Semi-Private Computation of Data Similarity with Controlled Leakage

René Bødker Christensen, Shashi Raj Pandey, and Petar Popovski

Abstract—Consider two data providers that want to contribute data to a certain learning model. Recent works have shown that the value of the data of one of the providers is dependent on the similarity with the data owned by the other provider. It would thus be beneficial if the two providers can calculate the similarity of their data, while keeping the actual data private. In this work, we devise multiparty computation-protocols to compute similarity of two data sets based on correlation, while offering controllable privacy guarantees. We consider a simple model with two participating providers and develop methods to compute exact and approximate correlation, respectively, with controlled information leakage. Both protocols have computational and communication complexities that are linear in the number of data samples. We also provide general bounds on the maximal error in the approximation case, and analyse the resulting errors for practical parameter choices.

Index Terms—data similarity, information leakage, multiparty computation, sample correlation, secure protocols

I. INTRODUCTION

WITH an increased roll-out of networked cyber-physical systems such as Internet of Things (IoT), there has been an unprecedented growth in collection and analysis of data. This development has also given rise to data markets [1], [2], [3], where data is exchanged between different entities. For instance, large quantities of high-quality data is fundamental during the training of machine learning models, and this may require the acquisition of data from external data holders. As a specific example, [4] considers a regression market, where a forecasting model is trained based on data distributed among different data providers such as IoT-devices. That is, the data providers – or simply referred to as participants – perform collaborative training in a market setup to improve forecasts.

There are, however, several challenging aspects in making such a market work. One such challenge is the fact that participants holding ‘similar’ data – in particular, correlated data – will be able to contribute similar knowledge to the model training. Once data from one participant is known, however, the data held by the other participant will have limited value to whoever is training the model [3], [5]. This may lead to data rivalry [2], [3], where participants are incentivized to sell earlier than others, potentially underbidding each other and thereby decreasing the overall price of the data. This injects mistrust in the data market and participants may opt out from exchanging their data at all.

With this in mind, it may be beneficial for participants to cooperate, sell the data collectively, and share the payment between them in the case where their data sets are correlated. In practice, though, these correlations are unknown a priori, and the participants are likely reluctant to reveal their data for computing correlations in the clear. Hence, there is a need for participants to determine if their data sets are similar without actually revealing their data sets to each other. A specific measure of similarity is the correlations, so in other words, the participants aim to compute correlation in a private fashion.

Several studies on incentive mechanism design have been conducted to handle the behaviour of strategic participants [6], [7], elicit data from them while manipulating the privacy costs, and for different contexts, derive collaboration opportunities for accurate computation in the data markets, such as in regression markets. Nevertheless, such techniques demand participants to reveal parts of their private information. In fact, the loss of privacy in data markets leads directly to monetary loss. This highlights the tension between data privacy and the benefits of data.

From these observation, we see that knowing correlation between data samples of the participants helps to explore the following outlined key research question: When is it beneficial for the participants to perform collaborative training? i.e., learning to learn when to collaborate or not.

Secure MPC is a term used to describe techniques that allow multiple participants to give inputs to and compute the value of a publicly known function without revealing the individual inputs. Traditionally, the functions under considerations in these MPC-problems are defined over a finite field. In many real-world problems, however, the inputs of each participant will not be elements of a finite field but instead real numbers. The literature already contains a number of suggestions for dealing with such cases. One way is to use MPC techniques designed for floating-point numbers [8], [9]. These methods are typically very computationally expensive, however. Another way is to assume that the inputs can be represented as fixed-point numbers, which can readily be mapped to a finite field where standard MPC techniques can be applied [10], [11], [12].

Nevertheless, certain functions are inherently real in the sense that even under the assumption that the inputs are exactly representable as fixed-point numbers, the output cannot be guaranteed to be representable as a fixed-point number. One such function is the sample correlation between two sets of data samples as in the data market example mentioned previously.

When exact computation with fixed-point numbers is not possible, we can in principle still use essentially the same...
approach by approximating the inputs by fixed-point numbers and obtaining the function value up to some approximation error. While this is an easy solution, it complicates security arguments as it will be necessary to prove that the error does not reveal more about the inputs than the desired function value [13].

In this work, we focus on computing the sample correlation in a way that is efficient, but still retains reasonable privacy guarantees. More precisely, we describe a protocol that allows the participants to approximate the sample correlation to a high degree of accuracy, and then show that this is at least as secure as a protocol where the participants reveal certain values related to the rounding errors when representing their inputs as fixed-point numbers.

The paper is structured as follows. Section II discusses related work. Section III recalls the basics of MPC and fixed-point numbers that will be used in subsequent sections. In Section IV we consider an exact correlation protocol where the participants reveal certain error values, and Section V treats the approximate protocol, whose leakage is bounded by the leakage in exact protocol. Afterwards, Section VI contains an analysis of the possible protocol parameters in a practical implementation. Finally, Section VII lists some open problems and Section VIII contains the concluding remarks.

II. RELATED WORK

There are numerous works that treat privacy-preserving computation on real or floating-point numbers. These take different approaches as some work directly with real numbers [14], some with floating-point numbers as specified by the IEEE 754 standard [15], and yet others using alternative number representations [16], [9], [17]. There has also been an interest in finding the best way to quantify the amount of leakage for protocols using real numbers [18].

An alternative approach to the problem at hand would be to consider differential privacy [19], [20], which would ensure that the output would essentially remain the same if a single sample point were to be left out. This thus provides privacy for the individual data samples. Differential privacy has for instance been used in training of machine learning models – see e.g. [21], [22] – which is a natural setting to consider a data market.

Following [2], a series of works [3], [23], [24] have explored the combinatorial nature of data, highlighting the negative effect of sharing correlated data on pricing and market efficiency. It was formally proven that correlated data can cause data trading problems [2], [23]; such information leakage leads to price depression and uncontrolled data exchanges, leaving poor communication and training efficiency during model training. A way out envisioned to avoid this unwanted scenario would be to – a priori – privately know data correlation between participants and accordingly develop the data revealing strategies. Yet, a majority of works [2], [3], [25] focus on mechanism design of data markets for data acquisition with a focus to incentivize truthful participation, keeping aside the aftermaths of data correlation in data trading. Several game-theoretic approaches or auction mechanism [3], [26], [27] are employed to formulate such interactions between the data market and the participants with some privacy guarantees, trading differentially private data samples. Nevertheless, efficient protocols to privately compute sample correlation, either on real or floating-point numbers, is fundamental to realize an efficient market design are left uninvestigated.

III. PRELIMINARIES

A. Secure Multiparty Computation

For a prime \( p \), let \( \mathbb{F}_p \) be the finite field of \( p \) elements, and consider two participants \( P_1 \) and \( P_2 \), each with private inputs \( x^{(1)} \in \mathbb{F}_p^2 \) and \( x^{(2)} \in \mathbb{F}_p^2 \), respectively. The goal of MPC is to devise a protocol for computing \( f(x^{(1)}, x^{(2)}) \) that reveals no more about the inputs \( x^{(1)}, x^{(2)} \) than what is revealed by the value of \( f(x^{(1)}, x^{(2)}) \) itself. Throughout, we will assume that the participants can be passively corrupted – sometimes referred to as ‘honest, but curious’. This means that they will follow protocols honestly, but try to use the information seen during the protocol to extract additional information about the inputs of the other participant.

An important tool in MPC is secret sharing. In this work, we will use additive secret sharing, meaning that a secret \( x \in \mathbb{F}_p \) is split into \( n \) shares \( (s_1, s_2, \ldots, s_n) \in \mathbb{F}_p^n \), where \( s_i \) is chosen uniformly at random for \( i = 1, 2, \ldots, n - 1 \) and \( s_n = x - \sum_{i=1}^n s_i \). This is a secret sharing scheme with \((n-1)\)-privacy and \( n \)-reconstruction. We use \([x]\) to denote a value shared in this way, and \([x]_i\) to denote the share held by the \( i \)’th participant. More explicitly, writing \([x]\) in the case of two participants means that \( P_1 \) and \( P_2 \) hold values \([x]_1\) and \([x]_2\), respectively, such that \([x]_1 + [x]_2 = x\). For a thorough introduction to MPC and secret sharing, see e.g. [28].

We assume that the participants have access to a functionality \( f_{BT} \) producing Beaver-triples [29], i.e., triples of shares \( ([u], [v], [w]) \) satisfying \( uv = w \), where \( u, v, w \) are in some finite field. This allows the participants to compute products of secretly shared values. Namely, to compute the product of \([x]\) and \([y]\), they compute and reveal \( x - u \) and \( y - v \). It can then be checked that if both participants compute \([z]_1 = [w]_1 + [x]_i(y - v) + [y]_i(x - u) - (x - u)(y - v)\) using their respective shares, this yields \([z]_1 + [z]_2 = xy\). That is, \(\{[z]_1, [z]_2\}\) is a sharing of \(xy\). Beaver triples can for instance be created using oblivious transfer. We omit the details, which can be found in e.g. [30] Sec. 4.1.

B. Fixed-point representation

Like [10], [11], we approach the computation of our public function using fixed-point arithmetic represented using elements of \( \mathbb{F}_p \). More precisely, given \( M \in \mathbb{N}_+ \) we let \( \mathbb{Z}_M^\delta = \{ x \in \mathbb{Z} \mid -M \leq x \leq M \} \). Then for a scaling factor \( \delta > 0 \), we let
\[
Q_{(M, \delta)} = \{ x\delta \mid x \in \mathbb{Z}_M^\delta \},
\]
which is also illustrated on Figure 1. There is a natural bijection between \( Q_{(M, \delta)} \) and \( \mathbb{Z}_M^\delta \), and in order to map fixed-point numbers in \( Q_{(M, \delta)} \) to elements of \( \mathbb{F}_p \), we use the map \( \varphi' : \mathbb{Z}_M^\delta \to \mathbb{F}_p \) given by \( \varphi'(x) = x \mod p \). This is injective as
long as $M < p - M$; that is, as long as $p > 2M$. In this case, the map $\psi : \mathbb{F}_p \rightarrow \mathbb{Z}_M^\pm \cup \{\perp\}$ defined by

$$
\psi(x) = \begin{cases} 
  x & x \in \{0, 1, \ldots, M\} \\
  x - p & x \in \{p - M, p - M + 1, \ldots, p - 1\} \\
  \perp & \text{otherwise}
\end{cases}
$$

is well-defined, and we have $\psi \circ \varphi' = \text{Id}$.  

**Remark 1:** If we let $M = \lfloor \frac{p}{2} \rfloor = \frac{p - 1}{2}$ for $p > 2$, each element of $\mathbb{F}_p$ corresponds to an element of $\mathbb{Z}_M^\pm$. In this case, we can omit $\perp$ from the definition of $\psi$, meaning that $\psi = (\varphi')^{-1}$. To simplify notation, we will assume this case in the remainder of the work.

Rather than working directly with $\varphi'$ and $\psi$, we define $\varphi_\delta : \mathbb{Q}(\langle M, \delta \rangle) \rightarrow \mathbb{F}_p$ by $\varphi_\delta(x) = \psi(x)$. Our assumption from Remark 1 further implies that $\varphi_\delta^{-1}(x) = \delta \psi(x)$. Later on, we will need the properties of $\varphi_\delta$ and appropriate inverses when applied to sums and products, so we record these in the following lemma.

**Lemma 1:** Let $p > 2$ be a prime, and let $M = (p - 1)/2$ as in Remark 1. If $a_1, a_2, \ldots, a_n \in \mathbb{Q}(\langle M, \delta \rangle) \subseteq \mathbb{R}$, the map $\varphi_\delta$ satisfies

1. If $\sum_{i=1}^{n} a_i \in \mathbb{Q}(\langle M, \delta \rangle)$, then $\varphi_\delta^{-1}(\sum_{i=1}^{n} \varphi_\delta(a_i)) = \sum_{i=1}^{n} a_i$.

2. If $\prod_{i=1}^{n} a_i \in \mathbb{Q}(\langle M, \delta \rangle)$, then $\varphi_\delta^{-1}(\prod_{i=1}^{n} \varphi_\delta(a_i)) = \prod_{i=1}^{n} a_i$.

**Proof:** Write $a_i = \delta x_i$ for some $x_i \in \mathbb{Z}_M^\pm$. We then have

$$
\varphi_\delta(a_i) = \sum_{i=1}^{n} (x_i \mod p)
$$

$$
= \left(\sum_{i=1}^{n} x_i \right) \mod p = \varphi_\delta \left(\sum_{i=1}^{n} a_i\right),
$$

where the last equality follows from the assumption $\sum_{i=1}^{n} a_i \in \mathbb{Q}(\langle M, \delta \rangle)$. The proof for products is similar.

**IV. SEMI-PRIVATE COMPUTATION OF CORRELATION**

We assume that participants $P_1$ and $P_2$ each hold samples $(x_j^{(1)})_{j=1}^{n}$ and $(x_j^{(2)})_{j=1}^{n}$ of their respective random variables with $x_j^{(i)} \in \mathbb{R}$. Their goal is to compute the sample correlation

$$
r = \frac{1}{n - 1} \sum_{j=1}^{n} \left(\frac{x_j^{(1)} - \bar{x}^{(1)}}{s^{(1)}}\right) \left(\frac{x_j^{(2)} - \bar{x}^{(2)}}{s^{(2)}}\right),
$$

where $s^{(i)} = \sqrt{\frac{1}{n - 1} \sum_{j=1}^{n} (x_j^{(i)} - \bar{x})^2}$ is the sample standard deviation for the data held by $P_i$. We use the notation $z_j^{(i)} = (x_j^{(i)} - \bar{x}) / s^{(i)}$, and note that $P_i$ can compute $z_j^{(i)}$ locally for every $j \in \{1, 2, \ldots, n\}$. In order to use techniques from MPC, the participants will round each $z_j^{(i)}$ to the nearest element of $\mathbb{Q}(\langle M, \delta \rangle)$ (where appropriate choices of $M$ and $\delta$ are discussed in a later section). That is, $P_i$ finds $\tilde{z}_j^{(i)} \in [-\delta, \delta]$ such that

$$
z_j^{(i)} = \tilde{z}_j^{(i)} + \varepsilon_j^{(i)}
$$

with $\tilde{z}_j^{(i)} \in \mathbb{Q}(\langle M, \delta \rangle)$. For convenience, we also define $\tilde{r}$ to be the approximation of the sample correlation when using $\mathbb{Q}(\langle M, \delta \rangle)$ to represent the $z_j^{(i)}$. That is, we let

$$
\tilde{r} = \frac{1}{n - 1} \sum_{j=1}^{n} \tilde{z}_j^{(1)} \tilde{z}_j^{(2)}.
$$

Based on these definitions, we can rewrite (1) as

$$
r = \frac{1}{n - 1} \left(\sum_{j=1}^{n} z_j^{(1)} z_j^{(2)}\right) = \tilde{r} + \frac{1}{n - 1} \left(\sum_{j=1}^{n} \sum_{i=1}^{n} \varepsilon_j^{(1)} \varepsilon_j^{(2)}\right)
$$

From this, we see that $r$ splits into the approximated sample correlation $\tilde{r}$ plus terms related to the rounding errors. Thus, this leads to a method of computing $r$ in which the participants do an approximation using fixed-point numbers and then accept the leakage of the remaining sums in $\mathbb{Q}(\langle M, \delta \rangle)$. Before presenting the details, we prove two lemmata that justify the parameter choices.

**Lemma 2:** Let $\delta > 0$ be fixed, and let $\tilde{r}$ be as in (3). If $R \in \mathbb{R}$ satisfies $|z_j^{(i)}| \leq R$ for all $i \in \{1, 2\}$ and all $j \in \{1, 2, \ldots, n\}$, then

$$
|r - \tilde{r}| \leq \frac{n \delta}{n - 1} \left( R + \frac{\delta}{4} \right).
$$

**Proof:** The bound follows by substituting all $\varepsilon_j^{(i)}$ in (3) by $\frac{\delta}{4}$ and all $z_j^{(i)}$ by $R$.

**Lemma 3:** Let $\delta > 0$ be fixed, and let $R$ be as in Lemma 2 with the additional assumption that $R \leq \frac{n \delta}{4}$. If $M \geq \frac{n - 1}{\delta} + n(\frac{R}{8} + \frac{1}{4})$, then

$$
\tilde{r} = \frac{1}{n - 1} \sum_{i=1}^{n} \varphi_\delta^{-1} \left(\sum_{j=1}^{n} \varphi_\delta(z_j^{(1)}) \varphi_\delta(z_j^{(2)})\right).
$$

**Proof:** First note that the assumptions imply that $M \geq R^2$. Thus, $z_j^{(i)} \in \mathbb{Q}(\langle M, \delta \rangle)$ for every $j$. Lemma 1 then ensures that $\varphi_\delta(z_j^{(1)}) \varphi_\delta(z_j^{(2)}) = \varphi_\delta(z_j^{(1)} \varphi_\delta(z_j^{(2)})$. Now, since the true sample correlation satisfies $|r| \leq 1$, we can combine this with Lemma 2 to obtain

$$
(n - 1)|\tilde{r}| \leq (n - 1)|r| + n \delta \left( R + \frac{\delta}{4} \right) \leq \delta^2 M.
$$

This implies that $(n - 1)|\tilde{r}| \leq \mathbb{Q}(\langle M, \delta \rangle)$, and the result follows from the first part of the proof combined with Lemma 1 applied to $(z_j^{(1)} \varphi_\delta(z_j^{(2)}))_{j=1}^{n}$.

The proposed procedure is summed up in Protocol [1]

**A. Correctness**

The correctness of Protocol [4] follows by noticing that the assumptions on $M$, $\delta$ and $R$ imply $\varphi_\delta^{-1}(a) = (n - 1)\tilde{r}$ by Lemma 3. Thus, the value output by the participants is exactly the same as in (4).
Fig. 1. Illustration of the fixed-point numbers in $\mathbb{Q}(M, \delta)$ within the real numbers.

**Protocol 1: Exact correlation with controlled leakage**

This protocol allows two participants $P_1$ and $P_2$ to compute their exact sample correlation $r$. Before the protocol, the participants have agreed on parameters $R, M, \delta$ such that $|\tilde{z}^{(i)}| \leq \frac{\delta}{3} \leq n$ for all $i, j$, and such that $(n-1)/\delta^2 + n(R/\delta + 1/4) \leq M = (p-1)/2$ for some prime $p$. The participants are assumed to have access to a functionality $F_{BT}$ for generating Beaver triples.

1. $P_1$ locally computes $s^{(i)} = \sqrt{\frac{1}{n-1} \sum_{j=1}^{n} (x_j^{(i)} - \bar{x})^2}$ and $z_j^{(i)} = (x_j^{(i)} - \bar{x})/s^{(i)}$. It then approximates each $z_j^{(i)}$ by a fixed-point number $\tilde{z}_j^{(i)} \in \mathbb{Q}(M, \delta)$ like in \( \Box \). $P_1$ sends $(\tilde{z}_j^{(i)})_{j=1}^{n}$ to $P_{3-i}$.

2. From the $z_j^{(3-i)}$ received in the previous step, $P_1$ locally computes $\sum_{j=1}^{n} z_j^{(i)} z_j^{(3-i)}$ and reveals the result to $P_{3-i}$.

3. For each $j \in \{1, 2, \ldots, n\}$, $P_1$ locally computes $\sum_{i=1}^{n} z_j^{(i)} z_j^{(3-i)}$ and reveals the result to $P_{3-i}$, and both participants recover $a$.

4. Each participant outputs $(\varphi^{(1)}_r(a) + \sum_{i=1}^{n} z_j^{(i)} z_j^{(3-i)})/\sum_{i=1}^{n} z_j^{(i)} z_j^{(3-i)}$.

**B. Privacy**

**Proposition 1:** After performing steps 1 and 2 steps 3-6 of Protocol 1 reveal no more than the true sample correlation.

**Proof:** We consider the real world, where participants follow Protocol 1 with the help of the functionality $F_{BT}$ for generating Beaver triples and compare this against the ideal world, where computation of $r$ is performed by the ideal functionality $F_r$, and protocol messages are generated by some simulator $S$. We give an explicit description of a simulator $S$ such that the two settings are perfectly indistinguishable. For notational ease, we assume that $P_1$ is passively corrupt. The case where $P_2$ is corrupt follows by symmetry.

In step 3, $S$ chooses $[\varphi_\delta(\tilde{z}_j^{(2)})]_1$ uniformly at random in $\mathbb{F}_p$. These are distributed identically to the shares that $P_1$ would receive in the ideal world.

Moving on to step 4, the Beaver triples $([u], [v], [w])$ are generated by $S$, and $P_1$ receives the corresponding shares. Since $\varphi_\delta(\tilde{z}_j^{(1)}) - u$ is revealed during computation of the product $[\varphi_\delta(\tilde{z}_j^{(1)})][\varphi_\delta(\tilde{z}_j^{(2)})]$ and $u$ is known to $S$, it can extract $\varphi_\delta(\tilde{z}_j^{(1)})$ and thereby compute $\tilde{z}_j^{(1)}$ for each $j \in \{1, 2, \ldots, n\}$. In addition, note that this information also allows $S$ to compute $[a]_1$. It passes $(\tilde{z}_j^{(1)})_{j=1}^{n}$ as inputs to $F_r$ and receives the output $r$.

In step 5, $S$ sets

$$[a]_2 = \varphi_\delta ((n-1)r - \sum_{i=1}^{n} z_j^{(3-i)} - \sum_{j=1}^{n} z_j^{(1)} z_j^{(2)}) - [a]_1$$

and sends this value to $P_1$. By 4, this implies $[a]_1 + [a]_2 = (n-1)r$ like in the real world, and furthermore, there is only one choice of $[a]_2$ given $r$ and $[a]_1$.

In summary, the messages produced by $S$ are perfectly indistinguishable from those seen in a real execution of Protocol 1, proving the result.

To illustrate the use of Protocol 1 we give two examples. The first focuses on the procedure itself and the second illustrates the kind of information that is leaked during the protocol.

**Example 1:** Assume that $P_1$ and $P_2$ hold samples $x^{(1)} = (2.113, -0.906, -0.546, -1.550, 1.770, 4.002, -0.135, 0.606)$ and $x^{(2)} = (0.647, -2.108, -0.479, -1.751, 0.442, 2.105, -0.836, -1.748)$ respectively. Computing their $z$-scores in Step 1 of Protocol 1, they obtain (if rounded to three decimal places)

$$z^{(1)} = (0.781, -0.852, -0.657, -1.200, 0.595, 1.802, -0.435, -0.034)$$

$$z^{(2)} = (0.765, -1.129, -0.009, -0.884, 0.624, 1.768, -0.254, -0.882)$$

Using scaling factor $\delta = 0.1$ and setting $R = 2.5$, we must choose $M \geq 902$ to satisfy the assumptions of the protocol. The smallest choice of prime is then $p = 1811$. As described in Remark 1 this implies $M = 905$.

With these parameters in place, $P_1$ rounds the entries of $z^{(1)}$ to the nearest elements in $\mathbb{Q}(M, \delta)$, obtaining

$$\tilde{z}^{(1)} = (0.8, -0.9, -0.7, -1.2, 0.6, 1.8, -0.4, 0.0).$$

When this is mapped to $\mathbb{F}_{1811}$, it yields $\varphi_\delta(\tilde{z}^{(1)}) = (8, 1802, 1804, 1799, 6, 18, 1807, 0) \in \mathbb{F}_{1811}^8$. Similarly, $P_2$ ends up computing $\varphi_\delta(\tilde{z}^{(2)}) = (8, 1800, 0, 1802, 6, 18, 1808, 1802) \in \mathbb{F}_{1811}^8$. While performing these computations, they have also recorded and revealed the error vectors $\varepsilon^{(1)} = z^{(1)} - \tilde{z}^{(1)}$ as well as the values $\sum_{j=1}^{n} z_j^{(1)} z_j^{(2)} \approx -0.079$ and $\sum_{j=1}^{n} z_j^{(2)} z_j^{(1)} \approx -0.029$. 


They then create sharings of these, and use the Beaver triples to compute shares of the entrywise product \( y = \varphi_y(\tilde{z}(1)) \circ \varphi_y(\tilde{z}(2)) \), giving

\[
y = (64, 99, 0, 108, 36, 324, 12, 0) \in \mathbb{F}_{1811}^8.
\]

Now, \( P_1 \) locally computes the sum \( \sum_{j=1}^n y_j \), and reveals this in step 5. Thus, both participants recover \( \sum_{j=1}^n y_j = 643 \). Mapping this back to \( \mathbb{Q}(M, \delta) \), they get \( \varphi_{\delta}^{-1}(643) = 6.43 \). Since the error vectors have previously been revealed, the participants are able to locally compute the sum \( \sum_{j=1}^n \varepsilon_j^{(1)} \varepsilon_j^{(2)} \approx 0.007 \). Combining all of this, they recover

\[
\frac{1}{n-1} \left( \varphi_{\delta}^{-1}(643) + \sum_{i=1}^{n} z_j^{(1)} \varepsilon_j^{(3-i)} + \sum_{j=1}^{n} \varepsilon_j^{(1)} \varepsilon_j^{(2)} \right) \\
\approx \frac{1}{7} (6.43 - 0.079 - 0.029 + 0.007) \approx 0.904,
\]

which is equal to the exact sample correlation apart from errors introduced by the rounding used in this exposition.

**Example 2**: We consider the same protocol execution as in Example 1 and analyse it from the view of \( P_2 \). During the protocol, \( P_2 \) learns \( \varepsilon_j^{(1)} \sum_{j=1}^n \varepsilon_j^{(2)} \), and \( \sum_{j=1}^n \varepsilon_j^{(1)} \varepsilon_j^{(2)} \). By combining these, \( P_2 \) can form the system

\[
\begin{cases}
\sum_{j=1}^n \varepsilon_j^{(1)} \varepsilon_j^{(2)} = \sum_{j=1}^n \varepsilon_j^{(1)} (1) - \sum_{j=1}^n \varepsilon_j^{(2)} (2) \\
\sum_{j=1}^n \varepsilon_j^{(1)} \varepsilon_j^{(2)} = \sum_{j=1}^n \varepsilon_j^{(1)} (2) - \sum_{j=1}^n \varepsilon_j^{(2)} (1)
\end{cases}
\]

where we note that the right-hand side is known to \( P_2 \). Hence, from the point of view of \( P_2 \), \( 5 \) is a linear system of two equations in \( n = 8 \) unknowns \( z_1^{(1)}, z_2^{(1)}, \ldots, z_8^{(1)} \). Solving this with the data samples from Example 1 yields six free variables \( \tilde{z}(1) = (3, 0, 1, 0, 0, 0, 0, 0) \), but not all assignments of values to these give a solution that is consistent with \( \tilde{z}^{(1)} \in \mathbb{Q}(M, \delta) \). In addition, \( P_2 \) will know the parameter \( R = 2.5 \) used in the protocol, meaning additionally that \( |z_j^{(1)}| \leq 2.5 \). Since this example is relatively small, we can go through all possible assignments of \( \tilde{z}(1) \) satisfying these conditions and record those that also imply \( \tilde{z}_1^{(1)}, \tilde{z}_2^{(1)} \in Q(M, \delta) \cap [-2.5, 2.5] \). There are a total of 3624355 such solutions, when accepting a small additive error (10^{-12}) in each variable to allow for precision loss in the floating-point computations. Figure 2 on page 6 shows the frequencies of values for each variable in these solutions as well as the corresponding analysis from the view of \( P_1 \). This illustrates, that the participants cannot determine the actual data held by the other participant, but may gain some insights about the distribution of the individual sample points.

Whether the leakage from Protocol 1 is acceptable or not depends on the use case as well as the distributions of the participants. As the example below shows, it may lead to a complete leakage in certain cases.

**Example 3**: Consider the case where the samples of \( P_2 \) come from a discrete distribution whose probability mass function has exactly two points in its support. That is, \( x_i^{(2)} \in \{a, b\} \) for some real numbers \( a \neq b \). In this example, we assume that \( P_1 \) somehow knows that the distribution of \( P_2 \) has this form.

Writing \( a = \tilde{a} + \varepsilon_a \) for \( \tilde{a} \in \mathbb{Q}(M, \delta) \) like in 4, and denoting by \( J_a \) the indices \( j \) such that \( x_j^{(2)} = a \), we have that \( \varepsilon_j^{(2)} = \varepsilon_a \) for all \( j \in J_a \). With similar notation for \( b \), we get \( \varepsilon_j^{(2)} = \varepsilon_b \) for all \( j \in J_b \). Thus, unless \( \varepsilon_a = \varepsilon_b \), \( P_1 \) can determine \( J_a \) and \( J_b \) from \( (\varepsilon_j^{(2)})^n_{j=1} \). During Protocol 1, \( P_1 \) learns the equation \( \sum_{j=1}^n \tilde{z}_j^{(1)} \varepsilon_j^{(2)} = r \) and the value of \( \sum_{j=1}^n \tilde{z}_j^{(1)} \varepsilon_j^{(2)} \). Combining this with the knowledge of \( J_a \) and \( J_b \) leads to the system

\[
\begin{cases}
\sum_{j \in J_a} \varepsilon_j^{(1)} + \sum_{j \notin J_a} \varepsilon_j^{(1)} = r \\
\sum_{j \in J_a} \varepsilon_j^{(2)} + \sum_{j \notin J_a} \varepsilon_j^{(2)} = \sum_{j=1}^n \tilde{z}_j^{(1)} \varepsilon_j^{(2)},
\end{cases}
\]

which from the point of view of \( P_1 \) is a system of two linear equations in two unknowns, \( a \) and \( b \). Unless these equations are linearly dependent, \( P_1 \) can determine \( a \) and \( b \) uniquely, and thus fully learns the data of \( P_2 \). Note, however, that this attack relies on \( P_1 \) knowing that the samples of \( P_2 \) have this special form.

**C. Complexity**

We now turn our attention to the cost of Protocol 1 in terms of computation and communication. For this analysis, we consider one addition or multiplication in \( \mathbb{R} \) or \( \mathbb{F}_p \) to be a single operation, and we will count \( \mathbb{R} \)-operations and \( \mathbb{F}_p \)-operations separately. Similarly, we count the transmission of symbols from \( \mathbb{R} \) and \( \mathbb{F}_p \) separately. We do not include the cost of generating the Beaver-triples in this analysis since that will depend on the specific implementation of \( \mathcal{F}_{BT} \).

**Proposition 2**: During Protocol 1 the participants consume \( n \) Beaver-triples. Furthermore, each participant

- performs \( O(n) \) \( \mathbb{R} \)-operations and \( O(n) \) \( \mathbb{F}_p \)-operations
- transmits \( O(n) \) elements of \( \mathbb{R} \) and \( O(n) \) elements of \( \mathbb{F}_p \)

**Proof**: In step 1., the computation of \( s^{(i)} \) and \( z_j^{(1)} \) takes a total of \( 6n + O(1) \) \( \mathbb{R} \)-operations in \( \mathbb{R} \) for each participant, where the constant term covers the cost of computing the square root in \( s^{(i)} \). In addition, each participant transmits the \( n \) error values. In the next step, computing the sum requires \( 2n - 1 \) \( \mathbb{R} \)-operations, and revealing it requires transmission of one element of \( \mathbb{R} \). Then in step 3., creating each of the \( n \) sharings require \( O(1) \) \( \mathbb{F}_p \)-operations, and distributing it requires transmission of a single \( \mathbb{F}_p \)-element. Moving to step 4., each product computed using Beaver triples costs 8 operations in \( \mathbb{F}_p \) as well as transmission of 2 \( \mathbb{F}_p \)-elements. Thus, the computation of \( a \) gives a total of \( 9n - 1 \) \( \mathbb{F}_p \)-operations and transmission of \( 2n \) \( \mathbb{F}_p \)-elements. When revealing \([a]\), afterwards, this is the transmission of a single element of \( \mathbb{F}_p \). Finally, for the computation of the output, observe that the participants already know the values of \( \sum_{j=1}^n \varepsilon_j^{(i)} \varepsilon_j^{(3-j)} \). Furthermore, the computational cost of computing \( \varphi_{\delta}^{-1}(a) \) is independent of \( n \), meaning that the participants can compute the result using \( 2n + O(1) \) \( \mathbb{R} \)-operations.

Summing all of these yields \( 10n + O(1) \) \( \mathbb{R} \)-operations, \( O(n) \) \( \mathbb{F}_p \)-operations as well as transmission of \( n + 1 \) and \( 3n + 1 \) elements of \( \mathbb{R} \) and \( \mathbb{F}_p \), respectively.

\[\text{In practice, the elements of } \mathbb{R} \text{ will be represented by floating-point numbers, but we ignore this detail here.}\]
Table 1: Leaked information in Example 2 from the view of $P_2$ (on the left) and $P_1$ (on the right). Each row corresponds to one of the variables $z_j^{(i)}$, and each column corresponds to one of the possible elements of $Q_i(m, d)$ as indicated by the scale on the left. The cell corresponding to $(a, z_j^{(i)})$ gives the relative frequency in per mille (%) of $z_j^{(i)} = a$ occurring in the valid solutions of Example 2. The cell colour indicates the ratio of the cell value to the maximum in the same column, with 0 being pure white and the maximum being pure black.

| $z_j^{(1)}$ | $z_j^{(2)}$ | $z_j^{(3)}$ | $z_j^{(4)}$ | $z_j^{(5)}$ | $z_j^{(6)}$ | $z_j^{(7)}$ | $z_j^{(8)}$ |
|------------|------------|------------|------------|------------|------------|------------|------------|
| 2.5        | 32.2       | 3.1        | 2.6        | 2.6        | 2.6        | 2.6        | 2.6        |
| 3.0        | 24.0       | 2.1        | 2.0        | 2.0        | 2.0        | 2.0        | 2.0        |
| 2.0        | 30.1       | 0.0        | 4.3        | 4.3        | 4.3        | 4.3        | 4.3        |
| 3.0        | 29.8       | 4.2        | 3.2        | 3.2        | 3.2        | 3.2        | 3.2        |
| 2.5        | 29.9       | 3.0        | 3.6        | 3.6        | 3.6        | 3.6        | 3.6        |
| 3.0        | 31.7       | 5.8        | 3.9        | 3.9        | 3.9        | 3.9        | 3.9        |
| 2.0        | 30.1       | 0.0        | 4.3        | 4.3        | 4.3        | 4.3        | 4.3        |
| 3.0        | 29.8       | 4.2        | 3.2        | 3.2        | 3.2        | 3.2        | 3.2        |
| 2.5        | 29.9       | 3.0        | 3.6        | 3.6        | 3.6        | 3.6        | 3.6        |
| 3.0        | 31.7       | 5.8        | 3.9        | 3.9        | 3.9        | 3.9        | 3.9        |

Fig. 2. Leaked information in Example 2 from the view of $P_2$ (on the left) and $P_1$ (on the right). Each row corresponds to one of the variables $z_j^{(i)}$, and each column corresponds to one of the possible elements of $Q_i(m, d)$ as indicated by the scale on the left. The cell corresponding to $(a, z_j^{(i)})$ gives the relative frequency in per mille (%) of $z_j^{(i)} = a$ occurring in the valid solutions of Example 2. The cell colour indicates the ratio of the cell value to the maximum in the same column, with 0 being pure white and the maximum being pure black.
V. APPROXIMATION OF THE CORRELATION

This section treats the case where the participants simply approximate their inputs as fixed-point numbers and performs MPC ‘as usual’ – that is, without ensuring functional privacy as in [13]. This essentially Protocol 2, but without the explicit disclosure of the error values. The resulting procedure is given in Protocol 2. Now, the result is no longer exact, but instead an approximation with an additive error as described by Lemma 2. As we show in Proposition 3, this approximation reveals no more than the error terms revealed in Protocol 1. In other words, Protocol 2 is at least as secure as Protocol 1. The correctness of Protocol 2 follows by Lemma 3 using similar arguments as in Section IV-A.

Proposition 3: Let $R, M, \delta$ be fixed. Protocol 2 reveals no more than Protocol 1 when the same parameters are used.

Proof: We consider a game, where a distinguisher inputs $(x_j^{(i)})_{j=1}^n$ for $i = 1, 2$ and receives a protocol transcript from the view of $P_i$ for some choice of $i \in \{1, 2\}$. The transcript is either from an execution of Protocol 2 or generated by some simulator $S$ that is given a transcript of Protocol 1 from the view of $P_i$ (with the same inputs). We give a simulator $S$ such that the two cases are perfectly indistinguishable.

First, notice that the shares revealed in steps 2 and 3 of Protocol 2 are perfectly indistinguishable from uniformly sampled elements of $\mathbb{F}_p$ by the security of the secret sharing scheme. In addition, observe that during the execution of Protocol 1, $P_i$ learns $a = \sum_{j=1}^n \varphi_i(x_j^{(1)}) \varphi_i(x_j^{(2)})$, which is identical to the $a$ recovered in step 3 of Protocol 2. Thus, by creating a random sharing of $a$, $S$ can generate something that has the same distribution as the shares $[a]_1$ and $[a]_2$ revealed in step 4 of Protocol 2.

Proposition 4: During Protocol 2, the participants consume $n$ Beaver-triples. Furthermore, each participant
- performs $O(n)$ $\mathbb{R}$-operations and $O(n)$ $\mathbb{F}_p$-operations
- transmits $O(n)$ elements of $\mathbb{F}_p$ (and no elements of $\mathbb{R}$)

We omit the proof since it is essentially the same considerations in the proof of Proposition 2.

Example 4: If the participants in Example 1 had instead used Protocol 2, they would not learn the information regarding the error values of the other participant. Thus, they would instead output $\varphi \approx (643)/7 \approx 0.92$ rather than the exact sample correlation. Since the scaling factor is relatively small ($\delta = 0.1$), the maximal error is relatively large at approximately 0.28 according to Lemma 2. Thus, the participants can only conclude that the sample correlation is in the interval $[0.64, 1]$.

VI. BOUNDS IN PRACTICAL IMPLEMENTATIONS

By choosing $p = 4294967291$, calculations in $\mathbb{F}_p$ can be implemented using standard 64-bit unsigned integers, and $p$ is the largest such prime. By rewriting the bound on $M$ from Protocols 1 and 2, we obtain

$$0 \leq \delta^2 \left(M - \frac{n}{4}\right) - \delta nR - n + 1. \quad (6)$$

Thus, by setting $M = \frac{p-1}{4}$ and picking specific choices for $n$ and $R$, we can find the smallest value $\delta_{\min}$ satisfying both $\delta_{\min} \geq 0$ and (6). Doing so yields

$$\delta_{\min} = \frac{-nR + \sqrt{n^2R^2 + 4(M - n/4)(n - 1)}}{2(M - n/4)},$$

and in Table I we list values of $\delta_{\min}$ for various values of $n$ and $R$ along with the maximal error as given in Lemma 2.

VII. OPEN PROBLEMS

In the more general case where each participant holds a multivariate random variable – i.e. $x_j^{(i)} \in \mathbb{R}^{n_j}$ for some $n_1, n_2$ – the correlation between them can be represented in a matrix

$$M = \begin{bmatrix} M_1 & M_{12} \\ M_{12} & M_2 \end{bmatrix}$$

where $M_1$ and $M_2$ depend only on the data held by $P_1$ and $P_2$, respectively, and $M_{12}$ contains the ‘mixed’ correlations. Hence, rather than computing a single correlation as in (1), the goal is to compute $M_{12}$. Of course, Protocols 1 and 2 can be used to compute each entry of this matrix. Before doing so, however, a more detailed of the information leakage is required, as each variable is part of the computation in several entries. For example, focusing on a single variable $(x_j^{(1)})_k$ held by $P_1$, computing the correlation with each of the $n_2$ variables of $P_2$ will reveal another two linear equations describing $(x_j^{(1)})_k$. Although these equations might not be linearly independent, they still represent a more severe information leakage than in the univariate case. We refrain from making such an analysis in the current work.

In addition to multidimensional data, another direction is the computation of other similarity measures, such as the statistical distance, e.g. the Kullback-Leibler (KL) divergence. This will also require computations on data points where each component is held by different participants. So this problem exhibits a similar structure, but will require a separate analysis of the necessary MPC techniques and the resulting error bounds.

VIII. CONCLUSION

In this work, we presented two methods for computing sample correlation. While not secure in the formal sense – as in functional privacy – the information that is leaked can be described precisely. In certain practical problems, such security guarantees might be sufficient. In addition, we demonstrated that even with practical field sizes, the approximation error is not too large.

REFERENCES

[1] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, “A survey on big data market: Pricing, trading and protection,” IEEE Access, vol. 6, pp. 15132–15154, 2018.
[2] D. Acemoglu, A. Makhdoumi, A. Malekian, and A. Ozdaglar, “Too much data: Prices and inefficiencies in data markets,” tech. rep., National Bureau of Economic Research, 2019.
[3] A. Agarwal, M. Dahleh, and T. Sarkar, “A marketplace for data: An algorithmic solution,” in Proceedings of the 2019 ACM Conference on Economics and Computation, pp. 701–726, 2019.
This protocol allows two participants $P_1$ and $P_2$ to compute an approximation of their sample correlation $r$. Before the protocol, the participants have agreed on parameters $R, M, \delta$ such that $|\hat{z}(i)| \leq R \leq \frac{\sqrt{n}}{\delta}$ for all $i,j$, and such that $(n-1)/\delta^2 + n(R/\delta + 1/4) \leq M = (p-1)/2$ for some prime $p$.

1. $P_i$ locally computes $s(i) = \sqrt{\frac{1}{n-1} \sum_{j=1}^n (x_j(i) - \bar{x})^2}$ and $z(i) = (x_j(i) - \bar{x}(i))/s(i)$.
2. For each $j \in \{1, 2, \ldots, n\}$, $P_i$ creates and distributes sharings $[\varphi_{\hat{z}(j)}]$.
3. The participants compute $[a] = \sum_{j=1}^n [\varphi_{\hat{z}(j)}][\varphi_{\hat{z}(j)}^2]$ using Beaver-triples and local addition of shares.
4. $P_i$ reveals its share $[a]$, to $P_{3-i}$, and each participant reconstructs and outputs $\varphi_{\hat{z}(j)}^{-1}(a)/(n-1)$.

### Table 1

| $R$ | $\delta_{\text{min}}$ | $\text{err}_{\text{max}}$ | $\delta_{\text{min}}$ | $\text{err}_{\text{max}}$ | $\delta_{\text{min}}$ | $\text{err}_{\text{max}}$ |
|-----|----------------------|--------------------------|----------------------|--------------------------|----------------------|--------------------------|
| 2.5 | 100                  | 2.15 \cdot 10^{-4}      | 5.42 \cdot 10^{-4}   | 2.15 \cdot 10^{-4}      | 1.08 \cdot 10^{-3}   | 2.14 \cdot 10^{-4}      | 1.27 \cdot 10^{-3}   |
| 5   | 1000                 | 6.81 \cdot 10^{-3}      | 3.41 \cdot 10^{-3}   | 6.80 \cdot 10^{-3}      | 6.80 \cdot 10^{-3}   |                          |
| 10  | 100000               | 2.15 \cdot 10^{-3}      | 1.07 \cdot 10^{-2}   | 2.13 \cdot 10^{-3}      | 2.13 \cdot 10^{-2}   |                          |
| 100 | 1000000              | 6.71 \cdot 10^{-3}      | 3.36 \cdot 10^{-2}   | 6.60 \cdot 10^{-3}      | 6.60 \cdot 10^{-2}   |                          |

Minimal scaling factor and maximal error for various choices of $n$ and $R$, when $p = 4294967291$. 

[4] P. Pinson, L. Han, and J. Kazempour, “Regression markets and application to energy forecasting,” 2022.

[5] A. Ghorbani and J. Zou, “Data shapely: Equitable valuation of data for machine learning,” in International Conference on Machine Learning, pp. 2242–2251, PMLR, 2019.

[6] D. Fudenberg and T. Tirole, Game theory. MIT press, 1991.

[7] A. E. Roth, “Game theory as a tool for market design,” in Game practice: Contributions from applied game theory, pp. 7–18, Springer, 2000.

[8] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele, “Secure computation on floating point numbers,” in Network & Distributed System Security Symposium, NDSS 2013, The Internet Society, 2013.

[9] L. Kanum and J. Willemsen, “Secure floating point arithmetic and private satellite collision analysis,” International Journal of Information Security, vol. 14, pp. 531–548, Nov 2015.

[10] O. Catrina and A. Saxena, “Secure computation with fixed-point numbers,” in Financial Cryptography and Data Security (R. Sion, ed.), (Berlin, Heidelberg), pp. 35–50, Springer Berlin Heidelberg, 2010.

[11] E. Roth, “Game theory as a tool for market design,” in Game practice: Contributions from applied game theory, pp. 7–18, Springer, 2000.

[12] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Dooer, S. Zahur, and D. Evans, “Privacy-preserving distributed linear regression on high-dimensional data,” Proceedings on Privacy Enhancing Technologies, vol. 2017, no. 4, pp. 345–364, 2017.

[13] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright, “Secure multiparty computation of approximations,” ACM Trans. Algorithms, vol. 2, pp. 435–472, Jul 2006.

[14] K. Tjell and R. Wisniewski, “Privacy in Distributed Computations based on Real Number Secret Sharing,” arXiv e-prints, p. arXiv:2107.09911, July 2021.

[15] P. Pullonen and S. Siim, “Combining secret sharing and garbled circuits for efficient private ioe: 754 floating-point computations,” in Financial Cryptography and Data Security (M. Brenner, N. Christin, B. Johnson, and K. Rohloff, eds.), (Berlin, Heidelberg), pp. 172–183, Springer Berlin Heidelberg, 2015.

[16] V. Dimitrov, L. Kerik, T. Krips, J. Randmets, and J. Willemsen, “Alternative implementations of secure real numbers,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, (New York, NY, USA), p. 553–564, Association for Computing Machinery, 2016.

[17] X. Liu, R. H. Deng, W. Ding, R. Lu, and B. Qin, “Privacy-preserving outsourced calculation on floating point numbers,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 11, pp. 2513–2527, 2016.

[18] B. Kuskonnaz, J. Skovsted Gundersen, and R. Wisniewski, “Investigation of Alternative Measures for Mutual Information,” arXiv e-prints, p. arXiv:2202.00956, Feb. 2022.