

Article

# A Novel Vertical Fragmentation Method for Privacy Protection Based on Entropy Minimization in a Relational Database

Tie Hong <sup>1,†,‡</sup>, SongZhu Mei <sup>2,\*†</sup>, ZhiYing Wang <sup>1,‡</sup> and JiangChun Ren <sup>1,‡</sup>

<sup>1</sup> College of Computer, National University of Defense Technology, Changsha 410073, China; hongtie@nudt.edu.cn (T.H.); zywang@nudt.edu.cn (Z.W.); jcren@nudt.edu.cn (J.R.)

<sup>2</sup> Science and Technology on Parallel and Distributed Laboratory, National University of Defense Technology, Changsha 410073, China

\* Correspondence: sz.mei@nudt.edu.cn; Tel.: +86-151-1120-9153

† Current address: The College of Computer, National University of Defense Technology, Changsha 410073, China.

‡ These authors contributed equally to this work.

Received: 16 October 2018; Accepted: 12 November 2018; Published: 14 November 2018



**Abstract:** Many scholars have attempted to use an encryption method to resolve the problem of data leakage in data outsourcing storage. However, encryption methods reduce data availability and are inefficient. Vertical fragmentation perfectly solves this problem. It was first used to improve the access performance of the relational database, and nowadays some researchers employ it for privacy protection. However, there are some problems that remain to be solved with the vertical fragmentation method for privacy protection in the relational database. First, current vertical fragmentation methods for privacy protection require the user to manually define privacy constraints, which is difficult to achieve in practice. Second, there are many vertical fragmentation solutions that can meet privacy constraints; however, there are currently no quantitative evaluation criteria evaluating how effectively solutions can protect privacy more effectively. In this article, we introduce the concept of information entropy to quantify privacy in vertical fragmentation, so we can automatically discover privacy constraints. Based on this, we propose a privacy protection model with a minimum entropy fragmentation algorithm to achieve minimal privacy disclosure of vertical fragmentation. Experimental results show that our method is suitable for privacy protection with a lower overhead.

**Keywords:** privacy protection; vertical fragmentation; information entropy; quantify privacy; relational database

---

## 1. Introduction

With the increase of human activities, more and more personal data needs to be stored. Some enterprises and individuals outsource their data to storage service providers. However, outsourcing storage gives rise to new security risks. A major problem is that the data is stored on external storage providers, so owners lose control over their data, potentially exposing them to improper access and dissemination. For example, in 2007, the well-known cloud service provider Salesforce [1] leaked a large amount of data due to security issues.

A current issue is how to ensure that outsourced data does not suffer from private leakage, even if the intruders obtain the original data. Currently, the most widely used method to protect outsourced data is encryption. At the same time, in order to ensure the availability of data, in the field of encryption technology, there are homomorphic encryption techniques [2], searchable encryption techniques [3,4] and the like. However, from the user's point of view, the use of encryption technology

is a huge burden. Encryption technology is very inefficient in operations such as database queries. Other commonly used data outsourcing methods include anonymization such as k-anonymity [5,6], l-diversity [7], and t-closeness [8]. Still, these methods limit the availability of the outsourced data.

An interesting method used to ensure both the availability and privacy of outsourced data is vertical fragmentation. Vertical fragmentation is not specifically designed to protect data security. It is primarily used in relational databases to improve database access performance and optimize storage. It splits a large table into multiple small tables and stores the information on multiple different servers. When the user submits a query request, the system will translate it into multiple requests which are executed concurrently on different servers. In this way, the efficiency of database querying is greatly improved. The vertical fragmentation method assumes that privacy is generated by the associations among data. For example, if the name and phone number in a database are leaked separately, the attacker could not obtain the knowledge of which number belongs to whom. However, if the name and phone number are leaked together, the attacker will obtain accurate privacy information. Vertical fragmentation splits the database into different servers, greatly reducing the risk of leaking the entire database. Thus, we can use vertical fragmentation to protect the privacy of an outsourced database. In addition, vertical fragmentation has a big advantage of maintaining outsourced data availability compared to the use of encryption and anonymization techniques, as these latter methods do not store the original data.

However, there are some problems with vertical fragmentation concerning privacy protection. First, it is very difficult for users to specify privacy constraints. In practice, users need to understand the semantics of each attribute and then determine if there are privacy constraints. Some weakly related privacy constraints are difficult for users to find, especially when there are many attributes in the database. Second, there are many vertical fragmentation methods that satisfy privacy constraints. In order to evaluate which method is better, De Capitani di Vimercat's approach simulates the frequency of access to all attributes and ultimately chooses the method with the best access efficiency to perform the vertical fragmentation. In fact, the access data for attributes are impossible to know in advance.

Compared with De Capitani di Vimercat's method, our research focuses on optimizing vertical fragmentation to eliminate user-defined privacy constraints and evaluate the effectiveness of privacy protection. We propose a new approach to vertically fragment a relational database to ensure the privacy and availability of the outsourced data. The approach consists of an entropy-based method to quantify the privacy of outsourcing data and a greedy algorithm to achieve privacy protection. In summary, we make the following contributions:

(1) We introduce the concept of information entropy and give an example of entropy calculation for a sample database, thus avoiding the need for the user to manually define privacy constraints in vertical fragmentation. We can calculate the entropy value of any number of attributes to determine which attributes together have a greater probability of having privacy constraints.

(2) We implement a vertical fragmentation method based on a greedy algorithm to minimize entropy. There are many solutions for vertically fragmenting databases. In order to minimize the risk of privacy leakage per fragmentation, we need to make the entropy of each fragment as small as possible. We propose a greedy algorithm to minimize the entropy for vertical fragmentation, which also works with low overhead.

The rest of this paper is structured as follows. Section 2 introduces related works. Section 3 describes the problem statement. We introduce our approach in Section 4. The experimental results are discussed in Section 5. In Section 6, we discuss the limitations and extensibility of our work. Conclusions are given in Section 7.

## 2. Related Work

In this section, we first introduce related work on vertical fragmentation with a focus on privacy protection (Section 2.1), and then we describe the research on information entropy (Section 2.2).

## 2.1. Vertical Fragmentation for Privacy Protection

With the development of encryption technology, encryption technology has been considered the greatest method for privacy protection. Encryption technology is inefficient at performing queries. To perform the query on encrypted data, researchers proposed the indexing methods. Hacigumus et al. [9] proposed the bucket-based index methods. Hore et al. [10] present an efficient bucket-based index method. Wang et al. [11] proposed a hash-based index method. However, the indexing methods can't resist inference attacks [12], and Order Preserving Encryption (OPE) [13] methods solve this problem by performing range query.

These encryption methods to protect privacy do not solve the problem of how to efficiently execute queries. There is no doubt that querying on encrypted data is very inefficient. An interesting method to achieve the goal of privacy protection and efficiently query is vertical fragmentation. Vertical fragmentation is mainly used to improve database access performance, since vertical fragmentation stores attributes together that are frequently accessed by queries. De Capitani di Vimercat et al. were the first to use vertical fragmentation for privacy protection. As reported in Refs. [14–20], De Capitani di Vimercat et al. used vertical fragmentation to protect the privacy of a relational database. They employed the user defined privacy constraints to complete the vertical fragmentation. Considering that there are many solutions to satisfy privacy constraints, they employed the user's query load to compute the query efficient optimal solution. Xiang et al. [21] suggested that the user's query load on the outsourced data was dynamically changing, so they proposed an adaptive partitioning strategy. Biskup et al. [22] used vertical fragmentation to protect confidentiality. Their model consists of two domains: one for storing encrypted data and one for trusted local domains which store fragments containing highly sensitive data without encrypting them. However, this method is not efficient. In order to solve the overhead in the query processing of data encryption methods, Ganapathy et al. [23] proposed distributing data storage for secure database services, in which data is stored on multiple servers. They also used the concept of privacy constraints to achieve a heuristic greedy algorithm for the distribution problem. Partitioning the queries for the servers was achieved using a bottom-up state-based algorithm. For multi-relational databases, Bkakria et al. [24] defined an approach for the privacy protection of sensitive information in outsourced data, based on a combination of fragmentation and encryption.

## 2.2. Information Entropy

Shannon's theory of information entropy [25] solves the basic theory of quantifying and communicating information. Earlier studies of information entropy to measure privacy are those by Diaz [26] and Serjantov et al. [27], who proposed the use of information entropy to measure the anonymity of anonymous communication systems. Assuming that the purpose of the attacker is to determine the true identity of the sender (or receiver) of the message, each user in the system can be estimated to be the true sender or receiver of the message with a certain probability. The attacker guesses that a user is a real sender or receiver, which is expressed as a random variable  $X$ . Let  $p(x)$  be the probability function of  $X$ , and  $S(X)$  be the set of possible values of  $X$ . In this way, we use the entropy  $H(x)$  to quantify the system's privacy level:

$$H(X) = - \sum p(x) \log_2 p(x). \quad (1)$$

For example, when the set  $S(X)$  is {1, 2} and the probability of each is 50%, then  $H(X) = 1$ . If the only value of the random variable  $X$  is 1, then the probability of taking the value of 1 is 100%. Accordingly,  $H(X) = 0$ . Therefore, the greater the entropy of the random variable  $X$ , the greater the dispersion of  $X$ . The more discrete the value, the greater the probability that a record is identified, and the higher the risk of information being leaked. If the entropy of the attribute is 0, it means that the value of the attribute may equally correspond to all the records equally, and thus the security is also the highest; it is hard for an attacker to identify the message sender or receiver.

### 3. Problem Statement

To provide privacy protection with vertical fragmentation, we first examine the meaning of vertical fragmentation for privacy protection, in general. We use an example to explain it and show that the current vertical fragmentation method is inefficient and unrealistic. Then, we introduce automated vertical fragmentation technology, which is the approach needed to quantify the privacy. Finally, we give the evaluation criteria of our method.

#### 3.1. Vertical Fragmentation with Privacy Constraints

De Capitani di Vimercat et al. first proposed a vertical fragmentation strategy to protect privacy in a relational database. In De Capitani di Vimercat's research, privacy is defined by the user, in terms of privacy constraints. A privacy constraint corresponds to a set of attributes of a relational database. The attributes in the set together may lead to privacy leakage.

**Definition 1.** *Privacy Constraints.* For a simple relationship  $R$ , it consists of attributes  $a_1, a_2, a_3, \dots, a_n$ . All privacy constraints are a subset of this attribute set, that is, the set of privacy constraints  $c \subseteq \{a_1, a_2, \dots, a_n\}$ .

For example, in a medical system, a patient's information sheet consists of their social security number (SSN), name, date of birth (DoB), zip code, disease, and physician, as shown in Table 1. De Capitani di Vimercat believed that some attributes together will cause the risk of a privacy leak. For example, the SSN and disease together form a privacy constraint, as they could disclose private information if leaked together. However, it is meaningless for only the SSN or only the disease information to be acquired by others; this does not cause a loss of privacy.

**Table 1.** An example of a patient's information sheet and the associated privacy constraints.

SSN	Name	DoB	Zip Code	Disease	Physician
123-56-1234	Alice	08/03	98112	Flu	M.White
234-56-7890	Bob	10/07	94778	Asthma	D.Warren
345-67-8901	David	02/12	94139	Gastritis	M.White
456-78-9012	Jery	08/03	94139	Flu	K.Jsery
567-89-0123	Semy	03/04	94141	Angina	D.Warren
678-90-1234	Fred	12/01	94142	Diabetes	M.Kity

$c_0 = \{\text{SSN, name}\}$ ,  $c_1 = \{\text{SSN, DoB}\}$ ,  $c_2 = \{\text{SSN, zip code}\}$ ,  $c_3 = \{\text{SSN, disease}\}$ ,  $c_5 = \{\text{SSN, physician}\}$ ,  $c_6 = \{\text{name, DoB}\}$ ,  $c_7 = \{\text{name, zip code}\}$ ,  $c_8 = \{\text{name, disease}\}$ ,  $c_9 = \{\text{name, physician}\}$ ,  $c_{10} = \{\text{DoB, zip code, disease}\}$ ,  $c_{11} = \{\text{DoB, zip code, physician}\}$ .

De Capitani di Vimercat reported that the SSN together with other attributes can be considered as privacy constraints, as described by  $c_0, \dots, c_5$ . Similarly, the name is also considered very sensitive. Privacy constraints  $c_6, \dots, c_9$  show that the name paired with other attributes presents a privacy risk. Privacy constraints  $c_{10}$  indicate that, with the birthday, zip code, and disease information together, one can infer the patient's name, acting as a quasi-identifier. Similarly, with the birthday, zip code, and physician together, one can also infer the name and SSN, as privacy constraint  $c_{11}$  describes. Privacy constraints  $c_0$  to  $c_{11}$  are manually defined by the user.

**Definition 2.** *Fragmentation.* Let  $R(a_1, a_2, \dots, a_n)$  be a relation schema, and the fragmentation result is  $F = \{f_1, f_2, \dots, f_p\}$ , where these fragments satisfy:

- (i)  $\forall f_i \in F, i \in [1, \dots, p], f_i \subseteq \{a_1, \dots, a_n\}$ ,
- (ii)  $\forall f_i, f_j \in F, i \neq j : f_i \cap f_j = \emptyset$ ,
- (iii)  $f_1 \cup f_2 \cup \dots \cup f_p = \{a_1, \dots, a_n\}$ .

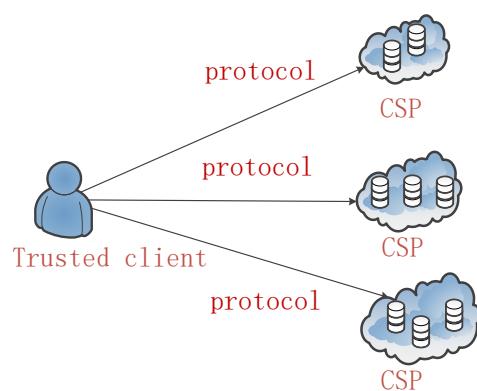
Condition (i) represents that only attributes of the relation  $R$  are considered by the fragmentation, condition (ii) ensures unlinkability between different fragments and condition (iii) guarantees that all attributes are fragmented.

Using these privacy constraints, De Capitani di Vimercat divided the patients' information sheet into four blocks { {SSN}, {name}, {DoB, zip code}, {disease, physician} }. Next, they were stored on four different servers that do not communicate with each other. Thus, even if one server is compromised due to an attack, the entire relational database would not be leaked; that is, "two can keep a secret" [28]. This is the core concept of vertical fragmentation for privacy protection in De Capitani di Vimercat's research: that is, to break the original relationships for the purpose of privacy protection.

However, De Capitani di Vimercat's research still has some flaws. First, we need to know the semantics of each attribute as well as which attributes together may cause privacy leaks. Just like the example above, we have to make sure that the SSN and name attributes are sensitive, and recognize that they will cause privacy leaks along with other attributes. Second, some privacy constraints are not easily found in semantics, such as the privacy constraint  $c_{10} = \{\text{DoB, zip code, and disease}\}$  in the above patients' information sheet. This is very difficult to apply to actual problem-solving. In reality, the number of attributes will be very large, and their relationships will be very complicated, so manually defined privacy constraints are not realistic. Third, there are many vertical fragmentation schemes that can satisfy the privacy constraints. For example, { {SSN}, {name}, {DoB, disease}, {zip code, physician} } is also a good scheme. However, De Capitani di Vimercat does not provide evaluation criteria to judge the quality of these schemes.

### 3.2. Evaluation Standard

We consider our model, as shown in Figure 1. It is composed of three main components: (1) The trusted client, which is the trusted party responsible for fragmenting and distributing the data to different cloud service providers (CSPs); (2) the CSPs, which are responsible for storing, querying, and retrieving data, and (3) the protocol that the trusted client and CSPs communicate by.



**Figure 1.** Storage model.

In our proposed research, we assume that the attacker can obtain the data of a certain CSP through the network, but that the CSPs do not communicate with each other, so the risk of privacy leakage is concentrated on a single CSP. Our goal is to ensure the protection of private information on a single CSP. Based on the analysis of the example of the patients' information sheet, below is a list of the types of tasks that must be accomplished with our approach.

(1) Correct Fragmentation: Verify that the privacy constraints are broken, so that fragmentation on a single CSP will not cause privacy leaks. Our approach must first ensure the most basic privacy protection requirements to break the potential privacy constraints that exist in relational databases. We must ensure there are no privacy leaks in the fragmentation of data, one can completely determine the records in the database, and so are very likely to disclose privacy when paired together.

**Definition 3.** *Correct Fragmentation.* Let  $R(a_1, a_2, \dots, a_n)$  be a relation schema, the fragmentation result be  $F = \{f_1, f_2, \dots, f_p\}$ , and  $C(c_0, c_1, \dots, c_t)$  be the privacy constraints. One correct fragmenting scheme must meet Definition 1 and the following requirements:

$$\forall c_i \in C, i \in [1, \dots, t], c_i \not\subseteq \{f_1, \dots, f_p\}.$$

As shown by example of the patients' information sheet, with the scheme  $F = \{\{\text{SSN}\}, \{\text{name}\}, \{\text{DoB, disease}\}, \{\text{zipcode, physician}\}\}$ , the privacy constraints  $c_0, \dots, c_{11}$  are not a subset of the fragment  $f_i$ .

(2) Minimal fragmentation: This is a simple metric of the fragmentation scheme to avoid excessive fragments. Intuitively, if there are  $n$  attributes of the data to be fragmented, we can simply divide it into  $n$  fragments. Thus, the risk of privacy leakage is also lower than other schemes. However, if we completely divide all attributes, it will lead to higher database query overhead.

**Definition 4.** *Minimal fragmentation.* Let  $R(a_1, a_2, \dots, a_n)$  be a relation schema. There are fragmentation schemes  $F$  satisfy definition 3,  $F_i$  is the minimal fragmentation scheme, iff

$$\forall F_j, j \neq i : \#F_i \leq \#F_j.$$

Here, # indicates the number of collection elements.  $\#F_i$  represents the number of fragments of scheme  $F_i$ . De Capitani di Vimercat et al. translate the problem into the problem of computing maximum weighted clique of the fragmentation graph and implemented it with the Ordered Binary Decision Diagrams(OBDDs) [29]. Their methods are based on heuristics algorithm and cannot get the optimal solution.

(3) Lower Risk of Information Disclosure: Each fragment contains a portion of private information that is less than the total amount of private information. There are many kinds of vertical fragmentation methods that can satisfy condition (1), but the various vertical fragmentation methods differ in that each fragment contains a different amount of information. In the patients' information sheet, both scheme (a)  $\{\{\text{SSN}\}, \{\text{name}\}, \{\text{DoB, zip code}\}, \{\text{disease, physician}\}\}$  and scheme (b)  $\{\{\text{SSN}\}, \{\text{name}\}, \{\text{DoB, disease}\}, \{\text{zip code, physician}\}\}$  are vertical fragmentation solutions. However, the amount of information they contain is different; thus, we need to reduce the amount of information contained in each fragment as much as possible. The greater the amount of information contained in the fragment, the greater the risk of information leakage. Suppose there are  $s$  fragments, and the amount of information contained in each fragment  $i$  is  $I(i)$ . To evaluate the risk of information leakage of the scheme, we need to minimize the sum of the squares of the information  $\sum I(i)^2$ .

**Definition 5.** *Lower Risk of Information Leakage.* Let  $R(a_1, a_2, \dots, a_n)$  be a relation schema. There are fragmentation schemes (a)  $F_a = \{fa_1, fa_2, \dots, fa_{p1}\}$  and (b)  $F_b = \{fb_1, fb_2, \dots, fb_{p2}\}$ , when

$$I_a = \sum_{i=1}^{p1} I_i^2, I_b = \sum_{i=1}^{p2} I_i^2, I_a < I_b.$$

$I_a$  is an evaluation of the risk of information disclosure of the scheme (a), and  $I_b$  is the evaluation of the risk of information disclosure of the scheme (b). This is to say scheme (a) has lower information than scheme (b).

The first statement indicates that our approach should achieve the goal of privacy protection, while the second statement indicates that the number of fragments needs to be minimized. The last statement indicates that we should minimize the potential risk of information disclosure. Our goal is to meet the above three requirements, which is equivalent to a multi-objective constrained optimization problem [30].

## 4. Approach

In this section, we describe the method to obtain the goal of privacy protection with minimal information disclosure. Our approach is implemented in three steps. First, we introduce the concept of entropy values to quantify private information. In this way, we can automatically discover privacy constraints to complete the fragmentation (Section 4.1). Second, we use the privacy constraints calculated in step 1 to get the number of minimal fragmentation (Section 4.2). Finally, we implement a minimum entropy fragmentation algorithm based on the greedy strategy, through which we can minimize the entropy of each fragment (Section 4.3).

### 4.1. Information Entropy to Quantify Privacy

To automatically discover privacy constraints, we need a method to quantify privacy. As an effective tool for measuring information, information entropy has proven to be an important contribution to the field of communications. As privacy is a kind of information, we can naturally consider using entropy to quantify it [31].

First, we assume that the set of attributes to be fragmentated is table A ( $a_1, a_2, a_3, \dots, a_n$ ), with  $n$  attributes kept in  $m$  records  $r_1, r_2, \dots, r_m$ . We can calculate the overall entropy of the table as follows:

$$EN(A) = - \sum_{i=1}^s p(r_i) \log_2 p(r_i), \quad (2)$$

$$s \leq m, 0 \leq p(r_i) \leq 1, \sum p(r_i) = 1.$$

$EN(A)$  represents the entropy of table A,  $s$  indicates the number of records that are not repeated in the record  $r$ , and  $p(r_i)$  represents the probability of  $r_i$  in table A. For example, the patient's information sheet has six records ( $r_1$  to  $r_6$ ). None of the records are repeated, thus:

$$p(r_i) = 1/6 (i = 1, 2, \dots, 6),$$

$$EN(A) = -(1/6 \log_2 1/6 + 1/6 \log_2 1/6 + 1/6 \log_2 1/6 +$$

$$1/6 \log_2 1/6 + 1/6 \log_2 1/6 + 1/6 \log_2 1/6) = \log_2 6 \approx 2.5850.$$

It can be proved that, if there is no repeat record, then the entropy value is  $\log_2 m$ . We can define a probability distribution that a record  $X_i$  in the database takes a value  $a$  via

$$\Pr(X_i = a) = \#\{r_j | i = a, 1 \leq j \leq m\} / m. \quad (3)$$

Here, the property  $X_i$  is a random variable of the attribute, and  $\#$  represents the number of elements in the collection. For example, in the patient's information sheet:

$$\Pr(disease = flu) = 2/6 = 1/3,$$

$$\Pr(physician = M.White) = 2/6 = 1/3.$$

Thus, the entropy of an attribute  $a_j$  is:

$$EN(a_j) = - \sum_{a \in a_j} \Pr(a_i = a) \log_2 \Pr(a_i = a). \quad (4)$$

As Table 1 shows for the patients' information sheet, the attributes' entropies can calculated as follows:

$$EN(SSN) = -(1/6 \log_2 1/6 + 1/6 \log_2 1/6 + 1/6 \log_2 1/6 +$$

$$1/6 \log_2 1/6 + 1/6 \log_2 1/6 + 1/6 \log_2 1/6) = \log_2 6 \approx 2.5850,$$

$$EN(physician) = -(2/6 \log_2 2/6 + 2/6 \log_2 2/6 + 1/6 \log_2 1/6 +$$

$$1/6 \log_2 1/6) = 1/3 \log_2 36 \approx 1.2925.$$

The joint entropy can be calculated as follows:

$$\begin{aligned} EN(SSN, name) &= \log_2 6 \approx 2.5850, \\ EN(DoB, disease) &\approx 2.2516. \end{aligned}$$

The joint entropy treats multiple attributes as a new attribute. As shown above  $EN(SSN, name)$ , we treat SSN and name as a new single attribute. That is to say,  $EN(A, B) = EN(B, A)$ .

**Theorem 1.** Let  $R$  be a relation schema,  $A$  and  $B$  are attributes,  $EN(A)$  and  $EN(B)$  represent their entropy, and  $EN(A, B)$  is the joint entropy of attributes  $A$  and  $B$ ; thus,

$$EN(A, B) \geq \max\{EN(A), EN(B)\}.$$

**Proof of Theorem 1.** Assume that attribute  $A$  consists of the records  $ra_1, ra_2, \dots, ra_m$ , and  $B$  consists of  $rb_1, rb_2, \dots, rb_m$ :

$$\begin{aligned} EN(A) &= EN(ra_1, ra_2, \dots, ra_m), \\ EN(B) &= EN(rb_1, rb_2, \dots, rb_m), \\ EN(A, B) &= EN(ra_1 \bowtie rb_1, ra_2 \bowtie rb_2, \dots, ra_m \bowtie rb_m). \end{aligned}$$

Here,  $\bowtie$  is a string concatenation function. From the definition of entropy, we know that the more non-repeating records there are, the larger the entropy. There are records  $ra_i$  and  $ra_j$  of attribute  $A$ . If  $ra_i = ra_j$ , after adding attribute  $B$ , it will become  $ra_i \bowtie rb_i$  and  $ra_j \bowtie rb_j$ . However,  $ra_i \bowtie rb_i = ra_j \bowtie rb_j$  is true only when  $ra_i = ra_j$  and  $rb_i = rb_j$ .  $EN(A, B)$  will reduce the number of repeating records, so  $EN(A, B) \geq \max\{EN(A), EN(B)\}$ .  $\square$

**Theorem 2.** Let  $R$  be a relation schema,  $A$  and  $B$  are attributes,  $EN(A)$  and  $EN(B)$  represent their entropy,  $EN(A, B)$  is the joint entropy of attributes  $A$  and  $B$ , and thus

$$EN(A, B) \leq EN(A) + EN(B).$$

**Proof of Theorem 2.** Mathematical induction. Assume that attribute  $A$  consists of the records  $ra_1, ra_2, \dots, ra_m$ ,  $B$  consists of  $rb_1, rb_2, \dots, rb_m$ , and  $n$  represents the number of non-repeating records of attribute  $B$ , thus:

When  $n = 1$ ,  $n = 1$  represents the records in attribute  $B$  exactly the same,  $rb_1 = rb_2 = \dots = rb_m$  and  $EN(B) = 0$ , thus:

$$\begin{aligned} EN(A, B) &= EN(ra_1 \bowtie rb_1, ra_2 \bowtie rb_2, \dots, ra_m \bowtie rb_m) \\ &= EN(ra_1, ra_2, \dots, ra_m) + 0 = EN(A) + EN(B). \end{aligned}$$

That is to say,

$$EN(A, B) \leq EN(A) + EN(B).$$

Suppose when  $n = t$ ,  $EN(A, B)_t \leq EN(A)_t + EN(B)_t$ , there are  $x$  duplicate  $rb_k$  records in attribute  $B$ ,  $\#BR = \#\{rb_i | rb_i = rb_k\} = x, x \geq 1$ . We mark them with  $BR = \{rnb_1, rnb_2, \dots, rnb_x\}$ , and the corresponding elements in attribute  $A$  are recorded as  $AR = \{rna_1, rna_2, \dots, rna_x\}$ . Here,  $rnb_1 = rnb_2 = \dots = rnb_x = rb_k$ . At the same time, we mark attributes  $A$  and  $B$  to remove these records as  $A - AR$  and  $B - BR$ :

$$EN(A, B)_t \leq EN(A)_t + EN(B)_t.$$

When  $n = t + 1$ , we know that the number of repeating records is minus one. There are  $x - 1$  duplicate  $rb_k$  records in attribute  $B$ ; we need to discuss in two cases:

(1)  $\#\{ra_i \bowtie rb_i | rb_i = rb_k\} = x$ , this means  $\#\{rna_1, rna_2, \dots, rna_x\} = x$ , thus

$$EN(B)_t = - \sum_{i=1}^{\#\{B-BR\}} p_i \log_2 p_i + \left(-\frac{x}{m}\right) \log_2 \left(\frac{x}{m}\right),$$

$$EN(B)_{t+1} = - \sum_{i=1}^{\#\{B-BR\}} p_i \log_2 p_i + \left(-\frac{x-1}{m}\right) \log_2 \left(\frac{x-1}{m}\right) + \left(-\frac{1}{m}\right) \log_2 \left(\frac{1}{m}\right).$$

When we change the duplicate record in attribute  $B$ ,  $ra_i$  and  $ra_i \bowtie rb_i$  will not change, thus

$$EN(A)_{t+1} = EN(A)_t,$$

$$EN(A, B)_{t+1} = EN(A, B)_t.$$

From the above, we know that we only need to prove  $EN(B)_{t+1} - EN(B)_t > 0$ :

$$EN(B)_{t+1} - EN(B)_t = \frac{1}{m} \log_2 \frac{x^x}{(x-1)^{x-1}}.$$

It is easy to prove that, when  $x > 1$  and  $m > 1$ , the above formula is greater than zero, so

$$EN(A, B)_{t+1} = EN(A, B)_t \leq EN(A)_{t+1} + EN(B)_{t+1}.$$

(2)  $\#\{ra_i \bowtie rb_i | rb_i = rb_k\} = y, y < x$ . We know that the number of duplicate records is  $y$ . There are at least two duplicate records in the set AR, thus  $y \geq 2$ , we mark them as RNA. When the number of repeating records in the set BR is minus one, if the records in the set RB corresponding to the RNA do not change, it will become situation (1). We only need to consider that the situation of the set RB corresponding to the RNA are changing, thus:

$$EN(A)_{t+1} = EN(A)_t,$$

$$EN(B)_t = - \sum_{i=1}^{\#\{B-BR\}} p_i \log_2 p_i + \left(-\frac{x}{m}\right) \log_2 \left(\frac{x}{m}\right),$$

$$EN(B)_{t+1} = - \sum_{i=1}^{\#\{B-BR\}} p_i \log_2 p_i + \left(-\frac{x-1}{m}\right) \log_2 \left(\frac{x-1}{m}\right) + \left(-\frac{1}{m}\right) \log_2 \left(\frac{1}{m}\right),$$

$$EN(A, B)_t = - \sum_{i=1}^s p_i \log_2 p_i + \left(-\frac{y}{m}\right) \log_2 \left(\frac{y}{m}\right),$$

$$EN(A, B)_{t+1} = - \sum_{i=1}^s p_i \log_2 p_i + \left(-\frac{y-1}{m}\right) \log_2 \left(\frac{y-1}{m}\right) + \left(-\frac{1}{m}\right) \log_2 \left(\frac{1}{m}\right).$$

This is equivalent to proof

$$EN(A, B)_{t+1} - EN(A, B)_t \leq EN(B)_{t+1} - EN(B)_t.$$

This is to say, we need to prove

$$\frac{1}{m} \log_2 \frac{y^y}{(y-1)^{y-1}} \leq \frac{1}{m} \log_2 \frac{x^x}{(x-1)^{x-1}}.$$

We know that, when  $m > 1$  and  $x > y > 1$ ,  $f(x) = \frac{1}{m} \log_2 \frac{x^x}{(x-1)^{x-1}}$  is a monotonically increasing function. Obviously,  $EN(A, B)_{t+1} \leq EN(A)_{t+1} + EN(B)_{t+1}$ .

In summary, we can get the following conclusion:

$$EN(A, B) \leq EN(A) + EN(B).$$

□

With this knowledge, we can automatically discover the privacy constraints that exist in Table 1:  $EN(SSN) = EN(name) = 2.585 = EN(total\ table)$ , which indicates that these attributes have privacy constraints, corresponding to  $c_0$  to  $c_9$ . Likewise,  $EN(DoB, zip\ code, disease) = EN(DoB, zip\ code, physician) = 2.585 = EN(total\ table)$  also indicates these attributes have privacy constraints, corresponding to  $c_{10}$  and  $c_{11}$ . We know from the definition of entropy that entropy is the amount of information an attribute contains. That is to say, both SSN and name contain the information amount of the entire table of the patients' information sheets, similar to the primary key. Thus, they are susceptible to privacy leaks along with other attributes. We proposed an algorithm that automatically generates privacy constraints, shown in Algorithm 1.

---

**Algorithm 1** Automatically Generates Privacy Constraints Algorithm (Table A, Constraints)

---

**Input:** Input parameters TableA : the table to be fragmented

**Output:** Output Constraints : the privacy constraints

```

1:  $[m, n] = size(TableA)$ 
2:  $Entropy\_tableA = calEntropy(TableA, m)$ 
3: for each  $A_i \in \{A_1, \dots, A_n\}$  do
4:    $Attri\_entropy = calEntropy(A_i, m);$ 
5: end for
6: for  $i = 2$  to  $n - 1$  do
7:    $Combinations(i) = calCombinations(i, Attrи_entropy, Entropy_tableA)$ 
8:   for  $j = 1$  to  $size(Combinations(i))$  do
9:      $Str = \emptyset$ 
10:    for each  $A_k \in Combinations(i, j)$  do
11:       $Str = Str + A_k$ 
12:    end for
13:     $Comb\_entropy(i, j) = calEntropy(Str, m)$ 
14:    if  $Comb\_entropy(i, j) > Entropy\_tableA \times threshold$  then
15:       $Constraints(i, j) = Combinations(j);$ 
16:    end if
17:   end for
18: end for
```

---

The automatically generated privacy constraints algorithm takes the table to be split as input, and the set of constraints result is saved in the array  $Constraints$ . Line 1 calculates the number of records and the number of attributes, marked as  $m$  and  $n$ , respectively. Line 2 calculates the entropy of the whole table. The function  $calEntropy$  takes the set of attributes and the number of records as input, using Equations (3) and (4) to calculate the entropy of the attribute. Lines 3 to 5 calculate the entropy of a single attribute in TableA. In lines 6 to 18, we calculate the constraints from 2 to  $n - 1$  dimension. In Line 7, the function  $calCombinations$  calculates the combination of possible privacy constraints for the number of different attributes based on the entropy of all attributes. If  $i = 3$ , it will return all ternary combinations that may have privacy constraints. The basis for determining whether it can become a privacy constraint is based on Theorems 1 and 2. In lines 8 to 17, we decide whether these

possible combinations will become privacy constraints. On line 13, we calculate the entropy of the possible combination of privacy constraints. On line 14, we set a  $threshold(0 < threshold \leq 1)$  that the user can adjust. The larger the  $threshold$ , the stronger the privacy constraint requirement. When the entropy of possible combination exceeds the entropy of entire table multiplied by threshold, we think that there is a privacy constraint so that we can automatically discover privacy constraints. There is no need for users to manually define privacy constraints at all.

#### 4.2. Calculate Minimal Fragmentation

Given a relational database, there are many correct fragmentation schemes. We can quickly calculate a simple minimal fragmentation scheme by Theorems 1 and 2.

**Theorem 3.** Let  $R(a_1, a_2, \dots, a_n)$  be a relation schema, and  $EN(R)$  represent the entropy of it.  $EN(a_i)$  is the entropy value of each attribute.  $F$  is a correct fragmentation and close to the optional minimal fragmentation schemes, and  $\lceil \rceil$  is the ceiling function,  $threshold$  is a parameter that allows users to adjust the strength of privacy protection, thus

$$\#F = \left\lceil \frac{\sum_{i=1}^n EN(a_i)}{EN(R) \times threshold} \right\rceil.$$

**Proof of Theorem 3.** We get correct fragmentation schemes  $F = \{f_1, f_2, \dots, f_p\}$ . From Definition 2, we know

$$f_1 \cup f_2 \cup \dots \cup f_p = \{a_1, a_2, \dots, a_n\},$$

$$\forall f_i, f_j \in F, i \neq j : f_i \cap f_j = \emptyset.$$

According to Theorem 2,

$$\begin{aligned} EN(f_1) + EN(f_2) + \dots + EN(f_p) &\leq EN(a_1) + EN(a_2) + \dots + EN(a_n) \\ EN\left(\sum_{i=1}^p EN(f_i)\right) &\leq \sum_{i=1}^n EN(a_i). \end{aligned} \quad (5)$$

We know from Section 4.1 that, when a fragmentation violates privacy constraints, its entropy must be greater than  $EN(R) \times threshold$ . When  $F$  is a correct fragmentation, thus

$$EN(f_i) < EN(R) \times threshold. \quad (6)$$

From Equations (5) and (6), we can conclude that

$$\#F = p = \left\lceil \frac{\sum_{i=1}^n EN(a_i)}{EN(R) \times threshold} \right\rceil.$$

□

According to the above method, we can quickly calculate a smaller number of fragmentation, but not an optimal solution.

#### 4.3. Minimum Entropy Fragmentation Algorithm

By using the entropy to quantify privacy, we can automatically discover the privacy constraints between attributes and the number of fragmentation. In this section, we achieve another goal, which is

to minimize the amount of information disclosed in each fragment using the minimum entropy fragmentation algorithm.

From the definition of information entropy, we know that if the entropy of an attribute is  $x$ , then the amount of information it contains is  $2^x$ . Thus, the information disclosed after fragmentation is:

$$I(n) = \sum_{i=1}^n 2^{EN(f_i)} \quad (7)$$

$EN(f(i))$  represents the entropy value of fragmentation  $f(i)$ , and  $n$  represents the number of fragments. In order for us to turn this problem into: given a relation  $R(a_1, a_2, \dots, a_n)$ ,  $F = \{f_1, f_2, \dots, f_p\}$  is a correct minimal fragmentation, this requires the following constraints:

$$\forall f_i, EN(f_i) < EN(R) * threshold$$

and make the  $I(p)$  as small as possible

$$I(p) = \sum_{i=1}^p 2^{EN(f_i)}.$$

This is the constrained minimum solution problem. To achieve the approximate optimal solution of disclosing the minimum amount of information, we employ a greedy algorithm, shown in Algorithm 2.

The minimum entropy fragmentation algorithm takes the table to be split as input, and the fragmentation result is saved in the array  $FS$ . Line 1 calculates the number of records and the number of attributes, marked as  $m$  and  $n$ , respectively. The function  $calEntropy$  takes the attribute and the number of records as input, using Equations (3) and (4) to calculate the entropy of the attribute. Lines 2 to 4 calculate the entropy of a single attribute in Table A. In Line 5, we calculate the entropy of entire table. In Line 6, we use Theorem 3 with the function  $calMinimalFragmentation$  to calculate the minimal number of fragments. In Line 7, all of the attributes are arranged in descending order. Lines 8 to 10 fill in the maximum attributes to the result array  $FS$ . Line 14 uses Equation (7) to calculate the current amount of information disclosure, while Lines 16 to 22 calculate the amount of information disclosure after adding the specified attribute to each fragment. On line 17, we use the function  $calSetSumEntropy$  to calculate the entropy of the set  $FS$  and determine if there is a risk of privacy leak after adding the attribute  $A_i$ . If there is a risk of privacy leak, we set the amount of information disclosure to the maximum value of the system; otherwise, we use the function  $calInfoIncrement$  to calculate the increment of information disclosure. On Line 23, the fragmentation that minimizes the amount of information disclosed is selected. Finally, the attributes are assigned to the target fragmentation. Thus, there is no need for users to specify privacy constraints at all, and an evaluation of the privacy level after fragmentation can be made.

**Algorithm 2** Minimum Entropy Fragmentation Algorithm (TableA, FS)

**Input:** Input parameters Table A : the table to be fragmented

**Output:** Output FS : the result array of fragmentation

```

1:  $[m, n] = \text{size}(\text{TableA})$ 
2: for each  $A_i \in \{A_1, \dots, A_n\}$  do
3:    $\text{Attri\_entropy} = \text{calEntropy}(A_i, m)$ 
4: end for
5:  $\text{Table\_entropy} = \text{calEntropy}(\text{TableA}, m)$ 
6:  $s = \text{calMinimalFragmentation}(\text{Table\_entropy}, \text{Attri\_entropy})$ 
7:  $[\text{AttriSort}, \text{Sindex}] = \text{SortByDescend}(\text{Attri\_entropy})$ 
8: for each  $i \in \{1, \dots, s\}$  do
9:    $FS[i] = \text{AttriSort}[i]$ 
10: end for
11:  $\text{AttriSort} = \text{AttriSort} - FS$ 
12: for each  $A_i \in \text{AttriSort}$  do
13:    $\text{preInformation} = \sum_{i=1}^s 2^{EN(FS[i])}$ 
14:   for each  $j \in \{1, \dots, s\}$  do
15:     if  $(\text{calEntropy}(A_i, m) + \text{calSetSumEntropy}(FS[j], m)) > \text{Table\_entropy} \times \text{threshold}$  then
16:        $\text{Information}[j] = \text{Max\_Number}$ 
17:     else
18:        $\text{Information}[j] = \text{calInfoIncrement}(\text{calEntropy}(A_i + FS[j], m), \text{preInformation})$ 
19:     end if
20:   end for
21:    $\text{target} = \text{minimal}(\text{Information})$ 
22:    $FS[\text{target}] = FS[\text{target}] \cup A_i$ 
23: end for

```

## 5. Experiments

Considering that there is no published data set for the problem defined in this paper, we tested the algorithm on the adult database of UCI datasets (<http://archive.ics.uci.edu/ml/datasets/Adult>). The UCI database is intended to be used for machine learning, presented by the University of California Irvine. Table 2 gives the detailed information of the test dataset. It has 14 attributes, which are age, workclass, fnlwgt (serial number), education, education\_num (time of education), marital\_status, occupation, relationship, race, sex, capital\_gain, capital\_loss, hours\_per\_week, and native\_country. In the following, we replace these attributes with the serial numbers 1–14.

**Table 2.** Detailed information of adult datasets.

Dataset	Number of Samples	Dimensions
Adult	32,561	14

### 5.1. Implementation and Usability Aspects

Our experiments were tested on an Intel(R) Core(TM) i5-4210U 1.7 GHz CPU (Intel Corporation, California, CA, USA) with 8 GB RAM. Entropy calculation and the minimum entropy fragmentation algorithm were run with Matlab 2014a (8.3.0.532, MathWorks, Massachusetts, MA, USA).

To break constraints, we could not use semantics to determine which attributes together would lead to privacy leak risk in the adult database. Instead, we propose that, if some attributes together can identify a record easily, then they have a privacy constraint. This is similar to a primary key and can completely identify a record. First, we calculated the original unfragmented adult database's entropy to be 14.9893. From the previous definition, we let  $\text{threshold} = 0.8$ , which means that when the entropy of the attribute set exceeds  $14.9893 \times 0.8$ , it is considered to have privacy constraints. Next, we calculated the entropy of the set of attributes for groups of attributes. For these sets of attributes, we listed the sets whose entropy exceeds  $14.9893 \times 0.8$ , which are the sets that have privacy constraints, as shown in Table 3. For example, the entropy of the first row attribute set (1, 2, 4, 7), corresponding

to age, workclass, education and occupation is 12.1658, which may represent a privacy constraint. The other attribute sets in the table represent other privacy constraints that exist in the adult database. We subjected the adult database to Algorithm 2. The minimum number of fragments is 4, and the last fragmentation result was  $F_1 = \{3\}$ ,  $F_2 = \{1, 11, 14\}$ ,  $F_3 = \{2, 4, 5, 7, 12\}$ , and  $F_4 = \{6, 8, 9, 10, 13\}$ . From the final fragmentation results, all possible privacy constraints in Table 3 have been broken, thus we have completely broken these potential privacy constraints.

**Table 3.** Calculate constraints.

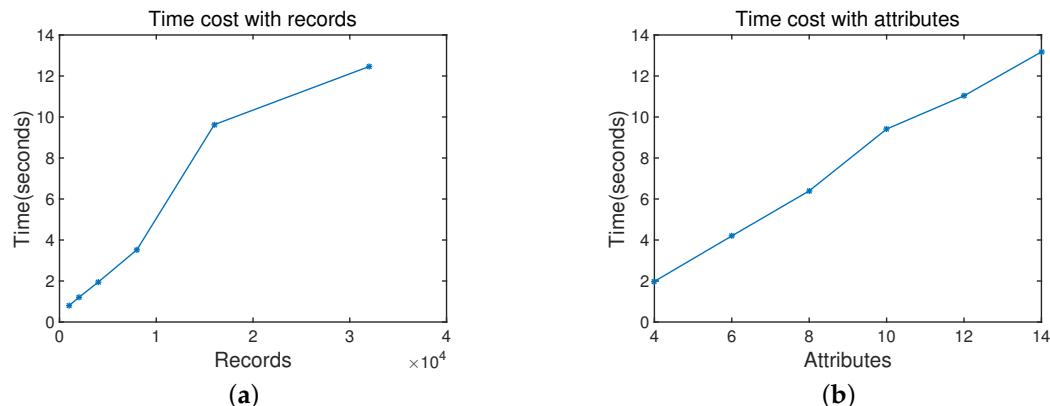
Attributes	Entropy	Attributes	Entropy
3	14.1583	1, 4, 6, 13	12.2393
1, 2, 4, 7	12.1658	1, 4, 7, 8	12.6782
1, 2, 4, 13	12.2012	1, 4, 7, 13	13.2738
1, 2, 5, 7	12.1658	1, 4, 8, 13	12.5190
1, 2, 5, 13	12.2012	1, 5, 6, 7	12.3872
1, 2, 7, 13	12.4623	1, 5, 6, 13	12.2393
1, 4, 6, 7	12.3872	1, 5, 7, 8	12.6782
1, 5, 7, 13	13.2738	1, 5, 8, 13	12.5190
1, 6, 7, 13	12.6807	1, 7, 8, 13	12.9189
1, 7, 9, 13	12.2523	1, 7, 10, 13	12.2812
1, 7, 11, 13	12.0908	1, 7, 13, 14	12.1851

Concerning the lower information aspect, we know the original unfragmented adult database's entropy is 14.9893. We also calculated the last entropy of each fragment  $F_1$  to  $F_4$  to be 14.1583, 7.2534, 7.6337, and 7.4301, respectively. Therefore, the entropy is reduced after fragmentation, and the risk of a privacy leak is significantly reduced. Since our minimum entropy fragmentation algorithm is a greedy algorithm, we have not yet obtained the optimal solution.

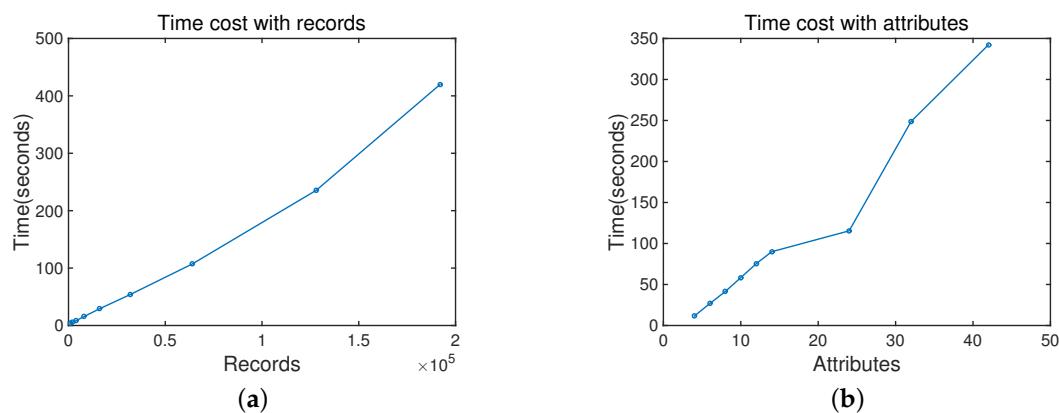
## 5.2. Performance Evaluation

In order to evaluate the minimum entropy fragmentation algorithm performance, we tested the time cost according to the number of samples and attributes. The adult database has 14 attributes and 32,561 records. We use the exhaustive method to calculate the minimum entropy fragmentation. In the experiment of the adult database, when there are only four attributes, it takes 17 seconds, but when the attribute is increased to 10, the time is increased to 1709 s. Thus, the exhaustive method takes time to grow exponentially and cannot be applied in reality. In order to observe the variation of our algorithm with the number of records, we divide the number of records into {1000, 2000, 4000, 8000, 16,000, 32,561} with a fixed number of attributes. In addition, in the case of fixed records, the attributes were divided into {4, 6, 8, 10, 12, 14}. Figure 2a shows the time cost when the adult database varies in the number of records, but the number of attributes is fixed at 14. The correlation coefficient between the number of records and the time cost was 0.9667, and thus a strong linear relationship between the number of records and the overhead time of the greedy algorithm was observed. Figure 2b shows the time cost when the number of attributes is varied, but the number of records is fixed at 32,561. We also calculated the correlation coefficient between the number of attributes and the time cost to be 0.9977.

In order to maintain the generalizability of our approach, we conducted experiments in the census-income database of the UCI database (<http://archive.ics.uci.edu/ml/datasets/Census+Income>), as depicted in Figure 3a,b. The census-income database has 199,523 records and 42 attributes. The correlation coefficients were calculated to be 0.9940 and 0.9806, respectively. Thus, the greedy fragmentation algorithm proposed in this paper is a polynomial time algorithm, which can quickly fragment the database with privacy protection.



**Figure 2.** Time cost for experiments on the adult database with respect to (a) changing the number of records, and (b) changing the number of attributes.



## 6. Discussion

Relational databases have always been the preferred solution for storing data. However, with the development of the internet, there must be more computing resources to handle fast-growing data. Extensions to computing resources include vertical scaling and horizontal scaling. The general method of vertical scaling is to replace the CPU, memory, etc., but this method is costly and has limited scalability. Another method is to use a small set of computers for large-scale computing tasks. Since relational databases are not suitable for building on clusters, NoSQL (Not Only SQL) databases have emerged.

NoSQL databases are divided into four categories according to storage methods [32]: (1) key-value storage, (2) columnar storage, (3) document storage, and (4) graphic storage. In the NoSQL data storage model, both columnar storage and document storage can be understood as extensions to key-value storage. These NoSQL databases are based on distributed storage, and the data is horizontally sharded and distributed to multiple nodes. The two major sharding strategies are order-preserving partitioning (OPP) and random partitioning (RP) [33]. OPP can perform range queries, but it may cause load-balancing problems. The current sharding method mainly considers load balancing and range query, regardless of privacy issues. We believe that our proposed entropy method can be used to protect privacy in NoSQL key-value storage databases because the information stored by each node can be quantified as  $EN(node_1), EN(node_2), \dots, EN(node_n)$ . However, from the current point of view, it is inefficient to calculate the entropy value in a NoSQL database. When a data record is generated, we need to recalculate which node is placed to minimize the entropy, so we need to fetch the data of each node and parse it to calculate the entropy, which is a time-consuming task. There may be some techniques such as delayed record write, or caching each node feature information for entropy

calculation to reduce the time overhead. However, these will always reduce the writing efficiency of the NoSQL database. For most NoSQL databases, it is necessary to support efficient writing operations, and our proposed entropy calculation method is only suitable for mass reading applications. To extend our method to NoSQL databases, we need to propose a new entropy calculation method. In the case of caching the key feature of each node, we can estimate the entropy value of the node after adding new data. This will be our next step in future work.

## 7. Conclusions

In this paper, we introduced information entropy as a powerful tool to quantify privacy; thus, we can automatically discover potential privacy constraints between attributes without the need for users to manually define such constraints. Based on this, we provided a greedy method of vertically fragmenting the data with the information entropy, which we called the minimum entropy fragmentation algorithm. With this approach, we can evaluate how much private information is contained in different vertical fragments. Our method is currently not applicable to NoSQL databases, but it is a feasible privacy protection scheme that enables us to optimize the efficiency of entropy calculation.

**Author Contributions:** T.H. contributed the design of the ideas, the analysis of results, and the writing of the paper. S.M., Z.W. and J.R. proofread the paper.

**Funding:** This research is supported by the National Natural Science Foundation of China (Grant No. 61303191 and No. 61402508) and the National High Technology Research and Development Program of China (Grant No. 2015AA016010).

**Acknowledgments:** We would like to acknowledge our families, friends, and colleagues for their support, suggestions, and reviews while conducting this study. We are also thankful to them for the successful completion of this part of the project.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Mowbray, M.; Pearson, S. A client-based privacy manager for cloud computing. In Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and MiddlewaRE, Dublin, Ireland, 16–19 June 2009; ACM: New York, NY, USA, 2009; p. 5.
2. Gentry, C. *A Fully Homomorphic Encryption Scheme*; Stanford University: Stanford, CA, USA, 2009.
3. Curtmola, R.; Garay, J.; Kamara, S.; Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. *J. Comput. Secur.* **2011**, *19*, 895–934. [[CrossRef](#)]
4. Kamara, S.; Papamanthou, C.; Roeder, T. Dynamic searchable symmetric encryption. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; ACM: New York, NY, USA, 2012; pp. 965–976.
5. Samarati, P.; Sweeney, L. *Protecting Privacy When Disclosing Information: K-Anonymity and Its Enforcement Through Generalization and Suppression*; Technical Report; SRI International: Menlo Park, CA, USA, 1998.
6. Sweeney, L. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2002**, *10*, 557–570. [[CrossRef](#)]
7. Machanavajjhala, A.; Gehrke, J. *l-Diversity: Privacy Beyond k-Anonymity*; IEEE: New York, NY, USA, 2006; p. 24.
8. Li, N.; Li, T.; Venkatasubramanian, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey, 15–20 April 2007; pp. 106–115.
9. Hacioglu, H.; Iyer, B.; Li, C.; Mehrotra, S. Executing SQL over encrypted data in the database-service-provider model. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, WI, USA, 3–6 June 2002; ACM: New York, NY, USA, 2012; pp. 216–227.
10. Hore, B.; Mehrotra, S.; Tsudik, G. A privacy-preserving index for range queries. In Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, ON, Canada, 31 August–3 September 2004; Volume 30, pp. 720–731.

11. Wang, Z.F.; Dai, J.; Wang, W.; Shi, B.L. Fast query over encrypted character data in database. In Proceedings of the International Conference on Computational and Information Science, Shanghai, China, 16–18 December 2004; Springer: Berlin, Germany, 2004; pp. 1027–1033.
12. Damiani, E.; Di Vimercati, S.D.C.; Foresti, S.; Samarati, P.; Viviani, M. Measuring inference exposure in outsourced encrypted databases. In *Quality of Protection*; Springer: Berlin, Germany, 2006; pp. 185–195.
13. Boldyreva, A.; Chenette, N.; Lee, Y.; O’Neill, A. Order-preserving symmetric encryption. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April 2009; Springer: Berlin, Germany, 2009; pp. 224–241.
14. Ciriani, V.; Di Vimercati, S.D.C.; Foresti, S.; Jajodia, S.; Paraboschi, S.; Samarati, P. Fragmentation and encryption to enforce privacy in data storage. In Proceedings of the European sYmposium on Research in Computer Security, Dresden, Germany, 24–26 September 2007; Springer: Berlin, Germany, 2007; pp. 171–186.
15. Di Vimercati, S.D.C.; Foresti, S.; Jajodia, S.; Paraboschi, S.; Samarati, P. Encryption policies for regulating access to outsourced data. *ACM Trans. Database Syst. (TODS)* **2010**, *35*, 12. [[CrossRef](#)]
16. Di Vimercati, S.D.C.; Erbacher, R.F.; Foresti, S.; Jajodia, S.; Livraga, G.; Samarati, P. Encryption and fragmentation for data confidentiality in the cloud. In *Foundations of Security Analysis and Design VII*; Springer: Berlin, Germany, 2014; pp. 212–243.
17. Di Vimercati, S.D.C.; Foresti, S.; Samarati, P. Selective and fine-grained access to data in the cloud. In *Secure Cloud Computing*; Springer: Berlin, Germany, 2014; pp. 123–148.
18. Di Vimercati, S.D.C.; Foresti, S.; Livraga, G.; Samarati, P. Selective and private access to outsourced data centers. In *Handbook on Data Centers*; Springer: Berlin, Germany, 2015; pp. 997–1027.
19. Di Vimercati, S.D.C.; Foresti, S.; Livraga, G.; Samarati, P. Practical techniques building on encryption for protecting and managing data in the cloud. In *The New Codebreakers*; Springer: Berlin, Germany, 2016; pp. 205–239.
20. Di Vimercati, S.D.C.; Foresti, S.; Livraga, G.; Paraboschi, S.; Samarati, P. Confidentiality Protection in Large Databases. In *A Comprehensive Guide through the Italian Database Research Over the Last 25 Years*; Springer: Berlin, Germany, 2018; pp. 457–472.
21. Xiong, L.; Goryczka, S.; Sunderam, V. Adaptive, secure, and scalable distributed data outsourcing: A vision paper. In Proceedings of the 2011 Workshop on Dynamic Distributed Data-Intensive Applications, Programming Abstractions, and Systems, San Jose, CA, USA, 8 June 2011; ACM: New York, NY, USA, 2011; pp. 1–6.
22. Biskup, J.; Preuß, M.; Wiese, L. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In Proceedings of the International Conference on Information Security, Xi’an, China, 26–29 October 2011; Springer: Berlin, Germany, 2011; pp. 246–261.
23. Ganapathy, V.; Thomas, D.; Feder, T.; Garcia-Molina, H.; Motwani, R. Distributing data for secure database services. In Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society, Uppsala, Sweden, 21–24 March 2011; ACM: New York, NY, USA, 2011; p. 8.
24. Bkakria, A.; Cappens, F.; Cappens-Boulahia, N.; Fernandez, J.M.; Gross-Amblard, D. Preserving Multi-relational Outsourced Databases Confidentiality using Fragmentation and Encryption. *JoWUA* **2013**, *4*, 39–62.
25. Shannon, C.E. A mathematical theory of communication. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2001**, *5*, 3–55. [[CrossRef](#)]
26. Diaz, C.; Seys, S.; Claessens, J.; Preneel, B. Towards measuring anonymity. In Proceedings of the International Workshop on Privacy Enhancing Technologies, San Francisco, CA, USA, 14–15 April 2002; Springer: Berlin, Germany, 2002; pp. 54–68.
27. Serjantov, A.; Danezis, G. Towards an information theoretic metric for anonymity. In Proceedings of the International Workshop on Privacy Enhancing Technologies, San Francisco, CA, USA, 14–15 April 2002; Springer: Berlin, Germany, 2002; pp. 41–53.
28. Aggarwal, G.; Bawa, M.; Ganeshan, P.; Garcia-Molina, H.; Kenthapadi, K.; Motwani, R.; Srivastava, U.; Thomas, D.; Xu, Y. Two can keep a secret: A distributed architecture for secure database services. In Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, USA, 4–7 January 2005.
29. Ciriani, V.; De Capitani di Vimercati, S.; Foresti, S.; Livraga, G.; Samarati, P. An OBDD approach to enforce confidentiality and visibility constraints in data publishing. *J. Comput. Secur.* **2012**, *20*, 463–508. [[CrossRef](#)]

30. Deming, S.N. Multiple-criteria optimization. *J. Chromatogr. A* **1991**, *550*, 15–25. [[CrossRef](#)]
31. Peng, C.G.; Ding, H.F.; Zhu, Y.J.; Tian, Y.L. Information entropy models and privacy metrics methods for privacy protection. *J. Softw.* **2016**, *27*, 1891–1903.
32. Cattell, R. Scalable SQL and NoSQL data stores. *ACM Sigmod Rec.* **2011**, *39*, 12–27. [[CrossRef](#)]
33. Okman, L.; Gal-Oz, N.; Gonen, Y.; Gudes, E.; Abramov, J. Security issues in nosql databases. In Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Changsha, China, 16–18 November 2011; pp. 541–547.



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).