Computer Security Symposium 2016
11 - 13 October 2016

Privately Evaluating Contingency Tables with Suppression

Wen-jie Lu[1] Jun Sakuma[1,2]

Abstract: Contingency tables are the most prevalent forms of statistical data, especially in the context of epidemiology and sample surveys. In this paper, we present a one-round secure protocol for privately conducting contingency tables with suppression. We operate in a multi-party outsourcing setting where an analyst wishes to obtain the contingency table from data collected from several data contributors. The computation of contingency tables and suppression are outsourced to a public server. Our protocol requires one round interaction between the analyst and the cloud server, that is, the analyst sends a query to the cloud and then the cloud responses the analyst with the suppressed contingency table. We assume the server behaves semi-honestly and then the security of our protocol follows even in the presence of a malicious analyst. We implement the protocol and demonstrate that we can evaluate the contingency table with thousands of data in 8 minutes.

Keywords: PWS, Contingency Table, Suppression, Fully Homomorphic Encryption

1. Introduction

Organizations such as research institutions, governmental agencies, and hospitals collect various person-specific data for research purposes. Using aggregated data from different sources can deliver better insights and lead to more precise analyses. For example, health information exchange networks such as NHIN [15], Common Well [5], and GaHIN [11] have been built for enhancing public health.

Nowadays, data analyses are straightforward and substantial if data are collected and stored in a centralized location, e.g., a third-party cloud server. However, this computation outsourcing raises concerns about the privacy of person-level sensitive information since data is stored in external, off-premise servers. In particular, in the context of medical data, sensitive patient records require to be kept confidential. Also, even we can prevent the cloud server from learning the private data, person-level information leakage might occur even from the analysis results. In particular in analyses that release aggregate statistics about the population.

In this work, we consider a particular aggregate statistic and aim to develop an efficient protocol for evaluating the statistic in a privacy-preserving way. Specifically, we examine contingency tables with suppression. A two-dimensional contingency table is a matrix over two categorical attributes. For each cell in the matrix, the table reports an aggregate value which represents the total number of individuals with a pair of attribute values (e.g., female and teacher) in the population. Despite their simple structure, contingency tables are the most prevalent forms of statistical data, especially in the context of epidemiology and sample surveys.

From data of N categorical attributes, we can derive $\mathcal{O}(N^2)$ contingency tables. However, it is usually hard to tell which of the contingency tables should be conducted since it depends on the type of applications. Once a new require for contingency tables comes up, the categorical data might should be re-uploaded to the cloud. This hinders the flexibility of outsourcing since data providers need to be standby and can not go offline. We can also require the data providers to upload all the $\mathcal{O}(N^2)$ combinations of categorical data so that once the data submission is finished, the data providers can go offline at any time. But this requires quadratically many storage spaces.

Intuitively, small values represent *rare* individuals in the population. In other words, we can consider that smaller aggregate values in the contingency table are at higher risk of information leakage than that of larger values. Similar to the K-anonymity [19], suppression is a common method for enforcing the privacy of rare individuals in the population. There are some strategies and methodologies [12], [13] for the contingency table suppression. Specifically, we consider a simple practice in this work: zero-out values in the contingency tables which are less than a small threshold $\mathcal{T}$.

The zero-out strategy naturally requires us to compare the aggregate values of the contingency table with the threshold $\mathcal{T}$ in a private manner. This comparison operation is usually implemented with *interactive* primitives such as the Yao's garbled circuit [21]. In this work, we develop a particular *one-round* protocol tailored for secure outsourcing the contingency table evaluation with suppression, using a fully homomorphic encryption (FHE) scheme. Our protocol requires only one data submission and linearly many storage spaces.

Related Work. This problem of privately evaluating contin-

1 筑波大学大学院システム情報工学研究科
University of Tsukuba
2 JST CREST

gency tables falls into the general area of privacy-preserving computation of statistics. During the last ten years, considerable effort has been made to developing privacy-preserving computations. One general approach is based on Yao's garbled circuits [6], [16], [21], where one party generates a garbled circuit representing the function they want to conduct and the other party evaluates the circuit. In our setting, it does not suffice to use a garbled circuit simply since we consider more than two parties involve in the computation.

The other general approach leverages homomorphic encryption [1], [2], [3], [14], [20]. Several studies that realize private outsourcing of descriptive and predictive statistics using FHE have been reported. Authors of [14] suggest to outsource mean and standard deviation privately. In [20], the authors introduced to evaluate the co-variance from FHE encrypted data. The private contingency table evaluation with suppression can be derived from the private data query system [1]. However, it might be impractical since [1] requires multiplicative depth of $\mathcal{O}(n)$ to perform the threshold test, where n is the number of data. We do not know of any existing practical implementation of a private contingency table evaluation protocol that also achieves suppression.

Our Contribution. First, we describe a secure one-round protocol for outsourcing the contingency table evaluation with suppression to a semi-honest cloud. This semi-honest model is well-suited for the cloud-based applications where we consider the cloud server is trying to offer a good service, and thus, is not motivated to give nonsensical output to the service users. Moreover, our protocol works in a one-round interaction pattern that is, once the encrypted data are submitted to the server, the server can independently perform the contingency table evaluation and suppression on the ciphertexts. No more interaction with the server is necessary before receiving the final result. The one-round interaction pattern enables us to apply our protocol to a useful three-party setting (discuss later in Section 2.3). Additionally, our protocol requires only linearly many storage spaces. We give a formal simulation-based proof of security of our protocol in the real-world/ideal-word paradigm [9].

Our protocol leverages a homomorphic encryption scheme and the homomorphic multiplication depth of our protocol is independent of the number of data. To show the practicality of our protocols, we implement our protocol using open-sourced libraries. We conduct experiments with over 4000 data points and with contingency table size up to 120. Our protocol can complete within 8 minutes for privately evaluating a contingency table with suppression in such scale of data. This work provides the first practical implementation of a private contingency table evaluation with the suppression functionality.

2. Preliminaries

We begin by introducing the notations used in this paper. For an integer $D > 0$, we write $[D]$ to denote the set of positive integers $\{0, 1, \dots, D-1\}$. We write $x \overset{\$}{\leftarrow} \mathcal{D}$ to denote that x is sampled uniformly at random from $\mathcal{D}$.

	$\mathsf{ind}(c_{ip})$		$\mathsf{ind}(c_{ip})$		$\mathsf{ind}(c_{ip})$	
element-wise multi.	$\mathsf{ind}(c_{iq})$			$\mathsf{ind}(c_{iq})$		
contribute to	λ_{00}	λ_{11}	λ_{02}	λ_{10}	λ_{01}	λ_{12}

Fig. 1 One multiplication gives 2×3 combinations of attributes of $c_{ip} \in \mathcal{C}_p$ and $c_{iq} \in \mathcal{C}_q$ where $|\mathcal{C}_p| = 2$ and $|\mathcal{C}_q| = 3$.

Let d_c be the number of categorical attributes. We write $\mathcal{C}_j \overset{\text{def}}{=} \{s_0^j, \dots, s_{|\mathcal{C}_j|-1}^j\}$ to denote the domain of the j^{th} categorical attribute. For instance, if $\mathcal{C}_j$ is *gender*, then s_0^j can denote *male*, meanwhile, s_1^j indicates *female*. We write $\mathcal{C} \overset{\text{def}}{=} \mathcal{C}_1 \times \cdots \times \mathcal{C}_{d_c}$ to denote the join domain of d_c categorical attributes. We use bold symbols e.g. $\boldsymbol{u}$ to denote vectors and the i^{th} element of $\boldsymbol{u}$ is denoted as u_i.

For the ring of integers modulo t, we write $\mathbb{Z}_t$. Let $\mathbb{Z}_t[X]$ be the ring of polynomials modulo t. Let $\mathcal{P}$ be a predicate. We use $\mathbf{1}\{\mathcal{P}(x)\}$ to indicate the indicator function for the predicate $\mathcal{P}$, that is $\mathbf{1}\{\mathcal{P}(x)\} = 1$ if and only if $\mathcal{P}(x)$ is true, and 0 otherwise.

2.1 Cryptographic Primitives

In this section, we give a brief overview of the cryptographic primitives that we use in our protocol.

Fully Homomorphic Encryption. In this paper, we use the Ring Learning-With-Error (RLWE) variant of [4], and its open-sourced implementation, i.e., HElib [17]. The message space of the RLWE variant of [4] is given as a quotient ring $\mathbb{A}_t \overset{\text{def}}{=} \mathbb{Z}_t[X]/\Phi_m(X)$ where t is a prime number and the $\Phi_m(X)$ is the m^{th} cyclotomic polynomial. By applying the Chinese-Remainder-Theorem (CRT), we can obtain an isomorphism $\mathbb{A}_t \cong \underbrace{\mathbb{Z}_{t^d} \otimes \cdots \otimes \mathbb{Z}_{t^d}}_{\ell \text{ copies}}$ where $\ell \times d = \phi(m)$ and $\phi(\cdot)$ is the Euler function. We designate the procedure $\mathcal{E} : \mathbb{Z}_t^\ell \to \mathbb{A}_t$ using the isomorphism as *CRT packing*. We refer to [18] for details about the CRT packing.

We denote the ciphertext of FHE with $[\![\cdot]\!]$. In addition, we write $[\![\boldsymbol{u}]\!]$ and $[\![\boldsymbol{v}]\!]$ to respectively denote $[\![\mathcal{E}(\boldsymbol{u})]\!]$ and $[\![\mathcal{E}(\boldsymbol{v})]\!]$ where integer vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_t^\ell$. For a vector whose length is less than ℓ, we append with 0s. The RLWE variant of [4] is an FHE scheme, as shown in the following equations.

$$\mathrm{Dec}([\![\boldsymbol{u}]\!] \oplus [\![\boldsymbol{v}]\!]) = \boldsymbol{u} + \boldsymbol{v} \mod t$$
$$\mathrm{Dec}([\![\boldsymbol{u}]\!] \odot [\![\boldsymbol{v}]\!]) = \boldsymbol{u} * \boldsymbol{v} \mod t,$$

where $\mathrm{Dec}(\cdot)$ is the decryption function and operators $\oplus$ and $\odot$ respectively denote the homomorphic addition and multiplication. The multiplication $*$ is performed *element-wisely*, that is, the j^{th} element of $\boldsymbol{u} * \boldsymbol{v}$ is $u_j v_j$. We can also operate *plaintext-ciphertext* additions and multiplications, e.g., $[\![\boldsymbol{u}]\!] \odot \mathcal{E}(\boldsymbol{v})$ homomorphically computes $\boldsymbol{u} * \boldsymbol{v}$.

Besides the addition and multiplication, we can homomorphically *rotate* the encrypted vectors with a given offset b: $\mathrm{Dec}([\![\boldsymbol{u}]\!] \gg b) = \tilde{\boldsymbol{u}}$ where $u_j = \tilde{u}_{j+b \mod \ell}$ for all positions $0 \le j < \ell$. In addition, the rotation in the opposite direction $[\![\boldsymbol{u}]\!] \ll b$ is also well-defined and available in [17].

2.2 Contingency Table

We view a contingency table for the p^{th} and q^{th} categorical attributes $\mathcal{CT} : \mathcal{C}^n \times \mathbb{Z}^2 \to \mathbb{Z}^{|\mathcal{C}_p| \times |\mathcal{C}_q|}$ as implementing a

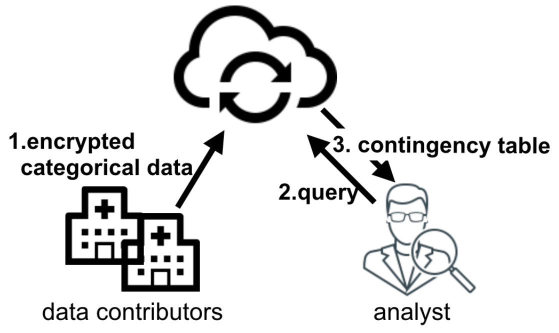


Fig. 2 Outsource the conduction of contingency tables to a cloud server.

function on n categorical data. Evaluation of a contingency table corresponds to counting combinations (s_u^p, s_v^q) for all possible (u, v) pairs. We write λ_{uv} to denote the counting of the combination (s_u^p, s_v^q). For instance, one categorical data $\boldsymbol{c} = [\cdots, s_2^p, \cdots, s_3^q, \cdots]$ contributes to the counting λ_{23} by 1.

Encoding. To encrypt a categorical value s_j^p, we use an indicator encoding ind which is defined as $\mathsf{ind} : \mathcal{C}_p \to \{0, 1\}^{|\mathcal{C}_p|}$. More specifically, ind takes as input a value s_j^p and outputs a vector which all elements of 0 except the j^{th} element, which is set to 1. We convert categorical values to integer vectors using ind then we can apply the CRT packing and encrypt the categorical attributes, i.e., $[\![\mathsf{ind}(s_j^p)]\!]$.

Contingency Table Evaluation. We can obtain the contingency table by element-wise multiplications and rotations of vectors due to the indicator encoding. Let $c_{ip} \in \mathcal{C}_p$ and $c_{iq} \in \mathcal{C}_q$ be two categorical values. The first element of $\mathsf{ind}(c_{ip}) * \mathsf{ind}(c_{iq})$ is 1 if and only if $c_{ip} = s_0^p$ and $c_{iq} = s_0^q$. Similarly, the first element of $(\mathsf{ind}(c_{ip}) \lll 2) * \mathsf{ind}(c_{iq})$ is 1 when $c_{ip} = s_2^p$ and $c_{iq} = s_0^q$. Generally, we need $\mathcal{O}(\max\{|\mathcal{C}_p|, |\mathcal{C}_q|\})$ multiplications and rotations to evaluate the contingency table of attributes $\mathcal{C}_p$ and $\mathcal{C}_q$.

To reduce the number of element-wise multiplications, we introduce a "Chinese-Remainder-Theorem" trick. We repeat the $\mathsf{ind}(c_{ip})$ for $|\mathcal{C}_q|$ times and repeat the $\mathsf{ind}(c_{iq})$ for $|\mathcal{C}_p|$ times. We take the element-wise multiplication of the two self-repeat vectors and let the product vector be $\boldsymbol{\mu}$. If $|\mathcal{C}_q|$ and $|\mathcal{C}_p|$ are *co-prime*, then $\boldsymbol{\mu}$ contains all counting in the contingency table [*1]. An example of this trick is given in Fig. 1. From the remark 1, we can identify λ_{uv} from $\boldsymbol{\mu}$.

Remark 1. *For a (u, v) pair, the counting λ_{uv} is given by μ_x, where $x \equiv u \mod |\mathcal{C}_p|$ and $x \equiv v \mod |\mathcal{C}_q|$.*

Suppression. To suppress a contingency table, we compare all the counting in the table with a given threshold $\mathcal{T} \in \mathbb{Z}^+$. For counting $\lambda_{uv} < \mathcal{T}$, we suppress them by setting the value of $\lambda_{uv} = 0$ while we do nothing to counting that $\lambda_{uv} \geq \mathcal{T}$. The threshold $\mathcal{T}$ can be decided by data contributors in advance.

2.3 Security Model

In our work, we consider three stakeholders, an analyst, data contributors, and a cloud server (Fig. 2). An analyst is an entity that wishes to conduct contingency tables. Data contributors, e.g., hospitals, are entities that provide private data for the contingency table conduction. The cloud server provides computation power and storage, for analysts and data contributors.

In our setting, the data contributors encrypt their data using a key that generated by the analyst, and they submit the ciphertexts to the cloud server. When the data submission is finished, the data contributors can go offline at any time since they do not involve in the further computation. The analyst queries the cloud and asks for a contingency table by designating two attributes. The cloud privately computes the contingency table on the encrypted data that collected from data contributors and responses the analyst with the computation result. In the end, the analyst decrypts and obtains the contingency table.

We design a two-party protocol that runs on the cloud server and the analyst. For private contingency table evaluation with suppression, we express this functionality $\mathcal{F}$ as the mapping $(\{\boldsymbol{c}_i\}^n, (p, q)) \mapsto (-, \mathcal{CT}(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T}))$, where $\{\boldsymbol{c}_i\}^n$ is the collection of categorical data of n data contributors, and (p, q) is the targeting attributes that designated by the analyst. We use '–' to indicate that at the end of the protocol execution, the cloud server learns nothing while the analyst obtains the suppressed contingency table $\mathcal{CT}$ of attributes $\mathcal{C}_p$ and $\mathcal{C}_q$.

The security definition we leverage in this paper follows the real-world/ideal-world paradigm of [9]. In next section, we present a *one-round* protocol to compute the contingency table with suppression. By saying one-round, we mean that our protocol needs one round of interaction between the cloud server and the analyst: the analyst sends the query and then the cloud server responses the ciphertexts it has computed. From that ciphertexts, the analyst can learn the suppressed contingency table. We prove that our protocol securely computes the functionality $\mathcal{F}$ and give the simulation-based proof in Appendix. In our work, we assume that the data contributors and cloud server behave semi-honestly, which means they follow the protocol specification but may try to learn additional information. Since our protocol works in the one-round pattern, the analyst is not required to be semi-honest.

3. One-round and Secure Contingency Table Evaluation with Suppression

In this section, we detail our one-round protocol for securely outsourcing the computation of contingency tables with suppression to a public cloud server.

3.1 Building Blocks

We describe two building blocks used in our construction: a procedure for repeating encrypted integer vectors, and a greater-than protocol for comparing two encrypted integers.

REPEAT Procedure. As described in Section 2.3, to evaluate a contingency table, we need to repeat ind-encoded vectors. More precisely, we need to *homomorphically* repeat the

[*1] We can use this trick, even the co-prime constraint is not satisfied. To do so, we append some 0s to $\mathsf{ind}(c_{ip})$ or $\mathsf{ind}(c_{iq})$, making the length of them are co-prime.

first $|\mathcal{C}_p|$ elements of $\mathsf{ind}(c_{ip})$ for $|\mathcal{C}_q|$ times. We introduce the procedure REPEAT($[\![\boldsymbol{u}]\!], k, R$) which repeats the first k elements of $[\![\boldsymbol{u}]\!]$ for R times as follows.

```
1.  $[\![\tilde{\boldsymbol{u}}]\!] \overset{\text{def}}{=} [\![\mathbf{0}]\!]$.
2.  $R = (b_\rho \cdots b_1 b_0)_2$ where $b_\rho$ is the most significant bit.
3.  For $0 \le i \le \rho$
      3.1  If $b_i$ is 1 then $[\![\tilde{\boldsymbol{u}}]\!] \leftarrow [\![\tilde{\boldsymbol{u}}]\!] \gg k$; $[\![\tilde{\boldsymbol{u}}]\!] \leftarrow [\![\tilde{\boldsymbol{u}}]\!] \oplus [\![\boldsymbol{u}]\!]$
      3.2  $[\![\boldsymbol{u}]\!] \leftarrow [\![\boldsymbol{u}]\!] \oplus ([\![\boldsymbol{u}]\!] \gg k)$
      3.3  $k \leftarrow k \times 2$
4.  return $[\![\tilde{\boldsymbol{u}}]\!]$
```

The procedure is very simple, and it requires $\mathcal{O}(\log R)$ homomorphic rotations and homomorphic additions. Two requirements are needed to make sure the procedure work correctly: Elements of $\boldsymbol{u}$ after the k^{th} position are 0; and the CRT packing offers enough spaces i.e., $k \times R \le \ell$.

Batch Greater-than Protocol. To achieve suppression, we need comparisons on encrypted integers. We newly instantiate a variant of the greater-than (GT) protocol from [10], specially for the suppression of the contingency table.

The protocol of [10] bases on the idea that for one pair of integers $a, b \in [D]$, we have $a \ge b \Leftrightarrow \exists w \in [D] \to a - b - w = 0$. Our batch greater-than (bGT) protocol (Alg. 1) uses the REPEAT procedure and the CRT packing to conduct the comparison of θ pairs of integers in a batch manner. The invocation of REPEAT makes sure that for all j, we can homomorphically conduct $a_j - b_j - w$ simultaneously. The $\alpha(j) \overset{\text{def}}{=} \theta \cdot \alpha + j$ makes sure that no collision between the θ comparisons. We now prove the security of our protocol.

Theorem 1. *The protocol in Alg. 1 securely computes the functionality* $(\boldsymbol{a}, \boldsymbol{b}) \mapsto \mathbf{1}\{a_j \ge b_j\}$ *for* $0 \le j < \theta$.

Proof. Without loss of generality, we show that the difference $|a_j - b_j|$ is not revealed by our bGT protocol. We let $\boldsymbol{\beta}(j) \overset{\text{def}}{=} [\beta_j, \ldots, \beta_{(D-1)\theta+j}]$. We have that $\boldsymbol{\beta}(j)$ contains one 0 if and only if $a_j \ge b_j$ and non-zero elements of $\boldsymbol{\beta}(j)$ reveal $|a_j - b_j|$. To hide these information, we choose a random vector $\boldsymbol{r}_j \overset{\$}{\leftarrow} (\mathbb{Z}_t/\{0\})^D$ and homomorphically compute the ciphertext of $\boldsymbol{r}_j * \boldsymbol{\beta}(j)$. If $\beta(j)_\alpha \ne 0$, then $r_{j\alpha}\beta(j)_\alpha$ is uniform over $\mathbb{Z}_t/\{0\}$ because t is a prime number. Since $\beta(j)_\alpha = a_j - b_j - w_\alpha$, we need to use unpredictable value of w_α to prevent leakage of $|a_j - b_j|$. The random permutation π_j enables us to do so. □

Our bGT protocol leverages the CRT packing for private comparisons. When the CRT packing does not offer enough space i.e. $\ell < \theta \cdot D$, we can not pack a $(\theta \cdot D)$-length vector into *one* ciphertext. Fortunately, we can use *multiple* ciphertexts to address this issue. In this case, the bGT generates $\lceil (\theta \cdot D)/\ell \rceil$ ciphertexts.

3.2 Secure Contingency Table Evaluation with Suppression in One-round interaction

Our protocol for privately evaluating a contingency table with suppression within one round of interaction is given in Alg. 2. The protocol is secure under the assumption with a semantically secure FHE scheme, a secure channel (e.g. Transport Layer Security) and a collusion-free server. We make the following assumptions about our setting.

Algorithm 1 Batch greater-than protocol.

- Input: $[\![\boldsymbol{a}]\!]$ and $[\![\boldsymbol{b}]\!]$ where $\boldsymbol{a}, \boldsymbol{b} \in [D]^\theta$ for $D, \theta \in \mathbb{Z}^+$.
- Output: $[\![\boldsymbol{\gamma}]\!]$ where $|\boldsymbol{\gamma}| = \theta \cdot D$.
- One can learn, for $0 \le j < \theta$,

$$\mathbf{1}\{a_j \ge b_j\} = \mathbf{1}\{0 \in \{\gamma_j, \gamma_{\theta+j}, \ldots, \gamma_{(D-1)\theta+j}\}\}.$$

```
1: Compute $[\![\tilde{\boldsymbol{a}}]\!] \leftarrow$ REPEAT($[\![\boldsymbol{a}]\!], \theta, D$); $[\![\tilde{\boldsymbol{b}}]\!] \leftarrow$ REPEAT($[\![\boldsymbol{b}]\!], \theta, D$)
2: Generate random permutations $\pi_j : [D] \to [D]$ for $0 \le j < \theta$.
3: Compute a $\theta \cdot D$ length vector $\boldsymbol{w}$ in which $w_{\alpha(j)} \leftarrow \pi_j(\alpha)$. Here
   $\alpha(j) \overset{\text{def}}{=} \theta \cdot \alpha + j$, for $\alpha \in [D]$ and $0 \le j < \theta$.
4: Compute $[\![\boldsymbol{\beta}]\!] \leftarrow [\![\tilde{\boldsymbol{a}}]\!] - [\![\tilde{\boldsymbol{b}}]\!] - \mathcal{E}(\boldsymbol{w})$.
5: Compute $[\![\boldsymbol{\gamma}]\!] \leftarrow [\![\boldsymbol{\beta}]\!] \otimes \mathcal{E}(\boldsymbol{r})$ where $\boldsymbol{r} \overset{\$}{\leftarrow} (\mathbb{Z}_t/\{0\})^{\theta \cdot D}$.
6: Output $[\![\boldsymbol{\gamma}]\!]$.
```

- The suppression threshold $\mathcal{T} \in \mathbb{Z}^+$ is decided by the data contributors in advance.
- The i^{th} data contributor's data consists of d_c categorical data $\boldsymbol{c}_i \overset{\text{def}}{=} (c_{i1}, \ldots, c_{id_c})$ where $c_{ij} \in \mathcal{C}_j$ is the value of the j^{th} categorical attribute. The data collection of n data contributors is denoted as $\{\boldsymbol{c}_i\}^n$.
- Data contributors already have the encryption key pk that is generated by the analyst.
- Data contributors encrypt their data using pk and send the ciphertexts to the server using the secure channel.

In the first part of the protocol (from Step 1 to Step 4), the server homomorphically repeats the encrypted vectors and performs the multiplications and additions. The server then obtains $[\![\boldsymbol{\mu}]\!]$ which is the ciphertext that encrypts all counting of the contingency table.

In the second part of the protocol, the server hides the counting of the contingency table with a random vector $\boldsymbol{\delta}$. In Step 6, the server invokes the bath greater-than protocol to compare the first Σ elements of $\boldsymbol{\mu}$ with the suppression threshold. After that, the server derives a vector $\boldsymbol{r}$ from $\boldsymbol{\delta}$ and adds $\boldsymbol{r}$ to the output of the bGT in Step 7. That enables the analyst to learn the counting whose values are above the suppression threshold but also hides the suppressed counting from the analyst.

We now show that the protocol is correct.

Lemma 1. *If the data contributors, the server and the analyst follow the protocol in Alg. 2, then at the end of the protocol, the analyst learns* $\mathcal{CT}(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T})$.

Proof. By using the co-prime numbers k_1 and k_2 in the REPEAT procedure, the Eq. 1 gives $[\![\boldsymbol{\mu}]\!]$ which is the ciphertext of the counting of the contingency table. For suppression, we use a blinding factor $\boldsymbol{\delta}$ and add it to $\boldsymbol{\mu}$. To learn the counting λ_{uv}, the analyst needs to figure out δ_x where x is decided according to the Remark 1. Whenever the analyst gets δ_x, he knows $\lambda_{uv} = \mu'_x - \delta_x$.

If $\lambda_{uv} \ge \mathcal{T}$ holds, $\boldsymbol{\gamma}(x)$ contains one and only one 0 due to the specification of our bGT protocol. In this case, let the j be the position such that $\gamma(x)_j = 0$. Then the analyst can identify the blinding factor as δ_x since $\gamma'(x)_j = \gamma(x)_j + \delta_x$.

In the opposite case, to prevent the analyst learning $\boldsymbol{\delta}$ directly through $\boldsymbol{\gamma}' - \boldsymbol{\gamma}$, we multiply a non-zero random vector $\boldsymbol{r}^*$ to $\boldsymbol{\gamma}$. If $\gamma(x)_j = 0$ then $\gamma^*(x)_j = 0$ still holds. The ana-

Algorithm 2 Secure contingency table protocol with suppression.

```
Let $(\mathsf{pk}, \mathsf{sk})$ be the encryption-decryption key-pair for the FHE scheme [4]. The analyst holds the decryption key $\mathsf{sk}$.
  • The $i^{\text{th}}$ data contributor input: $\mathsf{ctxt}_i^j \overset{\text{def}}{=} [\![\mathsf{ind}(c_{ij})]\!]$ for $1 \le j \le d_c$.
  • Analyst input: the targeting attributes $p$ and $q$.
  • Analyst output: $\mathcal{CT}(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T})$.
 1: Server: Check the validity of $p$ and $q$, if $p, q < 1$ or $p, q > d_c$ or $p = q$, it aborts the protocol and replies with $\perp$.
 2: Find $k_1, k_2$ where $k_1 \ge |\mathcal{C}_p|, k_2 \ge |\mathcal{C}_q|$ and $\mathsf{gcd}(k_1, k_2) = 1$. (Designate $\Sigma \overset{\text{def}}{=} k_1 k_2$)
 3: Compute the following ciphertexts using the Alg. 1.
        $\widetilde{\mathsf{ctxt}}_i^p \leftarrow \textsc{Repeat}(\mathsf{ctxt}_i^p, k_1, k_2)$
        $\widetilde{\mathsf{ctxt}}_i^q \leftarrow \textsc{Repeat}(\mathsf{ctxt}_i^q, k_2, k_1)$
 4: Compute the contingency table with homomorphic multiplications and homomorphic additions
        $[\![\boldsymbol{\mu}]\!] \leftarrow \sum_i \widetilde{\mathsf{ctxt}}_i^p \odot \widetilde{\mathsf{ctxt}}_i^q.$        (1)
 5: Compute $[\![\boldsymbol{\gamma}]\!] \leftarrow \mathsf{bGT}([\![\boldsymbol{\mu}]\!], \mathcal{T})$.    ▷ Batch compare the first $k_1 k_2$ elements of $\boldsymbol{n}$ with the threshold $\mathcal{T}$
 6: Sample $\boldsymbol{\delta} \overset{\$}{\leftarrow} \mathbb{Z}_t^{\Sigma}$ and compute $[\![\boldsymbol{\mu}']\!] \leftarrow [\![\boldsymbol{\mu}]\!] \oplus \mathcal{E}(\boldsymbol{\delta})$.    ▷ Hide the counting of the contingency table
 7: Compute $[\![\boldsymbol{\gamma}']\!] \leftarrow [\![\boldsymbol{\gamma}]\!] \oplus \mathcal{E}(\boldsymbol{r})$ where the length of $\boldsymbol{r}$ is $n\Sigma$ and elements of $\boldsymbol{r}$ satisfies    ▷ Hide the blinding factor $\boldsymbol{\delta}$
        $r_{xn+x'} = \delta_x$ for all $0 \le x, x' < \Sigma$.
 8: Compute $[\![\boldsymbol{\gamma}^*]\!] \leftarrow [\![\boldsymbol{\gamma}]\!] \odot \mathcal{E}(\boldsymbol{r}^*)$ where $\boldsymbol{r}^* \overset{\$}{\leftarrow} (\mathbb{Z}_t/\{0\})^{n\Sigma}$.    ▷ Prevent the blinding factor being learnt from $\boldsymbol{\gamma} - \boldsymbol{\gamma}'$
 9: Response to the analyst with ciphertexts $[\![\boldsymbol{\mu}']\!]$, $[\![\boldsymbol{\gamma}']\!]$, and $[\![\boldsymbol{\gamma}^*]\!]$.
10: Analyst: Decrypt $[\![\boldsymbol{\mu}']\!]$, $[\![\boldsymbol{\gamma}']\!]$ and $[\![\boldsymbol{\gamma}^*]\!]$.
11: For each pair of $u$ and $v$ that $0 \le u < |\mathcal{C}_q|, 0 \le v < |\mathcal{C}_p|$
    (1) Find the $0 \le x < \Sigma$ such that $x \equiv u \mod k_1$ and $x \equiv v \mod k_2$.    ▷ According to the Remark 1
    (2) Test whether $0 \in \boldsymbol{\gamma}^*(x)$. If so, designate the position of the 0 as $j$, then output the counting $\lambda_{uv} = \mu'_x - \boldsymbol{\gamma}'(x)_j$. Otherwise, i.e. $0 \notin \boldsymbol{\gamma}^*(x)$
        then output the counting $\lambda_{uv} = 0$. Here $\boldsymbol{\gamma}^*(x) \overset{\text{def}}{=} [\gamma_x^*, \gamma_{\Sigma+x}^* \cdots \gamma_{(n-1)\Sigma+x}^*]$.
```

lyst can still learn counting $\lambda_{uv} \ge \mathcal{T}$ while he can not know $\lambda_{uv} \le \mathcal{T}$. The suppression is achieved. □

We now prove that this protocol is secure against semi-honest adversaries.

Theorem 2. *If the data contributors and cloud server behave semi-honestly and the cloud server does not collude with the analyst, the protocol in Alg. 2 securely compute the functionality* $(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T}) \mapsto (-, \mathcal{CT}(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T}))$.

We give a simulation-based proof in Appendix.

Asymptotic Analysis. We briefly describe the asymptotic performance of the Alg. 2. Let n be the number of data contributors, d_c be the number of categorical attributes, $\Sigma \overset{\text{def}}{=} k_1 k_2$ be the size of the contingency table, ℓ be the maximum length that the CRT packing can handle at once. Consider the data contributor's computation. Each data contributor encrypts $\mathcal{O}(d_c)$ ciphertexts and uploads them to the cloud server.

On the server side, operating the Repeat requires $\mathcal{O}(n \log \Sigma)$ homomorphic rotations and additions. Computing Eq. 1 requires $\mathcal{O}(n)$ homomorphic multiplications and additions. The bGT protocol in Step 6 generates $\mathcal{O}(\lceil (\Sigma \cdot n)/\ell \rceil)$ ciphertexts which costs $\mathcal{O}(\lceil (\Sigma \cdot n)/\ell \rceil)$ homomorphic subtractions and plaintext-ciphertext multiplications. Thus, in Step 7, it costs $\mathcal{O}(\lceil (\Sigma \cdot n)/\ell \rceil)$ homomorphic additions to hide the blinding factor.

On the analyst side, the analyst needs to decrypt $\mathcal{O}(\lceil (\Sigma \cdot n)/\ell \rceil)$ ciphertexts. The communication cost between the analyst the cloud server is $\mathcal{O}(\lceil (\Sigma \cdot n)/\ell \rceil)$ ciphertexts.

4. Experiments

Our implementation was written in C++ and we used HElib [17] for the RLWE variant of [4]. We compile our program using g++ 4.8.2 on Ubuntu 14.04. We run our implementation on a physical machine with 12 2.60GHz Xeon CPU E5-4627 v3 processors and 1TB of RAM. We leverage parallelism to accelerate the computation: 36 threads were used on the server side, and 4 threads were used on the analyst side. We consider that the analyst usually can not offer too many cores for parallelism. We code the parallel program using the native stand thread library of C++ 11.

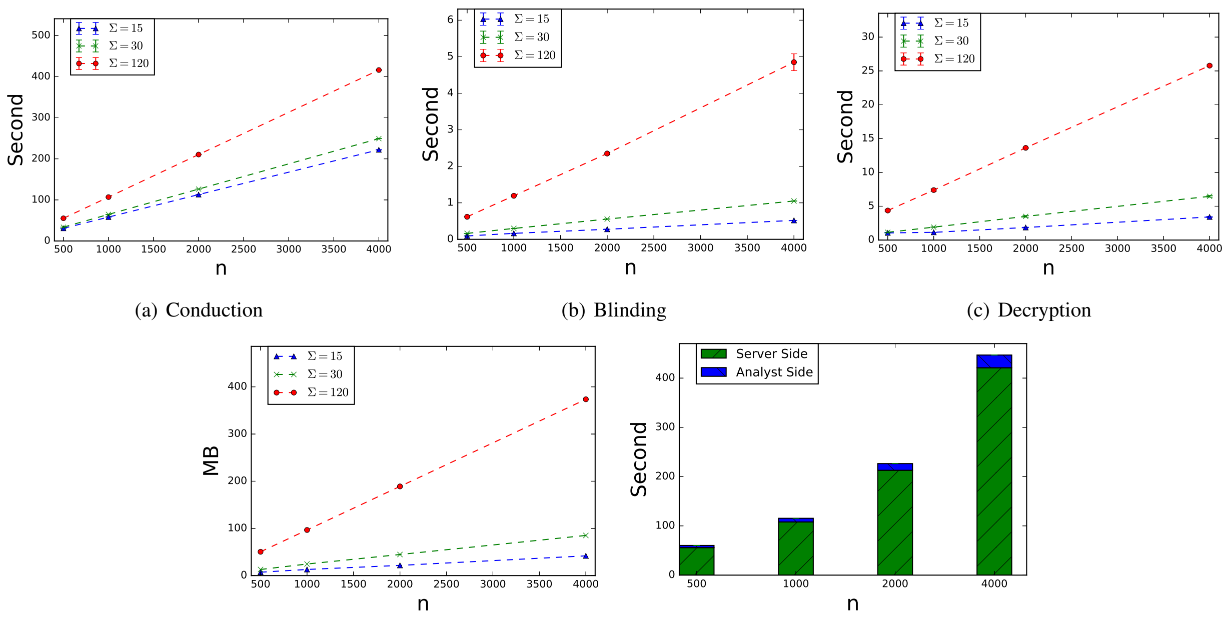
4.1 Parameters of HElib

The parameters of HElib include several variables, and we focus on three of them, i.e., m, t, and L and other parameters of HElib were set by default. Here t and m determine the message space of FHE, i.e. $\mathbb{A}_t$; and the parameters L and m together determine the security level of the FHE. We use L and m that provide at least 80-bit security. For the relation between the security level, L, and m, we refer to the design documentation of HElib [17].

To achieve the best performance, we need to choose the parameters of the HElib appropriately. We determine these parameters base on three concerns.

(1) provide the desired security level
(2) offer sufficient spaces of the CRT packing, i.e. ℓ
(3) operate the homomorphic rotation efficiently

In our experiments we used parameters that $m = 16384$, $t = 8191$, and $L = 10$ which provide at least 80-bit secu-



(a) Conduction (b) Blinding (c) Decryption

(d) Server-analyst communication (e) Server-analyst workload ($\Sigma = 120$)

Fig. 3 Running performances on conduction, blinding and decryption. Figure (e) shows the workloads on the server side and the analyst side. (Numbers are the average of 4 runs)

rity level and $\ell = 4096$. The homomorphic rotation on this message space $\mathbb{A}_t$ is efficient because we can find a good generator of this space. From the implementation aspect, we choose m and t so that[*2]

```
PAlgebra::numOfGens() == 1
&& PAlgebra::SameOrd(0) == true
```

We refer [7], [8], [17] for the homomorphic rotation.

The performance numbers of FHE on these parameters are summarized as follows (average of 10 runs).

- homomorphic addition: 0.005 sec
- homomorphic multiplication: 0.242 sec
- rotation: 0.153 sec
- plaintext-ciphertext addition: 0.018 sec
- plaintext-ciphertext multiplication: 0.025 sec
- encryption: 0.089 sec
- decryption: 0.025 sec

4.2 Experimental Results

In our benchmarks, we measure the server **conduction** (Step 2 to Step 5), server computation of **blinding** (Step 6 to Step 7), communication bandwidth between the server and analyst, and analyst **decryption** (Step 9 to Step 10). Remind that we used 36 parallels for the computation on the server side while used only 4 parallels on the analyst side.

To better understand the scalability of our protocol, we performed some experiments on randomly generated categorical data with different domain sizes. The number of data $n = 500, 1000, 2000$ and 4000. The size of contingency table $\Sigma = 15, 30$ and 120. We consider to compute contingency tables with several hundred counting is sufficient for real applications. The suppression threshold does not affect the running performance. Thus we use a fixed value. The execution time was measured in unit of second, and the communication was measured in the unit of the megabyte (MB). The benchmarks are shown in Fig. 3. To demonstrate the workload on both sides, we give Fig. 3(e).

[*2] PAlgebra is a C++ class in HElib.

4.3 Discussion

The benchmarks follow our asymptotic analysis. For example, the conduction time, i.e. Fig.3(a), grows linearly with the number of data that is the running time ($\Sigma = 120$) increases from 200s to 400s as n grows from 2000 to 4000. We can see that the gaps between each line are different, which confirm us that the blinding, decryption and communication cost grow linearly with the contingency table sizes Σ while the conduction cost increases logarithmically with Σ.

From the Fig.3(e), we can see that about 95% of the workload were performed on the server side, which satisfies our motivation of outsourcing computation to the cloud server. However, the workload of the analyst increases as the number of data grows, since he needs to decrypt more ciphertexts. Remind that we only used 4 parallels on analyst side, the decryption time can be easily reduced with more cores.

One improvement we can do is to decide a tight domain size for counting of the contingency table. In the current construction, we use the domain size as $[0, n]$ since the counting will never exceed the total number of data. However, this loose domain size leads to the linear complexity of n in blinding, decryption and communication. Of course, this cost can be reduced by choosing a larger ℓ.

5. Conclusion

In this work, we present a one-round protocol for privately evaluating the contingency table with suppression.

Our protocol is the first construction of private contingency table evaluation with suppression functionality. Our protocol works in a one-round interaction pattern . Thus we require the data contributors and the cloud server to behave semi-honestly. To some extent (we need to assume collusion-free servers), our protocol is secure against malicious analysts. Our results show that our protocol can conduct the evaluation of contingency table with a size of 120 from 4000 data within 8 minutes and about 95% of workload are performed by the cloud server. Even the performance of our protocol might seem passable; we conclude that our protocol has its merits and can work properly for some thousands of data. Also, the one-round interaction pattern of our protocol allows us to extend the protocol to other computation models.

For the current construction, we might not be able to reduce the execution time significantly, except leveraging the parallelism. To accelerate the execution of the conduction phase is one of our future work, and we hope to compare our protocol with the garbled circuit implementations.

Acknowledgments This work was supported by JST CREST and KAKENHI Grant Number 16H02864.

References

[1] Boneh, D., Gentry, C., Halevi, S., Wang, F., and Wu, D. J. Private database queries using somewhat homomorphic encryption. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings* (2013), pp. 102–118.

[2] Bos, J. W., Lauter, K. E., and Naehrig, M. Private predictive analysis on encrypted medical data. *IACR Cryptology ePrint Archive 2014* (2014), 336.

[3] Bost, R., Popa, R. A., Tu, S., and Goldwasser, S. Machine learning classification over encrypted data. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11* (2015).

[4] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012* (2012), pp. 309–325.

[5] commonwell health allicance. CommonWell. `http://www.commonwellalliance.org/`. Accessed: 2016-8-03.

[6] Constable, S. D., Tang, Y., Wang, S., Jiang, X., and Chapin, S. Privacy-preserving gwas analysis on federated genomic datasets. *BMC medical informatics and decision making 15*, 5 (2015), 1.

[7] Gentry, C., Halevi, S., and Smart, N. P. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings* (2012), pp. 465–482.

[8] Gentry, C., Halevi, S., and Smart, N. P. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings* (2012), pp. 850–867.

[9] Goldreich, O. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[10] Golle, P. A private stable matching algorithm. In *Financial Cryptography and Data Security, 10th International Conference, FC 2006, Anguilla, British West Indies, February 27-March 2, 2006, Revised Selected Papers* (2006), pp. 65–80.

[11] health information network, G. GaHIN. `http://www.gahin.org/`. Accessed: 2016-8-03.

[12] Kirkendall, N., and Sande, G. Comparison of systems implementing automated cell suppression for economic statistics. *Journal of Official Statistics 14*, 4 (1998), 513.

[13] Nabar, S. U., and Mishra, N. Releasing private contingency tables. *Journal of Privacy and Confidentiality 2*, 1 (2009), 9.

[14] Naehrig, M., Lauter, K. E., and Vaikuntanathan, V. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011* (2011), pp. 113–124.

[15] nationwide health information network. NHIN. `https://www.healthit.gov/policy-researchers-implementers/nationwide-health-information-network-nwhin`. Accessed: 2016-8-03.

[16] Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., and Taft, N. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013* (2013), pp. 334–348.

[17] Shai Halevi, V. S. HELib. `http://shaih.github.io/HElib/index.html`. Accessed: 2014-12-10.

[18] Smart, N. P., and Vercauteren, F. Fully homomorphic SIMD operations. *Designs, codes and cryptography 71*, 1 (2014), 57–81.

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

[20] Wu, D., and Haven, J. Using homomorphic encryption for large scale statistical analysis. `http://cs.stanford.edu/people/dwu4/FHE-SI_Report.pdf`, 2012.

[21] Yao, A. C. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986* (1986), pp. 162–167.

Appendix

A.1 Security Proof

Proof of Theorem 2. We prove security against the semi-honest analyst. Let $\mathcal{A}$ be the semi-honest analyst in the real protocol. We construct a semi-honest simulator $\mathcal{S}$ in the ideal world as follows:

(1) At the beginning of the protocol, $\mathcal{S}$ obtains the input p and q, the number of data n, the size of the contingency table Σ, and the suppressed contingency table $\mathcal{CT}(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T})$.

(2) Start running $\mathcal{A}$ on input p and q. Let pk be the analyst's public key.

(3) Arrange the counting λ_{uv} in $\mathcal{CT}(\{\boldsymbol{c}_i\}^n, p, q, \mathcal{T})$ to get $\boldsymbol{\mu} \in \mathbb{Z}^{\Sigma}$ according to the Remark 1.

(4) Sample $\hat{\boldsymbol{\delta}} \overset{\$}{\leftarrow} \mathbb{Z}_t^{\Sigma}$ and compute $\hat{\boldsymbol{\mu}}' \leftarrow \boldsymbol{\mu} + \hat{\boldsymbol{\delta}}$

(5) Sample $\hat{\boldsymbol{\gamma}} \overset{\$}{\leftarrow} (\mathbb{Z}_t/\{0\})^{n\Sigma}$. For each μ_x of $\boldsymbol{\mu}$, if $\mu_x > 0$ then set $\hat{\gamma}_j = 0$ where j is randomly chosen from $\{x, \Sigma + x, \dots, (n-1)\Sigma + x\}$.

(6) Compute $\hat{\boldsymbol{\gamma}}' \leftarrow \hat{\boldsymbol{\gamma}} + \boldsymbol{r}$, where $r_{xn+x'} = \hat{\delta}_x$ for $0 \le x, x' < \Sigma$.

(7) Compute $\hat{\boldsymbol{\gamma}}^* \leftarrow \hat{\boldsymbol{\gamma}} * \boldsymbol{r}^*$, where $\boldsymbol{r}^* \overset{\$}{\leftarrow} (\mathbb{Z}_t/\{0\})^{n\Sigma}$.

(8) When $\mathcal{A}$ sends query, check the validity of the query input. If the query is valid, reply with ciphertexts $[\![\hat{\boldsymbol{\mu}}']\!]$, $[\![\hat{\boldsymbol{\gamma}}']\!]$ and $[\![\hat{\boldsymbol{\gamma}}^*]\!]$. Otherwise, reply with $\perp$.

The view of the $\mathcal{A}$ in the real execution comprises three parts: $\boldsymbol{\mu}'$, $\boldsymbol{\gamma}^*$ and $\boldsymbol{\gamma}'$. Since the contingency table evaluation is deterministic, to show the security of our protocol, it sufficient to show that the view that $\mathcal{S}$ simulates for $\mathcal{A}$ is computationally indistinguishable from the view of $\mathcal{A}$ in the real execution.

In the real execution, $\boldsymbol{\mu}'$ distributes uniformly over $\mathbb{Z}_t^{\Sigma}$ due to the uniform randomness $\boldsymbol{\delta}$. It is easy to see that the simulated $\hat{\boldsymbol{\mu}}'$ also distributes the same with $\boldsymbol{\mu}'$ because $\hat{\boldsymbol{\delta}}$ is chosen uniformly at random from $\mathbb{Z}_t^{\Sigma}$. Thereby $\boldsymbol{\mu}' \approx^c \hat{\boldsymbol{\mu}}'$.

Next, consider the distribution of $\boldsymbol{\gamma}^*$ in the real protocol. $\boldsymbol{\gamma}^*$ is generated by multiplying a non-zero random vector to the output of the bGT protocol. According to Theorem 1, if the counting $\mu_x < \mathcal{T}$ then $\boldsymbol{\gamma}^*(x)$ distributes uniformly over $(\mathbb{Z}_t/\{0\})^n$. Otherwise, $\boldsymbol{\gamma}^*(x)$ contains one and only one 0 whose position also distributes uniformly. But this is precisely the same distribution from which $\mathcal{S}$ samples the $\hat{\boldsymbol{\gamma}}^*$. Thereby we have $\boldsymbol{\gamma}^* \approx^c \hat{\boldsymbol{\gamma}}^*$. Finally, it is easy to see that the distribution of $\boldsymbol{\gamma}'$ is an uniform distribution over $\mathbb{Z}_t^{n\Sigma}$ since we just add an uniform randomness $\boldsymbol{\delta}$ to $\boldsymbol{\gamma}$. Since $\mathcal{S}$ uses the same distribution of randomness $\hat{\boldsymbol{\delta}}$, we have $\boldsymbol{\gamma}' \approx^c \hat{\boldsymbol{\gamma}}'$.

It is oblivious to see that the distributions of $\boldsymbol{\mu}'$ and $\boldsymbol{\gamma}^*$ are independent and distributions of $\boldsymbol{\gamma}^*$ and $\boldsymbol{\gamma}'$ are independent since independent randomness are used. We survey the independence between the distribution of $\boldsymbol{\mu}'$ and that of $\boldsymbol{\gamma}'$. $\boldsymbol{\gamma}'$ can be seen as the sum of two random vectors. One is from the bGT protocol, and the other is $\boldsymbol{\delta}$. Since the randomness used in the bGT protocol is independent with $\boldsymbol{\delta}$, we have $\boldsymbol{\gamma}'$ is independent with $\boldsymbol{\mu}'$. Consequently, we have these three components independent with each other. To conclude, we have $\{\boldsymbol{\mu}', \boldsymbol{\gamma}^*, \boldsymbol{\gamma}'\} \approx^c \{\hat{\boldsymbol{\mu}}', \hat{\boldsymbol{\gamma}}^*, \hat{\boldsymbol{\gamma}}'\}$. So the view $\mathcal{S}$ simulates for $\mathcal{A}$ is computationally indistinguishable from the view $\mathcal{A}$ gets in the real execution; security follows. $\square$