Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 381361, 12 pages
http://dx.doi.org/10.1155/2014/381361

Research Article

Sharing Privacy Protected and Statistically Sound Clinical Research Data Using Outsourced Data Storage

Geontae Noh, Ji Young Chun, and Ik Rae Jeong

Center for Information Security Technologies (CIST), Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Republic of Korea

Correspondence should be addressed to Ik Rae Jeong; irjeong@korea.ac.kr

Received 14 November 2013; Accepted 28 April 2014; Published 18 May 2014

Academic Editor: Jongsung Kim

It is critical to scientific progress to share clinical research data stored in outsourced generally available cloud computing services. Researchers are able to obtain valuable information that they would not otherwise be able to access; however, privacy concerns arise when sharing clinical data in these outsourced publicly available data storage services. HIPAA requires researchers to deidentify private information when disclosing clinical data for research purposes and describes two available methods for doing so. Unfortunately, both techniques degrade statistical accuracy. Therefore, the need to protect privacy presents a significant problem for data sharing between hospitals and researchers. In this paper, we propose a controlled secure aggregation protocol to secure both privacy and accuracy when researchers outsource their clinical research data for sharing. Since clinical data must remain private beyond a patient's lifetime, we take advantage of lattice-based homomorphic encryption to guarantee long-term security against quantum computing attacks. Using lattice-based homomorphic encryption, we design an aggregation protocol that aggregates outsourced ciphertexts under distinct public keys. It enables researchers to get aggregated results from outsourced ciphertexts of distinct researchers. To the best of our knowledge, our protocol is the first aggregation protocol which can aggregate ciphertexts which are encrypted with distinct public keys.

1. Introduction

Researchers can accelerate their learning curve if they are able to freely access clinical data from other studies. Such clinical data sharing in outsourced publicly available services is crucial to scientific progress in clinical research. The benefits of clinical data sharing using these services have been widely reported, including reduced research costs, reduced management costs, improvement of quality control, and reduced time in discovering diseases and dealing with them effectively. Through shared data, researchers access valuable information that they would not ordinarily obtain. In its policy statement on grants, the U.S. National Institute of Health (NIH) supports data sharing by requiring investigators to include a plan for data sharing or explain why data sharing is not possible.

The problem with clinical data sharing in outsourced publicly available services for research is that researchers can inadvertently violate patient privacy. HIPAA (Health Insurance Portability and Accountability Act) offers protection of patients' personal health information, but it is difficult not to invade patient privacy while sharing clinical data in outsourced publicly available data storage services [1]. Therefore, researchers would rather not make their data publicly available than run the risk of violating HIPAA.

To mitigate privacy concerns, the HIPAA describes two ways to use and disclose clinical data for research purposes. Under the HIPAA Safe Harbor policy, clinical data should be deidentified so that patients are not individually identifiable. The HIPAA Safe Harbor policy stipulates that the data sharer should deidentify data by removing 18 specific data attributes, such as name, address, and all dates related to the individual patient, which may include birth date and date of death. (In addition, some researchers continue to assert that combinations of other data that are excluded from the HIPAA Safe Harbor policy could individually identify a specific person with nonnegligible probability, so they insist that there are more than 18 specific data attributes that should be included

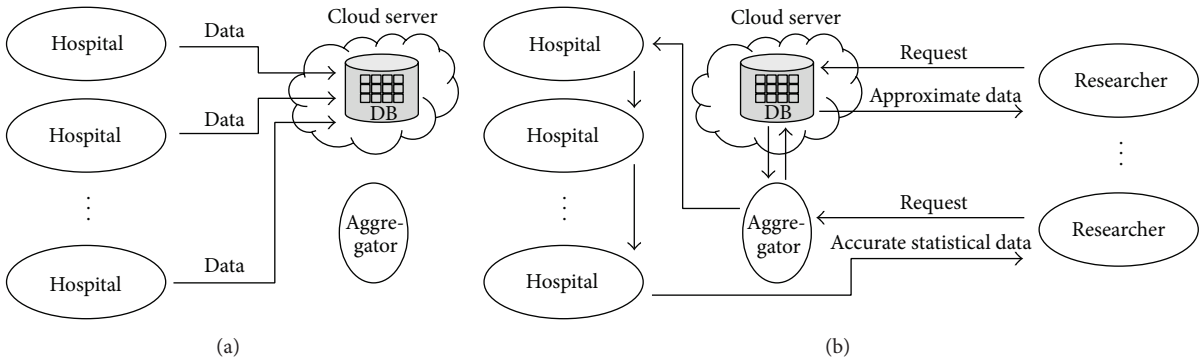

Figure 1: System environment ((a) store phase and (b) search phase).

in the Safe Harbor policy [2–4].) Once identifying information has been removed, the deidentified data are no longer subject to the Institutional Review Board (IRB) overview. Alternatively, researchers may use anonymity techniques to deidentify patient information instead of removing all of the 18 or more data attributes that are required to be deidentified. To date, anonymity techniques have been proposed, such as k-anonymity [5–7], ℓ-diversity [8], and t-closeness [9].

It is useful to protect patient privacy with deidentification formats when sharing clinical data in outsourced publicly available data storage services, but doing so degrades the statistical accuracy since it makes it difficult to get precise statistical results. However, in some cases where accurate statistical data on patients are critical, the anonymity techniques for deidentification are not sufficient. Due to poorly deidentified data, researchers can make bad decisions. Therefore, there needs to be a privacy-preserving method for accurate statistical data.

In this work, we propose how to outsource clinical research data securely and how to control the outsourced data against potential breaches of privacy, while not compromising the accuracy of statistical results. For example, a malicious researcher could circumvent any encryption by asking for one piece of data on one patient; in this way, the researcher could ultimately obtain each patient's private information. In this case, we propose a method that will foil such a malicious attempt.

The system environment we propose for hospitals, aggregator, and researchers is illustrated in Figure 1. In our system, each hospital outsources its own clinical data to cloud storage servers. The clinical data must be deidentified or encrypted to be stored publicly. We use a hybrid method to store the clinical data; that is, we deidentify the clinical data for approximate statistical data requests and encrypt numerical clinical data for accurate statistical data requests. Therefore, researchers can request both approximate and accurate statistical data. Researchers would obtain approximate statistical data directly from the cloud storage servers but cannot obtain accurate statistical data directly. When researchers would like to get accurate statistical data, they can get the data through the aggregator. The aggregator aggregates the requested data from the encrypted database stored in the cloud storage servers, and then asks each hospital to decrypt the aggregated data by consent. Hospitals can refuse the request of the aggregator, unless initial consents that have been obtained from patients allow the secondary research. Since there are ethical and practical issues associated with aggregating databases [10], hospitals should ensure that they are following "best practices" for their outsourced data, such as determining whether initial consents that have been obtained allow secondary research.

Since clinical data should remain private beyond a patient's lifetime, cryptographic long-term security is absolutely needed [11] in the area of managing clinical data. Therefore, we take advantage of a lattice-based homomorphic encryption in order to encrypt clinical data. Lattice-based cryptography is believed to be secure against quantum computing attacks and guarantees long-term security. RSA, ECC, and DLP cryptosystems, which have gained attention so far, could be attacked with quantum computers [12]. Quantum computing is not yet possible, but may become so in our lifetime. Furthermore, lattice-based cryptographic algorithms are more efficient than others in computational overhead because they require only linear operations on matrices such as addition, multiplication, and inverse.

In 2009, Gentry proposed the first fully homomorphic encryption scheme using ideal lattices [13]. In 2010, Gentry et al. have proposed a novel homomorphic encryption scheme (referred to as GHV homomorphic encryption scheme hereafter) that supports one multiplicative and polynomially many additive operations on encrypted data [14]. As a building block, we use a variant of the GHV homomorphic encryption scheme, which supports only additive operations. This can make it possible to aggregate ciphertexts which are encrypted under distinct public keys. Due to this property, the aggregator can aggregate the outsourced encrypted data from hospitals. Therefore, once hospitals outsource their clinical data, they do not need to encrypt the clinical data again for individual researchers. Each hospital only has to encrypt the clinical data, and then it outsources the encrypted data.

Contributions. In this paper, we propose a controlled secure aggregation protocol in sharing clinical research data to balance the interests between hospitals and researchers. The main contributions of this paper are as follows.

(i) Researchers can get approximate statistical data from deidentified clinical data directly. Researchers can also obtain accurate and aggregated clinical data from the encrypted database through the aggregator by obtaining each hospital's consent.

(ii) We take advantage of a lattice-based homomorphic encryption which is secure against quantum computing attacks. Therefore, our protocol resists quantum attacks and could remain secure in the long term.

(iii) The aggregator can aggregate encrypted clinical data which are encrypted with distinct public keys. Therefore, hospitals do not have to encrypt the clinical data again whenever researchers send requests.

To the best of our knowledge, our protocol is the first protocol which takes advantage of the lattice-based homomorphic encryption in order to share outsourced clinical research data.

Organization. The remainder of this paper is organized as follows. Section 2 provides related works and background. Section 3 presents our controlled secure aggregation protocol. We present our secure clinical data aggregation system in Section 4 and analyze it in Section 5. We provide our conclusions in Section 6.

2. Related Works and Background

In this section, we present related works and background.

2.1. Data Aggregation Based on Homomorphic Encryption. In 2004, Hacıgümüş et al. proposed an aggregation protocol over encrypted relational databases [15]. They designed the aggregation protocol using the PH (Privacy Homomorphism) which supports additive and multiplicative operations. In the aggregation protocol, permitted users can get the accurate and aggregated data. However, Mykletun and Tsudik showed that the aggregation protocol using the PH is not secure against ciphertext-only attacks [16]. Since then, various aggregation protocols over encrypted data have been proposed in the literatures [17–23]. Among those protocols, few literatures have focused on the health-care environment. In addition, most protocols considered the aggregation for a single provider's data.

Molina et al. [22] designed the aggregation protocol, HICCUPS, using homomorphic encryption in the health-care environment. In HICCUPS, clinical data of multiple providers can be aggregated as follows: caregivers who store clinical data on their own database are randomly chosen as the aggregator. When a researcher requests the aggregated result, the aggregator aggregates the encrypted clinical data from each caregiver and sends the aggregated result to the researcher.

Since HICCUPS is not based on the outsourcing system, caregivers have to provide clinical data whenever a researcher requests a certain data. In addition, HICCUPS requires each caregiver to aggregate and encrypt clinical data with the researcher's public key so that the aggregator can aggregate the encrypted clinical data. However, a malicious aggregator may want to have a researcher get a misleading result by intentionally excluding the encrypted clinical data from certain caregivers. Even though the malicious aggregator fabricates the aggregated result on purpose, there is no way for a researcher to detect the malicious behavior of the aggregator in HICCUPS.

To resolve the above issues, we design the controlled secure aggregation protocol which can aggregate outsourced ciphertexts under distinct public keys. Therefore, data providers (or hospitals) do not have to encrypt clinical data again, once they have outsourced their clinical data. Our protocol also enables a researcher to detect the malicious behavior of the aggregator. If the malicious aggregator excludes the encrypted clinical data from certain data providers on purpose, a researcher can detect that. Since each data provider (or hospital) collaboratively makes the aggregated data decryptable by a researcher, if the aggregated data is generated maliciously then the researcher cannot get a plausible result. The researcher gets the random result that cannot seem to be a meaningful result. Therefore, in our protocol, the researcher can be sure that the requested data are aggregated correctly.

2.2. Anonymity Techniques for Deidentification. Samarati and Sweeney introduced an anonymity technique called k-anonymity [5–7]. They considered a relational database that consists of unique identifiers, quasi-identifiers, and sensitive attributes. A unique identifier is any attribute that is able to identify only one private individual, such as a personal ID, an e-mail address, or a cell phone number. A quasi-identifier is any set of attributes that can be joined with additional information to identify only one private individual, such as a zip code and a birthday. A sensitive attribute is any attribute that a data owner does not want to publish, such as health-care data. In order to preserve privacy, all unique identifiers must be removed and all quasi-identifiers must be anonymized. In k-anonymity, each quasi-identifier is indistinguishable from at least $k - 1$ other quasi-identifiers. Tables 1 and 2 are good examples of the original health-care data and the 4-anonymous pieces of health-care data.

However, k-anonymity is not secure against homogeneity attacks and background knowledge attacks [8]. For example, suppose that Alice knows that Bob is in his twenties and his zip code is 13032; then, Alice can identify that Bob must have a gastric ulcer from Table 2.

To mitigate these attacks, Machanavajjhala et al. introduced a new anonymity technique, called ℓ-diversity [8]. In ℓ-diversity, all the equivalence classes that have the same quasi-identifiers must have ℓ or more different sensitive attributes. Table 3 shows the 3-diverse kinds of health-care data.

Since this result of ℓ-diversity, Li et al. showed that ℓ-diversity is insufficient for anonymity [9]. In ℓ-diversity, any information can be released if there exists a significant distribution difference between sensitive attributes of any equivalence class and all sensitive attributes. For example, if Alice knows Bob's personal information such as his age and zip code, she will be able to identify from Table 3 that Bob

Table 1: Original health-care data.

Number	Nonsensitive		Sensitive
	Zip code	Age	Condition
1	13033	22	Gastric ulcer
2	13062	25	Gastric ulcer
3	13032	27	Gastric ulcer
4	13067	28	Gastric ulcer
5	13065	31	Flu
6	13038	33	Stomach cancer
7	13035	35	Gastritis
8	13060	36	Gastritis
9	14856	43	Flu
10	14850	45	Gastritis
11	14852	50	Stomach cancer
12	14855	52	Gastric ulcer

Table 2: 4-anonymous health-care data.

Number	Nonsensitive		Sensitive
	Zip code	Age	Condition
1	130∗∗	<30	Gastric ulcer
2	130∗∗	<30	Gastric ulcer
3	130∗∗	<30	Gastric ulcer
4	130∗∗	<30	Gastric ulcer
5	130∗∗	3∗	Flu
6	130∗∗	3∗	Stomach cancer
7	130∗∗	3∗	Gastritis
8	130∗∗	3∗	Gastritis
9	148∗∗	≥40	Flu
10	148∗∗	≥40	Gastritis
11	148∗∗	≥40	Stomach cancer
12	148∗∗	≥40	Gastric ulcer

has stomach-related disease (e.g., gastric ulcer, gastritis, and stomach cancer.)

To mitigate such potential problem, Li et al. introduced another new anonymity technique, called t-closeness [9]. t-closeness requires the distribution of sensitive attributes of any equivalence class to be similar to that of all sensitive attributes.

2.3. GHV *Homomorphic Encryption Scheme.* GHV homomorphic encryption scheme supports one multiplicative and polynomially many additive operations on encrypted data [14]. The security of the GHV homomorphic encryption scheme is based on the learning with errors (LWE) problem [24] which is one of the hardest assumptions so far.

Let n be the security parameter, then other parameters are as follows:

(i) $c = c(n) > 0$,

(ii) p is a positive integer by setting a prime number $q = \omega(p^2 n^{3c+1} \log^5 n)$,

(iii) $m = \lfloor 8n \log q \rfloor$, and

Table 3: 3-diverse health-care data.

Number	Nonsensitive		Sensitive
	Zip code	Age	Condition
3	1303∗	<40	Gastric ulcer
1	1303∗	<40	Gastric ulcer
7	1303∗	<40	Gastritis
6	1303∗	<40	Stomach cancer
8	1306∗	<40	Gastritis
2	1306∗	<40	Gastric ulcer
5	1306∗	<40	Flu
4	1306∗	<40	Gastric ulcer
10	1485∗	≥40	Gastritis
11	1485∗	≥40	Stomach cancer
12	1485∗	≥40	Gastric ulcer
9	1485∗	≥40	Flu

(iv) $\beta = 1/(27n^{1+(3c/2)} \log n \log q \sqrt{qm})$ is a Gaussian parameter.

Then the IND-CPA secure [25] GHV homomorphic encryption scheme GHV = {GHV.Key, GHV.Enc, GHV.Dec, GHV.Add, GHV.Mul} is as follows.

(i) GHV.Key$(1^n, 1^m, q)$: given n, m, and q, output a public key $pk = \mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a secret key $sk = \mathbf{T} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{TA} = \mathbf{0} (\mathrm{mod} q)$, $\mathbf{T}$ is invertible, and the elements of $\mathbf{T}$ are bounded by $O(n \log q)$. (To generate two matrices $\mathbf{A}$ and $\mathbf{T}$, the trapdoor sampling algorithm in [26] can be used. For further details, please refer to [14].)

(ii) GHV.Enc$(pk, \mathbf{B})$: given pk and a plaintext $\mathbf{B} \in \mathbb{Z}_p^{m \times m}$, choose a uniformly random matrix $\mathbf{S} \in \mathbb{Z}_q^{n \times m}$ and a Gaussian error matrix $\mathbf{X} \in \mathbb{Z}_q^{m \times m}$. Then output a ciphertext $\mathbf{C} = \mathbf{AS} + p\mathbf{X} + \mathbf{B} (\mathrm{mod} q)$.

(iii) GHV.Dec$(sk, \mathbf{C})$: given sk and a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$, compute $\mathbf{E} = \mathbf{TCT}^t (\mathrm{mod} q)$. Then output a plaintext $\mathbf{B} = \mathbf{T}^{-1}\mathbf{E}(\mathbf{T}^t)^{-1} (\mathrm{mod} p)$.

In this algorithm,

$$\begin{aligned}
\mathbf{E} &= \mathbf{TCT}^t \left(\mathrm{mod} q\right) \\
&= \mathbf{TAST}^t + \mathbf{T}p\mathbf{XT}^t + \mathbf{TBT}^t \left(\mathrm{mod} q\right) \\
&= \mathbf{T}p\mathbf{XT}^t + \mathbf{TBT}^t \left(\mathrm{mod} q\right) \\
&= \mathbf{T}p\mathbf{XT}^t + \mathbf{TBT}^t, \qquad (1) \\
\mathbf{B} &= \mathbf{T}^{-1}\mathbf{E}\left(\mathbf{T}^t\right)^{-1} \\
&= \mathbf{T}^{-1}\mathbf{T}p\mathbf{XT}^t\left(\mathbf{T}^t\right)^{-1} + \mathbf{T}^{-1}\mathbf{TBT}^t\left(\mathbf{T}^t\right)^{-1} \\
&= p\mathbf{X} + \mathbf{B} = \mathbf{B} \left(\mathrm{mod} p\right).
\end{aligned}$$

(iv) $\mathsf{GHV.Add}(\mathbf{C}_1, \ldots, \mathbf{C}_n)$: given n ciphertexts, $\mathbf{C}_1 = \mathbf{AS}_1 + p\mathbf{X}_1 + \mathbf{B}_1 (\mathrm{mod} q)$, …, $\mathbf{C}_n = \mathbf{AS}_n + p\mathbf{X}_n + \mathbf{B}_n (\mathrm{mod} q)$, output

$$\begin{aligned}\mathbf{C}_1 + \cdots + \mathbf{C}_n &= \mathbf{AS}_1 + p\mathbf{X}_1 + \mathbf{B}_1 + \cdots \\ &\quad + \mathbf{AS}_n + p\mathbf{X}_n + \mathbf{B}_n \left(\mathrm{mod} q\right) \\ &= \mathbf{A}\left(\mathbf{S}_1 + \cdots + \mathbf{S}_n\right) \\ &\quad + p\left(\mathbf{X}_1 + \cdots + \mathbf{X}_n\right) \\ &\quad + \left(\mathbf{B}_1 + \cdots + \mathbf{B}_n\right) \left(\mathrm{mod} q\right) \\ &= \mathbf{AS}' + p\mathbf{X}' + \mathbf{B}' \left(\mathrm{mod} q\right).\end{aligned} \tag{2}$$

That is, the output of $\mathsf{GHV.Dec}(sk, \mathbf{C}_1 + \cdots + \mathbf{C}_n)$ is $\mathbf{B}' = \mathbf{B}_1 + \cdots + \mathbf{B}_n$.

(v) $\mathsf{GHV.Mul}(\mathbf{C}_1, \mathbf{C}_2)$: given two ciphertexts, $\mathbf{C}_1 = \mathbf{AS}_1 + p\mathbf{X}_1 + \mathbf{B}_1 (\mathrm{mod} q)$ and $\mathbf{C}_2 = \mathbf{AS}_2 + p\mathbf{X}_2 + \mathbf{B}_2 (\mathrm{mod} q)$, output

$$\begin{aligned}\mathbf{C}_1 \cdot \mathbf{C}_2^t &= \left(\mathbf{AS}_1 + p\mathbf{X}_1 + \mathbf{B}_1\right) \cdot \left(\mathbf{AS}_2 + p\mathbf{X}_2 + \mathbf{B}_2\right)^t \left(\mathrm{mod} q\right) \\ &= \mathbf{A} \cdot \left(\mathbf{S}_1 \mathbf{C}_2^t\right) + p\left(\mathbf{X}_1 \left(p\mathbf{X}_2 + \mathbf{B}_2\right) + \mathbf{B}_1 \mathbf{X}_2^t\right) \\ &\quad + \mathbf{B}_1 \mathbf{B}_2^t + \left(p\mathbf{X}_1 + \mathbf{B}_1\right) \mathbf{S}_2^t \cdot \mathbf{A}^t \left(\mathrm{mod} q\right) \\ &= \mathbf{AS}'' + p\mathbf{X}'' + \mathbf{B}'' + \mathbf{S}''' \mathbf{A}^t \left(\mathrm{mod} q\right).\end{aligned} \tag{3}$$

That is, the output of $\mathsf{GHV.Dec}(sk, \mathbf{C}_1 \cdot \mathbf{C}_2^t)$ is $\mathbf{B}'' = \mathbf{B}_1 \cdot \mathbf{B}_2^t$.

In this paper, we use, as a building block, a variant version of the GHV homomorphic encryption scheme which supports only additive operations. We call this variant version of the GHV homomorphic encryption scheme a GHV^* homomorphic encryption scheme hereafter. We can replace $\mathbf{S} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{X} \in \mathbb{Z}_q^{m \times m}$, $\mathbf{B} \in \mathbb{Z}_p^{m \times m}$, and $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$ of the GHV homomorphic encryption scheme with $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \in \mathbb{Z}_q^m$, $\mathbf{b} \in \mathbb{Z}_p^m$, and $\mathbf{c} \in \mathbb{Z}_q^m$ of the GHV^* homomorphic encryption scheme without any loss of security. Then the IND-CPA secure [25] GHV^* homomorphic encryption scheme $\mathsf{GHV}^* = \{\mathsf{GHV}^*.\mathsf{Key}, \mathsf{GHV}^*.\mathsf{Enc}, \mathsf{GHV}^*.\mathsf{Dec}, \mathsf{GHV}^*.\mathsf{Add}\}$ is as follows.

(i) $\mathsf{GHV}^*.\mathsf{Key}(1^n, 1^m, q)$: given n, m, and q, output a public key $pk = \mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a secret key $sk = \mathbf{T} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{TA} = \mathbf{0} (\mathrm{mod} q)$, $\mathbf{T}$ is invertible, and the elements of $\mathbf{T}$ are bounded by $O(n \log q)$.

(ii) $\mathsf{GHV}^*.\mathsf{Enc}(pk, \mathbf{b})$: given pk and a plaintext $\mathbf{b} \in \mathbb{Z}_p^m$, choose a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a Gaussian error vector $\mathbf{x} \in \mathbb{Z}_q^m$. Then output a ciphertext $\mathbf{c} = \mathbf{As} + p\mathbf{x} + \mathbf{b} (\mathrm{mod} q)$.

(iii) $\mathsf{GHV}^*.\mathsf{Dec}(sk, \mathbf{c})$: given sk and a ciphertext $\mathbf{c} \in \mathbb{Z}_q^m$, compute $\mathbf{e} = \mathbf{Tc} (\mathrm{mod} q)$. Then output a plaintext $\mathbf{b} = \mathbf{T}^{-1}\mathbf{e} (\mathrm{mod} p)$.

2.4. Ajtai's One-Way Function. Ajtai constructed a one-way function whose security is based on some well known approximation problems in lattices [27, 28].

Let n be the security parameter, m a positive integer, and q a positive integer. For a uniformly random matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{r} \in \{0,1\}^m$, the Ajtai's one-way function $\mathsf{h}_{\mathbf{M}} : \{0,1\}^m \to \mathbb{Z}_q^n$ is as follows:

$$\mathsf{h}_{\mathbf{M}}(\mathbf{r}) = \mathbf{Mr} \left(\mathrm{mod} q\right). \tag{4}$$

Note that the Ajtai's one-way function $\mathsf{h}_{\mathbf{M}}$ is regular [29]; that is, every output of $\mathsf{h}_{\mathbf{M}}$ is uniformly distributed over $\mathbb{Z}_q^n$ [30].

3. Controlled Secure Aggregation Protocol

In this section, we propose our controlled secure aggregation protocol (CSA protocol hereafter). Let n be the security parameter. Then we choose other parameters which are used in our CSA protocol as follows:

(i) $c = c(n) > 0$,

(ii) p is a positive integer by setting a prime number $q = \omega(p^2 n^{3c+1} \log^5 n)$,

(iii) $m = \lfloor 8n \log q \rfloor$, and

(iv) $\beta = 1/(27n^{1+(3c/2)} \log n \log q \sqrt{qm})$ is a Gaussian parameter.

Suppose that there are n users, $\mathcal{U}_i$ $(1 \le i \le n)$, a receiver $\mathcal{U}_{\mathscr{R}}$, and an aggregator $\mathscr{AGG}$. Each user $\mathcal{U}_i$ $(1 \le i \le n)$ outsources its own numerical data $\mathbf{b}_i$ with encrypted form. We assume that the receiver $\mathcal{U}_{\mathscr{R}}$ wants to know an aggregated value $\mathbf{b} = \sum_{j=1}^{j=|\mathscr{I}|} \mathbf{b}_{i_j}$, where $\mathscr{I} = \{i_1, \ldots, i_{|\mathscr{I}|}\} \subseteq \{1, \ldots, n\}$ and $|\mathscr{I}|$ is the number of elements in $\mathscr{I}$. We also assume that the receiver $\mathcal{U}_{\mathscr{R}}$ has a public key $pk_{\mathscr{R}} = \mathbf{A}_{\mathscr{R}}$ and a secret key $sk_{\mathscr{R}} = \mathbf{T}_R$ by performing $\mathsf{GHV}^*.\mathsf{Key}(1^n, 1^m, q)$. Then the receiver $\mathcal{U}_{\mathscr{R}}$ can get $\mathbf{b}$ by performing our CSA protocol.

Our CSA protocol consists of the following phases which are illustrated in Box 1: *Key Generation*, *Encryption*, *Aggregation*, *re-Aggregation*, and *dec-Aggregation*. In the *Key Generation* phase, each user generates a public key pair and a secret key. In the *Encryption* phase, each user encrypts its numerical data with his/her public key pair. In the *Aggregation* phase, ciphertexts generated under *distinct* public key pairs are aggregated. That is, to get an aggregated value, the receiver $\mathcal{U}_{\mathscr{R}}$ allows the aggregator $\mathscr{AGG}$ to know $\mathscr{I} = \{i_1, \ldots, i_{|\mathscr{I}|}\}$. Then an aggregator $\mathscr{AGG}$ aggregates each ciphertext on $\mathbf{b}_{i_j}$ $(1 \le j \le |\mathscr{I}|)$ in this phase. In the *re-Aggregation* phase, the user $\mathcal{U}_i$ eliminates $\mathbf{A}_i \mathbf{s}_i$ from a ciphertext $\mathbf{c}'$ and adds $\mathsf{GHV}^*.\mathsf{Enc}(\mathbf{A}_{\mathscr{R}}, \mathbf{0}) = \mathbf{A}_{\mathscr{R}} \mathbf{s}_i'' + p\mathbf{x}_i'' (\mathrm{mod} q)$ which is a ciphertext on $\mathbf{0}$ under the receiver's public key $\mathbf{A}_{\mathscr{R}}$. In this phase, a ciphertext under the public key $\mathbf{A}_i$ is converted into a ciphertext under the public key $\mathbf{A}_{\mathscr{R}}$ maintaining the same

Key Generation. Each user $\mathcal{U}_i$ $(1 \le i \le \mathsf{n})$ runs the following $\mathsf{CSA.Key}(1^n, 1^m, q)$ algorithm to get a public key pair $pk_i = (\mathbf{A}_i, \mathbf{M}_i)$ and a secret key $sk_i = \mathbf{T}_i$.

$\mathsf{CSA.Key}(1^n, 1^m, q)$: Given n, m, and q, output a public key pair $pk_i = (\mathbf{A}_i, \mathbf{M}_i)$ and a secret key $sk_i = \mathbf{T}_i$ by performing following steps:
(1) perform $\mathsf{GHV}^*.\mathsf{Key}(1^n, 1^m, q)$ to get $\mathbf{A}_i$ and $\mathbf{T}_i$,
(2) choose a uniformly random matrix $\mathbf{M}_i \in \mathbb{Z}_q^{n \times m}$.

Encryption. Each user $\mathcal{U}_i$ $(1 \le i \le \mathsf{n})$ runs the following $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$ algorithm to get a ciphertext pair $(\mathbf{c}_i, \mathbf{c}_i')$.

$\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$: Given a public key pair $pk_i = (\mathbf{A}_i, \mathbf{M}_i)$ and a plaintext $\mathbf{b}_i \in \mathbb{Z}_p^m$, output a ciphertext pair $(\mathbf{c}_i, \mathbf{c}_i')$ by performing following steps:
(1) choose $\mathbf{r}_i \in \{0,1\}^m$ at random,
(2) compute $\mathbf{s}_i = \mathsf{h}_{\mathbf{M}_i}(\mathbf{r}_i) \in \mathbb{Z}_q^n$,
(3) choose a uniformly random vector $\mathbf{s}_i' \in \mathbb{Z}_q^n$,
(4) choose Gaussian error vectors $\mathbf{x}_i \in \mathbb{Z}_q^m$ and $\mathbf{x}_i' \in \mathbb{Z}_q^m$,
(5) compute $\mathbf{c}_i = \mathbf{A}_i\mathbf{s}_i + p\mathbf{x}_i + \mathbf{b}_i \pmod q$,
(6) compute $\mathbf{c}_i' = \mathbf{A}_i\mathbf{s}_i' + p\mathbf{x}_i' + \mathbf{r}_i \pmod q$.

Aggregation. $\mathscr{AGG}$ aggregates ciphertext pairs $\{(\mathbf{c}_i, \mathbf{c}_i')\}_{i \in \mathcal{I}}$ generated under distinct public key pairs $\{pk_i\}_{i \in \mathcal{I}}$ by performing the following $\mathsf{CSA.Agg}(\{(\mathbf{c}_i, \mathbf{c}_i')\}_{i \in \mathcal{I}}, \mathcal{I})$.

$\mathsf{CSA.Agg}(\{(\mathbf{c}_i, \mathbf{c}_i')\}_{i \in \mathcal{I}}, \mathcal{I})$: Given ciphertext pairs $(\mathbf{c}_i, \mathbf{c}_i')$ where $i \in \mathcal{I}$, output $\mathbf{c}$, $\{\mathbf{c}_i'\}_{i \in \mathcal{I}}$ and $\mathcal{I}$ by performing following steps:
(1) Let $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$ where $|\mathcal{I}|$ is the number of elements in $\mathcal{I}$,
(2) $\mathbf{c} = \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_{|\mathcal{I}|}}$.

re-Aggregation. Each user $\mathcal{U}_i$ $(1 \le i \le \mathsf{n})$ can run the following $\mathsf{CSA.reAgg}(\mathbf{c}', \mathbf{c}_i', pk_i, sk_i, \mathbf{A}_{\mathscr{R}})$ algorithm to get a re-aggregated ciphertext.

$\mathsf{CSA.reAgg}(\mathbf{c}', \mathbf{c}_i', pk_i, sk_i, \mathbf{A}_{\mathscr{R}})$: Given an aggregated ciphertext $\mathbf{c}'$, a ciphertext $\mathbf{c}_i'$, a public key pair $pk_i = (\mathbf{A}_i, \mathbf{M}_i)$, a secret key $sk_i = \mathbf{T}_i$, and a public key $\mathbf{A}_{\mathscr{R}}$ of $\mathcal{U}_{\mathscr{R}}$, output $\bar{\mathbf{c}}$ by performing following steps:
(1) perform $\mathsf{GHV}^*.\mathsf{Dec}(sk_i, \mathbf{c}_i')$ to get $\mathbf{r}_i$,
(2) compute $\mathbf{s}_i = \mathsf{h}_{\mathbf{M}_i}(\mathbf{r}_i)$,
(3) choose a uniformly random vector $\mathbf{s}_i'' \in \mathbb{Z}_q^n$,
(4) choose a Gaussian error vector $\mathbf{x}_i'' \in \mathbb{Z}_q^m$,
(5) compute $\bar{\mathbf{c}} = \mathbf{c}' - \mathbf{A}_i\mathbf{s}_i + \mathbf{A}_{\mathscr{R}}\mathbf{s}_i'' + p\mathbf{x}_i'' \pmod q$.

dec-Aggregation. $\mathscr{AGG}$ gives an aggregated ciphertext $\mathbf{c} = \mathbf{c}_{i_1} + \cdots + \mathbf{c}_{i_{|\mathcal{I}|}}$ to $\mathcal{U}_{i_1}$, and a ciphertext $\mathbf{c}_{i_j}'$ and a public key $\mathbf{A}_{\mathscr{R}}$ of $\mathcal{U}_{\mathscr{R}}$ to each user $\mathcal{U}_{i_j}(1 \le j \le |\mathcal{I}|)$, respectively. Let $\bar{\mathbf{c}}_0 = \mathbf{c}$, then the receiver $\mathcal{U}_{\mathscr{R}}$ obtains $\mathbf{b} = \mathbf{b}_{i_1} + \cdots + \mathbf{b}_{i_{|\mathcal{I}|}}$ by performing following steps:
(1) Each user $\mathcal{U}_{i_j}(1 \le j \le |\mathcal{I}| - 1)$ in turn,
  (a) computes $\bar{c}_j = \mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{j-1}, \mathbf{c}_{i_j}', pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}})$,
  (b) sends $\bar{\mathbf{c}}_j$ to the next user $\mathcal{U}_{i_{j+1}}$.
(2) $\mathcal{U}_{i_{|\mathcal{I}|}}$ computes $\bar{\mathbf{c}}_{|\mathcal{I}|} = \mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{|\mathcal{I}|-1}, \mathbf{c}_{i_{|\mathcal{I}|}}', pk_{i_{|\mathcal{I}|}}, sk_{i_{|\mathcal{I}|}}, \mathbf{A}_{\mathscr{R}})$ and sends $\bar{\mathbf{c}}_{|\mathcal{I}|}$ to $\mathcal{U}_{\mathscr{R}}$.
(3) $\mathcal{U}_{\mathscr{R}}$ performs $\mathsf{GHV}^*.\mathsf{Dec}(sk_{\mathscr{R}}, \bar{\mathbf{c}}_{|\mathcal{I}|})$ to get $\mathbf{b} = \mathbf{b}_{i_1} + \cdots + \mathbf{b}_{i_{|\mathcal{I}|}}$.

Box 1: CSA Protocol.

plaintext. For example, suppose that $\mathbf{c}' = \mathbf{c}_i = \mathbf{A}_i\mathbf{s}_i + p\mathbf{x}_i + \mathbf{b}_i (\mathrm{mod} q)$ and $\mathbf{c}_i' = \mathbf{A}_i\mathbf{s}_i' + p\mathbf{x}_i' + \mathbf{r}_i (\mathrm{mod} q)$, then

$$\begin{aligned}
&\mathsf{CSA.reAgg}\left(\mathbf{c}', \mathbf{c}_i', pk_i, sk_i, \mathbf{A}_{\mathscr{R}}\right) \\
&= \mathbf{c}' - \mathbf{A}_i\mathbf{s}_i + \mathbf{A}_{\mathscr{R}}\mathbf{s}_i'' + p\mathbf{x}_i'' \left(\mathrm{mod} q\right) \\
&= \mathbf{A}_i\mathbf{s}_i + p\mathbf{x}_i + \mathbf{b}_i - \mathbf{A}_i\mathbf{s}_i + \mathbf{A}_{\mathscr{R}}\mathbf{s}_i'' + p\mathbf{x}_i'' \left(\mathrm{mod} q\right) \\
&= \mathbf{A}_{\mathscr{R}}\mathbf{s}_i'' + p\left(\mathbf{x}_i + \mathbf{x}_i''\right) + \mathbf{b}_i \left(\mathrm{mod} q\right) \\
&= \mathsf{GHV}^*.\mathsf{Enc}\left(\mathbf{A}_{\mathscr{R}}, \mathbf{b}_i\right).
\end{aligned} \tag{5}$$

As a result, a ciphertext which is decryptable by a user $\mathcal{U}_i$ is converted into a ciphertext which is decryptable by the receiver $\mathcal{U}_{\mathscr{R}}$ maintaining the same plaintext $\mathbf{b}_i$. This phase is needed in the *dec-Aggregation* phase to make an aggregated ciphertext $\mathbf{c}'$ decryptable by the receiver $\mathcal{U}_{\mathscr{R}}$. In the *dec-Aggregation* phase, each user $\mathcal{U}_{i_j}$ $(1 \le j \le |\mathcal{I}|)$ in turn makes an aggregated ciphertext $\mathbf{c}$ decryptable by the receiver $\mathcal{U}_{\mathscr{R}}$. Through these phases, the receiver $\mathcal{U}_{\mathscr{R}}$ can get an aggregated value $\mathbf{b} = \sum_{j=1}^{j=|\mathcal{I}|} \mathbf{b}_{i_j}$.

For example, we assume that $\mathsf{n} = 5$ users participating in our controlled secure aggregation protocol CSA and each

user $\mathcal{U}_i$ $(1 \le i \le 5)$ has its numerical data $\mathbf{b}_i$. Each user $\mathcal{U}_i$ $(1 \le i \le 5)$ outsources its numerical data with encrypted form $(\mathbf{c}_i, \mathbf{c}_i') = (\mathbf{A}_i\mathbf{s}_i + p\mathbf{x}_i + \mathbf{b}_i (\bmod q), \mathbf{A}_i\mathbf{s}_i' + p\mathbf{x}_i' + \mathbf{r}_i (\bmod q))$ using $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$ algorithm. Suppose that the receiver $\mathcal{U}_{\mathscr{R}}$ wants to know an aggregated value $\mathbf{b} = \mathbf{b}_2 + \mathbf{b}_5$. The receiver $\mathcal{U}_{\mathscr{R}}$ lets the aggregator $\mathscr{AGG}$ know $\mathcal{I} = \{2, 5\}$. Then $\mathscr{AGG}$ runs the $\mathsf{CSA.Agg}((\mathbf{c}_2, \mathbf{c}_2'), (\mathbf{c}_5, \mathbf{c}_5'), \{2, 5\})$ algorithm to get $\mathbf{c} = \mathbf{c}_2 + \mathbf{c}_5 = \mathbf{A}_2\mathbf{s}_2 + p\mathbf{x}_2 + \mathbf{b}_2 + \mathbf{A}_5\mathbf{s}_5 + p\mathbf{x}_5 + \mathbf{b}_5 (\bmod q)$. $\mathscr{AGG}$ gives $\mathbf{c}$, $\mathbf{c}_2'$, and $\mathbf{A}_{\mathscr{R}}$ to $\mathcal{U}_2$, and $\mathbf{c}_5'$ and $\mathbf{A}_{\mathscr{R}}$ to $\mathcal{U}_5$. Then $\mathcal{U}_2$ runs $\mathsf{GHV^*.reAgg}(\bar{\mathbf{c}}_0 = \mathbf{c}, \mathbf{c}_2', pk_2, sk_2, \mathbf{A}_{\mathscr{R}})$ to get

$$\begin{aligned}\bar{\mathbf{c}}_1 &= \bar{\mathbf{c}}_0 - \mathbf{A}_2\mathbf{s}_2 + \mathbf{A}_{\mathscr{R}}\mathbf{s}_2'' + p\mathbf{x}_2'' \,(\bmod q)\\ &= \mathbf{A}_2\mathbf{s}_2 + p\mathbf{x}_2 + \mathbf{b}_2 + \mathbf{A}_5\mathbf{s}_5 + p\mathbf{x}_5 + \mathbf{b}_5\\ &\quad - \mathbf{A}_2\mathbf{s}_2 + \mathbf{A}_{\mathscr{R}}\mathbf{s}_2'' + p\mathbf{x}_2'' \,(\bmod q)\\ &= p\left(\mathbf{x}_2 + \mathbf{x}_2''\right) + \mathbf{b}_2 + \mathbf{A}_5\mathbf{s}_5 + p\mathbf{x}_5 + \mathbf{b}_5\\ &\quad + \mathbf{A}_{\mathscr{R}}\mathbf{s}_2'' \,(\bmod q).\end{aligned} \tag{6}$$

$\mathcal{U}_2$ sends $\bar{\mathbf{c}}_1$ to $\mathcal{U}_5$; then $\mathcal{U}_5$ runs $\mathsf{GHV^*.reAgg}(\bar{\mathbf{c}}_1, \mathbf{c}_5', pk_5, sk_5, \mathbf{A}_{\mathscr{R}})$ to get

$$\begin{aligned}\bar{\mathbf{c}}_2 &= \bar{\mathbf{c}}_1 - \mathbf{A}_5\mathbf{s}_5 + \mathbf{A}_{\mathscr{R}}\mathbf{s}_5'' + p\mathbf{x}_5'' \,(\bmod q)\\ &= p\left(\mathbf{x}_2 + \mathbf{x}_2''\right) + \mathbf{b}_2 + \mathbf{A}_5\mathbf{s}_5 + p\mathbf{x}_5 + \mathbf{b}_5\\ &\quad + \mathbf{A}_{\mathscr{R}}\mathbf{s}_2'' - \mathbf{A}_5\mathbf{s}_5 + \mathbf{A}_{\mathscr{R}}\mathbf{s}_5'' + p\mathbf{x}_5'' \,(\bmod q)\\ &= \mathbf{A}_{\mathscr{R}}\left(\mathbf{s}_2'' + \mathbf{s}_5''\right) + p\left(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5''\right)\\ &\quad + \mathbf{b}_2 + \mathbf{b}_5 \,(\bmod q).\end{aligned} \tag{7}$$

$\mathcal{U}_5$ sends $\bar{\mathbf{c}}_2$ to $\mathcal{U}_{\mathscr{R}}$; then $\mathcal{U}_{\mathscr{R}}$ runs $\mathsf{GHV^*.Dec}(sk_{\mathscr{R}}, \bar{\mathbf{c}}_2)$ to get $\mathbf{b} = \mathbf{b}_2 + \mathbf{b}_5$. That is,

$$\begin{aligned}\mathbf{e} &= \mathbf{T}_{\mathscr{R}}\bar{\mathbf{c}}_2 \,(\bmod q)\\ &= \mathbf{T}_{\mathscr{R}}\mathbf{A}_{\mathscr{R}}\left(\mathbf{s}_2'' + \mathbf{s}_5''\right) + \mathbf{T}_{\mathscr{R}}p\left(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5''\right)\\ &\quad + \mathbf{T}_{\mathscr{R}}\mathbf{b}_2 + \mathbf{T}_{\mathscr{R}}\mathbf{b}_5 \,(\bmod q)\\ &= \mathbf{T}_{\mathscr{R}}p\left(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5''\right) + \mathbf{T}_{\mathscr{R}}\mathbf{b}_2\\ &\quad + \mathbf{T}_{\mathscr{R}}\mathbf{b}_5 \,(\bmod q)\\ &= \mathbf{T}_{\mathscr{R}}p\left(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5''\right) + \mathbf{T}_{\mathscr{R}}\mathbf{b}_2\\ &\quad + \mathbf{T}_{\mathscr{R}}\mathbf{b}_5,\\ \mathbf{b} &= \mathbf{T}_{\mathscr{R}}^{-1}\mathbf{e}\\ &= \mathbf{T}_{\mathscr{R}}^{-1}\mathbf{T}_{\mathscr{R}}p\left(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5''\right) + \mathbf{T}_{\mathscr{R}}^{-1}\mathbf{T}_{\mathscr{R}}\mathbf{b}_2\\ &\quad + \mathbf{T}_{\mathscr{R}}^{-1}\mathbf{T}_{\mathscr{R}}\mathbf{b}_5\\ &= p\left(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5''\right) + \mathbf{b}_2 + \mathbf{b}_5\\ &= \mathbf{b}_2 + \mathbf{b}_5 \,(\bmod p).\end{aligned} \tag{8}$$

$\mathbf{T}_{\mathscr{R}}p(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5'') + \mathbf{T}_{\mathscr{R}}\mathbf{b}_2 + \mathbf{T}_{\mathscr{R}}\mathbf{b}_5 (\bmod q)$ is the same as $\mathbf{T}_{\mathscr{R}}p(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5'') + \mathbf{T}_{\mathscr{R}}\mathbf{b}_2 + \mathbf{T}_{\mathscr{R}}\mathbf{b}_5$, since $\mathbf{T}_{\mathscr{R}}p(\mathbf{x}_2 + \mathbf{x}_2'' + \mathbf{x}_5 + \mathbf{x}_5'') + \mathbf{T}_{\mathscr{R}}\mathbf{b}_2 + \mathbf{T}_{\mathscr{R}}\mathbf{b}_5$ is a sufficiently short value [14].

In the *dec-Aggregation* phase, any user can refuse to perform the $\mathsf{CSA.reAgg}$ algorithm, unless initial consents that have been obtained from patients allow the secondary research. Then the receiver cannot get the result. The receiver can get the result only if all users perform the $\mathsf{CSA.reAgg}$ algorithm. That means the receiver can get an aggregated value that he/she is seeking only by the unanimous consent of all fusers who have the data aggregated. That is the reason why we use the term "controlled" in the CSA protocol.

3.1. Security. We now analyze the security of our controlled secure aggregation protocol.

First, we show that our encryption $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$ is IND-CPA secure. Intuitively, the only difference between our encryption scheme $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$ and the $\mathsf{GHV^*}$ homomorphic encryption scheme is how to generate a vector $\mathbf{s}_i \in \mathbb{Z}_q^n$. In the $\mathsf{GHV^*}$ homomorphic encryption scheme; the vector $\mathbf{s}_i$ is chosen uniformly, but in our encryption scheme $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$, it is generated by computing $\mathbf{s}_i = \mathsf{h}_{\mathbf{M}_i}(\mathbf{r}_i)$ using a randomly chosen vector $\mathbf{r}_i \in \{0,1\}^m$. Since every output of the Ajtai's one-way function $\mathsf{h}_{\mathbf{M}_i} : \{0,1\}^m \to \mathbb{Z}_q^n$ is uniformly distributed over $\mathbb{Z}_q^n$, a vector $\mathbf{s}_i = \mathsf{h}_{\mathbf{M}_i}(\mathbf{r}_i)$ from our encryption scheme is uniformly distributed over $\mathbb{Z}_q^n$. Therefore, the security of our encryption scheme $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$ is the same as the $\mathsf{GHV^*}$ homomorphic encryption scheme.

Theorem 1. *Our encryption scheme* $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$ *provides IND-CPA if the* $\mathsf{GHV^*}$ *homomorphic encryption scheme provides IND-CPA and every output of the Ajtai's one-way function* $\mathsf{h}_{\mathbf{M}_i} : \{0,1\}^m \to \mathbb{Z}_q^n$ *is uniformly distributed over* $\mathbb{Z}_q^n$.

Proof of Theorem 1. Formally, we show that if there exists an adversary $\mathscr{A}$ breaking the IND-CPA security of our encryption scheme $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_i)$, there exists a challenger $\mathscr{C}$ breaking the IND-CPA security of the $\mathsf{GHV^*}$ homomorphic encryption scheme.

Let $\{pk = \mathbf{A}, \mathsf{GHV^*.params}\}$ be an instance given to $\mathscr{C}$. $\mathscr{C}$ chooses a uniformly random matrix $\mathbf{M}_i \in \mathbb{Z}_q^{n\times m}$ and sends $\{pk_i = \mathbf{A}_i = \mathbf{A}, \mathsf{params} = (\mathsf{GHV^*.params}, \mathbf{M}_i)\}$ to $\mathscr{A}$.$\mathscr{A}$ chooses $\{\mathbf{b}_0, \mathbf{b}_1\}$ and sends $\{\mathbf{b}_0, \mathbf{b}_1\}$ to $\mathscr{C}$. $\mathscr{C}$ outputs $\{\mathbf{b}_0, \mathbf{b}_1\}$ and returns $\mathbf{c}_j$, where $j \in \{0,1\}$. $\mathscr{C}$ sends $\mathbf{c}_j$ to $\mathscr{A}$, and $\mathscr{A}$ outputs $d \in \{0,1\}$. Then $\mathscr{C}$ outputs $d \in \{0,1\}$. □

In our controlled secure aggregation protocol CSA, ciphertexts generated under distinct public key pairs can be aggregated. To decrypt the aggregated ciphertext $\mathbf{c}$ which is generated by ciphertexts of users $\mathcal{U}_{i_j}$ $(1 \le j \le |\mathcal{I}|)$, each user $\mathcal{U}_{i_j}$ $(1 \le j \le |\mathcal{I}|)$ needs to eliminate $\mathbf{A}_{i_j}\mathbf{s}_{i_j}$ from $\mathbf{c}$ and add $\mathsf{GHV^*.Enc}(\mathbf{A}_{\mathscr{R}}, \mathbf{0}) = \mathbf{A}_{\mathscr{R}}\mathbf{s}_i'' + p\mathbf{x}_i''(\bmod q)$ using the $\mathsf{CSA.reAgg}$ algorithm. Therefore, we should show that $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{j-1}, \mathbf{c}_{ij}', pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}})$ is secure.

Theorem 2. $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{j-1}, \mathbf{c}_{i_j}', pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}})$ *is an aggregation of secure ciphertexts if* $\bar{\mathbf{c}}_{j-1}$ *is an aggregation of secure*

ciphertexts including the ciphertext $\mathbf{c}_{i_j}$ *which is one of a pair of the ciphertexts* $(\mathbf{c}_{i_j}, \mathbf{c}'_{i_j})$.

Proof of Theorem 2. Let $\bar{\mathbf{c}}_j = \mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{j-1}, \mathbf{c}'_{i_j}, pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}})$ and $\bar{\mathbf{c}}_{j-1} = \mathbf{c}' + \mathbf{c}_{ij} = \mathbf{c}' + \mathbf{A}_{i_j}\mathbf{s}_{i_j} + p\mathbf{x}_{i_j} + \mathbf{b}_{i_j}$, then

$$\begin{aligned}
\bar{\mathbf{c}}_j &= \mathsf{CSA.reAgg}\left(\bar{\mathbf{c}}_{j-1}, \mathbf{c}'_{i_j}, pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}}\right) \\
&= \bar{\mathbf{c}}_{j-1} - \mathbf{A}_{i_j}\mathbf{s}_{i_j} + \mathbf{A}_{\mathscr{R}}\mathbf{s}''_{i_j} + p\mathbf{x}''_{i_j} \left(\mathrm{mod} q\right) \\
&= \mathbf{c}' + \mathbf{A}_{i_j}\mathbf{s}_{i_j} + p\mathbf{x}_{i_j} + \mathbf{b}_{i_j} - \mathbf{A}_{i_j}\mathbf{s}_{i_j} \\
&\quad + \mathbf{A}_{\mathscr{R}}\mathbf{s}''_{i_j} + p\mathbf{x}''_{i_j} \left(\mathrm{mod} q\right) \\
&= \mathbf{c}' + \mathbf{A}_{\mathscr{R}}\mathbf{s}''_{i_j} + p\left(\mathbf{x}_{i_j} + \mathbf{x}''_{i_j}\right) + \mathbf{b}_{i_j} \left(\mathrm{mod} q\right) \\
&= \mathbf{c}' + \mathsf{GHV}^*.\mathsf{Enc}\left(\mathbf{A}_{\mathscr{R}}, \mathbf{b}_{i_j}\right) \left(\mathrm{mod} q\right).
\end{aligned} \tag{9}$$

Therefore, $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{j-1}, \mathbf{c}'_{i_j}, pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}})$ is the aggregation of the secure ciphertexts. □

In the fifth step of the $\mathsf{CSA.reAgg}$ algorithm, $\mathsf{GHV}^*.\mathsf{Enc}(\mathbf{A}_{\mathscr{R}}, \mathbf{0})$ is added to be secure against an adversary $\mathscr{A}$ who can eavesdrop on our controlled secure aggregation protocol CSA. Assume that $\bar{\mathbf{c}}_j = \bar{\mathbf{c}}_{j-1} - \mathbf{A}_{i_j}\mathbf{s}_{i_j} (\mathrm{mod} q)$ in the fifth step of $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_{j-1}, \mathbf{c}'_{i_j}, pk_{i_j}, sk_{i_j}, \mathbf{A}_{\mathscr{R}})$, then any adversary $\mathscr{A}$ who can eavesdrop on our controlled secure aggregation protocol CSA is able to get $\bar{\mathbf{c}}_j = \bar{\mathbf{c}}_{j-1} - \mathbf{A}_{i_j}\mathbf{s}_{i_j} (\mathrm{mod} q)$, $\bar{\mathbf{c}}_{j-1}$, and $\mathbf{c}_{i_j} = \mathbf{A}_{i_j}\mathbf{s}_{i_j} + p\mathbf{x}_{i_j} + \mathbf{b}_{i_j} (\mathrm{mod} q)$. Then, $\mathscr{A}$ can compute the following:

$$\begin{aligned}
\mathbf{c}_{i_j} - \bar{\mathbf{c}}_{j-1} + \bar{\mathbf{c}}_j &= \mathbf{A}_{i_j}\mathbf{s}_{i_j} + p\mathbf{x}_{i_j} + \mathbf{b}_{i_j} \\
&\quad - \mathbf{A}_{i_j}\mathbf{s}_{i_j} \left(\mathrm{mod} q\right) \\
&= p\mathbf{x}_{i_j} + \mathbf{b}_{i_j} \left(\mathrm{mod} q\right) \\
&= p\mathbf{x}_{i_j} + \mathbf{b}_{i_j} \\
&= \mathbf{b}_{i_j} \left(\mathrm{mod} p\right).
\end{aligned} \tag{10}$$

Since $p\mathbf{x}_{i_j} + \mathbf{b}_{i_j}$ is a sufficiently short value, $p\mathbf{x}_{i_j} + \mathbf{b}_{i_j} (\mathrm{mod} q)$ is the same as $p\mathbf{x}_{i_j} + \mathbf{b}_{i_j}$ [14]. Therefore, $\mathscr{A}$ can decrypt $\mathbf{c}_{i_j}$ without the secret key sk_{i_j}.

In the *dec-Aggregation* phase, after all the users $\mathscr{U}_{i_j}$ $(1 \le j \le |\mathscr{I}|)$ eliminate $\mathbf{A}_{i_j}\mathbf{s}_{i_j}$ from $\mathbf{c}$, the result is the same form as a ciphertext generated under the public key $\mathbf{A}_{\mathscr{R}}$. Therefore, the receiver $\mathscr{U}_{\mathscr{R}}$ can decrypt it.

4. Secure Clinical Data Aggregation System

In this section, we provide an overview of our system and how it works.

4.1. System Overview. The proposed system environment consists of hospitals, an aggregator, and researchers. In our system, each hospital outsources its clinical data to cloud storage servers. Hospitals use the following hybrid method to store data when outsourcing their clinical research data in cloud servers: they make anonymous data publicly available in the cloud servers using anonymity techniques for deidentification in Section 2.1. In addition, hospitals also store their encrypted numerical data together with the anonymous data for statistical accuracy.

Table 4: Original clinical data.

	Nonsensitive		Sensitive
	Zip code	Age	Condition
$\mathscr{H}_1$	13062	25	Heart disease
	13035	31	Cancer
	14850	45	Viral infection
$\mathscr{H}_2$	13035	20	Heart disease
	14850	23	Heart disease
	14855	52	Viral infection
$\mathscr{H}_3$	13062	35	Cancer
	13035	22	Cancer
	13062	52	Heart disease
$\mathscr{H}_4$	13062	27	Heart disease
	13062	33	Viral infection
	14855	43	Cancer

Suppose that there are 4 hospitals, $\mathscr{H}_1, \mathscr{H}_2, \mathscr{H}_3$, and $\mathscr{H}_4$ that want to share their clinical data and have public and sescret key pairs $(pk_1, sk_1), (pk_2, sk_2), (pk_3, sk_3)$, and (pk_4, sk_4) of our CSA protocol, respectively. Suppose that there is an aggregator $\mathscr{AGG}$ and a researcher $\mathscr{R}$ who has a public and secret key pair $(pk_{\mathscr{R}}, sk_{\mathscr{R}}) = (\mathbf{A}_{\mathscr{R}}, \mathbf{T}_{\mathscr{R}})$ of the GHV^* homomorphic encryption scheme. The original clinical data of hospitals are shown in Table 4. Each hospital outsources its clinical data to cloud storage servers. That is, $\mathscr{H}_i$ stores deidentified nonsensitive data (such as zip code and age), sensitive data in the raw, and numerical data (such as age) using $\mathsf{CSA.Enc}(pk_i, \mathsf{Age})$ on cloud servers. Both anonymous and encrypted clinical data on cloud servers are shown in Table 5, where $(\mathbf{c}_{1,1}, \mathbf{c}'_{1,1})$ is an output of $\mathsf{CSA.Enc}(pk_1, 25)$, $(\mathbf{c}_{1,2}, \mathbf{c}'_{1,2})$ is an output of $\mathsf{CSA.Enc}(pk_1, 31)$, and so on.

When the researcher $\mathscr{R}$ wants to know the rough estimate of the age of the hospitals' cancer patients, $\mathscr{R}$ can directly get the estimate data from the cloud servers. When $\mathscr{R}$ wants to figure out the average age of the hospitals' cancer patients, $\mathscr{R}$ can ask the aggregator $\mathscr{AGG}$ for an aggregated age. $\mathscr{AGG}$ sums up the ages of cancer patients in each hospital, then totals the ages across hospitals. That is, $\mathscr{AGG}$ performs homomorphic additions to ciphertexts under the same public key, such as $\mathbf{c}_1 = \mathbf{c}_{1,2}, \mathbf{c}'_1 = \mathbf{c}'_{1,2}, \mathbf{c}_3 = \mathsf{GHV}^*.\mathsf{Add}(\mathbf{c}_{3,1}, \mathbf{c}_{3,2}), \mathbf{c}'_3 = \mathsf{GHV}^*.\mathsf{Add}(\mathbf{c}'_{3,1}, \mathbf{c}'_{3,2}), \mathbf{c}_4 = \mathbf{c}_{4,3}$, and $\mathbf{c}'_4 = \mathbf{c}'_{4,3}$. After performing homomorphic additions, $\mathscr{AGG}$ runs $\mathsf{CSA.Agg}((\mathbf{c}_1, \mathbf{c}'_1), (\mathbf{c}_3, \mathbf{c}'_3), (\mathbf{c}_4, \mathbf{c}'_4), \{1, 3, 4\})$ to get an aggregated ciphertext $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_3 + \mathbf{c}_4$. In order to allow $\mathscr{R}$ to know the aggregated age, each hospital in turn gives its consent. $\mathscr{H}_1, \mathscr{H}_3$, and $\mathscr{H}_4$ in turn perform $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_0 =$

Table 5: Anonymous and encrypted clinical data stored on cloud servers.

	Nonsensitive		Sensitive	Ciphertext pair
	Zip code	Age	Condition	CSA.Enc (pk_1, Age)
$\mathcal{H}_1$	130∗∗	<30	Heart disease	$(\mathbf{c}_{1,1}, \mathbf{c}'_{1,1})$ // 25
	130∗∗	3∗	Cancer	$(\mathbf{c}_{1,2}, \mathbf{c}'_{1,2})$ // 31
	1485∗	≥40	Viral infection	$(\mathbf{c}_{1,3}, \mathbf{c}'_{1,3})$ // 45
	Nonsensitive		Sensitive	Ciphertext pair
	Zip code	Age	Condition	CSA.Enc (pk_2, Age)
$\mathcal{H}_2$	130∗∗	<30	Heart disease	$(\mathbf{c}_{2,1}, \mathbf{c}'_{2,1})$ // 20
	1485∗	<30	Heart disease	$(\mathbf{c}_{2,2}, \mathbf{c}'_{2,2})$ // 23
	1485∗	≥40	Viral infection	$(\mathbf{c}_{2,3}, \mathbf{c}'_{2,3})$ // 52
	Nonsensitive		Sensitive	Ciphertext pair
	Zip code	Age	Condition	CSA.Enc (pk_3, Age)
$\mathcal{H}_3$	130∗∗	3∗	Cancer	$(\mathbf{c}_{3,1}, \mathbf{c}'_{3,1})$ // 35
	130∗∗	<30	Cancer	$(\mathbf{c}_{3,2}, \mathbf{c}'_{3,2})$ // 22
	130∗∗	≥40	Heart disease	$(\mathbf{c}_{3,3}, \mathbf{c}'_{3,3})$ // 52
	Nonsensitive		Sensitive	Ciphertext pair
	Zip code	Age	Condition	CSA.Enc (pk_4, Age)
$\mathcal{H}_4$	130∗∗	<30	Heart disease	$(\mathbf{c}_{4,1}, \mathbf{c}'_{4,1})$ // 27
	130∗∗	3∗	Viral infection	$(\mathbf{c}_{4,2}, \mathbf{c}'_{4,2})$ // 33
	1485∗	≥40	Cancer	$(\mathbf{c}_{4,3}, \mathbf{c}'_{4,3})$ // 43

$\mathbf{c}, \mathbf{c}'_1, pk_1, sk_1, pk_{\mathscr{R}})$, $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_1, \mathbf{c}'_3, pk_3, sk_3, pk_{\mathscr{R}})$ and $\mathsf{CSA.reAgg}(\bar{\mathbf{c}}_2, \mathbf{c}'_4, pk_4, sk_4, pk_{\mathscr{R}})$ to get $\bar{\mathbf{c}}_1$, $\bar{\mathbf{c}}_2$, and $\bar{\mathbf{c}}_3$, respectively. After the agreement procedure, $\mathscr{R}$ can get the aggregated ciphertext $\bar{\mathbf{c}}_3$ under his/her public key. Then, $\mathscr{R}$ can get an average age of the cancer patients, $131/4 = 32.75$, by performing $\mathsf{GHV}^*.\mathsf{Dec}(sk_{\mathscr{R}}, \bar{\mathbf{c}}_3) = \mathbf{b} = 31 + 35 + 22 + 43 = 131$, that is, the sum of the age of the cancer patients.

4.2. Attack Model. For designing a secure clinical data aggregation system, the following conditions should be considered.

(1) (*Anonymity*) Adversaries should not exactly identify only one private individual after looking ciphertexts on cloud storage servers.

(2) (*Confidentiality*) Adversaries should not reveal any information from the encrypted numerical data on cloud storage servers.

(3) (*External security*) The third parties (external adversaries) should not know any information with information flow.

(4) (*Internal security*) Hospitals and researchers (internal adversaries) except the researcher who sends a request should not know any information with information flow.

4.3. Our System. Now, we propose our secure clinical data aggregation system (SCDA system hereafter). Let n be the security parameter. Then we choose other parameters which are used in our SCDA system as follows:

(i) $c = c(n) > 0$,

(ii) p is a positive integer by setting a prime number $q = \omega(p^2 n^{3c+1} \log^5 n)$,

(iii) $m = \lfloor 8n \log q \rfloor$, and

(iv) $\beta = 1/(27n^{1+(3c/2)} \log n \log q \sqrt{qm})$ is a Gaussian parameter.

Suppose that there are $\mathsf{n}_{\mathscr{H}}$ hospitals $\mathscr{H}_i$ $(1 \le i \le \mathsf{n}_{\mathscr{H}})$, $\mathsf{n}_{\mathscr{R}}$ researchers $\mathscr{R}_j$ $(1 \le j \le \mathsf{n}_{\mathscr{R}})$, and an aggregator $\mathscr{AGG}$. We assume that the ith hospital $\mathscr{H}_i$ has $\mathsf{n}_{\mathsf{T}i}$ tuples and the relational database in the cloud servers has n_{C} numerical clinical data attributes.

The building blocks of our SCDA system are our controlled secure aggregation protocol CSA and the GHV^* homomorphic encryption scheme GHV^*. Our SCDA system consists of the following phases which are illustrated in Box 2: *Preparation*, *Data Publication*, *Query*, *Aggregation*, *Consent*, and *Acquisition*. In the *Preparation* phase, each hospital and each researcher generates a public key (pair) and a secret key. In the *Data Publication* phase, each hospital encrypts its numerical clinical data with his/her public key pair and makes anonymous data using anonymity techniques for deidentification. Then each hospital stores them in the cloud servers. In the *Query* phase, one of the researchers asks the aggregator $\mathscr{AGG}$ for an aggregated clinical data. In the *Aggregation* phase, ciphertexts generated under distinct hospitals are aggregated. In the *Consent* phase, each hospital goes through the procedure for consent. In the *Acquisition* phase, the researcher can get the aggregated clinical data.

5. Analysis

In this section, we analyze the security and efficiency in our protocol.

5.1. Secure Parameters. We follow parameters which are defined in our SCDA system from Section 4.2. Let n be the security parameter, then other parameters are as follows:

(i) $c = c(n) > 0$,

(ii) $p \ge n^{3c+1} \log^5 n$,

(iii) $q \approx p^3$ is a prime number,

(iv) $m = \lfloor 8n \log q \rfloor$, and

(v) $\beta = 1/27n^{1+(3c/2)} \log n \log q \sqrt{qm}$ is a Gaussian parameter.

Using the above parameters, a ciphertext pair is only six times as large as a plaintext because $q \approx p^3$ and the lengths of a plaintext and a ciphertext pair are $m \log_2 p$ bits and $2 \cdot m \log_2 q$ bits, respectively. Our SCDA system supports n^c additive operations in common with [14]. In the *Query* phase of our SCDA system, therefore, the number of including tuples in a request for an aggregated data must be less than n^c. Table 6 provides examples of secure parameters.

5.2. Security. We now analyze that our SCDA system is anonymous, confidential, and secure against external and internal adversaries.

Table 6: Examples of secure parameters.

n	m	c	q	p	β	n^c	Plaintext	Ciphertext pair
128	30,865	1	1.39×10^{30}	1.12×10^{10}	2.00×10^{-9}	128	37.86 KB	227.13 KB
128	69,705	3	1.18×10^{68}	4.91×10^{22}	4.33×10^{-16}	2.10×10^{6}	193.07 KB	1.13 MB
256	70,910	1	4.21×10^{34}	3.48×10^{11}	2.70×10^{-10}	256	99.90 KB	599.42 KB
256	159,688	3	9.39×10^{77}	9.79×10^{25}	7.31×10^{-18}	1.68×10^{7}	506.64 KB	2.97 MB
512	159,760	1	1.01×10^{39}	1.00×10^{13}	3.77×10^{-11}	512	253.55 KB	1.49 MB
512	359,509	3	5.90×10^{87}	1.81×10^{29}	1.28×10^{-19}	1.34×10^{8}	1.25 MB	7.52 MB
1,024	354,736	1	2.01×10^{43}	2.72×10^{14}	5.41×10^{-12}	1,024	0.61 MB	3.66 MB
1,024	798,622	3	3.08×10^{97}	3.13×10^{32}	2.29×10^{-21}	1.07×10^{9}	3.09 MB	18.56 MB

Preparation. Each hospital $\mathscr{H}_i$ $(1 \le i \le \mathsf{n}_{\mathscr{H}})$ runs the $\mathsf{CSA.Key}(1^n, 1^m, q)$ algorithm to get a public key pair $pk_i = (\mathbf{A}_i, \mathbf{M}_i)$ and a secret key $sk_i = \mathbf{T}_i$. Each researcher $\mathscr{R}_j (1 \le j \le \mathsf{n}_{\mathscr{R}})$ runs the $\mathsf{GHV}^*.\mathsf{Key}(1^n, 1^m, q)$ algorithm to get a public key $pk_j = \mathbf{A}_j$ and a secret key $sk_j = \mathbf{T}_j$.

Data Publication. For all k $(1 \le \mathsf{k} \le \mathsf{n}_{\mathsf{C}})$ and l $(1 \le l \le \mathsf{n}_{\mathsf{T}_i})$, each hospital $\mathscr{H}_i$ $(1 \le i \le \mathsf{n}_{\mathscr{H}})$ runs the $\mathsf{CSA.Enc}(pk_i, \mathbf{b}_{i,\mathsf{k},l})$ algorithm to get a ciphertext pair $(\mathbf{c}_{i,\mathsf{k},l}, \mathbf{c}'_{i,\mathsf{k},l})$, where $\mathbf{b}_{i,\mathsf{k},l}$ is the lth cell of the kth numeric clinical data attribute of the ith hospital $\mathscr{H}_i$. Then each hospital $\mathscr{H}_i$ $(1 \le i \le \mathsf{n}_{\mathscr{H}})$ makes its data anonymous using anonymity techniques for de-identification. Finally, each hospital $\mathscr{H}_i$ $(1 \le i \le \mathsf{n}_{\mathscr{H}})$ outsources its data in the cloud servers.

Query. The jth researcher $\mathscr{R}_j$ sends a request for an aggregated data to the aggregator $\mathscr{AGG}$. We assume that $\mathscr{R}_j$ is interested in the kth attribute and $|\mathscr{I}|$ hospitals, $\mathscr{H}_i$ $(i \in \mathscr{I})$, have the data in which $\mathscr{R}_j$ is interested. Each hospital $\mathscr{H}_i$ $(i \in \mathscr{I})$ has s_i tuples that meet the request, respectively.

Aggregation. $\mathscr{AGG}$ retrieves all ciphertext pairs satisfying $\mathscr{R}_j$'s request. For each $i \in \mathscr{I}$, $\mathscr{AGG}$ runs the $\mathsf{GHV}^*.\mathsf{Add}(\mathbf{c}_{i,\mathsf{k},l_1}, \ldots, \mathbf{c}_{i,\mathsf{k},l_{s_i}})$ and $\mathsf{GHV}^*.\mathsf{Add}(\mathbf{c}'_{i,\mathsf{k},l_1}, \ldots, \mathbf{c}'_{i,\mathsf{k},l_{s_i}})$ algorithm to get $(\mathbf{c}_i, \mathbf{c}'_i)$. Then $\mathscr{AGG}$ runs the $\mathsf{CSA.Agg}(\{(\mathbf{c}_i, \mathbf{c}'_i)\}_{i \in \mathscr{I}}, \mathscr{I})$ algorithm to get $\mathbf{c}$ and $\{\mathbf{c}'_i\}_{i \in \mathscr{I}}$.

Consent. $\mathscr{AGG}$ determines the order in which hospitals consented to $\mathscr{R}_j$'s request, then sends $\mathbf{c}$ to the first hospital and $(\mathbf{c}'_i, \mathbf{A}_j)$ to each hospital $\mathscr{H}_i$ $(i \in \mathscr{I})$. Each hospital $\mathscr{H}_i$ $(i \in \mathscr{I})$ in turn performs the *dec-Aggregation* phase in our CSA protocol. If any hospital $\mathscr{H}_i$ $(i \in \mathscr{I})$ does not want $\mathscr{R}_j$ to have the aggregated clinical data, it can deny the request by simply not performing the *dec-Aggregation* phase.

Acquisition. After the consent procedure, the last hospital $\mathscr{H}_{|\mathscr{I}|}$ sends $\bar{\mathbf{c}}_{|\mathscr{I}|}$ to $\mathscr{R}_j$. $\mathscr{R}_j$ runs the $\mathsf{GHV}^*.\mathsf{Add}(sk_{\mathscr{R}_j}, \bar{\mathbf{c}}_{|\mathscr{I}|})$ to get $\mathbf{b}$ that is an aggregated clinical data.

Box 2: SCDA Protocol.

Theorem 3 (anonymity). *Our* SCDA *system is anonymous if the anonymity techniques which are used in our* SCDA *system are anonymous.*

Proof of Theorem 3. We use anonymity techniques for deidentification, which guarantee anonymity. In our SCDA system, each hospital outsources its clinical data to cloud storage servers using these techniques for researchers. Therefore, researchers, other hospitals, and the third party cannot identify any individual using ciphertexts on cloud storage servers. □

Besides using the anonymity techniques that are mentioned in Section 2.2, we could use the technique that is used to make statistical database differentially private. In 2006, Dwork introduced the new concept, called "differential privacy," which provides a strong privacy guarantee in statistical databases [31]. To achieve the differential privacy, we could add appropriately chosen random noise in statistical databases.

Theorem 4 (confidentiality). *Our encrypted numerical data are confidential if the* GHV* *homomorphic encryption scheme is IND-CPA secure and every output of the Ajtai's one-way function* $\mathsf{h}_{\mathbf{M}_i} : \{0,1\}^m \to \mathbb{Z}_q^n$ *is uniformly distributed over* $\mathbb{Z}_q^n$.

Proof of Theorem 4. By Theorem 1 in Section 3.1, the encrypted numerical data are confidential. □

Theorem 5 (external and internal security). *Our* SCDA *system is secure against external and internal adversaries if the anonymity techniques for deidentification are anonymous and the* GHV* *homomorphic encryption scheme is secure.*

Proof of Theorem 5. All clinical data outsourced on cloud storage servers are anonymous and confidential since all hospitals use the anonymity techniques for deidentification

Table 7: Complexity analysis of our SCDA system.

	Computation cost	Communication cost
Preparation.	$O(\mathsf{n}_{\mathscr{H}}) \cdot \mathsf{CSA.Key} + O(\mathsf{n}_{\mathscr{R}}) \cdot \mathsf{GHV}^*.\mathsf{Key}$	·
Data Publication.	$O(\mathsf{n}_{\mathsf{C}} \cdot \mathsf{n}_{\mathscr{H}} \cdot \mathsf{n}_{\mathsf{T}}) \cdot \mathsf{CSA.Enc}$	$O(\mathsf{n}_{\mathsf{C}} \cdot \mathsf{n}_{\mathscr{H}} \cdot \mathsf{n}_{\mathsf{T}} \cdot \lvert\mathbf{c}\rvert)$
Query.	·	·
Aggregation.	$O(\mathscr{I} \cdot s) \cdot \mathsf{GHV}^*.\mathsf{Add}$	$O(\mathscr{I} \cdot s \cdot \lvert\mathbf{c}\rvert)$
Consent.	$O(\mathscr{I}) \cdot \mathsf{CSA.reAgg}$	$O(\mathscr{I} \cdot \lvert\mathbf{c}\rvert)$
Acquisition.	$O(1) \cdot \mathsf{GHV}^*.\mathsf{Dec}$	·

Let $\mathsf{n}_{\mathscr{H}}$ be the number of hospitals, $\mathsf{n}_{\mathscr{R}}$ the number of researchers, n_{C} the number of numeric clinical data attributes, n_{T} an average of the number of tuples owned in the ith hospital $\mathscr{H}_i$, $\mathscr{I}$ the number of hospitals such that the jth researcher $\mathscr{R}_j$ is interested, s an average of the number of tuples in the ith hospital $\mathscr{H}_i$ such that the jth researcher $\mathscr{R}_j$ is interested, and $\lvert\mathbf{c}\rvert$ the size of a ciphertext in the GHV^* homomorphic encryption scheme.

Table 8: Experimental results of our SCDA system.

n	m	q	p	*Data Publication.* (sec.)	*Aggregation.* (sec.)	*Consent.* (sec.)	*Acquisition.* (sec.)
128	30,865	1.39×10^{30}	1.12×10^{10}	72.571	0.895	330.070	3.267
128	69,705	1.18×10^{68}	4.91×10^{22}	213.052	1.888	1,244.058	11.260
256	70,910	4.21×10^{34}	3.48×10^{11}	199.752	1.768	1,290.020	11.941

and the GHV^* homomorphic encryption scheme. All transmitted data in our SCDA system are encrypted by the GHV^* homomorphic encryption scheme with fresh random numbers. Therefore, our SCDA system is secure against external and internal adversaries if the anonymity techniques for deidentification are anonymous and the GHV^* homomorphic encryption scheme is secure. □

5.3. Efficiency. Table 7 shows the complexity of our SCDA system. In Table 7, parameters $\mathsf{n}_{\mathscr{H}}$, $\mathsf{n}_{\mathscr{R}}$, n_{C}, $\mathsf{n}_{\mathsf{T}i}$, $\mathscr{I}$, and s_i follow in Box 2.

5.4. Experimental Results. To demonstrate the efficiency of our system, we use MATLAB on a computer with an Intel(R) Core(TM) i3-2100 CPU (3.10 GHz) processor and 4 GB of RAM. Table 8 gives our experimental results. We assume that there are 100 hospitals with 100 clinical data each. Each row in Table 8 represents the mean of 15 trials.

5.5. Handling Overflows. In our SCDA system, a numerical data is represented as $\mathbf{b}_{i,\mathsf{k},l} \in \mathbb{Z}_p^m$ (referred to as $\mathbf{b}_i \in Z_p^m$ in this part). For handling overflows, we want to restrict $\mathbf{b}_i \in \mathbb{Z}_2^m$, not $\mathbf{b}_i \in \mathbb{Z}_p^m$. That is, $\mathbf{b}_i \in \mathbb{Z}_2^m$ is represented in binary string. For example, $m = 5$, $\mathbf{b}_1 = 27 = \begin{bmatrix}1 & 1 & 0 & 1 & 1\end{bmatrix}_{(2)}$, and $\mathbf{b}_2 = 17 = \begin{bmatrix}1 & 0 & 0 & 0 & 1\end{bmatrix}_{(2)}$. In additive operation on the GHV^* homomorphic encryption scheme, $\mathbf{b}_1 + \mathbf{b}_2 = \begin{bmatrix}1 & 1 & 0 & 1 & 1\end{bmatrix} + \begin{bmatrix}1 & 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}2 & 1 & 0 & 1 & 2\end{bmatrix}$. Then we can decode $\mathbf{b}_1 + \mathbf{b}_2 = \begin{bmatrix}2 & 1 & 0 & 1 & 2\end{bmatrix} = 2 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 2 \times 2^0 = 44$ which is similar to the decoding method in binary string.

As illustrated in Section 5.1, $p > n^c$ because $p \geq n^{3c+1}\log^5 n$ and the number of including tuples in a request for an aggregated data should be less than n^c. If we use the above method (i.e., $\mathbf{b}_i \in \mathbb{Z}_2^m$), then our SCDA system has no overflow problem, because $\mathbf{b}_1 + \cdots + \mathbf{b}_{n^c} \in \mathbb{Z}_p^m$.

5.6. Long-Term Confidentiality. In the area of managing sensitive information, cryptographic long-term confidentiality is absolutely needed [11]. In 1996, Shor showed that the RSA cryptosystem is broken by quantum attacks [12]. And the DLP (Discrete Logarithm Problem) cryptosystem and ECC (Elliptic Curve Cryptography) which are important alternatives to the RSA cryptosystem are also broken by quantum attacks.

In our SCDA system, we use the GHV^* homomorphic encryption scheme which is secure if the LWE problem is hard. The LWE problem is hard if the SVP (Shortest Vector Problem) is hard, and the SVP is known to be hard to quantum attacks. Therefore, our SCDA system guarantees long-term confidentiality because all algorithms in our SCDA system are secure against quantum attacks.

6. Conclusion

In this paper, we have proposed how to outsource clinical research data securely and how to control the outsourced data against potential breaches of privacy. We also were able to share accurate statistical patient data. To achieve this, we design the controlled secure aggregation protocol that enables a researcher to get aggregated results from outsourced ciphertexts of distinct researchers. Since our protocol is designed by using the lattice-based GHV^* homomorphic encryption, it guarantees long-term security against quantum computing attacks and is very efficient in computational overhead.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was partly supported by Basic Science Research Programs through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2012R1A1A3005550, 2013R1A2A2A01068200).

References

[1] "Health Iinsurance Portability and Accountability Act of 1996," Public Law 104-191, 104th Congress, August 1996.

[2] Z. Lin, A. B. Owen, and R. B. Altman, "Genomic research and human subject privacy," *Science*, vol. 305, no. 5681, p. 183, 2004.

[3] G. Loukides, J. C. Denny, and B. Malin, "The disclosure of diagnosis codes can breach research participants' privacy," *Journal of the American Medical Informatics Association*, vol. 17, no. 3, pp. 322–327, 2010.

[4] K. El Emam, "Methods for the de-identification of electronic health records for genomic research," *Genome Medicine*, vol. 3, no. 4, article 25, pp. 1–25, 2011.

[5] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," Tech. Rep. SRI-CSL-98-04, SRI Computer Science Laboratory, 1998.

[6] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, 2001.

[7] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[8] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "ℓ-diversity: privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, article 3, pp. 1–3, 2007.

[9] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: privacy beyond k-anonymity and ℓ-diversity," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 106–115, April 2007.

[10] D. R. Karp, S. Carlin, R. Cook-Deegan et al., "Ethical and practical issues associated with aggregating databases," *PLoS Medicine*, vol. 5, no. 9, article e190, pp. 1333–1337, 2008.

[11] J. Buchmann, A. May, and U. Vollmer, "Perspectives for cryptographic long-term security," *Communications of the ACM*, vol. 49, no. 9, pp. 50–55, 2006.

[12] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[13] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09)*, pp. 169–178, May 2009.

[14] C. Gentry, S. Halevi, and V. Vaikuntanathan, "A simple BGN-type cryptosystem from LWE," in *Advances in Cryptology—EUROCRYPT 2010*, vol. 6110 of *Lecture Notes in Computer Science*, pp. 506–522, Springer, Berlin, Germany, 2010.

[15] H. Hacıgümüş, B. Iyer, and S. Mehrotra, "Efficient execution of aggregation queries over encrypted relational databases," in *Database Systems for Advanced Applications*, vol. 2973 of *Lecture Notes in Computer Science*, pp. 125–136, 2004.

[16] E. Mykletun and G. Tsudik, "Aggregation queries in the databaase-as-a-service model," in *Proceedings of 20th Annual on Data and Applications Security*, pp. 89–103, July 2006.

[17] Z. Yang, S. Zhong, and R. N. Wright, "Privacy-preserving queries on encrypted data," in *Proceedings of the 11th European Symposium On Research In Computer Security (ESORICS '06)*, pp. 479–495, September 2006.

[18] G. Amanatidis, A. Boldyreva, and A. O'Neill, "Provably-secure schemes for basic query support in outsourced databases," in *Data and Applications Security XXI*, pp. 14–30, 2007.

[19] T. Ge and S. Zdonik, "Answering aggregation queries in a secure system model," in *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*, pp. 519–530, September 2007.

[20] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD '09)*, pp. 139–152, June 2009.

[21] B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao, "Privacy-preserving computation and verification of aggregate queries on outsourced databases," in *Proceedings of the 9th Privacy Enhancing Technologies Symposium (PETS '09)*, pp. 185–201, August 2009.

[22] A. D. Molina, M. Salajegheh, and K. Fu, "HICCUPS: health information collaborative collection using privacy and security," in *Proceedings of the 1st ACM Workshop on Security and Privacy in Medical and Home-Care Systems (SPIMACS '09)*, pp. 21–30, November 2009.

[23] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "EPPA: an efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1621–1632, 2012.

[24] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, article 34, pp. 1–40, 2009.

[25] J. Katz and Y. Lindell, *Introduction to Modern Cryptography: Principles and Protocols*, Chapman & Hall, Boca Raton, Fla, USA, 1st edition, 2007.

[26] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," *Theory of Computing Systems*, vol. 48, no. 3, pp. 535–553, 2011.

[27] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC '96)*, pp. 99–108, 1996.

[28] O. Goldreich, S. Goldwasser, and S. Halevi, "Collision-free hashing from lattice problems," in *Studies in Complexity and Cryptography*, vol. 6650 of *Lecture Notes in Computer Science*, pp. 30–39, Springer, Heidelberg, Germany, 2011.

[29] O. Goldreich, H. Krawczyk, and M. Luby, "On the existence of pseudorandom generators," *SIAM Journal on Computing*, vol. 22, no. 6, pp. 1163–1175, 1993.

[30] S. C. Ramanna and P. Sarkar, "On quantifying the resistance of concrete hash functions to generic multicollision attacks," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4798–4816, 2011.

[31] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, vol. 4052 of *Lecture Notes in Computer Science*, pp. 1–12, 2006.