Definition and Efficient Construction of Encrypted k–anonymization Scheme

Masayuki Yoshino[1], Takayuki Suzuki[1], Ken Naganuma[1,2] and Hisayoshi Sato[1]

[1]*Hitachi, Ltd. Research & Development Group, Center for Technology Innovation-Systems Engineering, Kanagawa, Japan*

[2]*The University of Tokyo, Chiba, Japan*

Keywords: Encrypted Database, k–anonymization, Searchable Symmetric Encryption, Domain Generalization Hierarchy.

Abstract: In this paper, we propose an encrypted k–anonymization scheme (EAS) to k–anonymize an encrypted database using a domain generalized hierarchy while maintaining the encryption state. Preparation of the domain generalized hierarchy is optional; the proposed EAS can generate domain generalized hierarchies using a Huffman code tree from a database encrypted with searchable encryption. As a result, the user can delegate k–anonymization processing to a third party organization such as the cloud while retaining the confidentiality of the database without preparing a generalized hierarchy. In addition, third-party organizations that are entrusted also have the advantage to eliminate possible of misconduct such as information leakage. In a standard computer experiment, we performed a generalization process, which is the major procedure for our EAS. The generalization process takes around 168 seconds only to achieve k–anonymity with $k = 3$ on $1,000,000$ records consisting of 4 attributes. As a consequence, this high-speed performance means our EAS is applicable to not only batch processing but also real-time processing.

1 INTRODUCTION

Numerous studies have proposed technology to make individual identification hard, i.e., anonymize personal information. In these studies, k–anonymity is known as a representative index that quantifies the specific difficulty of identifying an individual (Samarati and Sweeney, 1998; Samarati, 2001; Sweeney, 2002b). k–anonymity means that "the value of the record must be converted so that there are more than $(k-1)$ records that all have the same attribute values" . Finding an optimal solution that satisfies k–anonymity is known to be a computationally hard problem, and some of computationally hard problems are even proven to be NP Hard (Meyerson and Williams, 2004). Therefore, a k–anonymization technique that gives up finding an optimal solution and obtains an approximate solution to work in polynomial time is widely used in practice (Sweeney, 2002a; LeFevre et al., 2005; LeFevre et al., 2006; Wang et al., 2004).

On the other hand, due to the progress of Internet of Things (IoT) technology in recent years, accumulated data has increased, and single systems have nearly reached their limit to store and manage this data. Therefore, in collaboration with external systems such as clouds that have abundant computational resources, more data management is being outsourced. One of the key primitives is searchable encryption, which realizes an encrypted data management system such as an encrypted database and encrypted file storage to protect sensitive information including personal data from being seen by not only system intruders but also cloud administrators. Searchable encryption is a cryptographic technique that enables matching of two kinds of encrypted data while maintaining encryption. Although encryption requires a secret key, the matching process does not. Several searchable encryption methods have been proposed (Boneh et al., 2004; Boneh and Waters, 2007; Curtmola et al., 2006; Song et al., 2000; Yoshino et al., 2011) that are classified into secret key cryptosystems and public key cryptosystems. In this paper, we target large-scale data and adopt the symmetric key cryptosystem (Yoshino et al., 2011), which is superior to high-speed processing.

Methods have been proposed to have external organizations conduct data anonymization processing while protecting privacy and utilizing data. For example, in the methods of Gentry (Gentry, 2009) and Ducas and Micciancio (Ducas and Micciancio, 2015), arbitrary arithmetic processing can be executed while

Yoshino, M., Suzuki, T., Naganuma, K. and Sato, H.
Definition and Efficient Construction of Encrypted k–anonymization Scheme.
DOI: 10.5220/0007354702930300
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 293-300
ISBN: 978-989-758-359-9

encrypting data so that anonymization processing can be delegated safely. However, in these methods, the overhead required for arithmetic processing in the encrypted state is still far from the practical level, so even anonymization of small databases is not realistic[1].

1.1 Our Contribution

We point out that outsourcing the anonymization process may lead to information leakage, thus we propose an encrypted k–anonymization scheme (EAS). Our contributions are briefly described as follows.

- We define the EAS played by a user and a server that can k–anonymize given encrypted data without a secret key, and define a semantic security model for EAS; an honest-but-curious server will not learn any useful information about the given encrypted database.
- We propose a construction of EAS and prove its security. We design EAS using domain generalization hierarchies, however the user does not need to prepare them. By combining generation technique for domain generalization hierarchy from database (Harada et al., 2012), and searchable symmetric encryption technique for an encrypted database (Kamara and Lauter, 2010; Yoshino et al., 2011; Popa et al., 2012), our construction is equipped with a method to generate domain generalization hierarchies from searchable encrypted database. Furthermore, our construction is proved to be secure under the security model.
- We implemented the proposed EAS on a general-purpose PC and carried out experiments, where a generalization technique achieving k–anonymity with $k = 3$ takes 168 seconds on $1,000,000$ records consisting of 4 attributes. Thanks to the high-speed processing, the proposed EAS is applicable to not only batch processing but also real-time processing.

2 PRELIMINARY

2.1 Table Notation

First, we define a plaintext table $\mathcal{PT}$ in a database to be k–anonymized.

- Let table $\mathcal{PT}$ be a combination of $(\mathcal{A}, \mathcal{C})$ where $\mathcal{A}$ is an array of n attributes $(a_1, \ldots a_n)$ and $\mathcal{C}$ is an array of n columns $(\mathcal{C}_1, \ldots \mathcal{C}_n)$.
- Each attribute a_i contains a word w called as quasi-identifier, which is selected from a dictionary $\mathcal{D}_a$: $w \in \mathcal{D}_a$.
- Each column $\mathcal{C}_i$ consists of m cells $(c_{1,i}, \ldots c_{m,i})$. Each cell $c_{i,j}$ contains a word w, which is selected from a dictionary $\mathcal{D}_{ci}$: $w \in \mathcal{D}_{ci}$.

Let an encrypted table $\mathcal{ET}$ be a same structure as $\mathcal{PT}$ except that each attribute $a_i \in \mathcal{A}$ and each cell $c_{i,j} \in \mathcal{C}_j$ contains an encrypted word ew.

2.2 k-anonymization Techniques

k–anonymization is a de-identification technique to achieve k–anonymity, which is an index to quantify the difficulty of individual identification proposed by Samarati and Sweeney in 1998 (Samarati and Sweeney, 1998). To satisfy k–anonymity, the value of the record must be converted so that there are more than $(k-1)$ records that all have the same attribute values. This conversion process is called recoding and can be roughly divided into a local recoding method and a global recoding method. Since the local recoding method calculates the distance between records for grouping, many calculations are required. Although precise recoding is performed, due to the high calculation volume, usage tends to be limited to use cases with a small number of records. On the other hand,many global recoding methods use auxiliary information called a generalized hierarchy[2], do not calculate distance, and regularly perform recoding. High speed is an advantage and is suitable for k–anonymization targeting large-scale data. Since this paper deals with large-scale data, we use a global recoding method with high speed.

In the global recording method, each attribute to be anonymized is associated with a domain generalized hierarchy (*DGH*) from which the values can be generalized to form a group of at least k tuples with identical values (Sweeney, 2002a). Examples of *DGH* and k–anonymized tables are given in Figures 1 and 2, respectively. The lowest values of *DGH* are called leaf nodes, and the highest node of *DGH* is called the root node. Relationships are defined between nodes from leaf nodes to root nodes. The upper node holds the generalized value of the lower node. Figure 1 shows the leaf node at the lowest level, which is the nationality unit $\{$(Japan, China), (Russia, England, Germany) $\}$, the more generalized regional unit

[1]To execute the 1-NAND operation on a general-purpose computer, Gentry's method takes about 30 minutes whereas Ducas and Micciancio's method takes about 1 second.

[2]There is k–anonymization technology without a generalized hierarchy such as (LeFevre et al., 2006).

$\{$Asia, Europe $\}$, and the top root, which is located in the most generalized unit $\{*\}$. Figure 2 shows the two attributes (occupation and nationality) associate with their DGH. Figure 2 shows $\mathcal{PT}$ where the combinations of four attributes are 2-anonymity, i.e., there are at least 2 identical records for every four attribute combinations. We refer to the k–anonymized $\mathcal{PT}$ as $k\mathcal{PT}$ and the k–anonymized $\mathcal{ET}$ as $k\mathcal{ET}$.

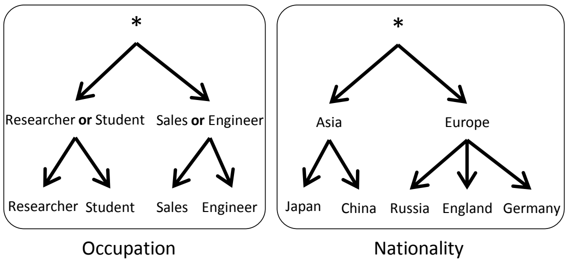

Figure 1: Example of DGH.

Occupation	Nationality	Gender	Birth date
Researcher or Student	Asia	*	19**
Sales or Engineer	Europe	*	20**
Sales or Engineer	Europe	*	20**
Researcher or Student	Asia	*	19**
Sales or Engineer	Europe	*	20**

Figure 2: Example of $k\mathcal{PT}$ with $k = 2$.

2.3 Searchable Symmetric Encryption

In this paper, we use searchable symmetric encryption SSE as cryptographic primitives. An SSE scheme has two phases called as a store phase, which is performed once, and a search phase, which is performed a polynomial number of times. In the store phase, an user encrypts all data and stores them on a server. In the search phase, the user sends a trapdoor of a word w, the server returns the encrypted word ew, which is matched by a comparison function for a trapdoor of a word and an encrypted word.

Formally, the searchable symmetric encryption SSE consists of five polynomial-time algorithms $\mathtt{SSE} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Trpdr}, \mathtt{Cmpr}, \mathtt{Dec})$ as follows (Curtmola et al., 2011):

- $(sk, pp) \leftarrow \mathtt{Gen}(1^{\lambda})$: is a probabilistic algorithm that takes security parameter λ as input and outputs a public parameter pp and a secret key sk.
- $ew \leftarrow \mathtt{Enc}(sk, w)$: is a probabilistic algorithm that takes a secret key sk and word w as input and outputs an encrypted word ew. We sometimes write $\mathsf{ew} \leftarrow \mathtt{Enc}(sk, \mathsf{w})$ where w is a set of w and ew is a set of ew.
- $td(w') \leftarrow \mathtt{Trpdr}(sk, w')$: is a deterministic algorithm that takes a secret key sk and a word w' as input and outputs a trapdoor $td(w')$. We sometimes write $\mathsf{td}(\mathsf{w}') \leftarrow \mathtt{Trpdr}(sk, \mathsf{w}')$ where w' is a set of w' and $\mathsf{td}(\mathsf{w}')$ is a set of $td(w')$.
- 0 or $1 \leftarrow \mathtt{Cmpr}(ew, td(w'))$: is a deterministic algorithm that takes encrypted word ew and a trapdoor $td(w')$ as input and outputs either 1 (if $w = w'$) or 0 (if $w \neq w'$) with provability $1 - \varepsilon(\lambda)$ and ε is a negligible function.
- $w \leftarrow \mathtt{Dec}(sk, ew)$: is a deterministic algorithm that takes a secret key sk and encrypted word ew as input and outputs a word w.

In SSE, the server should learn almost no information on w and w'. Let $\mathcal{L}_1(\mathsf{w})$ denote the information that the server can learn in the store phase, and let $\mathcal{L}_2(\mathsf{w}, \mathsf{w}')$ denote that in the search phase.

Most SSE reveals $\mathcal{L}_1(\mathsf{w}) = (\ldots, |w_i|, \ldots)$ and a dictionary $\mathcal{D}$ such that $w_i \in \mathcal{D}$, and $\mathcal{L}_2(\mathsf{w}, \mathsf{w}')$ consists of search result $\{(i, j) | w_i = w'_j, w_i \in \mathsf{w}, w'_j \in \mathsf{w}'\}$ and the search pattern such that $\mathtt{searchpattern}((w'_1, \ldots w'_q), w') = (b_1, \ldots b_q)$ where $b_j = 1$ if $(w'_j = w')$ and $b_j = 0$ if $(w'_j \neq w')$. The search pattern caused by deterministic Trpdr algorithm.

We introduce definitions of semantic security against an adversary in (Curtmola et al., 2006). The security is defined by using two games: $\mathbf{Real}_{SSE,\mathbf{A}}$ is played by an adversary $\mathbf{A}$ and a challenger $\mathbf{C}$, and $\mathbf{Ideal}_{SSE,\mathbf{A},\mathbf{S}}$ is played by $\mathbf{A}$, $\mathbf{C}$ and a simulator $\mathbf{S}$.

Definition 1. *((Curtmola et al., 2006)) Let* $\mathtt{SSE} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Trpdr}, \mathtt{Cmpr}, \mathtt{Dec})$ *be a searchable symmetric encryption, and consider the following probabilistic experiments where* $\mathbf{A}$ *is an adversary,* $\mathbf{C}$ *is a challenger,* $\mathbf{S}$ *is a simulator and* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *are leakage algorithms:*

Real$_{SSE,\mathbf{A}}(\lambda)$ *: The adversary* $\mathbf{A}$ *chooses a set of word* w *and sends it to the challenger* $\mathbf{C}$*.* $\mathbf{C}$ *begins by running* $\mathtt{Gen}(1^{\lambda})$ *to generate a secret key sk and a public parameter pp and send them to* $\mathbf{A}$*.* $\mathbf{A}$ *outputs a set of word* w *and receives* ew *from* $\mathbf{C}$ *by* $\mathsf{ew} \leftarrow \mathtt{Enc}(sk, \mathsf{w})$*.* $\mathbf{A}$ *gives a polynomial number of words* w *to* $\mathbf{C}$*, and receives trapdoors* $\mathsf{td}(\mathsf{w}') \leftarrow \mathtt{Trpdr}(sk, \mathsf{w}')$ *from* $\mathbf{C}$*. Finally,* $\mathbf{A}$ *returns a bit* b*, which is output by the experiment.*

Ideal$_{SSE,\mathbf{A},\mathbf{S}}(\lambda)$ *: The adversary* $\mathbf{A}$ *chooses a set of word* w *and sends it to the challenger* $\mathbf{C}$*.* $\mathbf{C}$ *sends* $\mathcal{L}_1(\mathsf{w})$ *to the simulator* $\mathbf{S}$*.* $\mathbf{S}$ *generates* ew *from* $\mathcal{L}_1(\mathsf{w})$*, and send it to* $\mathbf{C}$*.* $\mathbf{C}$ *relays* ew *to* $\mathbf{A}$*.* $\mathbf{A}$ *gives a polynomial number of words* w *to* $\mathbf{C}$*.* $\mathbf{C}$

sends $\mathcal{L}_2(\mathtt{w},\mathtt{w}')$ to **S**. **S** *generates a set of trapdoor* $\mathtt{td}(\mathtt{w}')$ *from* $\mathcal{L}_2(\mathtt{w},\mathtt{w}')$ *and sends them to* **C**. **C** *relays them to* **A**. *Finally,* **A** *returns a bit* $\mathtt{b}$*, which is output by the experiment.*

We say that SSE *is ($\mathcal{L}_1$, $\mathcal{L}_2$)-secure against chosen-keyword attacks if for all PPT adversaries* **A***, there exists a PPT simulator* **S** *such that*

$$Pr[\boldsymbol{Real}_{SSE,\mathbf{A}}(\lambda)=1]-Pr[\boldsymbol{Ideal}_{SSE,\mathbf{A},\mathbf{S}}(\lambda)=1]\leq \mathtt{negl}.$$

3 DEFINITION

We begin by reviewing the formal definition of an encrypted k–anonymization scheme. The participants include a user that wants to store sensitive information $\mathcal{PT}$ on an honest-but-curious server in such a way that (1) the server will not learn any useful information about $\mathcal{PT}$ and (2) the server is given the ability to k–anonymize $\mathcal{ET}$ and return $k\mathcal{ET}$ to the user.

3.1 Encrypted k–anonymization Scheme

In this subsection, we define an encrypted k–anonymization scheme.

Definition 2. EAS *consists of five polynomial time algorithms.*

$$\mathtt{EAS} = (\mathtt{Gen},\mathtt{Enc},\mathtt{Trpdr},\mathtt{Annmz},\mathtt{Dec})$$

such that

- $K \leftarrow \mathtt{Gen}(1^\lambda)$*: is a probabilistic algorithm, which is run by the user. It takes security parameter λ as input and outputs a secret key K.*
- $\mathcal{ET} \leftarrow \mathtt{Enc}(K,\mathcal{PT})$*: is a probabilistic algorithm run by the user. It takes a secret key K and a table $\mathcal{PT}$ as input and outputs an encrypted table $\mathcal{ET}$.*
- $\mathtt{td}(\mathtt{w}') \leftarrow \mathtt{Trpdr}(K,\mathtt{w}')$*: is a deterministic algorithm run by the user. It takes a secret key K and a set of word $\mathtt{w}'$ and outputs a set of trapdoor $\mathtt{td}(\mathtt{w}')$.*
- $k\mathcal{ET}$ *or* $\perp \leftarrow \mathtt{Annmz}(\mathcal{ET},\mathtt{td}(\mathtt{w}'), kv)$*: is a deterministic algorithm run by the server to k–anonymize an encrypted table $\mathcal{ET}$ with selected attributes. It takes an encrypted table $\mathcal{ET}$, a set of trapdoor $\mathtt{td}(\mathtt{w}')$, an integer kv. If it is not possible to generate an encrypted k–anonymized table $k\mathcal{ET}$, then output is $\perp$. Otherwise, it outputs $k\mathcal{ET}$ with provability $1-\varepsilon(\lambda)$ and ε is a negligible function.*
- $k\mathcal{PT} \leftarrow \mathtt{Dec}(K,k\mathcal{ET})$*: is a deterministic algorithm run by the user. It takes a secret key K and an encrypted k–anonymized table $k\mathcal{ET}$ and outputs a k–anonymized table $k\mathcal{PT}$.*

We say that an EAS *satisfies correctness if for any K output by* $\mathtt{Gen}(1^\lambda)$*, any $\mathcal{PT}$, any set of word $\mathtt{w}'$ in $\mathcal{PT}$, and any positive integer kv,*

$$k\mathcal{ET} \text{ or } \perp = \mathtt{Annmz}(\mathtt{Enc}(K,\mathcal{PT}),\mathtt{Trpdr}(K,\mathtt{w}'),kv) \quad (1)$$

and

$$k\mathcal{PT} = \mathtt{Dec}(K,k\mathcal{ET}), \quad (2)$$

where Equation (1) holds with provability $1-\varepsilon(\lambda)$ and ε is a negligible function, and Equation (2) holds with probability 1.

Similar with SSE, EAS has two phases called as a store phase and which is performed once, and an anonymization phase, which is performed a polynomial number of times. In the store phase, an user generates a plain table, encrypts it and stores it on a server. In the search phase, the user selects attributes on the table, and sends a trapdoor and an integer k for k–anonymization. The server k–anonymized the encrypted table and send it to the user.

Definition 3. EAS *played by a user $\mathcal{U}$ and a server $\mathcal{S}$ consists of two phases:*

- *Store phase*
  1. *$\mathcal{U}$ generates a secret key $K \leftarrow \mathtt{Gen}(1^\lambda)$.*
  2. *$\mathcal{U}$ encrypts a plain table $\mathcal{PT}$ by $\mathcal{ET} \leftarrow \mathtt{Enc}(K,\mathcal{PT})$.*
  3. *$\mathcal{U}$ gives an encrypted table $\mathcal{ET}$ to $\mathcal{S}$.*
- *Anonymization phase*
  1. *$\mathcal{U}$ selects s attributes from $\mathcal{A}$.*
  2. *$\mathcal{U}$ generates a trapdoor set $\mathtt{td}(\mathtt{w}') \leftarrow \mathtt{Trpdr}(K,\mathtt{w}')$ and send them and an integer kv to $\mathcal{S}$.*
  3. *$\mathcal{S}$ receives $\mathtt{td}(\mathtt{w}')$ and kv, generates an encrypted k–anonymized table $k\mathcal{ET} \leftarrow \mathtt{Annmz}(\mathcal{ET},\mathtt{td}(\mathtt{w}'),kv)$ and give $k\mathcal{ET}$ to $\mathcal{S}$.*
  4. *$\mathcal{U}$ gets a k–anonymized table $k\mathcal{PT}$ by $\mathcal{PT} \leftarrow \mathtt{Dec}(K,k\mathcal{ET})$.*

3.2 Security Definition

Let $\mathcal{L}_A(\mathtt{w}|$ all $w_i \in \mathtt{w}$ are stored in attributes or cells of $\mathcal{PT})$ denote the information that the server can learn in the store phase, and we abbreviate this leakage information $\mathcal{L}_A(\mathtt{w})$ as for simplicity. Furthermore, let $\mathcal{L}_B(\mathtt{w}|$ all $w_i \in \mathtt{w}$ are stored in attributes or cells of $\mathcal{PT},\mathtt{w}',kv)$ denote that in the anonymization phase, and we abbreviate this leakage information as $\mathcal{L}_B(\mathtt{w},\mathtt{w}',kv)$ for simplicity. Using these leakage information, we next define the semantic security of EAS.

Definition 4. *Let* $\mathtt{EAS} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Trpdr}, \mathtt{Annmz}, \mathtt{Dec})$ *be an* EAS, *and consider the following probabilistic experiments where* **A** *is an adversary,* **C** *is a challenger,* **S** *is a simulator and* $\mathcal{L}_A$ *and* $\mathcal{L}_B$ *are leakage algorithms:*

$\boldsymbol{Real}_{EAS,\mathbf{A}}(\lambda)$ *: The adversary* **A** *chooses a plain table* $\mathcal{PT}$ *and sends it to the challenger* **C**. **C** *begins by running* $\mathtt{Gen}(1^\lambda)$ *to generate a secret key* K *and send it to* **A**. **A** *outputs a plain table* $\mathcal{PT}$ *and receives* $\mathcal{ET}$ *from* **C** *by* $\mathcal{ET} \leftarrow \mathtt{Enc}(K, \mathcal{PT})$. **A** *repeats the following Step 1–3 polynomial times.*

1. **A** *selects attributes of* $\mathcal{PT}$.
2. **A** *gives a polynomial number of words* $\mathtt{w}'$ *for the attributes to* **C**.
3. **A** *receives trapdoors* $\mathtt{td}(\mathtt{w}') \leftarrow \mathtt{Trpdr}(K, \mathtt{w}')$ *from* **C**.

Finally, **A** *returns a bit* $\mathtt{b}$, *which is output by the experiment.*

$\boldsymbol{Ideal}_{EAS,\mathbf{A},\mathbf{S}}(\lambda)$ *: The adversary* **A** *chooses a plain table* $\mathcal{PT}$ *and sends it to the challenger* **C**. **C** *sends* $\mathcal{L}_A(\mathtt{w})$ *to the simulator* **S**. **S** *generates* $\mathcal{ET}$ *from* $\mathcal{L}_A(\mathtt{w})$, *and send it to* **C**. **C** *relays* $\mathcal{ET}$ *to* **A**. **A** *repeats the following step 1–3 polynomial times.*

- **A** *selects attributes of* $\mathcal{PT}$.
- **A** *sends a polynomial number of words* $\mathtt{w}'$ *for the attributes to* **C**. **C** *sends* $\mathcal{L}_B(\mathtt{w}, \mathtt{w}', kv)$ *to* **S**. **S** *generates trapdoors* $\mathtt{td}(\mathtt{w}')$ *from* $\mathcal{L}_B(\mathtt{w}, \mathtt{w}', kv)$ *and sends them to* **C**.
- **A** *receives them from* **C**.

Finally, **A** *returns a bit* $\mathtt{b}$, *which is output by the experiment.*

We say that EAS *is (*$\mathcal{L}_A$, $\mathcal{L}_B$*)-secure against chosen-keyword attacks if for all PPT adversaries* **A**, *there exists a PPT simulator* **S** *such that*

$$Pr[\boldsymbol{Real}_{EAS,\mathbf{A}}(\lambda) = 1] - Pr[\boldsymbol{Ideal}_{EAS,\mathbf{A},\mathbf{S}}(\lambda) = 1] \leq \mathtt{negl}.$$

4 EAS CONSTRUCTION

k–anonymization techniques are roughly classified into methods that use a generalized hierarchy, often called global recoding, and methods that do not, often called local recoding. If the purpose of anonymous data is clear, it is more advantageous to use a generalized hierarchy that can clarify the policies of anonymization. Section 4.1 describes a concrete EAS. In practice, generalized hierarchies are troublesome to create. Section 4.3 describes an algorithm to generate a domain generalization hierarchy from $\mathcal{ET}$.

4.1 EAS Construction using SSE

We now present our EAS construction.

- $K \leftarrow \mathtt{Gen}(1^\lambda)$: generate a secret key $sk \leftarrow \mathtt{SSE.Gen}(1^\lambda)$ and output $K = sk$.
- $\mathcal{ET} \leftarrow \mathtt{Enc}(K, \mathcal{PT})$
  1. encrypt each word in all attributes and cells of $\mathcal{PT}$.
     - $ew \leftarrow \mathtt{SSE.Enc}(K, w)$ for all $w \in \mathcal{A} \cup \mathcal{C}$
  2. output the table consisting of ew as $\mathcal{ET}$.
- $\mathtt{td}(\mathtt{w}') \leftarrow \mathtt{Trpdr}(K, \mathtt{w}')$:
  1. select attributes from $\mathcal{A}$. Without losing generality, we assume that $(a_1, \ldots a_s)$ are selected from $\mathcal{A}$ for simplicity.
  2. generate trapdoors of each word in the selected attributes and the corresponding columns.
     - $td(w') \leftarrow \mathtt{SSE.Trpdr}(K, w')$ for all $w' \in a_1 \cup \ldots a_s \cup \mathcal{D}_{c1} \cup \ldots \mathcal{D}_{cs}$.
     - $ew' \leftarrow \mathtt{SSE.Enc}(K, w')$ for all $w' \in \mathcal{D}_{c1} \cup \ldots \mathcal{D}_{cs}$.
  3. output the set of $td(w')$ and ew' as $\mathtt{td}(\mathtt{w}')$.
- $k\mathcal{ET}$ or $\perp \leftarrow \mathtt{Annmz}(\mathtt{td}(\mathtt{w}'), \mathcal{ET}, kv)$:
  1. for all $j \in [1, s]$
     (a) find an attribute a_i such that $\mathtt{SSE.Cmpr}(ew_i, td(w'_j)) = 1$ for $i \in [1, n]$ where ew_i is stored in an attribute a_i. If the a_i is not found, output $\perp$ and stop.
     (b) replace ew stored in all cells in $\mathcal{C}_j$ with ew' if $\mathtt{Cmpr}(ew, td(w')) = 1$ holds for all $w' \in \mathcal{D}_{cj}$.
     (c) count frequency $f(w')$ of each trapdoor such that $f(w') = \sum_{ew \in \mathcal{C}_i} \mathtt{Cmpr}(ew, td(w'))$ for all $w' \in \mathcal{D}_{cj}$ and all ew stored in $\mathcal{C}_j$, and generate a domain generalization hierarchy by Algorithm 1 described in Section 4.3.
  2. this generalization step depends on selected generalization algorithms. Here we give it based on the algorithm proposed by Wang et al. (Wang et al., 2004).
     (repeat the following step (a)–(c) until k-anonymity is satisfied or one node in every domain generation hierarchy is left)
     (a) select two leaf nodes among $DGH_1, \ldots DGH_s$ with the smallest information entropy lost by generalization to the nearest parent node, i.e. two leaf nodes containing ew'_1 or ew'_2 are selected by using the measure of information entropy and the parent node contains $(ew'_1 \texttt{ or } ew'_2)$.
     (b) replace a value on the cells to that of the parent node, i.e. ew'_1 and ew'_2 in the cells are replaced with $(ew'_1 \texttt{ or } ew'_2)$.

(c) delete the two leaf nodes from a DGH selected at Step 2.(a) and set the parent node as a new leaf node of the DGH.

3. if the table is satisfied with k–anonymity for the selected attributes, shuffle record order randomly, and output it as $k\mathcal{ET}$. Otherwise, output $\perp$.

- $k\mathcal{PT} \leftarrow \texttt{Dec}(K, k\mathcal{ET})$:
  1. decrypt each word in all attributes and cells of $k\mathcal{ET}$.
     - $w \leftarrow \texttt{SSE.Dec}(K, ew)$ for all $ew \in \mathcal{A} \cup \mathcal{C}$
  2. output the table as $k\mathcal{PT}$.

Step 2.(a)–(c) applies the greedy method proposed by Wang et al. (Wang et al., 2004). The method sequentially selects optimal anonymization target data in accordance with a given generalized hierarchy. Through this sophisticated anonymization process, it is expected to output higher quality anonymous data than the method of LeFerve et al.(LeFevre et al., 2005), which is a representative global re-encoding method and performs rough anonymization processing.

4.2 Security

We define $\mathsf{w}'_{[\mathsf{s}]}$ consists of words for queries of k–anonymization with s attributes: $\mathsf{w}'_{[\mathsf{s}]} = \{w' |$ all w' stored in $(a_1, \ldots a_s)$ and $w' \in \mathcal{D}_{c1} \cup \ldots \mathcal{D}_{cs}\}$.

Theorem 1. *If* SSE *is* $(\mathcal{L}_1, \mathcal{L}_2)$*-secure against chosen-keyword attacks, then the proposed* EAS *construction is* $(\mathcal{L}_A, \mathcal{L}_B)$*-secure against chosen-keyword attacks such that* $\mathcal{L}_A(\mathsf{w}) = \mathcal{L}_1(\mathsf{w})$*, and* $\mathcal{L}_B(\mathsf{w}, \mathsf{w}'_{[\mathsf{s}]}, kv) = \mathcal{L}_1(\mathsf{w}'_{[\mathsf{s}]})$*,* $\mathcal{L}_2(\mathsf{w}, \mathsf{w}'_{[\mathsf{s}]})$ *and* kv*.*

Proof. Let $\mathbf{S}'$ be a simulator of the $(\mathcal{L}_1, \mathcal{L}_2)$-secure SSE scheme. We construct a simulator $\mathbf{S}$ of EAS, which achieves $(\mathcal{L}_A, \mathcal{L}_B)$-secure as follows.

(Store phase) In $\mathbf{Ideal}_{EAS,\mathbf{A},\mathbf{S}}$, $\mathbf{S}$ takes $\mathcal{L}_A(\mathsf{w}) = \mathcal{L}_1(\mathsf{w})$ as input. $\mathbf{S}$ runs $\mathbf{S}'(\mathcal{L}_1(\mathsf{w}))$ and gets its output ew from $\mathbf{S}'$. $\mathbf{S}$ constructs $\mathcal{ET}$ from ew and sends it to $\mathbf{C}$.

(Anonymization phase) In $\mathbf{Ideal}_{EAS,\mathbf{A},\mathbf{S}}$, $\mathbf{S}$ takes $\mathcal{L}_B(\mathsf{w}, \mathsf{w}'_{[\mathsf{s}]}, kv)$ as input. $\mathbf{S}$ runs $\mathbf{S}'(\mathcal{L}_2(\mathsf{w}, w'_i | w'_i$ stored in $a_i))$, $\mathbf{S}'(\mathcal{L}_2(\mathsf{w}, w'_i |$ all $w'_i \in \mathcal{D}_{ci}))$ for $i = 1, \ldots s$, and get its trapdoor $td(w')$ respectively. $\mathbf{S}$ runs $\mathbf{S}'(\mathcal{L}_1(w'_i |$ all $w'_i \in \mathcal{D}_{ci}))$ for $i = 1, \ldots s$, and get its trapdoor ew' respectively. $\mathbf{S}$ constructs $k\mathcal{ET}$ by $\texttt{EAS.Annmz}(\texttt{td}(\mathsf{w}'), \mathcal{ET}, kv)$, and send $k\mathcal{ET}$ to $\mathbf{C}$. □

4.3 Generation of Domain Generalization Hierarchy

We design Algorithm 1, which generate domain generalization hierarchies from searchable encrypted database. This work is inspired by the research of Harada et al., which use a data compression rule such as Huffman code (Huffman, 1952) to create a domain generalization hierarchy from a table $\mathcal{PT}$ (Harada et al., 2012).

Algorithm 1: Generation of Domain Generalization Hierarchy.

INPUT: trapdoor set $\texttt{td}(\mathsf{w}'_\mathsf{i})$, attribute a_i, column $\mathcal{C}_i$, frequency set $f(\mathsf{w}'_\mathsf{i})$;
OUTPUT: domain generalization hierarchy DGH_i;

1. extract all ew' from $\texttt{td}(\mathsf{w}'_\mathsf{i})$ and set each ew' as a value of leaf nodes of DGH_i
2. store each combination of ew' and a frequency $f(w')$ in a list Q in ascending order of $f(w')$, i.e. $(ew'_1, f(w'_1)), (ew'_2, f(w'_2)), \ldots$ with $f(w'_1) \leq f(w'_2) \leq \ldots$.
3. (repeat the following (a)–(d) until the number of nodes in the list Q is 1)
   (a) retrieve two nodes with the lowest and the 2nd lowest frequencies such as ew'_1, ew'_2, and delete them from the list Q.
   (b) create a new parent node of children ew'_1 and ew'_2 such as $(ew'_1 \texttt{ or } ew'_2)$
   (c) assigns frequency of the parent node to the sum of frequencies of the child nodes. e.g. frequency of the node $(ew'_1 \texttt{ or } ew'_2)$ is $f(w'_1) + f(w'_2)$.
   (d) add the parent node to the list Q and sort it again.
4. the last attribute value in list Q is taken as the root node. Output this tree as the domain generalization hierarchy DGH_i

On the basis of the results of aggregation, upper nodes of the generalized hierarchy are gradually formed from nodes with low appearance frequency. The appearance frequency of the upper node is the sum of the appearance frequencies of the lower nodes, and so that the degree of generalization is not too strong, Huffman code or the like is used. Also, an upper node generalizing lower nodes is represented by a logical sum (or) of values of lower nodes.

In this paper, for simplicity, we ignore the order relation of attributes and use Huffman code for generalized hierarchy. In the case of handling numeric column such as age and geographical information, instead of Huffman code and SSE, one can apply Hu-Tucker code (Hu and Tucker, 1971) and order preserving encryption to maintain the order relation.

5 EXPERIMENTAL PERFORMANCE EVALUATION

We implement the proposed scheme to utilize 256-bit AES and SHA-256 for SSE (Yoshino et al., 2011). The performance is evaluated on a conventional computer, which is equipped with 3.4 GHz Core i7 6700 CPU, 32GB memory, Cent OS 7.4, and JVM 1.8.0. To measure the mounting performance for data scalability, we generated dummy data, which is shown in Table 1. Three attributes (occupation, gender, and address) are randomly selected in these value ranges, and the last attribute (birth date) is sampled from statistics based on Japanese population estimates (Bureau, 2016).

Table 1: Test Dataset Consisting of Attributes and Possible Values.

attribute	possible value
occupation	integer in $[1,24]$
gender	female or male
address	5000 types
birth date	DD/MM/YYYY

Performance of the `Enc`, `Trpdr`, or `Dec` function of the proposed `EAS` depends on `SSE.Enc`, `SSE.Trpdr`, or `SSE.Dec`, respectively, and the running time increases linearly to data size of $\mathcal{PT}$, $\mathcal{ET}$, or message space. Thus, we measure the last function `Annmz` of the proposed `EAS`, in which running time is not easily estimated. The generalization process consisting of Step 2.(a)–(c) is the major process for `Annmz`. Figure 3 shows performance of the generalization process on the dummy data with 4 attributes and $k = 3$. It takes about 440 milliseconds for 10^3 records, about 6 seconds for 10^4 records, about 80 seconds for 10^5 records, and about 168 seconds for 10^6 records. Compared with the case of the plaintext (Wang et al., 2004), the generalization step of the proposed `EAS` is comparable and almost equivalent. `Annmz` has a branch process: if domain generation hierarchies are not given, then Step 2 generates them. Figure 4 shows the generation time of the generalized hierarchy: about 77 milliseconds to 229 milliseconds for 10^3 to 10^6 records. That is negligible compared with the generalization time.

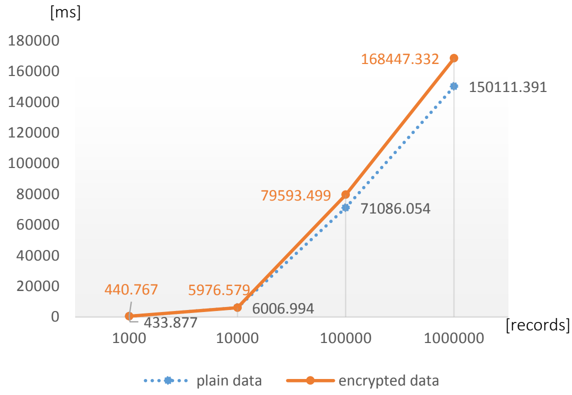

Figure 3: Performance on 3–anonymization

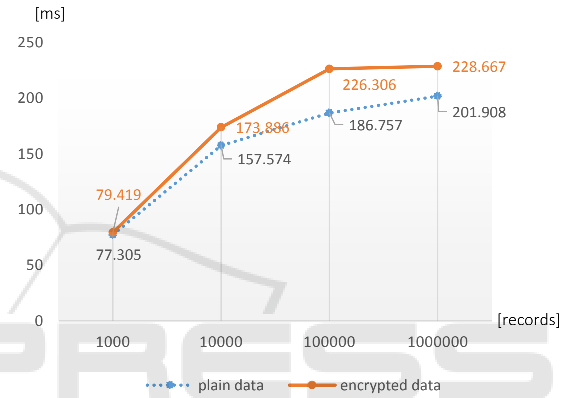

Figure 4: Performance on *DGH* Generation

6 CONCLUSION

We pointed out that outsourcing k–anonymization processes may lead to information leakage, thus we defined an encrypted k–anonymization scheme (`EAS`) and a semantic security model of `EAS`. Furthermore, we give a construction of `EAS` and prove the security under the semantic security model. Finally, we implemented the proposal on a general-purpose PC and demonstrated its efficiency. As a consequence, our high-speed `EAS` makes it feasible not only to prevent information leakage from database but also to gain the advantage that even the server can prevent unintended observation of the given database.

REFERENCES

Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT*, pages 506–522. Springer.

Boneh, D. and Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference*, pages 535–554. Springer.

Bureau, S. (2016). Ministry of internal affairs and communications. *Government of Japan*.

Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2006). Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings* *of the 13th ACM Conference on Computer and Communications Security*, CCS, pages 79–88, New York, NY, USA. ACM.

Curtmola, R., Garay, J., Kamara, S., and Ostrovsky, R. (2011). Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934.

Ducas, L. and Micciancio, D. (2015). FHEW: bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology – EUROCRYPT*, pages 617–640.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, STOC, pages 169–178, New York, NY, USA. ACM.

Harada, K., Sato, Y., and Togashi, Y. (2012). Reducing amount of information loss in k-anonymization for secondary use of collected personal information. In *SRII Annual Global Conference*, pages 61–69. IEEE.

Hu, T. C. and Tucker, A. C. (1971). Optimal computer search trees and variable-length alphabetical codes. *SIAM Journal on Applied Mathematics*, 21(4):514–532.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. In *International Conference on Financial Cryptography and Data Security*, pages 136–149. Springer.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. (2005). Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60. ACM.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 25–25. IEEE.

Meyerson, A. and Williams, R. (2004). On the complexity of optimal k-anonymity. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 223–228. ACM.

Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2012). Cryptdb: Processing queries on an encrypted database. *ACM Communications*, 55(9):103–111.

Samarati, P. (2001). Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027.

Samarati, P. and Sweeney, L. (1998). Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International.

Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 44–55. IEEE.

Sweeney, L. (2002a). Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588.

Sweeney, L. (2002b). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

Wang, K., Philip, S. Y., and Chakraborty, S. (2004). Bottom-up generalization: A data mining solution to privacy protection. In *Proceedings of the IEEE International Conference on Data Mining*.

Yoshino, M., Naganuma, K., and Satoh, H. (2011). Symmetric searchable encryption for database applications. In *International Conference on Network-Based Information Systems (NBiS)*, pages 657–662. IEEE.