Confidential Truth Finding with Multi-Party Computation

Angelo Saadeh[1,6], Pierre Senellart[2,4,5,6,7], and Stéphane Bressan[3,6,7]

[1] LTCI, Télécom Paris, IP Paris, Palaiseau, France
angelo.saadeh@telecom-paris.fr
[2] DI ENS, ENS, PSL University, CNRS, Paris, France
pierre@senellart.com
[3] National University of Singapore, Singapore
steph@nus.edu.sg
[4] Inria, Paris, France
[5] Institut Universitaire de France
[6] CNRS@CREATE LTD, Singapore
[7] IPAL, CNRS, Singapore

Abstract. Federated knowledge discovery and data mining are challenged to assess the trustworthiness of data originating from autonomous sources while protecting confidentiality and privacy. Truth-finding algorithms help corroborate data from disagreeing sources. For each query it receives, a truth-finding algorithm predicts a truth value of the answer, possibly updating the trustworthiness factor of each source. Few works, however, address the issues of confidentiality and privacy. We devise and present a secure secret-sharing-based multi-party computation protocol for pseudo-equality tests that are used in truth-finding algorithms to compute additions depending on a condition. The protocol guarantees confidentiality of the data and privacy of the sources. We also present a variants of a truth-finding algorithm that would make the computation faster when executed using secure multi-party computation. We empirically evaluate the performance of the proposed protocol on a state-of-the-art truth-finding algorithm, 3-Estimates, and compare it with that of the baseline plain algorithm. The results confirm that the secret-sharing-based secure multi-party algorithms are as accurate as the corresponding baselines but for proposed numerical approximations that significantly reduce the efficiency loss incurred.

Keywords: truth finding · secure multi-party computation · secret-sharing · uncertain data · privacy.

1 Introduction

Truth-finding algorithms [6] help corroborate data from disagreeing sources. For each query it receives, a truth-finding algorithm predicts a truth value of the answer, possibly updating the trustworthiness factor of each source. Few works, however, address the issues of confidentiality and privacy. We consider the design

and implementation of truth-finding algorithms that protect the confidentiality of sources' data, using secret-sharing-based secure multi-party computation [3], or simply secure multi-party computation (MPC).

We devise and present a secure multi-party pseudo-equality protocol that securely computes additions depending on a condition – we call them conditioned additions – for truth-finding algorithms. In particular, we present a secure equality test alternative that uses a polynomial evaluation to reduce the number of communication; this is used for conditioned additions, an operation that is an essential building block of many truth-finding algorithms. The protocol guarantees the confidentiality of the data. We also devise several variants of privacy-preserving truth-finding algorithms; ones that implement the truth-finding algorithms without changes, and others with modifications that aim to make the computation more efficient.

The secure multi-party protocols are then implemented with two servers. We empirically evaluate the performance of the proposed protocol on a state-of-the-art truth-finding algorithm, 3-Estimates [4, Algorithm 4] (see also [2, 5] for further experiments on this algorithm), and compare it with that of the non-secure baseline algorithms. The results confirm that the secure multi-party algorithm is as accurate as the corresponding baseline except for proposed modifications to reduce the efficiency loss incurred.

Set $n \in \mathbb{N}^*$, and let $\mathcal{V}$ be a set of n sources. The client would like to label k queries (or facts) $\{f^1, ..., f^k\}$. A truth-finding algorithm outputs a truth value for a query when different data sources (or sources) provide disagreeing information on it. Concretely, the truth-finding algorithm takes $v^1, ..., v^n$ as input with $v^i \in \{-1, 0, 1\}^k$, and outputs estimated truth values in $[-1, 1]^k \subset \mathbb{R}^k$ or $[0, 1]^k \subset \mathbb{R}^k$ depending on the truth-finding algorithm.

Truth-finding (or truth discovery) algorithms [6] are usually run by the client in order to know the truth value of a given query when the sources give disagreeing answers. That is, for each of the client's queries, each source in $\mathcal{V}$ delivers an answer v^i such that an output of 1 corresponds to a positive answer, -1 to a negative one, and 0 if the source does not wish to classify the data point. 3-Estimates [4] is a truth-finding algorithms that given a number of queries k, output a truth value in the range $[-1, 1]^k \subset \mathbb{R}$ and a trust coefficient in each of the sources, or sources. In addition, 3-Estimates computes an estimate of the difficulty of each query.

The goal of this work is to execute truth-finding algorithms that protect sources' data using secure multi-party computation (MPC) [1,3]. More generally, given a function F and a set of private inputs $x^1, ..., x^m$ respectively owned by $P_1, ..., P_m$, MPC is a cryptographic approach that makes it possible to compute the output of the function $F(x^1, .., x^m)$ without resorting to a third party that would compute the function F and would send the result back. MPC will be used to implement the 3-Estimates algorithm without having any source disclose their answer.

Because of lack of space, details are ommitted. An extended version is available as [8], which also covers another truth finding algorithm, Cosine, from [4].

2 Proposed Approach

The first task we wish to achieve is private voting, i.e., the client sends queries to each source, and the source classifies the query. In the case where the query is a vector of features and the models are logistic regressions, existing MPC works [7] can keep the query private. We suppose that the answers are already computed and secret-shared on two servers P_1 and P_2 using a two-party additive secret sharing. In other words, P_1 holds v_1^{ij} and P_2 holds v_2^{ij} such that $v^{ij} = v_1^{ij} + v_2^{ij}$ is the ith source's answer for the query f^j and is equal to -1, 0, or 1.

The second step which is the aggregation of the data (the answers) is computed on the two servers P_1 and P_2. The problem is now constructing a secure two-party computation algorithm with additively shared data that implements the truth-finding algorithms using their arithmetic circuits. Once the circuits are evaluated, the two servers (P_1 and P_2) send their share of the output to the client who reconstructs it by adding the received shares together.

Other than additions and multiplications, the truth-finding algorithm we implement – 3-Estimates – uses existing real-number operations like division, and square root, which are dealt with in standard ways [8]. We focus on computing conditioned sums by replacing equality tests with degree-two polynomial evaluations.

The truth-finding algorithms we use require conditioned additions. Given two vectors of same size $t = (t^1, ..., t^k) \in \mathbb{R}^k$, $z = (z^1, ..., z^k) \in \{-1, 0, 1\}^k$, and an element $\kappa \in \{-1, 0, 1\}$, we define the following operation: $S \coloneqq \sum_{i:z^i=\kappa} t^i$. In other words, the ith element of t, t^i, is added to the sum only if the ith element of z, z^i, is equal to κ. The difficulty is that even though κ is public, z^i is private. To achieve this in MPC we start by defining the following function, for $i \in \{1, ..., r\}$:

$$\mathcal{E}(z^i, \kappa) = \begin{cases} 1 \text{ if } z^i = \kappa \\ 0 \text{ if not.} \end{cases}$$

A naive way to compute the sum S is as follows: $S = \sum_i \mathcal{E}(z^i, \kappa) \cdot t^i$. This way to compute S requires an equality test which is costly in MPC. To this end, we propose an alternative that makes good use of the fact that $z^i, \kappa \in \{-1, 0, 1\}$. The goal is to express the function $\mathcal{E}$ as a polynomial so that it can be computed using the smallest number of additions and multiplications possible. We define and use the following expressions of $\mathcal{E}(z^i, \kappa)$.

If $\kappa = -1$, we compute S as follows: $S = \sum_i \frac{1}{2}((z^i)^2 - z^i) \cdot t^i$. We have:

$$\frac{1}{2}((z^i)^2 - z^i) = \begin{cases} 1 & \text{if } z^i = -1 \\ 0 & \text{if } z^i = 0 \\ 0 & \text{if } z^i = 1 \end{cases}$$

Hence, by multiplying $\frac{1}{2}((z^i)^2 - z^i)$ by t^i, the only elements considered in the sum are the ones such that $z^i = -1$. The function $\frac{1}{2}((z^i)^2 - z^i)$ is equal to $\mathcal{E}(z^i, -1)$. If $\kappa = 0$ we similarly compute S as: $S = \sum_i (1 - (z^i)^2) \cdot t^i$. It is also straightforward that the function $1 - (z^i)^2$ is equal to $\mathcal{E}(z^i, 0)$ because it

outputs 1 if $z^i = 0$ and 0 otherwise. If $z = 1$, in the same way, S is computed as: $S = \sum_i \frac{1}{2}((z^i)^2 + z^i) \cdot t^i$.

Lemma 1 (Conditioned additions). *Denote by $\Pi_{\mathcal{E}}$ the MPC protocol implementing the function $\mathcal{E}$ using the three previously defined degree-2 polynomials. $\Pi_{\mathcal{E}}$ does not reveal information about the other's player's share.*

Proof. The three conditioned sums defined in this section do not need comparisons and they are expressed using only additions and multiplications, so their security level is the same as Π_{add} and Π_{mul}. □

See [8] for how this allows us to reformulate the truth finding algorithm for 3-Estimates with MPC.

Normalization in 3-Estimates. In the 3-Estimates algorithm, the truth value, trust factor, and difficulty score need to be normalized at each step. This could be done using a secure comparison protocol to securely compute the minimum and the maximum of each value, and then normalize them as it is done in [4]. Secure comparisons however are very costly in MPC. To reduce the amount of communication we replace the normalization based on finding the maximum and minimum by a pre-computed linear transformation which forces the values to stay between 0 and 1. Concretely we apply the function $h(x) = 0.5x + 0.25$ to all the values after each update. We evaluate the impact of this change in the experiments. The chosen function, h, is not perfect. Indeed, if we have information about the distribution of the parameters, we can pre-compute a linear normalization for every iteration. Using any public pre-computed or predefined normalizing function improves the efficiency of the algorithm because it would translate to using multiplication and addition by public constants, which is communication-free.

3 Experimental Results

We evaluate our protocol on two computing servers. We suppose that the sources have already answered and secret-shared their answers. We use the ring $\mathbb{Z}_{2^{60}}$ with 20 bits of fixed precision. The two servers communicate via a local socket network implemented in Python on an Intel Core i5-9400H CPU (2.50 GHz × 8) and a RAM of 15.4 GiB. For the sake of the experiment, these communications are not encrypted or authenticated.

We implement our solution using the dataset Hubdub from [4].[8] This dataset is constructed from 457 questions from a Web site where users had to bet on future events. As the questions had multiple answers, they have been increased to 830 questions to obtain binary questions with answers $-1, 0$ or 1. The client sends the 830 queries to be classified by each source, and after the classification,

[8] Datasets used, as well as the source code of our implementation, are available at `https://github.com/angelos25/tf-mpc/`.

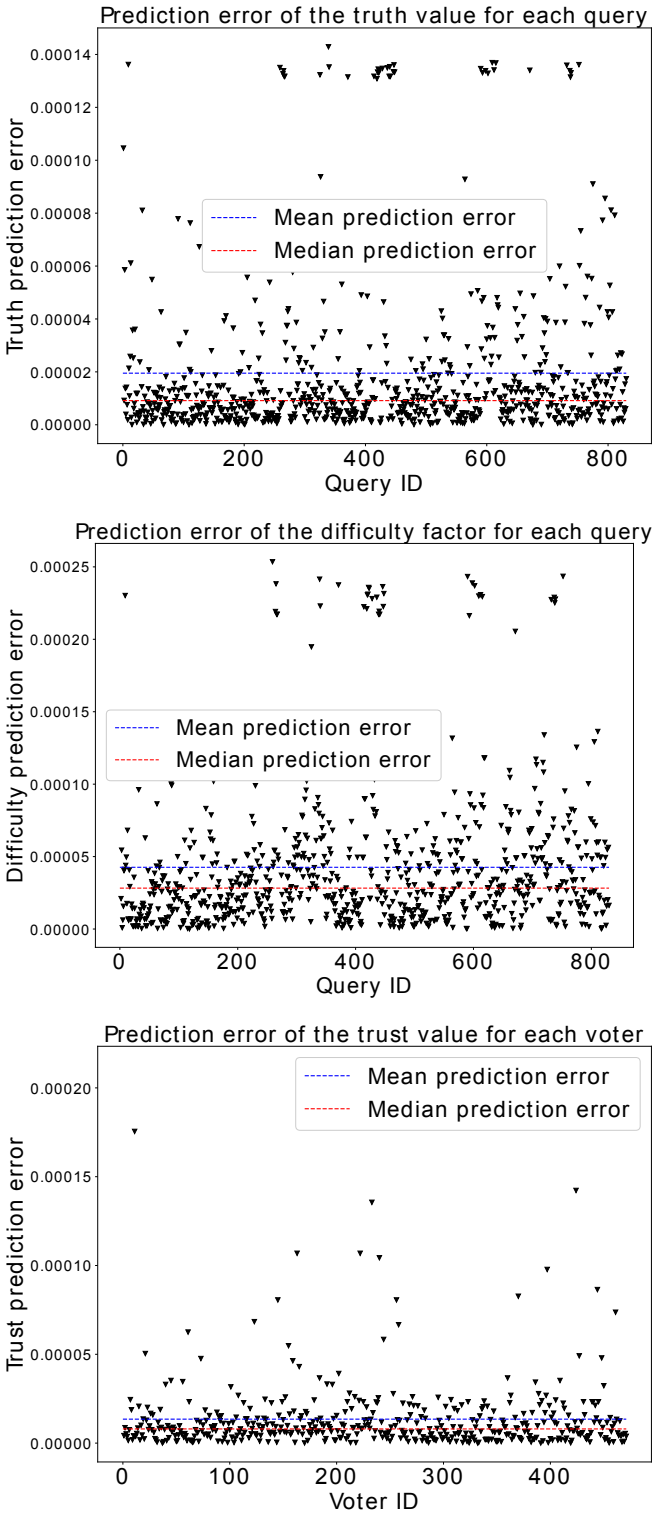

Fig. 1. Prediction errors between secure multi-party computation and the base model results with 3-Estimates on Hubdub dataset.

the sources secret-share them on two servers to evaluate using MPC the 3-Estimates truth-finding algorithm. At the end of the evaluation, the results are reconstructed by the client. The results include the truth value for each query (the label), a difficulty score for each query, and a trustworthiness factor for each of the 471 sources. In Fig. 1 we show the difference between the predictions from the base model and the predictions from the MPC evaluation. The base model corresponds to the 3-Estimates algorithm implemented without MPC on the plain data. The MPC evaluation contains errors compared to the base model, and these errors are mostly below 10^{-4}. To evaluate the impact of the errors induced by MPC, we look at label prediction. The MPC method labels all the questions exactly the same way as the baseline method, so both methods made the same number of errors, i.e., 269 (as shown in [4], this is less than majority voting and some other methods). On average, the execution of each iteration took $52.85s$ wall-clock time, or $39.58s$ CPU time. The MPC model is 2 000 times slower than the base model, this is due to the high number of comparisons that should be made to normalize the three factors.

If we use the pre-computed linear function h presented at the end of Sec. 2 the outputs will be very different of course because of the aforementioned reasons, but wall-clock time of each iteration is reduced to $0.58s$ and the CPU time to $0.48s$ making it almost 100 times faster. This normalization alternative increases the number of queries labeled differently by the MPC to 5, however, it yields 266 errors in total. For this specific dataset, the pre-computed normalization used happens to gives better results than the original baseline.

Acknowledgments This research is part of the program DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC (1988)
2. Berti-Équille, L.: Data veracity estimation with ensembling truth discovery methods. In: BigData (2015)
3. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press (2015)
4. Galland, A., Abiteboul, S., Marian, A., Senellart, P.: Corroborating information from disagreeing views. In: WSDM (2010)
5. Li, X., Dong, X.L., Lyons, K., Meng, W., Srivastava, D.: Truth finding on the deep Web: Is the problem solved? PVLDB **6**(2) (2013)
6. Li, Y., Gao, J., Meng, C., Li, Q., Su, L., Zhao, B., Fan, W., Han, J.: A survey on truth discovery. SIGKDD Explorations (2016)
7. Mohassel, P., Rindal, P.: Aby^3: A mixed protocol framework for machine learning. In: CCS (2018)
8. Saadeh, A., Senellart, P., Bressan, S.: Confidential truth finding with multi-party computation (extended version). CoRR **abs/2305.14727** (2023)