shown in Fig. 4a. When a new homomorphically encrypted tuple indexed by t_i is added by a user, the CSP conducts the following steps:

1- The CSP decrypts J_t and reorganizes the records of J_t in their chronological order using the permutation algorithm parameterized with the secret key K_π. Then the CSP looks for the two previous last added tuples (that is to say the two last lines of the database DB_e^w – see Fig. 4a) accordingly to J_t.

2- As exposed in Fig. 4b, the CSP concatenates the new tuple to the two previous ones and computes the relative attribute subset partitioning. This partition depends on the position of the tuple in the database and can be simply computed based on t_i; computation we cannot detail due to paper length limitation.
3- The cloud watermarks this set of tuples accordingly the two following sub-steps:
   a. Bits of the message M are extracted from pre-existing but incomplete sub-sets (i.e. subsets some attributes of which do not exist - e.g. B_1 in Fig. 4a) and are next re-inserted into these subsets once these ones completed with the attribute values of the new added tuple (see new version of B_1 on Fig. 4b).
   b. New subsets, created after the addition of the new tuple (e.g. B_4 in Fig. 4b), are watermarked. To do so, the CSP secretly generates a sequence of bits based on the watermarking key K_w, i.e. a sub-message m_i.

After watermarking, the CSP adds to J_t the record R_{t_i} such as $R_{t_i} = <A, Id_i = E[t_i.PK, r_{iPK}], m_i>$. J_t is next secretly permuted before is encrypted.

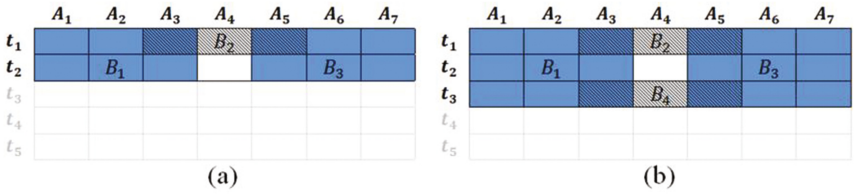

Fig. 4. **(a)** Initial protected database constituted of two tuples. Blue areas represent the uncomplete subsets B_1 and B_3 while grey areas represents the subset B_2. t_3, t_4 and t_5 correspond to empty positions where new tuples should be added. **(b)** The database after concatenation of new tuples to the two previous ones. Blue areas represent the subsets B_1 and B_3 completed. B_4 is a new subset created after the addition of new tuple in the database. (Color figure online)

Protection on the Fly When Modifying an Attribute Value. Let us consider a protected database DB_e^w. If an attribute value $E[t_i.A_j, r_{ij}]$ is updated by its user, the CSP renews the database protection as follows:

1- The CSP decrypts J_t and permutes its records based on K_π.
2- The database records are ordered accordingly the J_t and the CSP finds the position of the tuple t_i as well as the subset partition that corresponds to the attribute value that is updated by its owner.
3- The CSP extracts the message bit embedded from the corresponding subset and re-inserts it once attribute value updated.

Notice that as such an update does not remove or add a new tuple, J_t is not modified.

Protection on the Fly When Suppressing One Tuple. Let us now consider a protected database DB_e^w and that a user modifies it by suppressing the tuple t_i. To update the protection, the CSP proceeds as follows:

1- It decrypts J_t and reorganizes its records. Then, based on J_t, it reorganizes the database and finds the position of the tuple t_i in the database.
2- The CSP extracts the message from the subsets concerned by the suppression and replaces it by an "empty" tuple, that is to say a tuple the encrypted attribute values of which are set to zero or any other predefined value.
3- The CSP re-watermarks subsets using the extracted message while distinguishing two distinct cases:
   a. If the suppressed tuple contains the centers of some subsets, as illustrated in the Fig. 5a, where t_i contains the center of B_l^w, then these subsets are re-watermarked by re-inserting the bit of the message by modifying with the help of iterative procedure presented in Sect. 3.1, one of the encrypted attributes of the tuples t_{i-1} and t_{i+1} out of the intersection of two subsets (as for example one of the attributes identified in B_l^w by a red cross in Fig. 5a).
   b. If the suppressed tuple t_i does not contain the centers of subsets, see Fig. 5b, then the message is extracted from these subsets and re-embedded into them by modifying their center element using iterative procedure presented in Sect. 3.1.

After message embedding, the CSP adds to J_t, the record R_{t_i} associated to the suppressed tuple t_i, and J_t is secretly permuted before is encrypted.

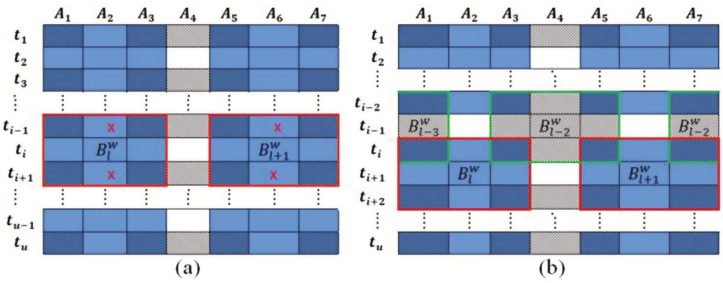

Fig. 5. **(a)** Update of the database protection in the case the suppressed tuple contains the centers of subsets. Red outlines represent subsets concerned by the suppression of the tuple t_i and red cross indicates encrypted attribute values to modify when re-watermarking the data subset. **(b)** Update of the database protection in the case the suppressed tuple do not contain centers of subsets. Red and green outlines represent subsets concerned by the suppression of the tuple t_i. (Color figure online)

Message Extraction and Integrity Verification. Let us consider a protected database $\widehat{DB_e^w}$ the CSP wants to verify the integrity. To do so, the CSP has to conduct following three steps:

1- The CSP decrypts and reorganizes J_t and accordingly reorganizes $\widehat{DB_e^w}$.
2- The CSP starts by verifying the latest tuple updated in the database. For each tuple t_i to verify, the CSP finds tuple's neighbors i.e. $\{t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}\}$.
3- The integrity verification is conducted according to the action applied to t_i and reported in J_t:
   a. If t_i has been added, only t_{i-1} and t_{i+1} are needed so as to compute the subset partition around to t_i. The CSP retrieves these tuples from J_t, identifies the subsets and extracts from them the message it next compares with the ones stored in J_t. Any difference will raise an alarm, indicating an unauthorized alteration of t_i, t_{i-1} or t_{i+1} and the position of the suspicious subsets.
   b. If t_i has been suppressed, the CSP adds an "empty" tuple so as to compute the subset partition. Two cases have to be considered depending on whether the suppressed tuple contains subset centers or not. In the former case, the CSP has to retrieve the tuples t_{i-1} and t_{i+1} (see Fig. 5a) while in second, it needs to access the tuples t_{i-2}, t_{i-1}, t_{i+1} and t_{i+2} (See Fig. 5b). Once subsets constituted, the CSP extract the message bits and compare them to the ones stored in J_t.

It is important to notice that the above procedure allows detecting the tampering of encrypted attributes. Other non-authorized modifications such as addition or suppression of tuples will be identified with the help of J_t. Indeed, added tuple will not be reorganized and will appear as extra data, while suppressed tuples will not be retrieved $\widehat{DB_e^w}$.

4 Experimental Results and Discussion

The proposed watermarking scheme was experimented on a relational database constituted of one table of 10 000 tuples issued from a real genetic database containing pieces of information related to genetic variants of 57 individuals. Each tuple or line containing information about a position in the genome. As shown in Table 2, each tuple is represented by eight attributes that are chromosome (*#chrom*), position (*pos*), identifier (*id*), reference (*ref*), alternative base(s) (*alt*), quality (*qual*), filter status (*filter*) and additional information (*info*). In the sequel, the attribute *pos*, or more clearly its encrypted version, is considered as the primary key as it uniquely identify one tuple.

Table 2. Sample view of the genetic database used in these experiments. One tuple containing information about a position in the genome

#chrom	*pos*	*id*	*ref*	*alt*	*qual*	*filter*	*Info*
21	9825790	.	C	CT, G	1130.64	PASS	CSQ=G
21	9825796	.	C	CGCGT	1179.35	PASS	CSQ=GCGT
21	9825809	.	G	A	1079.59	PASS	CSQ=A

This database was encrypted using Paillier cryptosystem with a public key encoded in 2048 bits so as to ensure a high level of security. In the case of static database watermarking, this table was divided into 11 390 subsets where a uniformly distributed binary message of 11 390 bits was inserted (see Sect. 3.1). The permutation of the journal table J_t records was conducted using the permutation technique based on indices vectors and described in [20]. In the sequel, we evaluate the performance of our scheme in terms of: computation complexity, modification detection and localization precision.

4.1 Computation Complexity

As shown in Sects. 2 and 3, message embedding and integrity verification processes are performed by the CSP on encrypted databases. Whatever the scheme, static or dynamic, message embedding consists in the insertion of one message bit into one subset of encrypted attributes. To do so, an iterative procedure is applied (see Sect. 3.1) such that the v^{th} bit of the hash value corresponds to the bit of the message. At each iteration, the center of the subset (a Paillier encrypted attribute) is multiplied by the Paillier encrypted version of the value zero with a different random value (see iterative procedure in Sect. 3.1). Since the encryption of zero is of higher complexity than a modular multiplication, the computation complexity when watermarking one subset is bounded by $O(L)$ encryptions where L represents the number of iterations. Based on the fact, at each iteration there is one chance out of two that the bit hash value corresponds to the message bit, we thus have in average $L = 2$ with as consequence a watermarking computation complexity bounded by $O(2)$. Considering a static or dynamic database of n subsets, the computation complexity is thus bounded by $O(2n)$ encryptions. Regarding the verification process, its computation complexity mainly depends on the calculation of the subset hash values. However, such computation remains negligible compared to the homomorphic encryption operations. Table 3 provides the computation time for the

Table 3. Computation time for message insertion and integrity verification in the case of the protection of our test database.

Watermarking scheme	Computation step	Computation time
Watermarking of a static encrypted database	Database encryption	24 min 43 s
	Message embedding in the database	7 min 14 s
	Database integrity verification	8 s
Watermarking of a dynamic encrypted database	Message embedding if one tuple is added	0.1 s
	Message embedding if one tuple is suppressed	0.1 s
	Integrity verification for one added tuple	0.07 s
	Integrity verification for one suppressed tuple	0.09 s

message embedding and the integrity verification of our test database. Notice that our method was implemented in C/C++ with GMP library and all experiments were conducted using a machine equipped with 8 GB RAM running on Ubuntu 18.04 LTS.

4.2 Dynamic Database Watermarking and Attack Detection

As stated in Sect. 3.1, three kinds of attacks have to be considered: "*tuple suppression attack*", "*tuple addition attack*" and "*Encrypted attribute value modification attack*". They can be the result of an intrusion in the system or of transmission errors.

***"Tuple suppression"* and *"tuple addition"* Attacks.** For sake of simplicity, let us consider the simple case where a hacker suppresses or adds one tuple in the database DB_e^w. As seen in Sect. 3.2, the integrity verification process relies on pieces information stored in the journal table J_t. In the case of an added tuple, the CSP will not find its identifier in J_t (see Table 1, Sect. 3.2) and will consequently raise an alarm. In the case of a suppressed tuple, the CSP will not be able to retrieve it in the database and will raise an alarm. In this case, in order to pursue the integrity verification of the whole database, the CSP just has to add an empty tuple in $\widehat{DB_e^w}$ (i.e. attacked version of DB_e^w). Based on the tuple identifiers stored in J_t, the detection rate of such attacks is of 100%.

"Encrypted attribute value modification attack". In this attack a non-authorized user conducts homomorphic operations so as to damage or falsify some database elements. Indeed, due to the fact data are homomorphically encrypted with the user public key; he/she can make some operations so as to modify the database attribute values.

One can distinguish different integrity level checking: the subset level and the database level. At the subset level, depending on the subset partitioning (Sect. 3.2 – Fig. 4), if the altered attribute value is not at the intersection of two subsets, the probability of not detecting such a modification is ½ due to the fact there is a half chance that the message bit inserted in the hash subset changes (see Sect. 3.1 – performance detection of SHA-2). If now the modified attribute belongs to two subsets then the probability of non-detection is ¼. In any case, the probability of non-detection in one subset is bounded by ½. At the database level, if a hacker modifies k subsets of DB_e^w, the probability of detection is bounded by $1 - (1/2)^k$ which converges rapidly to 1 with the increase of k.

To experimentally evaluate the performance of our method against this attack, a given percentage of attributes of our protected database DB_e^w were randomly modified: 0.001% (that is to say one attribute value has been modified in the database), 0.003% (three attribute values modified), 25%, 50%, 75%, and 100%. As it can be seen in Fig. 6 the successful detection of a tampered database depends on the number of modified attribute values. For instance, if a hacker only modifies one attribute value we have detection rate of 70%. We also compare in Fig. 6, the experimental detection rate with the theoretical limit. Experimental results provide better performance.

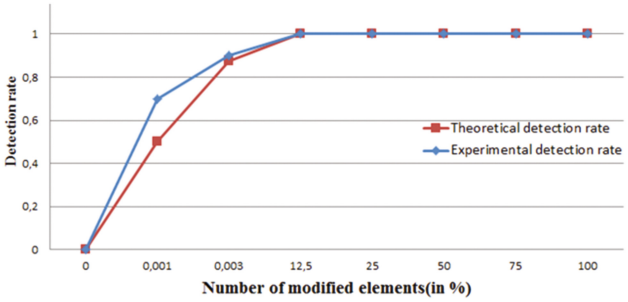

Fig. 6. Theoretical and experimental detection rates of our scheme in the case of the attribute modification attack. Experimental results are given in average.

4.3 Security Analysis

The proposed watermarking scheme allows the CSP to control the integrity of encrypted databases outsourced by different users. In the following, we discuss its security in terms of data confidentiality and data integrity. We first start by analyzing our scheme considering cryptographic attacks, which aims at breaking the data confidentiality, and then focus on watermarking attacks which aims at breaking the integrity protection of the database.

Encryption operations are performed using Paillier cryptosystem. The security analysis of this cryptosystem has been investigated in [18]. Because we only exploit its semantic security and homomorphic properties, there is no access to private parameters like user's data and private key. Even though, a hacker knows some watermarking parameters (e.g. K_w, partitioning), this gives him no additional information about the security parameters of the Paillier cryptosystem. Furthermore, the message embedding does not modify the clear data thanks to properties of homomorphic encryption. Therefore, the decryption operation is not compromised.

Regarding the integrity of the database, there are different attacks that a hacker can perform over the database. For a static database, the message embedding in DB_e and the integrity verification of DB_e^w depend on the watermarking key K_w. Without the knowledge of K_w, it is extremely difficult for an attacker to identify and reorganize the database subsets and find the location of the watermark. In fact, the partitioning and the location of the bits of the message M within the hash of the watermarked subset depend on K_w. Without this key, a hacker cannot distinguish the bits of M from the others. Even though he knows the structure of the message M (the way it is generated), he can only try an exhaustive search until he finds a valid message.

The security of dynamic database watermarking depends on the secret watermarking key K_w and on the security of the journal J_t. To ensure the confidentiality of J_t, this one is encrypted by the CSP which has to keep secret the encryption keys. Notice that if J_t is encrypted using a deterministic cryptosystem, some cipher-texts may leak information about the plain-texts. That will be the case of the first column of the J_t which indicates if a tuple has been added or suppressed. Such an information leak can support chosen plain-text attacks. Beyond, our scheme relies on a secret permutation

algorithm of the records of J_t using a secret permutation key K_π so as to mask the chronological order of users' operations. Without K_π, it should be extremely difficult for an attacker to identify the chronological order of records in J_t and of the tuples in the database neither. In this work we use the algorithm issued from [20] the security analysis of which has been established. Anyway, even though a hacker knows the secret permutation key K_π, he cannot permute J_t because it is encrypted. Similarly, even if a hacker knows the secret decryption key being able to decrypt the J_t, without K_π he cannot conduct the inverse permutation operations and reorganize DB_e^w. He will thus not be able to modify the database while ensuring the correct detection of the watermark. As conclusion, in order to break the integrity of the journal table, a hacker should dispose of both the secret permutation key K_π and the secret journal decryption key of the CSP.

5 Conclusion

In this paper, we have proposed the first watermarking scheme which allows verifying the integrity of homomorphically encrypted databases taking advantage of the homomorphic and semantic security properties of such cryptosystems. Another main originality of this scheme is that it is dynamic, making possible to protect databases that are updated by their owners (e.g. tuple additions, tuple suppressions and encrypted attribute value modifications). Experimental results conducted on a genetic database show that the proposed scheme provides very high detection and localization performance capabilities; better alteration localization performance than watermarking schemes for clear data. Future works will focus on adapting our method on genetic data so as to ensure data integrity control when processing data.

Acknowledgements. This work has received a French government support granted to the Labex CominLabs and managed by the ANR in the "Investing for the future" program under reference ANR-10-LABX-07-01, and to the Labex GenMed, ANR-10-LABX-0013, through the project PrivGen.

References

1. Bellafqira, R., Coatrieux, G., Bouslimi, D., Quellec, G.: Content-based image retrieval in homomorphic encryption domain. In: 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 2944–2947 (2015)
2. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. ACM Trans. Storage **2**(2), 107–138 (2006)
3. Almulla, S.A., Yeun, C.Y.: Cloud computing security management. In: 2nd International Conference on Engineering Systems Management and Its Applications, pp. 1–7 (2010)
4. Agrawal, R., Kiernan, J.: Watermarking relational databases. In: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002), pp. 155–166 (2002)
5. Franco-Contreras, J., Coatrieux, G., Cuppens, F., Cuppens-Boulahia, N., Roux, C.: Robust lossless watermarking of relational databases based on circular histogram modulation. IEEE Trans. Inf. Forensics Secur. **9**(3), 397–410 (2014)

6. Wang, C., Wang, J., Zhou, M., Chen, G., Li, D.: ATBaM: an Arnold transform based method on watermarking relational data. In: International Conference on Multimedia and Ubiquitous Engineering, pp. 263–270. IEEE (2008)
7. Kamel, I., Kamel, K.: Toward protecting the integrity of relational databases. In: World Congress on Internet Security, pp. 258–261. IEEE (2011)
8. Prasannakumari, V.: A robust tamperproof watermarking for data integrity in relational databases. Res. J. Inf. Technol. **1**(3), 115–121 (2009)
9. Guo, H., Li, Y., Liu, A., Jajodia, S.: A fragile watermarking scheme for detecting malicious modifications of database relations. Inf. Sci. **176**, 1350–1378 (2006)
10. Chang, J.N, Wu, H.C.: Reversible fragile database watermarking technology using difference expansion based on SVR prediction. In: International Symposium on Computer, Consumer and Control, pp. 690–693 (2012)
11. Coatrieux, G, Chazard, E., Beuscart, R., Roux, C.: Lossless watermarking of categorical attributes for verifying medical data base integrity. In: 33rd IEEE Annual International Conference of the Engineering in Medicine and Biology Society, pp. 8195–8198 (2011)
12. Memon, N., Wong, P.: A buyer–seller watermarking protocol. IEEE Trans. Image Process. Image **10**, 643–649 (2001)
13. Bouslimi, D., Coatrieux, G., Roux, C.: A joint watermarking/encryption algorithm for verifying medical image integrity and authenticity in both encrypted and spatial domains. In: IEEE Annual International Conference of the Engineering in Medicine and Biology Society, pp. 8066–8069 (2011)
14. Bouslimi, D., Bellafqira, R., Coatrieux, G.: Data hiding in homomorphically encrypted medical images for verifying their reliability in both encrypted and spatial domains. In: Engineering in Medicine and Biology Society. pp. 2496–2499. IEEE (2016)
15. Xiang, S., He, J.: Database authentication watermarking scheme in encrypted domain. IET Inf. Secur. **12**(1), 42–51 (2017)
16. Chen, B., Wornell, G.W.: Quantization index modulation: a class of provably good methods for digital watermarking and information embedding. IEEE Trans. Inf. Theory **47**, 1423–1443 (2001)
17. Xiao, L., Yen, I.L.: Security analysis for order preserving encryption schemes. In: 46th Annual Conference on Information Sciences and Systems (CISS), pp. 1–6 (2012)
18. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, Jacques (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
19. Bakhtiari, S., Safavi-Naini, R., Pieprzyk, J.: Cryptographic hash functions: a survey, Technical report 95–09, Department of Computer Science, University of Wollongong (1995)
20. Radwan, A.G., AbdElHaleem, S.H., AbdElHafiz, S.K.: Symmetric encryption algorithms using chaotic and non-chaotic generators: a review. J. Adv. Res. **7**(2), 193–208 (2016)