Privacy-Preserving Distribution of Statistical Computation to a Semi-Honest Multi-Cloud

Aida Calviño, Sara Ricci, Josep Domingo-Ferrer
UNESCO Chair in Data Privacy, Dept. of Comp. Eng. and Maths
Universitat Rovira i Virgili, Tarragona
{aida.calvino,sara.ricci,josep.domingo}@urv.cat

***Abstract*—We present the problem of privacy-preserving distributed statistical computing (PPDSC) in which one party vertically splits a data set among a set of honest-but-curious clouds and wishes to use the clouds' processing power to perform statistical computation on the overall data set. The cornerstone is to compute covariances and, more specifically, scalar products. Existing protocols for computing scalar products on split data are identified and compared, and new variants specifically designed for PPDSC are presented that improve privacy and performance.**

Keywords*—*Privcy, data splitting, covariance matrix, cloud computing, scalar product.

I. Introduction and motivation

Privacy-preserving distributed statistical computing (PPDSC) is about running statistical and data mining analyses on a data set partitioned into several fragments stored at different locations (e.g. clouds in a multi-cloud), without reassembling the various fragments.

Our work in this new problem is motivated by the European project CLARUS [1]. This project is developing a secure framework for storing and *processing* sensitive data outsourced to the cloud so that end users can monitor, audit and control their stored data while benefiting from the computational power and functionality of the cloud [2]. To achieve this, the CLARUS solution is envisioned as a proxy located in a domain trusted by the end user (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 cloud service provider. In what follows, we will simply write CLARUS instead of CLARUS proxy unless there is risk of confusion.

Under this setting, we propose to split the data and store them in different clouds (each provided by a different cloud service provider (CSP)) or in different cloud accounts with the same CSP. Data splitting has long been used as a privacy-preserving technique (see [3] for early proposals or the recent [4] for an application to storing sensitive healthcare data in the cloud). Underlying assumptions are:

- Splitting is performed in such a way that data fragments are less disclosive than the entire data, even if fragments are stored in the clear: for example, just learning the values of an attribute "Diagnosis" is useless to an intruder if he cannot associate each diagnosis to the subject to whom it corresponds.
- The various clouds at which fragments are stored are honest-but-curious, but they do not collude to reconstruct the original data from the fragments. One might assume that the clouds are honest enough to refrain from colluding, but this is too strong an assumption, especially when storing fragments at various accounts with the same CSP. It is more secure to thwart collusion by withholding from the clouds the metadata specifying how fragments must be combined to obtain the entire original data.

A natural alternative to data splitting would be to encrypt the whole data set and store it in the cloud. However, encryption substantially hampers processing data in the cloud (i.e, using the cloud computing power to perform queries and compute some mathematical operations on the outsourced data). Admittedly, searchable and homomorphic encryption also allow performing some of these operations, but with substantial functionality and performance restrictions (only some operations can be performed and special methods must be used for them). Furthermore, searchable and/or homomorphically encrypted data entail important storage overheads (encrypted data take much more storage than the clear data). Another shortcoming of encryption relates to key management: CLARUS ought to locally store all keys used.

Due to the above downsides of encryption, data splitting seems an option worth considering. Yet, computing over split/distributed data is not an easy task. In fact, [5] acknowledged that mining data from distributed sources remains a challenging job and [6] identified correlating the data at the various locations as the main challenge. The literature on parallel processing for statistical computation has partly treated this topic: the way to combine partial results obtained from independent processors may

provide guidance on how to treat distributed data. On the other hand, privacy-preserving data mining (PPDM) over partitioned data can be of use too, as its main objective is to mine data owned by different parties who are willing to collaborate in order to get better results but do not want or cannot share the raw original data. In the remainder of the paper we evaluate and enhance the proposals in the literature on these topics to come up with protocols adapted to the PPDSC scenario in CLARUS.

The paper is organized as follows. In Section II, we revise some of the literature on data splitting. Section III is devoted to reviewing and comparing protocols previously proposed in the literature that might be suitable for computing over vertically partitioned data with a special focus on their suitability for the CLARUS scenario. After identifying the main weaknesses and strengths of existing protocols, in Section IV we propose some new variants enhancing privacy and performance. In Section V we compare the best protocols found in the previous sections with the trivial benchmark consisting of CLARUS downloading and reassembling the whole data set and performing the computations locally. Finally, Section VI lists some conclusions and future research lines.

II. Data splitting

Data splitting (a.k.a. data partitioning or fragmentation) is a privacy-preserving technique that splits a sensitive data set into fragments such that each fragment considered in isolation is no longer sensitive (it does not allow linking confidential information to specific individuals). Fragments are then stored at different locations to enforce their isolation. Compared to encryption, data splitting incurs much less query processing overhead (see [7]). Splitting can be horizontal (fragments consist of sets of records), vertical (fragments consist of sets of attributes) or mixed (fragments contain the values of some attributes for some records).

Horizontal splitting is not very frequent, as it is of limited use in enabling privacy-preserving decomposition, given that all the information of the same record is stored together [7]. Whenever preserving anonymity and preventing attribute disclosure is a key concern (as it is in CLARUS), horizontal partitioning is not suitable.

Vertical splitting methods were proposed in [7] and [8] to ensure confidentiality at the record level. Both methods rely on predefined rules that express risky attribute combinations. An example of risky pair is passport number and disease, whereas blood pressure and disease is a safe pair. The goal of both methods is to partition the original data set into several vertical fragments in such a way that, if some risky attributes are stored together in a fragment, some of them need to be encoded/encrypted. Different algorithms are proposed to solve the problem of determining the best decomposition in terms of minimizing the cost of the queries.

Regarding mixed splitting, in general it does not improve on vertical splitting in terms of privacy and it complicates distributed computation (no local computation on entire attributes is possible any more at the locations holding fragments). For this reason, it is not very attractive for PPDSC.

In the CLARUS setting, vertical splitting is very interesting as it permits anonymizing dynamic data in a secure (privacy-preserving) and fast manner. Indeed, each time one wants to add/update a new record, uploading can be done separately to each fragment, which preserves the anonymizing isolation inherent to splitting. Also, additions/updates are fast, because nothing needs to be done to the rest of records (those that do not change). On the contrary, if data stored in the cloud are masked for anonymity rather than split, any record addition/update is likely to require re-anonymizing the original data set with the added/updated record and re-uploading the entire re-anonymized data set. With most masking methods, no utility/privacy guarantees subsist if the anonymized data set is partially changed[1]. Re-anonymizing and re-uploading is more costly and also less privacy-preserving than data splitting, because the cloud might be able to infer the value of some original records by comparing the successive anonymized versions.

III. Computation over vertically partitioned data

For vertical splitting, we resort to the literature on privacy-preserving data mining over partitioned data. Many protocols have been proposed in this area to enable statistical analysis on data owned by different entities. The general setting is that several entities hold data on a common set of subjects, but each entity holds a different subset of attributes on those subjects. Thus, protocols for computation over vertically partitioned data can be used for privacy-preserving distributed statistical computing (with some modifications that we will later discuss).

Computation over vertically partitioned data can use cryptography or not. Cryptographic approaches may use a variety of techniques, including homomorphic encryption (see [9]). Non-cryptographic approaches are rather based on modifying 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 (e.g. [10], [11], [12]). In the CLARUS context, we are interested in protocols that imply less computational effort for CLARUS than downloading the data and directly performing the operations on the downloaded data.

[1] Although record-level independent masking might be an alternative to splitting (e.g. separately adding noise to each record), it can lead to poor protection, as it is not possible to know in advance the amount of masking noise to be added to a record to keep the data set anonymized enough for any possible additions/updates of other records.

The vast majority of protocols proposed for computing on vertically split data do without a trusted third party (with the exception of the commodity server solution of [12]). While avoiding a TTP is technically elegant, it normally takes more computation and communication and it may not be needed in CLARUS (where the proxy can be regarded as a TTP with respect to the various clouds).

Many statistical analyses, such as regression, classification, principal component analysis, etc., are based on the covariance matrix $\mathbf{\Sigma}$ of the data set. Therefore, a central issue in PPDM on partitioned data sets is to find efficient privacy-preserving protocols to compute this matrix. For this reason, we devote the next subsection to the analysis of solutions proposed for this problem.

A. Computation of the covariance matrix $\mathbf{\Sigma}$

The covariance matrix has as the i-th diagonal element the variance of the i-th attribute $\boldsymbol{x}_i$ and as (i,j)-th off-diagonal elements the covariance of attributes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. This matrix can be computed in scalar form, by separately computing all variances and covariances and filling the matrix elements, or in matrix form, using the formula $\mathbf{\Sigma} = \boldsymbol{X}^T\boldsymbol{X}$, where $\boldsymbol{X}$ is the entire original data set.

In the CLARUS setting, assume that a data set $\boldsymbol{X}$ with n records and m attributes has been vertically split into p different fragments, with the k-th fragment $\boldsymbol{X}_k$ containing m_k attributes, for $k = 1, \ldots, p$. Each fragment has subsequenty been stored in a different cloud. In this case, $\mathbf{\Sigma}$ can be computed using one of the two following approaches:

- *Scalar approach*. Each cloud computes the variances and covariances of its attributes and gives them to CLARUS. To compute the remaining $\sum_{k=1}^{p}\sum_{j=k+1}^{p} m_k m_j$ elements of the covariance matrix, it is necessary to securely compute the scalar product[2] of all possible pairs of centered attributes each from a different cloud.
- *Matrix approach*. The covariance matrix can be divided into blocks as follows. If $\boldsymbol{X}_k$ is the $n \times m_k$ data matrix stored at cloud k, for $k = 1, \ldots, p$, then,

$$\mathbf{\Sigma} = \boldsymbol{X}^T\boldsymbol{X} = (\boldsymbol{X}_1|\cdots|\boldsymbol{X}_p)^T(\boldsymbol{X}_1|\cdots|\boldsymbol{X}_p)$$
$$= \left(\begin{array}{c|c|c|c} \boldsymbol{X}_1^T\boldsymbol{X}_1 & \boldsymbol{X}_1^T\boldsymbol{X}_2 & \cdots & \boldsymbol{X}_1^T\boldsymbol{X}_p \\ \hline \boldsymbol{X}_2^T\boldsymbol{X}_1 & \boldsymbol{X}_2^T\boldsymbol{X}_2 & \cdots & \boldsymbol{X}_2^T\boldsymbol{X}_p \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \boldsymbol{X}_p^T\boldsymbol{X}_1 & \boldsymbol{X}_p^T\boldsymbol{X}_2 & \cdots & \boldsymbol{X}_p^T\boldsymbol{X}_p \end{array}\right).$$

  Note that the products $\boldsymbol{X}_k^T\boldsymbol{X}_k$, for $k = 1, \ldots, p$ (the shaded ones) can be internally computed by each cloud and thereafter be sent to CLARUS. The remaining blocks can be obtained by means of $\frac{p(p-1)}{2}$ secure matrix products, because the covariance matrix is symmetric.

[2]Recall that $Cov(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{1}{n}\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)$.

Therefore, the problem of obtaining the covariance matrix is reduced to performing either secure distributed scalar products or secure distributed matrix products. In the following subsections we will review some protocols for those ends that have been previously proposed in the literature for PPDM over vertically partitioned data.

We choose to explore non-cryptographic protocols, because we want to minimize the computation by CLARUS, and cryptographic options would require the proxy to encrypt all data before splitting them to the clouds and then decrypt the results. In fact, as mentioned in the introduction above, if one is willing to accept the computational overheads and the functional limitations of using encryption, the covariance matrix could be securely computed without data splitting.

The privacy guarantees of the non-cryptographic protocols we will review amount to each participant being unable to learn the data matrices held by the other participants (at most each participant learns some linear relationships between her matrix and the other participants' data matrices). In the remainder of this paper we assume, without loss of generality, that the data set is distributed between two clouds or cloud accounts Alice and Bob (the results can be easily extended to more clouds/accounts, by iterating them until the complete $\mathbf{\Sigma}$ has been obtained).

It is important to highlight that being able to securely compute scalar products permits obtaining, in addition to covariance matrices, contingency tables (a.k.a. cross-tabulations) in a very simple way. For each cross-tabulation cell, the clouds can compute auxiliary binary attributes aux from the stored records where $aux_i = 1$ if the i-th record meets the criteria of the cell, and $aux_i = 0$ otherwise; then they can compute the scalar product of their auxiliary attributes. For instance, imagine we want to find the number of patients aged $40 - 50$ who suffer from a specific disease. In that case, the clouds holding attributes "age" and "disease" should independently compute their corresponding auxiliary binary attributes (the former cloud would set $aux_i^1 = 1$ iff the i-th record corresponds to a patient aged $40-50$ and the latter cloud would set $aux_i^2 = 1$ iff the the i-th record corresponds to a patient suffering from the specific disease). Then both clouds would engage in a scalar product protocol of their attributes aux^1 and aux^2 to find the desired number of patients. Note that this procedure is applicable to continuous, discrete and categorical attributes.

B. Secure scalar product without cryptography

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two vectors with n components, for n even, owned by Alice and Bob, respectively. The problem

is to securely compute the product $\boldsymbol{x}^T\boldsymbol{y}$. The following protocol was proposed in [10]:

Protocol 1:

i. Alice and Bob jointly generate a random $n \times n/2$ matrix $\boldsymbol{Q}$.

ii. Alice generates $\boldsymbol{p}$, a random $n/2$-vector, computes $\boldsymbol{u} = \boldsymbol{x} + \boldsymbol{Q}\boldsymbol{p}$ and sends it to Bob.

iii. Bob computes $S = \boldsymbol{u}^T\boldsymbol{y}$ and $\hat{\boldsymbol{y}} = \boldsymbol{Q}^T\boldsymbol{y}$ and sends both to Alice.

iv. Alice obtains $\boldsymbol{x}^T\boldsymbol{y} = S - \hat{\boldsymbol{y}}^T\boldsymbol{p}$.

As previously noted, Protocol 1's privacy relies on the fact that the original vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are not shared at any time; only linear transformations of them are. In particular, getting $\boldsymbol{u}$ allows Bob to learn a set of n linear equations involving $\boldsymbol{x}$ and $\boldsymbol{p}$. However, the number of unknowns in those two vectors ($\frac{3n}{2}$) is higher than the number of equations and, therefore, Bob cannot reconstruct $\boldsymbol{x}$. On the other hand, getting S, $\hat{\boldsymbol{y}}$ and the final result $\boldsymbol{x}^T\boldsymbol{y}$ gives Alice $2 + \frac{n}{2}$ equations with n unknowns, so Alice does not learn $\mathbf{y}$ as long as $n > 4$ (in fact $n \geq 6$ since n must be even).

Alternatively, the following protocol was proposed in [13], again for n even:

Protocol 2:

i. Alice and Bob agree on an $n \times n$ invertible matrix, $\boldsymbol{M}$.

ii. Alice computes $\boldsymbol{x}' = \boldsymbol{x}^T\boldsymbol{M}$, sends $\bar{\boldsymbol{x}} = (x'_1, ..., x'_{n/2})$ to Bob, and keeps $\underline{\boldsymbol{x}} = (x'_{n/2+1}, ..., x'_n)$ to herself. .

iii. Bob computes $\boldsymbol{y}' = \boldsymbol{M}^{-1}\boldsymbol{y}$, sends $\underline{\boldsymbol{y}} = (y'_{n/2+1}, ..., y'_n)$ to Alice, and keeps $\bar{\boldsymbol{y}} = (y'_1, ..., y'_{n/2})$ to himself.

iv. Alice sends $\underline{s} = \underline{\boldsymbol{x}}^T\underline{\boldsymbol{y}}$ to Bob.

v. Bob sends $\bar{s} = \bar{\boldsymbol{x}}^T\bar{\boldsymbol{y}}$ to Alice.

vi. They both compute $\bar{s} + \underline{s} = \boldsymbol{x}^T\boldsymbol{y}$.

Protocol 2's privacy relies on the fact that both participants obtain only $\frac{n}{2}+2$ linear equations for n unknowns; hence, for $n > 4$ the protocol is privacy-preserving. Essentially, the protocol breaks the computation of the product $\boldsymbol{x}^T\boldsymbol{y}$ into two pieces ($\bar{s}$ and $\underline{s}$) held each by one participant, so that no participant learns the product unless the other participant shares his/her piece. However, if n is large enough, $\bar{s}$ and $\underline{s}$ are likely to be similar and $\boldsymbol{x}^T\boldsymbol{y}$ can be one-sidedly and accurately estimated by either participant as $2\bar{s}$ or $2\underline{s}$.

Another interesting protocol for the scalar product is the one proposed in [14]. It is based on the idea of hiding the actual vector $\boldsymbol{y}$ within a matrix so that, even if this matrix is reconstructed, it is not possible to know which row corresponds to $\boldsymbol{y}$. The protocol follows:

Protocol 3:

i. Bob chooses an integer privacy parameter s and randomly generates an $s \times s$ matrix $\boldsymbol{K}$ and a scalar $r, 1 \leq r \leq s$. Bob also randomly generates $s-1$ vectors $\boldsymbol{w}_i$ with $n+1$ components, for $1 \leq i \leq s$, $i \neq r$ and creates a new vector $\boldsymbol{w}_r$ whose n first components equal $\boldsymbol{y}$ and the $(n+1)$-th one equals 1. Then, Bob builds an $s \times (n+1)$ matrix $\boldsymbol{W}$ whose i-th row is $\boldsymbol{w}_i^T$ and computes:

$$b = \sum_{i=1}^{s} K_{ir} \quad \text{and} \quad \boldsymbol{c} = \sum_{i=1, i\neq r}^{s} \left(\boldsymbol{w}_i^T \sum_{j=1}^{s} K_{ji} \right).$$

ii. Bob randomly chooses three numbers r_1, r_2 and r_3 and a vector $\boldsymbol{f}$ with $n+1$ nonzero components and sends to Alice:

$$\boldsymbol{K}\boldsymbol{W}, \quad \boldsymbol{c}' = \boldsymbol{c} + \boldsymbol{f}^T r_1 r_2 \quad \text{and} \quad \boldsymbol{g} = \boldsymbol{f} r_1 r_3.$$

iii. Alice creates a new vector $\boldsymbol{x}'$ whose n first components equal $\boldsymbol{x}$ and the $(n+1)$-th one equals a random number α. Then Alice computes:

$$\boldsymbol{v} = \boldsymbol{K}\boldsymbol{W}\boldsymbol{x}', \quad z = \sum_{i=1}^{s} v_i \quad \text{and} \quad a = z - \boldsymbol{c}'\boldsymbol{x}'.$$

iv. Alice sends a and $h = \boldsymbol{g}^T\boldsymbol{x}'$ to Bob.

v. Bob computes

$$\beta = \frac{a + h\frac{r_2}{r_3}}{b}$$

and sends it to Alice.

vi. Alice obtains $\boldsymbol{x}^T\boldsymbol{y} = \beta - \alpha$.

For the privacy analysis of Protocol 3 we take into account that both Alice and Bob know the equations that relate all parameters, vectors and matrices involved in the protocol. For that reason, Bob gets two equations (when he gets a and h) with $n+1$ unknowns (both equations determining a and h can be expressed in terms of the $n+1$ components of vector $\boldsymbol{x}'$). Similarly, if $s = kn$ where k is a positive integer, Alice gets $kn^2 + (k+2)n + 4$ equations with $n^2(k^2+k) + n(k+1) + 4$ unknowns[3]. Further, note that, even if reconstruction of $\boldsymbol{W}$ were possible, Alice still could not guess which of the rows of $\boldsymbol{W}$ corresponds to $\boldsymbol{y}$ (as long as $\boldsymbol{y}$ is not the only row with a "1" in its last component). Thus, for the privacy of the protocol, it is important to obtain a matrix $\boldsymbol{W}$ with several ones in the last column. Moreover, $\boldsymbol{f}$ cannot contain any zeros, because this would automatically disclose some of the elements of vector $\boldsymbol{c}$.

[3]Note that the number of equations is strictly smaller that the number of unknowns as long as k is strictly larger than $\frac{1}{\sqrt{n}}$.

Note that the selection of $\boldsymbol{Q}$, $\boldsymbol{M}$ and $\boldsymbol{K}$ in Protocols 1, 2 and 3, respectively, needs to be done carefully in order to prevent leakage. For instance, the presence of a column or row with all elements but one equal to zero directly leaks some elements of the multiplying vector. Moreover, previously generated matrices should be reused in successive instances of the protocols with the same original data vector in order to avoid leaking new equations that would make reconstruction of the original vector possible.

C. Secure matrix product without cryptography

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two matrices of sizes $(n \times m_x)$ and $(n \times m_y)$, respectively, whose columns are attributes. Assume also that $\boldsymbol{X}$ is stored in a cloud Alice and $\boldsymbol{Y}$ in a cloud Bob. The following protocol was proposed in [11] in order to securely compute the product $\boldsymbol{X}^T\boldsymbol{Y}$.

Protocol 4:

i. Alice generates an $n \times g$ matrix $\boldsymbol{Z}$ such that $\boldsymbol{X}^T\boldsymbol{Z} = \boldsymbol{0}$.

ii. Alice sends $\boldsymbol{Z}$ to Bob.

iii. Bob computes $\boldsymbol{W} = (\boldsymbol{I} - \boldsymbol{Z}\boldsymbol{Z}^T)\boldsymbol{Y}$, where $\mathbf{I}$ is the identity matrix of size n.

iv. Bob sends $\boldsymbol{W}$ to Alice.

v. Alice computes $\boldsymbol{X}^T\boldsymbol{W} = \boldsymbol{X}^T(\boldsymbol{I} - \boldsymbol{Z}\boldsymbol{Z}^T)\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{Y}$.

The authors also study the best g^* (in the sense that both participants leak the same amount of information) and conclude that it needs to equal $g^* = \frac{nm_y}{m_x+m_y}$. However, *this protocol is not privacy-preserving for Bob because matrix $(\boldsymbol{I} - \boldsymbol{Z}\boldsymbol{Z}^T)$ is invertible, except from the fortuitous case where the eigenvalue of $\boldsymbol{Z}\boldsymbol{Z}^T$ equals* 1 (we will later propose a variant that fixes this problem). Regarding Alice, Bob needs to get the orthogonal complement of $\boldsymbol{X}$ ($Ort(\boldsymbol{X})$), whose dimension is $n - m_x$, in order to get the span of $\boldsymbol{X}$. It is important to highlight that Bob cannot recover $\boldsymbol{X}$, but its span and, therefore, learning g equations of the orthogonal complement gives Bob approximately the same information as if they were random. For that reason, from now on we will assume that these g equations are random. In this case, as long as $n \gg m_x$, Alice's data stays private as she only reveals g equations of $Ort(\boldsymbol{X})$. See [11] on how to obtain matrix $\boldsymbol{Z}$.

Another matrix product protocol was presented in [12]. It is based on what they call a commodity server. Let Alice and Bob be as previously defined and let Charlie be the commodity server (in our setting, CLARUS or a third cloud):

Protocol 5:

i. Charlie generates a random $n \times m_x$ matrix $\boldsymbol{R}_x$ and another $n \times m_y$ matrix $\boldsymbol{R}_y$ and lets $\boldsymbol{P}_x + \boldsymbol{P}_y = \boldsymbol{R}_x^T\boldsymbol{R}_y$, where $\boldsymbol{P}_x$(or $\boldsymbol{P}_y$) is a randomly generated $m_x \times m_y$ matrix.

ii. Charlie sends $\boldsymbol{P}_x$ and $\boldsymbol{R}_x$ to Alice and $\boldsymbol{P}_y$ and $\boldsymbol{R}_y$ to Bob (or equivalently the seeds for a common random generator).

iii. Alice computes $\hat{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{R}_x$ and sends it to Bob.

iv. Bob generates a random $m_x \times m_y$ matrix $\boldsymbol{V}_y$, computes $\boldsymbol{T} = \hat{\boldsymbol{X}}^T\boldsymbol{Y} + (\boldsymbol{P}_y - \boldsymbol{V}_y)$ and sends $\boldsymbol{T}$ and $\hat{\boldsymbol{Y}} = \boldsymbol{Y} + \boldsymbol{R}_y$ to Alice and $\boldsymbol{V}_y$ to Charlie.

v. Alice computes $\boldsymbol{V}_x = \boldsymbol{T} + \boldsymbol{P}_x - \boldsymbol{R}_x^T\hat{\boldsymbol{Y}}$ and sends it to Charlie.

vi. Charlie computes $\boldsymbol{V}_x + \boldsymbol{V}_y = \hat{\boldsymbol{X}}^T\boldsymbol{Y} + (\boldsymbol{P}_y - \boldsymbol{V}_y) + \boldsymbol{P}_x - \boldsymbol{R}_x^T\hat{\boldsymbol{Y}} + \boldsymbol{V}_y = \boldsymbol{X}^T\boldsymbol{Y} + (\boldsymbol{P}_x + \boldsymbol{P}_y - \boldsymbol{R}_x^T\boldsymbol{R}_y) = \boldsymbol{X}^T\boldsymbol{Y}$.

By following Protocol 5, Bob gets $m_x(n+m_y)$ linear equations with $m_x(2n+m_y)$ unknowns. Similarly, Alice gets $m_y(n+2m_x)$ linear equations with $2m_y(n+m_x)$ unknowns. Hence, neither Alice's data set $\boldsymbol{X}$ can be computed by Bob, nor Bob's data set $\boldsymbol{Y}$ can be computed by Alice.

D. Comparison among methods

In this section we will compare the protocols reviewed in the previous subsections. Since the first three ones were aimed at scalar product and the last two were aimed at matrix product, we will assume $m_x = m_y = 1$ in the matrix product protocols to get comparable results.

Table I lists different features of Protocols 1, 2, 3, 4 and 5. In particular, the number of equations revealed and the number of random unknowns to each participant is shown[4]. As previously noted, as long as the number of equations is smaller than the number of unknowns (including both random and private unknowns), it is not possible to solve the system of equations for a unique solution and, therefore, the protocol preserves the privacy of the other participant's data set.

Shannon wrote in [15]: *"It is shown that perfect secrecy is possible but requires, if the number of messages is finite, the same number of possible keys."* Following this idea, we have created a new Privacy Index (PI) that permits measuring how much each participant learns about the other's data set. It is defined as the quotient between the number of equations revealed (n_e) minus the number of random unknowns (n_r) on one side, and the number of private unknowns (n_p), that is, $PI = \max\left(0, \frac{n_e - n_r}{n_p}\right)$.

[4]We have taken $s = kn$ in Protocol 3 and $g = g^* = \frac{n}{2}$ in Protocol 4 (because $m_x = m_y = 1$).

If PI takes a value equal to or larger than 1, then the protocol is not privacy-preserving because the participant can fully recover the other participant's data set. On the contrary, if PI is 0, the privacy of the other participant's data set is fully guaranteed because of the large quantity of randomness introduced in the protocol.

In order to compare the above protocols, we have computed the PI index for both Alice and Bob for each protocol. Further we give the average of Alice's and Bob's indices as a global measure of the protocol (we assume n is large for this average, in order to get an asymptotic value).

It can be seen that Protocols 1 and 2 yield a similar level of privacy (in fact asymptotically it is the same level). Nevertheless, it is important to highlight that n needs to be greater than 4 for the protocols to give any privacy. On the other hand, Protocol 3 gives a fair level of privacy to both participants as long as k is large enough ($k > \frac{1}{\sqrt{n}}$). Furthermore, Bob's PI equals 0 if $k \geq \frac{1}{n}\sqrt{2n+1}$. Protocol 4 is entirely privacy-preserving for Alice, but not privacy-preserving at all for Bob, which why the average PI is meaningless (and we have not computed it). Finally, Protocol 5 is entirely privacy-preserving for Alice and Bob; for large data sets, the only protocol among the other ones that offers comparable privacy to Alice and Bog is Protocol 3.

Beyond the privacy level, protocols must also be compared in terms of computational and communication cost. This is done in Table II for Protocols 1, 2, 3, 4 and 5. In particular, we have evaluated the computational cost for Alice, for Bob, for CLARUS and the total computation. To assess this cost, we have excluded the cost of elementary operations, such as addition of two values or scalar multiplications. Moreover, operations that do not need to be repeated each time the protocol is executed, like the generation of matrices $\boldsymbol{Q}, \boldsymbol{M}, \boldsymbol{K}, \boldsymbol{Z}$, are not considered either. Assuming that the clouds have unlimited storage, it is reasonable to assume as well that those matrices 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 seed and generates the matrices when needed. Note that Protocol 5 is the one with smallest computational cost but it is also the only one with nonzero computational cost for CLARUS (which plays the role of commodity server). Protocols 1 and 2 are the ones with medium computational costs but they are also the ones offering lowest privacy (highest PI).

Finally, the right half of Table II shows the communication cost of the protocols. We have associated the communication cost with the sender and we take into account that final results need to be communicated to CLARUS. Moreover, we use a parameter γ to represent the maximum length of the numbers in the vectors and matrices used in the protocols. We consider that whenever possible the participants send the seeds of random vectors and matrices, rather than the vectors and matrices themselves. Again, Protocols 1, 2 and 5 are the ones with less cost.

[5]All the equations revealed are linear except from the ones by Bob in Protocol 3.

IV. Enhanced protocol variants

In this section, we propose variants of the above protocols to improve their privacy and performance.

A. Preventing the clouds from learning the covariances

In the CLARUS context, and in order to preserve privacy as much as possible, it is preferable to prevent the clouds from learning the resulting covariance. Indeed, Alice could combine the attributes she stores with the covariances to estimate the attributes stored by Bob by using simple linear regressions, which would partly undo the protection provided by splitting. The last column of Table I shows which participants learn the covariance in the various protocols. We now show how to modify them so that only CLARUS learns the covariance. From now on, Protocol i.j refers to the j-th adapted version of original Protocol i.

Protocol 1.1:

iii. Bob computes $S = \boldsymbol{u}^T\boldsymbol{y}$ and sends it to CLARUS (not to Alice). Bob sends $\hat{\boldsymbol{y}}$ to Alice.

iv. Alice computes $S' = \hat{\boldsymbol{y}}^T\boldsymbol{p}$ and sends it to CLARUS.

v. CLARUS computes $S - S' = \boldsymbol{x}^T\boldsymbol{y}$.

Protocol 1.1 prevents Alice from learning the result and reduces the number of equations known to Alice to $\frac{n}{2}$.

Protocol 2.1:

iv. Alice sends $\bar{s} = \bar{\boldsymbol{x}}\bar{\boldsymbol{y}}$ to CLARUS.

v. Bob sends $\underline{s} = \underline{\boldsymbol{x}}\underline{\boldsymbol{y}}$ to CLARUS.

vi. CLARUS computes $\bar{s} + \underline{s} = \boldsymbol{x}^T\boldsymbol{y}$.

Protocol 2.1 also prevents Alice from learning the result. However, for large n Alice and Bob can still get quite a precise estimation, in the same way pointed out after Protocol 2.

Regarding Protocol 3, it is quite straightforward to prevent Alice from learning the result if we let Alice and Bob send to CLARUS α and β, rather than Bob sending β to Alice (not getting β also reduces the number of equations known to Alice).

TABLE I. PRIVACY COMPARISON OF PROTOCOLS 1, 2, 3, 4 AND 5.

	# of equations revealed[5]		# of (random, private) unknowns		PI			Who learns
	by Alice to Bob	by Bob to Alice	to Bob	to Alice	Alice vs Bob	Bob vs Alice	Avg.	$\boldsymbol{x}^T\boldsymbol{y}$?
Prot. 1	n	$\frac{n}{2}+2$	$(\frac{n}{2}, n)$	$(0, n)$	$\frac{1}{2}$	$\frac{1}{2}+\frac{2}{n}$	$\frac{1}{2}$	Alice
Prot. 2	$\frac{n}{2}+2$	$\frac{n}{2}+2$	$(0, n)$	$(0, n)$	$\frac{1}{2}+\frac{2}{n}$	$\frac{1}{2}+\frac{2}{n}$	$\frac{1}{2}$	Alice and Bob
Prot. 3	2	$4+(k+2)n+kn^2$	$(1, n)$	$(3+kn+(k+k^2)n^2, n+1)$	$\frac{1}{n}$	$\frac{-k^2n^2+2n+1}{n+1}$	0	Alice
Prot. 4	$\frac{n}{2}$	$n+1$	$(\frac{n}{2}, n)$	$(0, n)$	0	$1+\frac{1}{n}$	N/A	Alice
Prot. 5	$n+1$	$n+2$	$(n+1, n)$	$(n+2, n)$	0	0	0	Charlie

TABLE II. COST COMPARISON OF PROTOCOLS 1, 2, 3, 4 AND 5.

	Computational cost				Communication cost			
	for Alice	for Bob	for CLARUS	Total	for Alice	for Bob	for CLARUS	Total
Prot. 1	$\frac{n^2}{2}+n$	$\frac{n^2}{2}+n$	0	$O(n^2)$	$(n+1)\gamma$	$(\frac{n}{2}+1)\gamma$	0	$O(n\gamma)$
Prot. 2	$n^2+\frac{n}{4}$	$n^\tau+n^2+\frac{n}{4}$	0	$O(n^\tau)$	$(\frac{n}{2}+2)\gamma$	$(\frac{n}{2}+2)\gamma$	0	$O(n\gamma)$
Prot. 3	$3kn^2+(k+2)n+2$	$k^2(n^3+n^2)$	0	$O(n^3)$	3γ	$(kn^2+(k+2)n+1)\gamma$	0	$O(n^2\gamma)$
Prot. 4	n	$\frac{n^3}{2}+n^2$	0	$O(n^3)$	$(\frac{n^2}{2}+1)\gamma$	$n\gamma$	0	$O(n^2\gamma)$
Prot. 5	n	n	$3n+1$	$O(n)$	$(n+1)\gamma$	$(n+2)\gamma$	2γ	$O(n\gamma)$

As for Protocol 4, an easy way to prevent Alice from learning the covariances is to let Alice and Bob send $\boldsymbol{X}$ and $\boldsymbol{W}$, respectively, to CLARUS. The downside is that now the workload for CLARUS is the same as downloading the whole $\boldsymbol{X}$ and $\boldsymbol{Y}$ and performing $\boldsymbol{X}^T\boldsymbol{Y}$. Hence, this variant is not suitable for the CLARUS setting. Finally, Protocol 5 already hides the covariance matrix from both Alice and Bob, but the work to be performed by Charlie-CLARUS can be computationally heavy.

B. Enhancing privacy

As noted in Section III-B, in Protocol 3 matrix $\boldsymbol{W}$ should not have a single "1" in its last column in order to really hide $\boldsymbol{y}$. However, randomly generating all rows of matrix $\boldsymbol{W}$ other than $\boldsymbol{y}$, as done in Protocol 3, makes it very unlikely to have more than a single "1" in the last column. We now propose a variant of Protocol 3 that fixes this problem. Furthermore, the modified protocol still withholds the covariance matrix from Alice and Bob. Finally, we also remove the hiding of vector $\boldsymbol{c}$ in order to reduce the communication cost of the protocol. This removal implies a slightly worse PI, but the increase is negligible for large n.

Protocol 3.1:

i. Let Bob choose a privacy parameter s and randomly generate an $s \times s$ matrix $\boldsymbol{K}$ and a scalar r, with $1 \leq r \leq s$. Bob also randomly generates $s-1$ vectors $\boldsymbol{w}_i$, $1 \leq i \leq s, i \neq r$ with n components and lets $\boldsymbol{w}_r = \boldsymbol{y}$. Then, Bob builds an $s \times n$ matrix $\boldsymbol{W}$ whose i-th row is $\boldsymbol{w}_i^T$ and computes:

$$b = \sum_{i=1}^{s} K_{ir} \quad \text{and} \quad \boldsymbol{c} = \sum_{i=1, i\neq r}^{s} \left(\boldsymbol{w}_i^T \sum_{j=1}^{s} K_{ji} \right).$$

ii. Bob sends b to CLARUS and $\boldsymbol{KW}$ and $\boldsymbol{c}$ to Alice.

iii. Alice then computes:

$$\boldsymbol{v} = \boldsymbol{KWx}, \quad z = \sum_{i=1}^{s} v_i \quad \text{and} \quad a = z - \boldsymbol{cx}.$$

iv. Alice sends a to CLARUS.

v. CLARUS obtains $\boldsymbol{x}^T\boldsymbol{y} = \frac{a}{b}$.

In this protocol, Bob does not learn anything about Alice and Alice learns n^2k+n equations with $n^2k^2+n^2k$ unknowns (where we assume again that $s = kn, k > 0$). As long as $k > \frac{1}{\sqrt{n}}$, Alice cannot reconstruct $\boldsymbol{y}$. Moreover, since Alice cannot guess which row of $\boldsymbol{W}$ corresponds to $\boldsymbol{y}$, the linear equations in terms of the components $\boldsymbol{W}$ that Alice can obtain are useless.

Regarding Protocol 4, we have already mentioned that Alice can invert matrix $(\boldsymbol{I} - \boldsymbol{ZZ}^T)$ and learn $\boldsymbol{Y}$. To fix this privacy problem, we propose the following variant.

Protocol 4.1:

i. Alice generates an $n \times g$ matrix $\boldsymbol{Z}$ such that $\boldsymbol{X}^T\boldsymbol{Z} = \boldsymbol{0}$.

ii. Alice sends $\boldsymbol{Z}$ to Bob.

iii. Bob generates a random $n \times m_y$ matrix $\boldsymbol{B}$.

iv. Bob computes $\boldsymbol{W} = \boldsymbol{Y} - \boldsymbol{ZZ}^T\boldsymbol{B}$ and sends $\boldsymbol{W}$ to Alice.

v. Alice computes $\boldsymbol{X}^T\boldsymbol{W} = \boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{ZZ}^T\boldsymbol{B}) = \boldsymbol{X}^T\boldsymbol{Y}$.

Introducing $\boldsymbol{B}$ makes Bob private vs Alice, as Alice now gets nm_y equations with $2nm_y$ unknowns. As previously noted in Section III-C, Alice remains private vs Bob as long as $n \gg m_x$.

C. Reducing the computation cost associated to CLARUS

In Protocol 5, generating matrices $\boldsymbol{R}_x, \boldsymbol{P}_x, \boldsymbol{R}_y$ and $\boldsymbol{P}_y$ can be computationally heavy for CLARUS. The following is a simpler and faster variant that avoids the use of $\boldsymbol{P}_x$, $\boldsymbol{P}_y$, and still prevents Alice and Bob from learning the result:

Protocol 5.1:

i. CLARUS generates a random $n \times m_x$ matrix $\boldsymbol{R}_x$ and another $n \times m_y$ matrix $\boldsymbol{R}_y$ and computes $\boldsymbol{P} = \boldsymbol{R}_x^T \boldsymbol{R}_y$, hence $\boldsymbol{P}$ is a $m_x \times m_y$ matrix.

ii. CLARUS sends $\boldsymbol{R}_x$ to Alice and $\boldsymbol{R}_y$ to Bob (or equivalently the seeds for a common random generator).

iii. Alice computes $\hat{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{R}_x$ and sends it to Bob.

iv. Bob sends $\boldsymbol{T} = \hat{\boldsymbol{X}}^T \boldsymbol{Y}$ to CLARUS and sends $\hat{\boldsymbol{Y}} = \boldsymbol{Y} + \boldsymbol{R}_y$ to Alice.

v. Alice computes $\boldsymbol{S}_x = \boldsymbol{R}_x^T \hat{\boldsymbol{Y}}$ and sends it to CLARUS.

vi. CLARUS computes $\boldsymbol{T} - \boldsymbol{S}_x + \boldsymbol{P} = (\boldsymbol{X} + \boldsymbol{R}_x)^T \boldsymbol{Y} - \boldsymbol{R}_x^T (\boldsymbol{Y} + \boldsymbol{R}_y) + \boldsymbol{R}_x^T \boldsymbol{R}_y = \boldsymbol{X}^T \boldsymbol{Y}$.

With the above variant, CLARUS needs to generate two matrices instead of three. Privacy is maintained: Alice and Bob get both a number of equations that is half the number of unknowns. Alternatively, $\boldsymbol{R}_x$ and $\boldsymbol{R}_y$ could be generated by a third cloud.

D. Comparison of the variants

Here we compare the above protocol variants in terms of privacy, computational and communication cost. Thus, Tables III and IV show the same information as Tables I and II, respectively, but for Protocols 1.1, 2.1, 3.1, 4.1 and 5.1.

As it can be seen in Table III, all protocols are privacy-preserving now and Protocols 3.1, 4.1 and 5.1 have an average PI equal to 0. If $k > \sqrt{\frac{2}{n}}$, then Protocol 3.1 also yields $PI = 0$ for both Alice and Bob and, therefore, it offers the same privacy as Protocol 5.1.

In terms of costs, Table IV shows that, except for Protocol 5.1, protocols with larger PIs have smaller costs. As to Protocol 5.1, it is still the only one with nonzero cost for CLARUS.

All in all, the most suitable protocols for the CLARUS scenario seem to be Protocols 3.1 and 5.1, which yield both similar privacy. We discard Protocol 4.1, because Alice still gets the result and this should be avoided. In terms of costs, although Protocol 5.1 is less heavy, it requires CLARUS to perform some operations, whereas Protocol 3.1 is heavier for the clouds but requires nothing from CLARUS.

TABLE V. TIMES REQUIRED BY PROTOCOLS 3.1, 5.1 AND THE BENCHMARK PROTOCOL 6 TO COMPUTE THE SCALAR PRODUCT $\boldsymbol{x}^T \boldsymbol{y}$.

Protocol 3.1	$(k^2n^3 + (k^2 + k)n^2 + n)/\mu + (kn^2 + n + 2)/\delta$
Protocol 5.1	$2n/\mu + 3n/\eta + (2n + 4)/\delta$
Protocol 6	$n/\eta + 2n/\delta$

V. COMPARISON WITH THE BENCHMARK SOLUTION

In this section we compare the best two protocols identified in the previous section with the benchmark solution that consists of CLARUS downloading the whole data set and computing scalar products/covariances locally:

Protocol 6:

i. Alice and Bob send $\boldsymbol{x}$ and $\boldsymbol{y}$ to CLARUS, respectively.

ii. CLARUS locally computes $\boldsymbol{x}^T \boldsymbol{y}$.

The last row of Table IV shows the communication and computation cost of Protocol 6. In order to compare Protocols 3.1 and 5.1 with Protocol 6, we need to parameterize the download/communication and computation speeds. In particular, we assume that the download/communication speed from the cloud is δ values of size γ per second. Moreover, the computation speed by the cloud and CLARUS are given by μ and η, respectively, where $\mu > \eta$. Table V shows the times for Protocols 3.1, 5.1 and 6.

It is easy to see that Protocol 6 is faster than Protocol 5.1 for any value of the parameters. However, if $\eta \ll \mu$, then Protocol 3.1 is faster than Protocol 6.

The fact that downloading and computing by CLARUS may be faster does not necessarily invalidate the protocols proposed in this paper, because there may be other reasons to compute in the cloud. For instance, if the cloud service is for free (or nearly so), then it is preferable to use the cloud rather than the local computers (that need to be purchased, replaced, upgraded, maintained, etc.) even if using the cloud is a bit slower.

Moreover, even if the cloud service must be paid for, it may still make sense to use it in spite of it being a bit slower than the local computing approach. The reason is that computing capacity in the cloud can be easily obtained on demand at any moment. In contrast, if more local computing capacity is required, buying more computers is needed, which entails a longer-term investment.

VI. CONCLUSIONS AND FUTURE WORK

We have presented the PPSDC problem, with a focus on the computation of scalar products and covariances. We have explored the extent to which existing protocols

TABLE III. PRIVACY COMPARISON OF PROTOCOLS 1.1, 2.1, 3.1, 4.1 AND 5.1.

	# of equations revealed		# of (random, private) unknowns		PI			Who learns
	by Alice to Bob	by Bob to Alice	to Bob	to Alice	Alice vs Bob	Bob vs Alice	Avg.	$\boldsymbol{x}^T\boldsymbol{y}$?
Prot. 1.1	n	$\frac{n}{2}$	$(\frac{n}{2}, n)$	$(0, n)$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	CLARUS
Prot. 2.1	$\frac{n}{2}$	$\frac{n}{2}$	$(0, n)$	$(0, n)$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	CLARUS
Prot. 3.1	0	$n + kn^2$	$(0, n)$	$(n^2(k^2 + k) - n, n)$	0	$-k^2n + 2$	0	CLARUS
Prot. 4.1	$\frac{n}{2}$	$n + 1$	$(\frac{n}{2}, n)$	(n, n)	0	$\frac{1}{n}$	0	Alice
Prot. 5.1	n	n	(n, n)	(n, n)	0	0	0	CLARUS

TABLE IV. COST COMPARISON OF PROTOCOLS 1.1, 2.1, 3.1, 4.1, 5.1 AND 6.

	Computational cost				Communication cost			
	for Alice	for Bob	for CLARUS	Total	for Alice	for Bob	for CLARUS	Total
Prot. 1.1	$\frac{n^2}{2} + n$	$\frac{n^2}{2} + n$	0	$O(n^2)$	$(n + 1)\gamma$	$(\frac{n}{2} + 1)\gamma$	0	$O(n\gamma)$
Prot. 2.1	$n^2 + \frac{n}{4}$	$n^\tau + n^2 + \frac{n}{4}$	0	$O(n^\tau)$	$(\frac{n}{2} + 2)\gamma$	$(\frac{n}{2} + 2)\gamma$	0	$O(n\gamma)$
Prot. 3.1	$kn^2 + n$	$k^2(n^3 + n^2)$	0	$O(n^3)$	γ	$(kn^2 + n + 1)\gamma$	0	$O(n^2\gamma)$
Prot. 4.1	n	$\frac{n^3}{2} + n^2$	0	$O(n^3)$	$(\frac{n^2}{2} + 1)\gamma$	$n\gamma$	0	$O(n^2\gamma)$
Prot. 5.1	n	n	$3n$	$O(n)$	$(n + 1)\gamma$	$(n + 1)\gamma$	2γ	$O(n\gamma)$
Prot. 6	0	0	n	$O(n)$	$n\gamma$	$n\gamma$	0	$O(n\gamma)$

for computing on vertically split data can be leveraged. Existing proposals have been compared and new variants with enhanced privacy and performance have been presented. Furthermore, the best variants have been compared in terms of computation and communication with the benchmark alternative of downloading and locally computing.

As future work, we plan a deeper comparative analysis including economic costs (i.e., the price of computing power in the cloud and the price of investing in local computers). Also, the best identified protocols will be implemented in the CLARUS framework.

ACKNOWLEDGMENTS AND DISCLAIMER

The following funding sources are gratefully acknowledged: European Commission (project H2020 644024 "CLARUS"), Government of Catalonia (ICREA Acadèmia Prize to the last author and grant 2014 SGR 537) and Spanish Government (project TIN2011-27076-C03-01 "CO-PRIVACY"). The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are their own and do not necessarily reflect those of UNESCO.

REFERENCES

[1] "CLARUS - a Framework for User Centred Privacy and Security in the Cloud," http://www.clarussecure.eu.

[2] D. Sánchez and J. Domingo-Ferrer, "CLARUS - a Framework for User Centred Privacy and Security in the Cloud," in *Position paper at Cloudscape VII*, 2015.

[3] G. Trouessin, "Traitements fiables de données confidentielles par fragmentation-redondance-dissémination," Ph.D. dissertation, Université de Toulouse 3, 1991.

[4] A. Dubovitskaya, V. Urovi, M. Vasirani, K. Aberer, and M. Schumacher, "A cloud-based ehealth architecture for privacy preserving data integration," in *ICT Systems Security and Privacy Protection*. Springer, 2015, pp. 585–598.

[5] G. Weiss, "Data mining in the real world: Experiences, challenges, and recommendations." in *DMIN*, 2009, pp. 124–130.

[6] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology & Decision Making*, vol. 5, no. 04, pp. 597–604, 2006.

[7] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep a secret: A distributed architecture for secure database services," in *CIDR 2005*, 2005, pp. 186–199.

[8] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R. Motwani, "Distributing data for secure database services," *Transactions on Data Privacy*, vol. 5, no. 1, pp. 253–272, 2012.

[9] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *Information Security and Cryptology ICISC 2004*, ser. Lecture Notes in Computer Science, C. Park and S. Chee, Eds. Springer Berlin Heidelberg, 2005, vol. 3506, pp. 104–120.

[10] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu, "Tools for privacy preserving distributed data mining," *ACM SiGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[11] A. Karr, X. Lin, A. Sanil, and J. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *Journal of Official Statistics*, vol. 25, no. 1, p. 125, 2009.

[12] W. Du, Y. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification." in *SDM*, vol. 4. SIAM, 2004, pp. 222–233.

[13] Y. Chiang, D. Wang, C. Liau, and T. Hsu, "Secrecy of two-party secure computation," in *Data and Applications Security XIX*. Springer, 2005, pp. 114–123.

[14] I. Ioannidis, A. Grama, and M. Atallah, "A secure protocol for computing dot-products in clustered and distributed environments," in *Proceedings of the International Conference on Parallel Processing*. IEEE, 2002, pp. 379–384.

[15] C. Shannon, "Communication theory of secrecy systems." in *Bell System Technical Journal*, 1949, vol. 28, pp. 656–715.