DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch131

Real-Valued Negative Databases

Dongdong Zhao[1, 2] and Wenjian Luo [1, 2, *]

[1] School of Computer Science and Technology,
University of Science and Technology of China, Hefei 230027, Anhui, China
[2] Anhui Province Key Laboratory of Software Engineering in Computing and Communication,
University of Science and Technology of China, Hefei 230027, Anhui, China
zdd@mail.ustc.edu.cn, wjluo@ustc.edu.cn

Abstract

The negative database (*NDB*) is the negative representation of original data. Existing work has demonstrated that *NDB* can be used to preserve privacy and hide information. However, most work about *NDB* is based on binary representation. In some applications which are naturally descripted in real-valued space, the binary negative database is hard to be applied appropriately. Therefore, the real-valued negative database is proposed in this paper, and reversing the real-valued negative database is proved to be an *NP*-hard problem. Moreover, an effective algorithm for generating real-valued negative databases is given. Finally, an example of applying the real-valued negative database to the privacy-preserving data publication is descripted, and it shows that the real-valued negative database is valuable in practice.

Introduction

Nowadays, databases have become basic tools for storing data. As the privacy of data is widely concerned, the techniques which can preserve privacy while keeping the database services available are urgently needed. Traditional databases store the data with the form what it actually is. This way is called the positive representation of data, and the databases are called positive databases. The privacy of traditional databases is easy to be revealed when the databases are leaked. Although some cryptography methods can be applied to the positive databases, it is time-consuming to encrypt every entry in the databases and the encrypted databases cannot support basic database operations efficiently. Another way is to control the access of the positive database, but this way cannot eliminate all the security risks as there may be some internal attacks.

The negative database, which is inspired by Natural Immune System, was proposed by Esponda and his colleagues (Esponda et al., 2004a; Esponda et al., 2004b; Esponda et al., 2005; Esponda et al., 2007a; Esponda et al., 2009). In contrast to traditional databases, the negative database only stores the information in the complementary set of the original data. This way is called the negative representation of data. It has been proved that reversing the negative database with the binary representation (i.e. recovering the corresponding binary positive database) is *NP*-hard (Esponda et al., 2004b; Esponda et al., 2009). Therefore, the binary negative database could be employed to protect data privacy. Some algorithms for generating binary negative databases from binary positive databases have been proposed, such as the prefix algorithm (Esponda et al., 2004b; Esponda et al., 2009), the *RNDB* algorithm (Esponda et al., 2004b; Esponda et al., 2009), the q-hidden algorithm (Jia et al., 2005; Esponda et al., 2007a) and the hybrid-*NDB* algorithm (Liu et al., 2011). Furthermore, some basic operations upon the negative database have been proposed, such as the negative Cartesian product, negative join and negative intersection (Esponda et al., 2004a; Esponda et al., 2005; Esponda et al., 2007b).

So far, most work about the negative database is based on the binary representation. However, in some applications which are naturally descripted in real-valued space, the negative database with the binary representation is not appropriate. Therefore, the real-valued negative database is proposed in this paper.

The negative database has already been introduced to some applications such as privacy preserving (Esponda et al., 2004b; Esponda et al., 2007a; Esponda et al., 2009), sensitive data collection (Esponda, 2006; Horey et al., 2007) and authentication (Dasgupta and Azeem, 2007; Dasgupta and Azeem, 2008). In this paper, an example of applying the real-valued negative database to the privacy-preserving data publication is given. This example demonstrates that the real-valued negative database is appropriate for the privacy-preserving data publication.

Existing Work about Negative Databases

The negative database (*NDB*) was proposed by Esponda and his colleagues (Esponda et al., 2004a; Esponda et al., 2004b). Presently, most negative databases are based on the binary representation. The details of the binary negative database are descripted as follows (Esponda et al., 2004b; Esponda et al., 2009).

Assume the original data is a database which consists of n entries, i.e. $DB = \{x_1, x_2, \ldots, x_n\}$, and each entry in DB is a binary string with length m. The universal set is $U = \{0, 1\}^m$.

* Corresponding author. Tel: 86-551-63602824

The complementary set of *DB* is denoted as *U-DB*, and the negative database *NDB* only stores the elements that belong to *U-DB*. As there are usually too many binary strings belong to *NDB*, a "don't care" symbol '*' is introduced to compress *NDB* to a reasonable size. Each entry in *NDB* is a string defined upon the alphabet {0, 1, *} with length *m*. The positions with value 0 or 1 are called specified positions, and those with symbol '*' are called unspecified positions. The symbol '*' represents 0 or 1 at a given position. If all the entries in *U-DB* are covered by *NDB*, *NDB* is said to be complete. Any binary string *s* is said to be matched with (or covered by) an entry *y* in *NDB* if and only if the value at each position of *s* is identical to that of *y* or the corresponding value of *y* is '*'. With the unspecified value '*", multiple different negative databases can be generated from the same positive database.

It has been proved that reversing the binary negative database (i.e. recovering the corresponding binary positive database) is an *NP*-hard problem (Esponda et al., 2004b; Esponda et al., 2009). If reversing a negative database is computationally infeasible, the negative database is said to be hard-to-reverse, otherwise it is said to be easy-to-reverse.

Some algorithms for generating the binary negative database from a binary positive database have been proposed. The prefix algorithm (Esponda et al., 2004b; Esponda et al., 2009) is the first algorithm for generating binary negative databases, and it is compact and efficient. The binary negative database generated by the prefix algorithm is complete but easy-to-reverse. In order to overcome this shortcoming, the *RNDB* algorithm (Esponda et al., 2004b; Esponda et al., 2009) was proposed. The *RNDB* algorithm embeds some random factors for generating binary negative databases which are possibly hard-to-reverse. However, the hard-to-reverse property of the binary negative databases generated by the *RNDB* algorithm could not be guaranteed, and the size of those binary negative databases could be too large. The *q*-hidden algorithm (Jia et al., 2005; Esponda et al., 2007a) was proposed for the binary positive databases that contain only one entry, and it is very efficient. The generated binary negative databases are not complete, but hard-to-reverse on average. The hybrid-*NDB* algorithm (Liu et al., 2011) combines the prefix algorithm with the *q*-hidden algorithm to generate binary negative databases that are both complete and hard-to-reverse on average. It is noted that the "hard-to-reverse" property mentioned here means that the SAT solvers with local search strategy (e.g. WalkSAT (Selman et al., 1995)) could not reverse the negative databases on average.

In real-world applications, real-valued databases are often used. However, it is not convenient to employ the binary negative database to represent a real-valued database. Therefore, the real-valued negative database is studied in this paper. It is noted that earlier work about the negative database is the negative selection algorithm (Forrest et al., 1994; Ji and Dasgupta, 2007). The binary negative database is closely related to the negative selection algorithm with the binary representation (Forrest et al., 1994; Ji and Dasgupta, 2007), while their objectives and generation algorithms are obviously different. Hence, the real-valued negative database is also related to (but different from) the negative selection algorithm with the real-valued representation (González et al., 2003; Ji and Dasgupta, 2004; Ji and Dasgupta, 2006; Ji and Dasgupta, 2007).

The Real-Valued Negative Database

Assume real-valued positive database (*DB*) contains *n* entries, i.e. $DB = \{x_1, x_2, \ldots, x_n\}$. There are *m* attributes $\{R_1, R_2, \ldots, R_m\}$ in *DB*, and the domain of each attribute R_k ($k = 1 \ldots m$) is $I_k = [l_k, u_k]$. l_k is the lower bound and u_k is the upper bound. The bounds l_k and u_k are both real values, i.e. $l_k \in R$, $u_k \in R$. Each entry x_i ($i = 1 \ldots n$) is a vector of *m* real values, and each value $x_i[k]$ ($k = 1 \ldots m$) belongs to the domain of k^{th} attribute, i.e. $x_i[k] \in I_k$.

The real-valued negative database only stores the information that belongs to the complementary set of the real-valued positive database. Since the instances covered by the real-valued negative database are usually too many to be presented exactly, intervals are introduced to compress them.

Suppose *a* is an entry with *m* real values, and *v* is an entry with *m* intervals. Entry *a* is matched with (or covered by) entry *v* if and only if following condition is satisfied.

$$a[k] \in v[k], \quad k = 1, 2, \ldots, m \tag{1}$$

Based on above matching rule, the real-valued negative database for *DB* can be defined as follows.

Definition 1. (*Real-Valued Negative Database*) Giving the real-valued positive database *DB* and the universal set $U = I_1 \times I_2 \times \ldots \times I_m$, the real-valued negative database (*RvNDB*) for *DB* is a compressed representation of *U–DB*. Each entry in *RvNDB* consists of *m* intervals, and does not cover any entries in *DB*.

If *RvNDB* covers the whole complementary set of *DB*, *RvNDB* is said to be complete. Otherwise, *RvNDB* is said to be incomplete. A simple database query can be processed directly upon the real-valued negative database. For any *s* (a vector with *m* real values), if it is covered by *RvNDB*, it does not belong to *DB*; if *s* is not covered by *RvNDB* and *RvNDB* is complete, it belongs to *DB*.

As any two entries in the real-valued negative database may intersect with each other, one real-valued positive database can be mapped to multiple real-valued negative databases. An example is given in table 1.

Table 1. An example of *RvNDB*s

DB	NDB_1	NDB_2
0.2, 0.8	[0, 0.2), [0, 1.0]	[0, 0.2), [0.8, 1.0]
	[0.21, 1.0], [0, 1.0]	[0.21, 1.0], [0.8, 1.0]
	[0, 1.0], [0, 0.8)	[0, 1.0], [0, 0.8)
	[0, 1.0], [0.81, 1.0]	[0, 1.0], [0.81, 1.0]

Notes: There are two attributes in *DB*. The domains of the two attributes are both [0, 1.0].

The *NP*-Hard Property of *RvNDB*s

In this section, reversing the real-valued negative database (*RvNDB*) is proved to be an *NP*-hard problem. The proofs are

similar to the work in (Esponda et al., 2004b; Esponda et al., 2009). Based on the hardness of reversing the real-valued negative database, the real-valued negative database can be used to preserve privacy.

Problem 1. Is the positive database of *RvNDB* non-empty? That is, is there any entry that is not covered by *RvNDB*?

Problem 2. Can *RvNDB* be reversed to obtain the entries in the corresponding positive database? That is, can any entry that is not covered by *RvNDB* be found?

Lemma 1. Problem 1 is *NP*.

Proof. Giving any entry w which consists of m real values, if an algorithm can check whether it is the solution in polynomial time, problem 1 is *NP*. Obviously, the complexity of checking whether the entry w is matched with (or covered by) an entry y_i in *RvNDB* is $O(m)$ (m is the number of attributes). Then the complexity of checking whether *RvNDB* covers the entry w is $O(m \cdot |RvNDB|)$ ($|RvNDB|$ is the number of entries in *RvNDB*). Therefore, checking whether an entry w is the solution of problem 1 can be done in $O(m \cdot |RvNDB|)$, and problem 1 is *NP*.

Lemma 2. Any CNF-SAT instance ϕ can be converted to a real-valued negative database $RvNDB_\phi$.

Proof. Giving any CNF-SAT instance ϕ with n clauses and m variables $x_1, x_2, \ldots, x_m$, $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$, a real-valued negative database $RvNDB_\phi = \{y_1, y_2, \ldots, y_n\}$ with m attributes and n entries can be constructed as follows.

(1) Divide the domain of each attribute into two segments: $[l_k, p_k)$ and $[p_k, u_k]$ ($k = 1, 2, \ldots, m$). Then encode three intervals as follows.

$$\begin{cases} [l_k, p_k) = 0 \\ [p_k, u_k] = 1 \\ I_k = * \end{cases} \quad (2)$$

As the interval I_k covers both interval $[l_k, p_k)$ and $[p_k, u_k]$, the symbol '*' represents either 0 or 1.

(2) Each clause C_i is mapped to an entry y_i of $RvNDB_\phi$, and a binary negative database denoted as $eNDB_\phi = \{e_1, e_2, \ldots, e_n\}$ is constructed according to $RvNDB_\phi$.
   (a) If the k^{th} variable is presented as x_k in C_i, $[l_k, p_k)$ is assigned to $y_i[k]$, and $e_i[k]$ is set as 0.
   (b) If the k^{th} variable is presented as $\bar{x}_k$ in C_i, $[p_k, u_k]$ is assigned to $y_i[k]$, and $e_i[k]$ is set as 1.
   (c) If the k^{th} variable does not appear in C_i, I_k is assigned to $y_i[k]$, and $e_i[k]$ is set as '*'.

After all the clauses of the CNF-SAT instance are mapped to entries, the real-valued negative database $RvNDB_\phi$ denoted with intervals is constructed. The database $eNDB_\phi$ is the binary form of $RvNDB_\phi$, and they can be converted to each other easily. The $eNDB_\phi$ has the same structure with the binary negative database defined in (Esponda et al., 2004b; Esponda et al., 2009).

Lemma 3. Any entry in $eNDB_\phi$ is not the true assignment of ϕ.

Proof. Obviously, if assign $e_i[k]$ ($k = 1 \ldots m$) to the k^{th} variable x_k, the entry e_i is not a true assignment of C_i. Because each entry in $eNDB_\phi$ cannot satisfy at least one clause of ϕ, it is not the true assignments of ϕ.

Lemma 4. Each true assignment of the CNF-SAT instance ϕ corresponds to a real-valued entry not covered by $RvNDB_\phi$, and vice versa.

Proof. For any true assignment a of the CNF-SAT instance ϕ, as every clause of ϕ is satisfied by a and every entry in $eNDB_\phi$ is not the true assignment of ϕ, at least one bit of a is different from each entry in $eNDB_\phi$. That is to say, a is not covered by $eNDB_\phi$. According to equation 2, the assignment a can be converted to an entry v that consists of intervals, and obviously the entry v is not covered by $RvNDB_\phi$.

For any entry w consists of m real values and not covered by any entries in $RvNDB_\phi$, it could be encoded to a binary string a as follows.

$$a[k] = \begin{cases} 0 & w[k] \in [l_k, p_k) \\ 1 & w[k] \in [p_k, u_k] \end{cases}, \quad k = 1, 2, \ldots, m \quad (3)$$

As w is not covered by any entry y_i ($i = 1, \ldots, n$) in $RvNDB_\phi$, there is at least one attribute k that $w[k]$ is not covered by $y_i[k]$, and the encoding result $a[k]$ is different from $e_i[k]$ as well, i.e. $a[k] = \overline{e_i[k]}$. According to the encoding of $RvNDB_\phi$, if assign $a[k]$ to x_k, the clause C_i will be satisfied. Moreover, since w is not covered by all the entries in $RvNDB_\phi$, all the clauses in ϕ are satisfied by a, and a is a true assignment of the CNF-SAT instance ϕ.

Theorem 1. Problem 1 is *NP*-complete.

Proof. According to lemma 4, the problem of checking the satisfiability of the instance ϕ is equivalent to the problem 1 for $RvNDB_\phi$. Furthermore, due to the instance ϕ is chosen arbitrarily, any instance of the CNF-SAT can be converted to a special real-valued negative database. Therefore, problem 1 is *NP*-complete.

Theorem 2. Problem 2 is *NP*-hard.

Proof. Based on lemma 1, 2, 3, 4 and theorem 1, this theorem is immediately proved.

Generation Algorithm for *RvNDBs*

Some generation algorithms for the binary negative database have been proposed (Esponda et al., 2004b; Jia et al., 2005; Esponda et al., 2007a; Esponda et al., 2009; Liu et al., 2011). Based on these generation algorithms, an algorithm for generating real-valued negative databases is proposed in this section.

Giving a positive database $DB = \{x_1, x_2, \ldots, x_n\}$, and there are m attributes in DB. Each entry in DB is a vector of m real values. The procedure of the generation algorithm for the real-valued negative database from DB is described as follows.

(1) Preprocessing: Divide the domains of attributes in *DB*, and convert *DB* to a real-valued database DB^I which consists of intervals.
(2) Encoding: Encode DB^I to a binary positive database DB^2.
(3) Generating: Input DB^2 to an algorithm for generating a binary negative database from the binary positive database such as the *q*-hidden algorithm or the prefix algorithm, and output a binary negative database NDB^2.
(4) Decoding: Decode NDB^2 to a real-valued negative database *RvNDB* which consists of intervals.

Phase 1: Preprocessing

The preprocessing phase contains two processes: the dividing process and the converting process. In the dividing process, the domain of each attribute in *DB* is divided into several distinct intervals. In the converting process, the values of each entry in *DB* are converted to the intervals which they belong to.

Dividing Process. The domain of each attribute in *DB* is divided to a set of intervals. For any k ($k = 1\ldots m$), the interval set $P_k = \{p_{k,1}, p_{k,2}, \ldots, p_{k,num}\}$, where num_k is the number of intervals in P_k.

The set P_k should be generated according to the requirements for real-life applications and satisfy following basic conditions.

(1) The union of all the intervals in P_k equals to I_k, i.e.

$$p_{k,1} \cup p_{k,2} \cup \ldots \cup p_{k,num} = I_k \tag{4}$$

(2) The intersection between any two different intervals in P_k is the empty set, i.e.

$$p_{k,i} \cap p_{k,j} = \varnothing, \quad \forall 1 \le i < j \le num_k \tag{5}$$

(3) Since *DB* will be encoded to a binary database, ideally, the number of intervals in P_k should be the exponent of 2.

Divide algorithm
Input: $I = \{I_1, I_2, \ldots, I_m\}$, $Num = \{num_1, num_2, \ldots, num_m\}$
Output: $P = \{P_1, P_2, \ldots, P_m\}$

1. **For** the k^{th} ($k = 1\ldots m$) attribute **do**
2. $low \leftarrow l_k$, $unit \leftarrow (u_k - l_k)/num_k$
3. **For** $i = 1$ to $num_k - 1$ **do**
4. $up \leftarrow low + unit$
5. Add $[low, up)$ to P_k
6. $low \leftarrow up$
7. Add $[low, u_k]$ to P_k

Figure 1. An algorithm for dividing process

Although the dividing process depends on the requirements of real-life applications, a simple algorithm is given in figure 1. The algorithm in figure 1 equally divides each domain I_k ($k = 1\ldots m$) into num_k intervals. This algorithm can be applied to some applications such as the privacy-preserving data publication.

Converting Process. According to above dividing process, *DB* can be converted to a real-valued positive database DB^I which consists of intervals as follows.

Let $DB^I = \{t_1, t_2, \ldots, t_n\}$. For each entry x_i ($i = 1\ldots n$) in *DB*, the value of the k^{th} ($k = 1\ldots m$) attribute is converted to the interval which $x_i[k]$ belongs to in P_k, i.e.

$$t_i[k] = p_{k,j} \quad \textit{iff} \quad x_i[k] \in p_{k,j} \tag{6}$$

Phase 2: Encoding

In order to generate real-valued negative databases, the real-valued positive database DB^I is encoded to a binary database, and then an algorithm for generating negative databases from binary positive databases can be employed.

For the k^{th} ($k = 1\ldots m$) attribute, since any two different intervals are not intersected with each other and the number of the intervals in P_k is the exponent of 2, it is easy to encode num_k intervals in P_k as num_k binary strings with length $log_2(num_k)$. According to the encoding of intervals in P_k, the entries in DB^I can be converted to binary strings. The details of the encoding phase are shown in figure 2.

The algorithm shown in Figure 3 is used for generating the binary code from an integer. If the length of the binary code is less than l, some zeros will be attached after it. It follows that all the generated binary strings have the same length. In the encoding phase, this algorithm is employed to encode the intervals in P_k ($k = 1\ldots m$) according to their indexes.

Encode algorithm
Input: $DB^I = \{t_1, t_2, \ldots, t_n\}$, $P = \{P_1, P_2, \ldots, P_m\}$
Output: $DB^2 = \{s_1, s_2, \ldots, s_n\}$

1. **For** each entry t_i ($i = 1\ldots n$) in DB^I **do**
2. **For** the k^{th} ($k = 1\ldots m$) attribute **do**
3. $l \leftarrow log_2(num_k)$
4. **If** $t_i[k]$ is the j^{th} interval in P_k **then**
5. $s_i[k] \leftarrow binaryCode(j-1, l)$

Figure 2. The algorithm for encoding phase

***binaryCode*(*v*, *l*)**
Input: an integer v and length l
Output: a binary string *str* with length l

1. $i \leftarrow 1$
2. **While** $i \le l$ **do**
3. $str[i] \leftarrow v \bmod 2$
4. $v \leftarrow v/2$
5. $i \leftarrow i + 1$

Figure 3. Generating the binary code from an integer

Phase 3: Generating

In the encoding phase, a binary database DB^2 has been generated from the real-valued positive database DB^1. In the generating phase, DB^2 is inputted to an algorithm for generating negative databases from the binary positive database, such as the prefix algorithm (Esponda et al., 2004b; Esponda et al., 2009), the *RNDB* algorithm (Esponda et al., 2004b; Esponda et al., 2009) and the *q*-hidden algorithm (Jia et al., 2005; Esponda et al., 2007a), and the generation algorithm outputs a binary negative database $NDB^2 = \{z_1, z_2, \ldots, z_N\}$.

Phase 4: Decoding

In the generating phase, a binary negative database NDB^2 is obtained from the binary positive database DB^2. It is not convenient to use the binary negative database in the real-valued space. Therefore, in the decoding phase, the binary negative database NDB^2 is converted to a real-valued negative database *RvNDB*.

Decode algorithm
Input: $NDB^2 = \{z_1, z_2, \ldots, z_N\}$, $P = \{P_1, P_2, \ldots, P_m\}$
Output: *RvNDB*

1. **For** each entry z_i ($i = 1 \ldots N$) in NDB^2 **do**
2. **For** the k^{th} ($k = 1 \ldots m$) attribute **do**
3. $Q_k \leftarrow Extend\_Pattern(z_i[k], P_k)$
4. $RvNDB \leftarrow RvNDB \cup (Q_1 \times \ldots \times Q_m)$ **

Figure 4. The algorithm for decoding phase

Extend_Pattern(str, P_k)
Input: A string *str* defined upon alphabet {0, 1, *}, and P_k
Output: A set *W* of intervals which is decoded from *str*

1. Initialize *W* as the empty set
2. Set B_p as the unspecified positions of *str*
3. **For** every possible assignment *T* of B_p **do**
4. Let *str'* be the same with *str* but the unspecified positions are assigned according to *T*
5. Let *temp* be the decimal value of *str'*
6. Add $p_{k,\, temp+1}$ to *W*
7. Merge the adjacent intervals in *W*

Figure 5. Decoding a string defined upon alphabet {0, 1, *} to a set of intervals

The algorithm for the decoding phase is given in figure 4. Since the entries in NDB^2 are defined upon the alphabet {0, 1, *}, and the symbol '*' represents either 0 or 1 at a given position, each entry may cover multiple strings of specified values (i.e. 0 and 1). An extra algorithm for decoding a string defined upon the alphabet {0, 1, *} to a set of intervals is given in figure 5.

The algorithm in figure 5 enumerates every specified string which is covered by the string *str*, and converts these specified strings to intervals. Finally, the adjacent intervals in *W* are merged.

Application to the Privacy-Preserving Data Publication

As sensitive data has been involved in many applications nowadays, the privacy preserving of data has been widely concerned. The privacy-preserving data publication is a technique which can both preserve the privacy and maintain the utility of the published data.

The data generalization is an important technique for protecting sensitive data and preserving privacy (Fung et al., 2010). In the preprocessing phase of the generation algorithm for the real-valued negative database, the conversion from real values to intervals can be regarded as the generalization of real values, and the dividing of domains determines the generalized intervals. Therefore, when apply the real-valued negative database to the privacy-preserving data publication, the first phase can be replaced by some generalization techniques, such as some algorithms that can satisfy the *k*-anonymity principle (Sweeney, 2002). Then, a real-valued negative database can be generated from the generalized positive database through the generation algorithm descripted in the former section.

An example of applying the real-valued negative database to the privacy-preserving data publication is given as follows. The original data is shown in table 2. There are four attributes in the original positive database, and the attribute "Name" is the explicit identifier. The combination of attributes <Age, Postcode> is regarded as the quasi-identifiers. The sensitive attribute is "Salary". The domains of the last three attributes (i.e. *I*(Age), *I*(Postcode) and *I*(Salary)) are divided as follows.

I(Age) = [0, 150]
Divided as: {[0, 20), [20, 40), [40, 70), [70, 150]}
Encoded as: {00, 01, 10, 11}.

I(Postcode) = [00000, 50000]
Divided as: {[00000, 10000), [10000, 20000), [20000, 30000), [30000, 50000]}
Encoded as: {00, 01, 10, 11}.

I(Salary) = [0, 100.0]
Divided as: {[0, 10.0), [10.0, 20.0), [20.0, 50.0), [50.0, 100.0] }
Encoded as: {00, 01, 10, 11}.

Table 2. The original database

Name	Age	Postcode	Salary (k$)
Alice	16	21000	5.5
Bob	10	25000	65.0
John	55	16000	50.5
Bill	62	11000	25.3
David	42	13000	15.5

** It is noted that the cross product operation $(Q_1 \times \ldots \times Q_m)$ could be compressed to a new type of entry $y_Q = (Q_1, Q_2, \ldots, Q_m)$ for decreasing the size of *RvNDB*. Consequently, an entry in *RvNDB* could consist of *m* sets of intervals.

The generalized data which satisfies 2-anonymity principle (the *k*-anonymity principle demands that each entry in the published database cannot be distinguished from at least other *k*-1 entries (Sweeney, 2002)) is shown in table 3. The binary positive database is shown in table 4. The binary negative database generated by the prefix algorithm (Esponda et al., 2004b; Esponda et al., 2009) from the binary positive database is shown in table 5. Finally, the real-valued negative database decoded from the binary negative database is shown in table 6.

Table 3. The real-valued positive database which consists of intervals

Age	Postcode	Salary (k$)
[0, 20)	[20000, 30000)	[0, 10.0)
[0, 20)	[20000, 30000)	[50.0, 100.0]
[40, 70)	[10000, 20000)	[50.0, 100.0]
[40, 70)	[10000, 20000)	[20.0, 50.0)
[40, 70)	[10000, 20000)	[10.0, 20.0)

Table 4. The binary positive database

Age	Postcode	Salary (k$)
00	10	00
00	10	11
10	01	11
10	01	10
10	01	01

Table 5. A binary negative database

Age	Postcode	Salary (k$)
01	**	**
00	0*	**
00	11	**
00	10	01
00	10	10
11	**	**
10	1*	**
10	00	**
10	01	00

Table 6. The real-valued negative database

Age	Postcode	Salary (k$)
[20, 40)	[00000, 50000]	[0, 100.0]
[0, 20)	[00000, 20000)	[0, 100.0]
[0, 20)	[30000, 50000]	[0, 100.0]
[0, 20)	[20000, 30000)	[10.0, 20.0)
[0, 20)	[20000, 30000)	[20.0, 50.0)
[70, 150]	[00000, 50000]	[0, 100.0]
[40, 70)	[20000, 50000]	[0, 100.0]
[40, 70)	[00000, 10000)	[0, 100.0)
[40, 70)	[10000, 20000)	[0, 10.0)

Discussion

The real-valued negative database can be applied to the privacy-preserving data publication. The preprocessing phase of the generation algorithm for the real-valued negative database could be replaced by an existing generalization algorithm. The privacy of the published data is preserved through not only the generalization but also the real-valued negative database. If high data precision is expected, the generalized intervals can be controlled to small ranges. Even if the sensitive data is not generalized, it is still under the protection of the negative representation. If the real-valued negative database is complete, it can be considered as "equivalent" to the generalized positive database and no extra information is lost. Furthermore, since the relationship between the real-valued positive database and the real-valued negative database is one-to-many, and it is hard to check whether two real-valued negative databases correspond to the same positive database (the hardness could be roughly controlled through the generation algorithm for negative databases). Therefore, the real-valued negative database could be properly applied to the privacy-preserving data republication (Xiao and Tao, 2007) and the privacy-preserving publication of dynamic data (Jian et al., 2007; Xiao and Tao, 2007; Bu et al., 2008).

Conclusions and Future Work

Since the data in some applications is naturally represented in real-valued space, it is difficult to apply binary negative databases properly. Therefore, the real-valued negative database is proposed in this paper. Reversing the real-valued negative database is proved to be an *NP*-hard problem, and it follows that the real-valued negative database could be employed to protect data privacy. Based on the generation algorithms for the binary negative database, an effective algorithm for generating real-valued negative databases is proposed in this paper.

The real-valued negative database is applied to the privacy-preserving data publication in this paper. The privacy of the published data is under the protection of both the generalization and the negative representation. Furthermore, the balance between security and data precision could be controlled through the level of generalization and the generation algorithm for the real-valued negative database.

Although the definition and a generation algorithm for the real-valued negative database are given in this paper, some further work is expected. Since the generation algorithm for the real-valued negative database is based on the generation algorithms for the binary negative database, some more efficient generation algorithms which are dedicated to the real-valued negative database are expected to be proposed. Some operations for the real-valued negative database such as select, delete, insert, project, union, intersection, set difference, Cartesian product and join need to be designed urgently. These database operations are critical for extending the applications of the real-valued negative database. Moreover, some concrete and practical solutions of applying the real-valued negative database to the privacy-preserving data publication will be considered in future as well.

Acknowledgements

This work is partly supported by National Natural Science Foundation of China (No. 61175045).

References

Bu, Y., Fu, A. W. C., Wong, R. C. W., Chen, L. and Li, J. (2008). Privacy Preserving Serial Data Publishing by Role Composition. *Proceedings of the VLDB Endowment*, 1(1): 845-856.

Dasgupta, D. and Azeem, R. (2007). A Negative Authentication System. Technical Report, Department of Computer Science, The University of Memphis: CS-07-001.

Dasgupta, D. and Azeem, R. (2008). An Investigation of Negative Authentication Systems. *Proceedings of the Third International Conference on i-Warfare & Security (ICIW 2008)*, Omaha, Nebraska, USA.

Esponda, F. (2006). Negative Surveys. *Arxiv: math/0608176*.

Esponda, F., Ackley, E. S., Forrest, S. and Helman, P. (2004a). Online Negative Databases. *Proceedings of the Third International Conference on Artificial Immune Systems*, Catania, Sicily, Italy.

Esponda, F., Ackley, E. S., Forrest, S. and Helman, P. (2005). Online Negative Databases (with Experimental Result). *International Journal of Unconventional Computing*, 1(3): 201-220.

Esponda, F., Ackley, E. S., Helman, P., Jia, H. and Forrest, S. (2007a). Protecting Data Privacy through Hard-to-Reverse Negative Databases. *International Journal of Information Security*, 6(6): 403-415.

Esponda, F., Forrest, S. and Helman, P. (2004b). Enhancing Privacy through Negative Representations of Data. Technical Report, Department of Computer Science, University of New Mexico: A667894.

Esponda, F., Forrest, S. and Helman, P. (2009). Negative Representations of Information. *International Journal of Information Security*, 8(5): 331-345.

Esponda, F., Trias, E. D., Ackley, E. S. and Forrest, S. (2007b). A Relational Algebra for Negative Databases. Technical Report, Department of Computer Science, University of New Mexico: TR-CS-2007-2018.

Forrest, S., Perelson, A. S., Allen, L. and Cherukuri, R. (1994). Self-Nonself Discrimination in a Computer. *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society.

Fung, B. C. M., Wang, K., Chen, R. and Yu, P. S. (2010). Privacy-Preserving Data Publishing: A Survey of Recent Developments. *ACM Computing Surveys*, 42(4): 1-53.

González, F., Dasgupta, D. and Niño, L. (2003). A Randomized Real-Valued Negative Selection Algorithm. *Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS 2003)*, Springer Berlin Heidelberg.

Horey, J., Groat, M., Forrest, S. and Esponda, F. (2007). Anonymous Data Collection in Sensor Networks. *Proceedings of the Fourth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Philadelphia, USA.

Ji, Z. and Dasgupta, D. (2004). Real-Valued Negative Selection Algorithm with Variable-Sized Detectors. *Proceedings of the 2004 Conference on Genetic and Evolutionary Computation (GECCO 2004)*, Springer Berlin Heidelberg.

Ji, Z. and Dasgupta, D. (2006). Applicability Issues of the Real-Valued Negative Selection Algorithms. *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005)*, ACM, Seattle, Washington, USA.

Ji, Z. and Dasgupta, D. (2007). Revisiting Negative Selection Algorithms. *Evolutionary Computation*, 15(2): 223-251.

Jia, H., Moore, C. and Strain, D. (2005). Generating Hard Satisfiable Formulas by Hiding Solutions Deceptively. *Proceedings of the Twentieth National Conference on Artificial Intelligence*, AAAI Press, Pittsburgh, Pennsylvania.

Jian, P., Jian, X., Zhibin, W., Wei, W. and Ke, W. (2007). Maintaining k-Anonymity against Incremental Updates. *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*.

Liu, R., Luo, W. and Wang, X. (2011). A Hybrid of the Prefix Algorithm and the q-hidden Algorithm for Generating Single Negative Databases. *Proceedings of the 2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS 2011)*, IEEE Computer Society, Paris, France.

Selman, B., Kautz, H. and Cohen, B. (1995). Local Search Strategies for Satisfiability Testing. *Proceedings of the Second DIMACS Challange on Cliques, Coloring, and Satisfiability*.

Sweeney, L. (2002). k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5): 557-570.

Xiao, X. and Tao, Y. (2007). m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets. *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ACM, Beijing, China.