Privacy-Preserving Cloud-Based Statistical Analyses on Sensitive Categorical Data

Sara Ricci, Josep Domingo-Ferrer, and David Sánchez

UNESCO Chair in Data privacy
Department of Computer Science and Mathematics
Universitat Rovira i Virgili
Av. Països Catalans 26
43007 Tarragona, Catalonia
{sara.ricci,josep.domingo,david.sanchez}@urv.cat

Abstract. We consider the problem of privacy-preserving cloud-based statistical computation on sensitive categorical data. Specifically, we focus on protocols to obtain the contingency matrix and the sample covariance matrix of the categorical data set. A multi-cloud is used not only to store the sensitive data but also to perform computations on them. However, the multi-cloud is semi-honest, that is, it follows the protocols but is not authorized to learn the sensitive data. Hence, the data must be stored and computed on by the multi-cloud in a privacy-preserving format, which we choose to be vertical splitting among the various clouds. We give a comparison of our proposals, based on the secure scalar product, against a benchmark protocol consisting of downloading plus local computation.
Keywords: Data splitting, privacy, categorical data, cloud computing, contingency tables, distance covariance.

1 Introduction

Data have become a crucial asset of many enterprises, organizations and public administrations. Collecting and analyzing large amounts of data related to individuals does not only improve research, but it also drives a tremendous business [21]. However, local storage and processing of such *big* data is often unfeasible for the data controllers because of the associated costs (software, hardware, energy, maintenance). The cloud offers a suitable alternative for these data-intensive scenarios, by providing large and highly scalable storage/computation resources at a low cost and with ubiquitous access. However, most controllers holding (potentially) sensitive data are reluctant to embrace the cloud because of security and privacy concerns regarding the cloud service provider (CSP) [3]. On the one hand, CSPs may read, use or even sell the data outsourced by their customers (especially those CSPs that offer their services for free expecting to monetize users' data). On the other hand, CSPs may suffer attacks, accidents or data leakages that may compromise the privacy of the subjects to whom the outsourced data refer.

To allay these issues and win the trust of potential customers in cloud computing, there is a need for secure, efficient and privacy-preserving storage *and processing* methods for the (sensitive) data outsourced to the cloud. This is precisely the main goal of the European project CLARUS [6] in which the current work is framed. CLARUS consists in a proxy located in a domain trusted by the data controller (e.g., a server in her company's intranet or a plug-in in her device) that implements security and privacy-enabling features towards the CSP so that i) the CSP only receives privacy-protected versions of the controller's (or the controller's users') data, ii) CLARUS makes the access to such data transparent to the controller's users (by adapting their queries and reconstructing the results retrieved from the cloud) and iii) it remains possible for the users to leverage the cloud to perform accurate computations on the outsourced data without downloading them.

To do so, CLARUS particularly relies on *data splitting* as a data protection technique: data are partitioned into several fragments, each of which is stored in the *clear* in a cloud provided by a different CSP, see [1] and [5]. Data splitting is an alternative that is more efficient and functionality-preserving than encryption-based methods (e.g., CipherCloud, PerspecSys, SecureCloud, etc.). In general, even though searchable and homomorphic encryption allow performing some operations on ciphertext [10], computing on encrypted data is extremely limited and costly [15], and it requires careful management of encryption keys. In contrast, the *vertical* data splitting implemented by CLARUS protects privacy (confidential information on an individual is partitioned into fragments that cannot be linked) and allows computation to be performed on clear data.

Yet, computing on split/distributed data is not easy. In fact, [23] acknowledge that mining data from distributed sources remains a challenge and [24] identify correlating the data from the various sources as the main hurdle. In this context, *we assume the CSPs to be semi-honest*: they are not entitled to see the entire data set, but they neither deviate from the protocols nor collude to aggregate the data fragments they hold.

Contribution and plan of this paper

In [4], we evaluated several non-cryptographic proposals for statistical computation (basically correlations) on split data, and we enhanced and proposed some protocols adapted to the CLARUS scenario. All these protocols and methods were designed for numerical data. However, many of the (personal) data currently collected from a variety of sources (social networks, surveys, B2C transactions, etc.) are not numerical (for example, user profiles [22], health records [13] or transaction logs [21]).

In this paper, we adapt some of the methods proposed for split numerical data to categorical data. Specifically, we describe two protocols (with and without cryptography, respectively) to compute the contingency table and the sample covariance matrix needed to measure the correlation between two categorical attributes stored in different clouds. We also compare their computational and communication costs against a benchmark consisting of the CLARUS proxy

downloading the entire data set and locally computing on the downloaded data set.

The rest of this paper is organized as follows. In Section 2, we review some methods to measure the statistical dependence of categorical attributes. Section 3 focuses on the statistical analysis on vertically partitioned data; we give a non-cryptographic protocol and a cryptographic protocol for computing contingency tables and the sample covariance matrix. In Section 4, we compare the computational and communication costs of the protocols described in Section 3; a benchmark protocol is taken that consists of the CLARUS proxy downloading the entire data set and computing locally on the downloaded data. Finally, Section 5 lists some conclusions and future research lines.

2 Statistical dependence analyses on categorical data

Numerical data (continuous or ordinal) are easy to analyze: correlations, covariances, regressions and classifications can be computed using the standard arithmetic operators. In contrast, analyzing categorical data, consisting of categorical values lacking a total order, is more difficult: they must be either mapped to numbers in some way [8] or they must be processed using methods specifically designed to measure their differences [17], analyze their distributions [18] or estimate their dependence [20]. A well-known, albeit simple, procedure to measure the statistical dependence between two categorical attributes is the χ^2-test of independence [2]. This test uses the contingency tables associated to the categorical attributes as the input for a linear regression analysis. Even if it can measure some degree of statistical dependence, the χ^2-test only considers the similarities between the distributions of categorical labels, but it fails to capture the semantic similarity between the categories themselves.

In [20], a more recent and accurate way to measure the dependence between two categorical attributes is proposed: the *distance covariance/correlation* measure, meant to be an alternative to the standard numerical covariance/correlation. The numerical covariance requires the values of attributes to be totally ordered, and it measures dependence by checking whether greater values of one attribute correspond to greater values of the other attribute, and smaller values to smaller values. This does not work for non-ordinal categorical attributes (i.e., nominal/textual), that lack total order. The *distance covariance* is a viable alternative that quantifies to what extent the two attributes are independently dispersed, where dispersion is measured according to the pairwise distances between all pairs of values of each attribute. Moreover, unlike statistical tests based on contingency tables (i.e., value distributions), pairwise distances can capture the *semantics* inherent to categorical values, which is crucial to properly measure the correlation of non-numerical data [16]. To do so, the pairwise distance can be calculated using similarity/distance measures [17], that quantify how similar are the meanings of the concepts associated to the categorical values, based on the semantic evidences gathered from one or several knowledge sources (e.g., ontologies, corpora).

3 Computation on vertically partitioned data

When storing dynamically changing sensitive data in the cloud, vertical splitting is very convenient: additions/updates are fast, because the other records (those that do not change) do not need to be modified. On the contrary, if data stored in the cloud are masked rather than split, any record addition/update requires re-anonymizing the original data set including the added/updated record and re-uploading the entire re-anonymized data set. Furthermore, if fragments are stored at different CSPs and these do not collude, splitting is more privacy-preserving than masking for dynamic data, because in masking the (single) CSP might infer the value of some original records by comparing the successive anonymized versions of the data set.

In vertical splitting, analyses that involve single attributes (e.g., mean, variance) or attributes within a single fragment are fast and easy to compute: the cloud storing the fragment can compute and send the output of the analysis to the CLARUS proxy. However, statistical dependence analyses may involve attributes stored in different fragments, and thus communication between several clouds. In [4] we focused on computing the sample covariance matrix because many of the statistical dependence analyses on numerical data are based on it. Obtaining the sample covariance matrix in vertical splitting among several clouds can be decomposed into several secure scalar products to be conducted between pairs of clouds. Secure scalar products can be based on cryptography (the protocol in [11] involves homomorphic encryption), or not ([7], [12] modify the data before sharing them in such a way that the original data cannot be deduced from the shared data but the final results are preserved).

As discussed in Section 2, for categorical data the problem is more complicated. We focus here on the computation of the contingency tables needed for the χ^2-test and of the *distance covariance* matrix needed to measure the statistical dependence between categorical attributes. Section 3.1 reviews the best two computation protocols on split numerical data, one that uses cryptography and another that does not, adapted to the multi-cloud scenario considered in CLARUS [4]. In Section 3.2 and 3.3, we modify these protocols to compute contingency tables and the distance covariance matrix for categorical data.

3.1 Secure scalar product

Let $\mathbf{x}$ and $\mathbf{y}$ be two n-component vectors, respectively owned by Alice and Bob (who can be two CSPs). The goal is to securely compute the product $\mathbf{x}^T\mathbf{y}$. Shannon wrote in [19]: *"It is shown that perfect secrecy is possible, but requires, if the number of messages is finite, the same number of possible keys."* The privacy of the following protocol relies on the fact that the original vectors $\mathbf{x}$ and $\mathbf{y}$ are not shared at any time; only linear transformations of them are, such that the number of unknowns (randomness) added by the transformations is greater than or equal to the number of private unknowns.

In [9], the protocol is based on what they call a commodity server. Let Alice and Bob be as previously defined and let a third, non-colluding cloud Charlie play the role of the commodity server. In [4], we suggested the following variant:

Protocol 1

i. Charlie generates two random n-vectors $\mathbf{r_x}$ and $\mathbf{r_y}$ and computes $p = \mathbf{r_x}^T\mathbf{r_y}$ (note that p is a number).
ii. Charlie sends $\mathbf{r_x}$ to Alice, $\mathbf{r_y}$ to Bob (or equivalently sends them the seeds for a common random generator). Also, Charlie sends p to CLARUS.
iii. Alice computes $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{r_x}$ and sends it to Bob.
iv. Bob sends $t = \hat{\mathbf{x}}^T\mathbf{y}$ to CLARUS and sends $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{r_y}$ to Alice.
v. Alice computes $s_x = \mathbf{r_x}^T\hat{\mathbf{y}}$ and sends it to CLARUS.
vi. CLARUS computes $t - s_x + p = (\mathbf{x} + \mathbf{r_x})^T\mathbf{y} - \mathbf{r_x}^T(\mathbf{y} + \mathbf{r_y}) + \mathbf{r}_x^T\mathbf{r_y} = \mathbf{x}^T\mathbf{y}$.

Security. Charlie receives nothing from the other clouds. Bob gets n linear equations with n degrees of randomness. Similarly, Alice gets n linear equations with n degrees of randomness. Therefore, neither Alice's vector $\mathbf{x}$ can be computed by Bob, nor Bob's vector $\mathbf{y}$ can be computed by Alice and they are both protected according to the aforementioned Shannon's principle.

In [11], the authors propose and justify the security of a cryptographic protocol based on the Paillier homomorphic cryptosystem [14]. The following variant is proposed in [4]:

Protocol 2

Set-up phase:
i. Alice generates a private and public key pair (s_k, p_k) and sends p_k to Bob.
Scalar product of Alice's $\mathbf{x} = (x_1, \ldots, x_n)^T$ and Bob's $\mathbf{y} = (y_1, \ldots, y_n)^T$:
ii. Alice generates the ciphertexts $c_i = Enc_{p_k}(x_i; r_i)$, where r_i is a random number in $\mathbb{F}_N$, for every $i = 1, \ldots, n$, and sends them to Bob.
iii. Bob computes $\omega = \prod_{i=1}^{n} c_i^{y_i}$.
iv. Bob generates a random plaintext s_B, a random number r', sends $\omega' = \omega Enc_{p_k}(-s_B; r')$ to Alice and sends s_B to CLARUS.
v. Alice sends $s_A = Dec_{s_k}(\omega') = \mathbf{x}^T\mathbf{y} - s_B$ to CLARUS.
vi. CLARUS computes $s_A + s_B = \mathbf{x}^T\mathbf{y}$.

Protocol 2 works in a finite field $\mathbb{F}_N$, where the order N is the product of two primes p and q of the same length and such that $\gcd(pq, (p-1)(q-1)) = 1$. In case Alice and Bob need to execute this protocol several times, they can reuse public and private keys and thus the set-up step (first step) needs to be executed only once. The complexity of all these operations depends on N: the larger N, the more computationally demanding they are. Since we are computing $\mathbf{x}^T\mathbf{y}$ mod N, if we do not want the result to be modified by the modulus, it must hold that $N > \mathbf{x}^T\mathbf{y}$. Let $M_\mathbf{x} = \max_{x_i \in \mathbf{x}} x_i$ and $M_\mathbf{y} = \max_{y_i \in \mathbf{y}} y_i$. It is sufficient to choose $N > nM_\mathbf{x}M_\mathbf{y}$.

Security. The only modification with respect to the Protocol in [11] is that Alice and Bob do not share their results s_A and s_B, but they send these values to CLARUS. Hence, the security of the protocol is preserved and follows from the security of the Paillier cryptosystem (see [14] for more details).

3.2 Contingency table computation

A contingency table (or cross-classification table) is a type of table containing the (multivariate) frequency distributions of the categorical attributes. Let $\mathbf{a}$ and $\mathbf{b}$ denote two categorical attributes, $\mathbf{a}$ with h categories $c_1(\mathbf{a}), \ldots, c_h(\mathbf{a})$ and $\mathbf{b}$ with k categories $c_1(\mathbf{b}), \ldots, c_k(\mathbf{b})$. The contingency table has h rows and k columns displaying the sample frequency counts of the $h \times k$ category combinations.

To obtain the contingency table in vertical splitting among several clouds, one just needs to compute the table cells. Let $(a_1, \ldots, a_n)^T$ and $(b_1, \ldots, b_n)^T$ be the vectors of values from the categorical attributes $\mathbf{a}$ and $\mathbf{b}$, owned by Alice and Bob (who can be two CSPs), respectively. A cell C_{ij} (for every $i = 1, \ldots, h$ and $j = 1, \ldots, k$) is computed by counting the number of records in the original data set containing both the categories $c_i(\mathbf{a})$ and $c_j(\mathbf{b})$. Alice creates a new vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ such that

$$x_l = 1 \text{ if } a_l = c_i(\mathbf{a}), \text{ and } x_l = 0 \text{ otherwise, for } l = 1, \ldots, n. \quad (1)$$

Bob creates $\mathbf{y} = (y_1, \ldots, y_n)^T$ such that

$$y_l = 1 \text{ if } b_l = c_j(\mathbf{b}), \text{ and } y_l = 0 \text{ otherwise, for } l = 1, \ldots, n. \quad (2)$$

The scalar product $\mathbf{x}^T\mathbf{y}$ gives the number C_{ij} of records in the original data set containing both the categories $c_i(\mathbf{a})$ and $c_j(\mathbf{b})$. Hence, Alice and Bob can use Protocol 1 or Protocol 2 to securely compute C_{ij} by just adding two preliminary steps to the scalar product computation part: one step by Alice to generate $\mathbf{x}$ from $(a_1, \ldots, a_n)^T$ using Expression (1), and another step by Bob to generate $\mathbf{y}$ from $(b_1, \ldots, b_n)^T$ using Expression (2).

Security. The only modification with respect to the Protocols 1 and 2 is that Alice and Bob compute $\mathbf{x}$ and $\mathbf{y}$, respectively. These computations are done by the clouds without exchange of information, hence the security of the protocol is preserved.

3.3 Distance covariance matrix computation

As explained in Section 2, first of all we need to measure the pairwise (semantic) distance between the categorical values of each attribute; see [17] for a survey of semantic distance measures. Let $\mathbf{x^1} = (x_1^1, \ldots, x_n^1)^T$ and $\mathbf{x^2} = (x_1^2, \ldots, x_n^2)^T$ be vectors of values of two categorical attributes owned by Alice and Bob, respectively. We compute the matrices $X^1 = [x_{ij}^1]_{i,j \leq n}$ and $X^2 = [x_{ij}^2]_{i,j \leq n}$ where

$x^1_{ij} = |x^1_i - x^1_j|$ and $x^2_{ij} = |x^2_i - x^2_j|$ are the semantic distances between two values of the same attribute $\mathbf{x^1}$ and $\mathbf{x^2}$, respectively. We define

$$X^1_{kl} = x^1_{kl} - \bar{x}^1_{k\cdot} - \bar{x}^1_{\cdot l} + \bar{x}^1_{\cdot\cdot} \quad \text{for } k, l = 1, \ldots, n, \tag{3}$$

where

$$\bar{x}^1_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} x^1_{kl}, \quad \bar{x}^1_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} x^1_{kl}, \quad \bar{x}^1_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} x^1_{kl}.$$

Similarly, we define $X^2_{kl} = x^2_{kl} - \bar{x}^2_{k\cdot} - \bar{x}^2_{\cdot l} + \bar{x}^2_{\cdot\cdot}$ for $k, l = 1, \ldots, n$.

Definition 1. *The squared* sample distance covariance *is obtained as the arithmetic average of the products* $X^1_{kl}X^2_{kl}$*:*

$$d\mathcal{V}^2_n(\mathbf{x^1}, \mathbf{x^2}) = \frac{1}{n^2}\sum_{k,l=1}^{n} X^1_{kl}X^2_{kl} \tag{4}$$

and the squared sample distance variance *is obtained as*

$$d\mathcal{V}^2_n(\mathbf{x^1}) = d\mathcal{V}^2_n(\mathbf{x^1}, \mathbf{x^1}) = \frac{1}{n^2}\sum_{k,l=1}^{n} X^1_{kl}X^1_{kl}.$$

See [20] for details and justification on the above definition. In general, if $\mathbf{X} = (\mathbf{x}^1, \ldots, \mathbf{x}^m)$ is a data set with m attributes $\mathbf{x}^j$, $j = 1, \ldots, m$, the distance covariance matrix $\hat{\mathbf{\Sigma}}$ of $\mathbf{X}$ is

$$\hat{\mathbf{\Sigma}} = \left(\begin{array}{c|c|c|c} d\mathcal{V}_n(\mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^1, \mathbf{x}^2) & \cdots & d\mathcal{V}_n(\mathbf{x}^1, \mathbf{x}^m) \\ \hline d\mathcal{V}_n(\mathbf{x}^2, \mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^2) & \cdots & d\mathcal{V}_n(\mathbf{x}^2, \mathbf{x}^m) \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline d\mathcal{V}_n(\mathbf{x}^m, \mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^m, \mathbf{x}^2) & \cdots & d\mathcal{V}_n(\mathbf{x}^m) \end{array}\right).$$

Note that $d\mathcal{V}_n(\mathbf{x}^i, \mathbf{x}^j)$ is the square root of the number $d\mathcal{V}^2_n(\mathbf{x}^i, \mathbf{x}^j)$, for $i, j = 1, \ldots, m$, and that X^j, X^j_{kl}, $d\mathcal{V}_n(\mathbf{x^j})$, for $j = 1, \ldots, m$, are separately computed by the cloud storing the respective attribute. The most challenging task is therefore calculating the squared sample distance covariance, i.e. Expression (4), which requires performing a secure scalar product of n vectors, each held by two different parties (where "secure" means without any party disclosing her vector to the other party). In fact, calling $\mathbf{X^1_k} = (X^1_{k1}, \ldots, X^1_{kn})$ and $\mathbf{X^2_k} = (X^2_{k1}, \ldots, X^2_{kn})$ for $k = 1, \ldots, n$, we can rewrite Expression (4) as

$$d\mathcal{V}^2_n(\mathbf{x^1}, \mathbf{x^2}) = \frac{1}{n^2}\sum_{k=1}^{n}\left(\sum_{l=1}^{n} X^1_{kl}X^2_{kl}\right) = \frac{1}{n^2}\sum_{k=1}^{n}(\mathbf{X^1_k})^T\mathbf{X^2_k}, \tag{5}$$

where the n scalar products are $(\mathbf{X^1_k})^T\mathbf{X^2_k}$ for $k = 1, \ldots, n$. Therefore, obtaining the distance covariance matrix in vertical splitting among several clouds can be decomposed into several secure scalar products to be conducted between pairs of clouds. Protocols 1 and 2 are perfectly suited to compute $(\mathbf{X^1_k})^T\mathbf{X^2_k}$ for $k = 1, \ldots, n$ (see Section 3.1). The only adaptation needed is to add two preliminary steps: one step by Alice to compute $\mathbf{x} = \mathbf{X^1_k}$ from $\mathbf{x^1}$, an another step by Bob to compute $\mathbf{y} = \mathbf{X^2_k}$ from $\mathbf{x^2}$.

Security. The two preliminary steps added before the secure scalar product are done separately by Alice and Bob, so there is no additional exchange of information between the clouds. Hence the security of Protocols 1 and 2 is preserved.

4 Comparison among methods

We compare here the performance of the methods presented in Sections 3.2 and 3.3 with the following benchmark solution:

Protocol 3

Set-up phase:

i. CLARUS encrypts the original data set $\mathbf{E} = Enc(\mathbf{X})$.

ii. CLARUS sends $\mathbf{E}$ *to a cloud Alice for storage.*

Computation phase:

iii. CLARUS downloads $\mathbf{E}$ *from Alice, decrypts* $\mathbf{X} = Dec(\mathbf{E})$ *and performs the desired computation.*

Encryption and decryption can be performed using a fast symmetric cryptosystem, such as the Advanced Encryption Standard (AES), which takes time linear in the number of records/vector components n, as well as ciphertexts similar in size to the corresponding plaintexts.

For simplicity, we consider a data set $\mathbf{X}$ with two attributes owned by Alice and Bob, respectively (the generalization to more attributes and clouds is straightforward). We now evaluate the computational cost for Alice, Bob, CLARUS and the total computation under each protocol. Just giving the order of magnitude of the complexity is not accurate enough (e.g. n additions are faster than n multiplications, even if we have $O(n)$ computation in both cases); therefore, we give the complexity in terms of the costliest operation performed in each case. For instance, "read" means reading the vector, "AESdecr" means AES decryption of the vectors and "RNDgen" is the random key generation for the AES encryption. Moreover, operations that do not need to be repeated each time the protocol is executed, e.g. the generation of cryptographic keys in Protocol 2, are separately counted as set-up costs. Assuming that the clouds have unlimited storage, it is reasonable to assume as well that those matrices or vectors need to be generated only once and can be stored for subsequent reuse. In contrast, we do not assume unlimited storage at CLARUS; therefore, we assume the proxy just stores the random seeds and generates the matrices when needed. Also, we have associated the communication cost with the sender and we use a parameter γ to represent the maximum length of the numbers in the vectors and matrices used in the protocols. For the case of Protocol 2, lengths are a function of the size N of the field used by the Paillier cryptosystem: the public key is $3\log_2 N$ bits long, the secret key is $\log_2 N$ bits long, ciphertexts are $2\log_2 N$ bits long and plaintexts are $\log_2 N$ bits long. We consider that whenever possible the participants send the seeds of random vectors and matrices, rather than the vectors and matrices themselves.

4.1 Comparison for contingency table computation

In Section 3.2, we adapted Protocols 1 and 2 to compute one cell value of the contingency table. We compare here these adaptations with Protocol 3, where, in step (iii), "desired computation" means "count the number of records in $\mathbf{X}$ containing both the categories $c_i(\mathbf{a})$ and $c_j(\mathbf{b})$, where $\mathbf{a}$ and $\mathbf{b}$ are the two attributes in $\mathbf{X}$".

Table 1. Long-term and temporary storage costs of Protocols 1, 2 and 3. n is the number of records/vector components; γ represents the maximum length of the numbers in the vectors and matrices used in the protocols; N is the size of the plaintext field used by Paillier. Note: In protocols not requiring the presence of Bob or Charlie, their costs are indicated with "$-$".

	Storage							
	Long-term				Temporary			
	Alice	Bob	Charlie	CLARUS	Alice	Bob	Charlie	CLARUS
Prot. 1	$(n+1)\gamma$	$(n+1)\gamma$	0	0	$(3n+1)\gamma$	$(3n+1)\gamma$	$(2n+3)\gamma$	3γ
Prot. 2	$n\gamma+$ $4\log_2 N$	$n\gamma$	$-$	0	$(3n+2)\gamma$	$(2n+4)\gamma$	$-$	3γ
Prot. 3	$2n\gamma$	$-$	$-$	$2n\gamma$	0	$-$	$-$	$(4n+1)\gamma$

Only Protocol 2 and Protocol 3 require a set-up phase. When the stored data are updated (that is, records are added, changed or removed), the set-up phase (key generation) of Protocol 2 does not need to be repeated: since $\mathbf{x}$ and $\mathbf{y}$ are binary vectors, we can fix a small order $N > 2$ of the finite field in use (see Section 3.1). Protocol 3 requires a set-up phase, that is, the key generation for the AES cryptosystem and the encryption of the private vectors. Compared to Protocol 2, the set-up phase of Protocol 3 needs to be repeated every time that the private vectors are changed. Only Protocol 2 and Protocol 3 present a communication cost of the set-up phase, due to the exchange of the public key for the two former protocols and the transmittal of the encrypted private vectors for the latter one.

Table 1 shows the long-term and temporary data storage costs (temporary storage is the one needed only to conduct a certain calculation at some point). In Protocol 1, Alice and Bob need long-term storage for $(a_1, \ldots, a_n)^T$ and $(b_1, \ldots, b_n)^T$, respectively, and for the seed to create the random vector. Charlie needs only temporary storage for $\mathbf{r_x}$, $\mathbf{r_y}$, their seeds and their product p. Alice and Bob also need temporary storage to share $(\mathbf{x}, \mathbf{r_x}, \hat{\mathbf{x}})$ and $(\mathbf{y}, \mathbf{r_y}, \hat{\mathbf{y}})$ and create s_x and t, respectively. In Protocol 2, long-term storage is required by Alice and Bob to store their respective private vectors, as well as to store the public-private key of the Paillier cryptosystem; Alice needs $4\log_2 N$ space for the key pair. The temporary storage depends on the computations as before. All data splitting protocols need the same long-term storage space. On the other hand, only the benchmark Protocol 3 requires CLARUS to store a large amount of data, namely the decrypted data. Note that Protocol 3's long-term storage is $2n\gamma$ as both n-vectors $\mathbf{a}$ and $\mathbf{b}$ are stored.

Table 2. Execution costs for Protocols 1, 2 and 3. n is the number of records/vector components; γ represents the maximum length of the numbers in the vectors and matrices used in the protocols. Charlie only appears in Protocol 1 and Bob does not appear in Protocol 3; we indicate the absence of a cloud with "−". The computation cost is presented in terms of the costliest operations performed in each case; the communication cost is the exact amount of transmitted data.

	Computational cost				Communication cost				Crypt.
	Alice	Bob	CLARUS	Charlie	Alice	Bob	CLARUS	Charlie	package
Pr.1	n prod. + n read	n prod. + n read	2 sum.	$2n$ RNDgen.	$(n+1)\gamma$	$(n+1)\gamma$	0	3γ	none
Pr.2	n RNDgen. + n encr. + n read	n prod. + n read	1 sum.	−	$(n+1)\gamma$	2γ	0	−	Alice
Pr.3	0	−	n read + n AESdecr.	−	$2n\gamma$	−	0	−	CLARUS

Table 2 shows the computational and communication costs incurred by the execution of the above mentioned protocols (after set-up). In Protocol 1, the commodity server Charlie computes the scalar product of the two random vectors $(\mathbf{r_x}, \mathbf{r_y})$, stores the result and sends the seeds of $\mathbf{r_x}$ and $\mathbf{r_y}$ to Alice and Bob. Protocols 1 and 2 require reading vectors $(a_1, \ldots, a_n)^T$ and $(b_1, \ldots, b_n)^T$ to compute $\mathbf{x}$ and $\mathbf{y}$. Protocols 1 and 2 have all similar costs for CLARUS, but Protocol 2 needs encryption. Therefore, all in all, Protocol 1 seems to be the most advantageous one. Nevertheless, if a cryptographic package is available, Protocol 2 is suitable and fast.

4.2 Comparison for the distance covariance matrix computation

Section 3.3 showed that the distance covariance matrix calculation can be decomposed into computing several secure scalar products, each of which is conducted between a pair of clouds. Therefore, we focus on the secure scalar product between two vectors $\mathbf{x} = \mathbf{X_k^1}$ and $\mathbf{y} = \mathbf{X_k^2}$ computed by Alice and Bob, respectively (see Expression (5)). We compare here these adaptations with Protocol 3, where, in Step (iii), "desired computation" means "compute $\mathbf{x}^T\mathbf{y}$".

Note that all the computations to obtain $\mathbf{x}$ and $\mathbf{y}$ are performed by Alice and Bob in all the protocols except for Protocol 3, where they are left to CLARUS. Compared to the protocols in Section 4.1, these protocols differ only in the creation of $\mathbf{x}$ and $\mathbf{y}$. Therefore, in Table 1 the long-term storage remains unchanged, but the temporary storage requires a supplement of storage related to Equation (3). In Table 2, showing the execution costs, the sentences "n read" needs to be changed by the complexity of the $\mathbf{x}$ and $\mathbf{y}$ computations: "$(n^2 + 3n)$ sums + semantic distance complexity".

Like in Section 4.1, Protocols 1 and 2 present all similar costs for CLARUS, but Protocol 2 needs encryption. Therefore, all in all, Protocol 1 seems to be the most advantageous one. Nevertheless, if a cryptographic package is available, Protocol 2 is suitable and fast.

5 Conclusions and future work

We have presented two protocols, a cryptographic one and a non-cryptographic one, that allow statistical computation on protected sensitive categorical data stored in semi-honest clouds. For the sake of flexibility and efficiency, we have considered data splitting as a non-cryptographic method for data protection, rather than the heavier fully homomorphic encryption. We have provided complexity analyses and benchmarking for all proposed protocols, in order to show their computational advantages for the cloud user. In this way, clouds are not only used to store sensitive data, but also to perform computations on them in a privacy-aware manner. This is especially interesting for large sensitive data sets.

In future research, it would be interesting to design protocols for computations other than contingency tables and distance covariance matrices. Also, here we have considered only categorical attributes and in [4] we considered only numerical attributes. Dealing with cloud-stored sensitive data sets containing both types of attributes and using semi-honest clouds to perform computations involving both attribute types would be highly relevant.

Acknowledgments and disclaimer

Partial support to this work has been received from the European Commission (projects H2020-644024 "CLARUS" and H2020-700540 "CANVAS"), from the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer and grant 2014 SGR 537), and from the Spanish Government (project TIN2014-57364-C2-1-R "SmartGlacis" and TIN 2015-70054-REDC). The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are the authors' own and are not necessarily shared by UNESCO.

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In CIDR 2005, pp. 186–199 (2005).
2. Agresti, A., Kateri, M.: Categorical Data Analysis. Springer (2011).
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Communications of the ACM, vol. 53, no. 4, pp. 50–58 (2010).
4. Calviño, A., Ricci, S., Domingo-Ferrer, J.: Privacy-preserving distributed statistical computation to a semi-honest multi-cloud. In IEEE Conf. on Communications and Network Security (CNS 2015). IEEE (2015).
5. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Selective data outsourcing for enforcing privacy. Journal of Computer Security, vol. 19, no. 3, pp. 531–566 (2011).
6. CLARUS - A Framework for User Centred Privacy and Security in the Cloud, H2020 project (2015-2017). `http://www.clarussecure.eu`

7. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.: Tools for privacy preserving distributed data mining. ACM SiGKDD Explorations Newsletter, vol. 4, no. 2, pp. 28–34 (2002).
8. Domingo-Ferrer, J., Sánchez, D., Rufian-Torrell, G.: Anonymization of nominal data based on semantic marginality. Information Sciences, vol. 242, pp. 35–48 (2013).
9. Du, W., Han, Y., Chen, S.: Privacy-preserving multivariate statistical analysis: linear regression and classification. In SDM, vol. 4. SIAM, pp. 222–233 (2004).
10. Dubovitskaya, A., Urovi, V., Vasirani, M., Aberer, K., Schumacher, M.: A Cloud-Based eHealth Architecture for Privacy Preserving Data Integration. In ICT Systems Security and Privacy Protection. Springer, pp. 585–598 (2015).
11. Goethals, B., Laur, S. , Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In Information Security and Cryptology - ICISC 2004, Lecture Notes in Computer Science, vol. 3506, pp. 104–120, Springer (2005).
12. Karr, A., Lin, X., Sanil, A., Reiter, J.: Privacy-preserving analysis of vertically partitioned data using secure matrix products Journal of Official Statistics, vol. 25, no. 1, p. 125 (2009).
13. Martínez, S., Sánchez, Valls, A.: A semantic framework to protect the privacy of electronic health records with non-numerical attributes. Journal of Biomedical Informatics, vol. 46, no. 2, pp. 294-303 (2013).
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cryptology — EUROCRYPT '99, ser. Lecture Notes in Computer Science, vol. 1592, pp. 223–238, Springer (1999).
15. Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. IEEE Internet Computing, no. 1, pp. 69–73 (2012).
16. Rodríguez-García, M., Batet, M., Sánchez, D.: Semantic noise: privacy-protection of nominal microdata through uncorrelated noise addition. In 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1106–1113 (2015).
17. Sánchez, D., Batet, M., Isern, D., Valls, A.: Ontology-based semantic similarity: A new feature-based approach. Expert Systems with Applications 39.9: 7718-7728 (2012).
18. Sánchez, D., Batet, M., Martínez, S., Domingo-Ferrer, J.: Semantic variance: An intuitive measure for ontology accuracy evaluation. Engineering Applications of Artificial Intelligence, vol. 39, pp. 89–99 (2015).
19. Shannon, C.: Communication theory of secrecy systems. Bell System Technical Journal, vol. 28, pp. 656–715 (1949).
20. Székely, G.J., Rizzo, M.L.: Brownian distance covariance. The Annals of Applied Statistics 3.4: 1236-1265 (2009).
21. U.S. Federal Trade Commission: Data Brokers, A Call for Transparency and Accountability (2014).
22. Viejo, A., Sánchez, D., Castellà-Roca, J.: Preventing automatic user profiling in Web 2.0 applications. Knowledge-based Systems, vol. 36, pp. 191-205 (2012).
23. Weiss, G.: Data mining in the real world: experiences, challenges, and recommendations. In DMIN, pp. 124–130 (2009).
24. Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making, vol. 5, no. 4, pp. 597–604 (2006).