Efficient Unconditionally Secure Comparison and Privacy Preserving Machine Learning Classification Protocols *

Bernardo David[1], Rafael Dowsley[2], Raj Katti[3], and Anderson C. A. Nascimento[3]

[1] Aarhus University
bernardo@cs.au.dk
[2] Karlsruhe Institute of Technology
rafael.dowsley@kit.edu
[3] University of Washington Tacoma
rajkatti@uw.edu; andclay@uw.edu

Abstract. We propose an efficient unconditionally secure protocol for privacy preserving comparison of ℓ-bit integers when both integers are shared between two semi-honest parties. Using our comparison protocol as a building block, we construct two-party generic private machine learning classifiers. In this scenario, one party holds an input while the other holds a model and they wish to classify the input according to the model without revealing their private information to each other. Our constructions are based on the setup assumption that there exists pre-distributed correlated randomness available to the computing parties, the so-called commodity-based model. The protocols are storage and computationally efficient, consisting only of additions and multiplications of integers.

Keywords. Secure Comparison, Private Machine Learning, Unconditional Security, Commodity Based Model.

1 Introduction

We propose protocols for privacy preserving machine learning classification. Our protocols are information theoretically secure and work in the commodity-based cryptographic model [3, 5], where a trusted center pre-distributes correlated randomness to Alice and Bob during a setup phase. The trusted initializer never knows the players's inputs, as the pre-distributed data is independent from the actual inputs to the protocol. Moreover, the trusted initializer never engages in the protocol execution after the setup phase. Our main technique is to reduce

* Bernardo David was supported by European Research Council Starting Grant 279447. The author acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and also from the CFEM research centre (supported by the Danish Strategic Research Council) within which part of this work was performed.

the fundamental building blocks used frequently in privacy preserving machine learning classification (distributed protocols for securely computing comparison, inner products and argmax) to protocols that can be efficiently computed in the commodity-based cryptography model. In the case a trusted initializer is not available or desirable, we sketch how to substitute the TI by a pre-processing phase where Alice and Bob engage in a protocol for emulating the trusted initializer behavior; but this comes at the cost of achieving only computational security.

Private Machine Learning Classification: Supervised machine learning algorithms usually have two phases: training and classification. We focus on the privacy issues that arise during the classification phase [12, 11, 31]. In the classification phase, Alice holds an input vector v that represents, for example, her personal health records. Bob (a health care provider) holds a classifier C and a model w so that $C(w, v)$ represents, for example, Alice's estimated health care related expenses for the current year. For obvious reasons, Alice does not want to reveal her complete personal health records to Bob, while Bob worries that knowledge about the model might reveal something about the training dataset used to obtain w (previously stored personal health records, for example).

Comparison Protocols: One building block often used in classifiers are comparison protocols. In a secure comparison protocol, Alice and Bob hold inputs x and y and would like to know whether $y > x$ while leaking no additional information on the inputs. Given an active trusted third party, secure comparisons (or any other secure computation for that matter) are trivially implementable: Alice and Bob give the inputs to the trusted third party who, in turn, computes the desired function of the inputs and announces the result. In cryptography, we are interested in emulating this *ideal* protocol without a trusted third party actively engaging and computing with Alice and Bob. Computing secure comparisons finds diverse applications, such as solving the classic Millionaires Problem proposed by Yao [48], implementing secure auctions [20], computing the median [1] and solving minimum spanning tree and a variant of shortest paths [13]. In this paper, we are interested in applying secure comparisons to perform privacy preserving machine learning [14, 37, 12, 11] and will present a new protocol that is efficient and unconditionally secure.

Commodity-based Cryptography: We obtain our results based on the existence of pre-distributed correlated data to Alice and Bob, the so-called commodity-based model. This model was introduced by Beaver [3, 5] and is inspired by the client-server distributed computation model. It is an alternative for obtaining efficient secure multi-party computation. In this model a trusted initializer (TI) distributes values (i.e., the commodities) to the parties before the start of the protocol execution, possibly long before the inputs are known. The values are correlated and distributed according to a joint probability distribution. The TI does not perform any subsequent communication with the parties. Since complex operations can be delegated to the TI (the parties need only to de-randomize the

pre-distributed computations to match the actual inputs which are not known to the TI), it is possible to obtain very efficient protocols in this model. In cases where the presence of a trusted initializer is undesirable (or infeasible), it can be substituted by a secure two-party computation protocol that generates the necessary correlated randomness. For example, there are a number of known approaches [23, 22, 25, 40] for generating multiplicative triples that are used for performing secure multiplication [4].

1.1 Related Works

Bost et al. [12, 11] proposed protocols for computing $C(w, v)$ privately for the case of some general classifiers. They used protocols for securely computing comparison, inner products and argmax as building blocks. The resulting protocols are computationally secure and require many modular exponentiations. Graepel et al. [31] constructed computationally secure privacy preserving procedures for both training and classification phases from somewhat homomorphic encryption (SHE). The SHE scheme proposed by the authors incurs in high computational and communication overheads, requiring expensive operations (*e.g.* modular exponentiations) and that ciphertexts grow with the number of multiplications performed.

Clearly, assumptions are needed in order to perform secure comparisons. Computationally secure protocols assume, for example, that adversaries are computationally bounded and that some specific cryptographic related computational problems are intractable. Most of the previously proposed protocols for secure comparison are computationally secure. Secure comparison protocols in the plain model (without assuming correlated randomness) have been implemented using Yao's garbled circuits [34], using encryption of bits as quadratic and non-quadratic residues modulo an RSA modulus [28], homomorphic encryption [20, 21, 29, 33], and other adhoc techniques [7, 8]. In all of these protocols modular exponentiations, public key operations or computations of comparable complexity are required.

Information-theoretically secure (or unconditionally secure) protocols do not make any assumptions whatsoever on the computing power of an adversary. But for obtaining an unconditionally secure comparison protocol it is necessary to make some assumption, e.g., the existence of a secure multiplication protocol, which can be obtained from pre-distributed correlated randomness, or that the majority of the players are honest. In this setting previous works [19, 38, 43, 10] reduced the task of secure comparison to that of secure multiplication. For ℓ-bit values that are shared in form of bit-wise secret sharings, it is possible to use only $O(\ell)$ instances of the multiplication protocol and it is even possible to obtain constant round protocols [19, 38, 43].[4] Since these protocols

[4] Errata: In the conference version of this paper the complexity of these solutions was unfairly compared to that of our comparison protocol. More specifically, the reported complexity for the previous solutions was for the case of the full-fledged comparison protocol which takes the inputs as a single secret shares in $\mathbb{Z}_q$ and have to perform

information-theoretically reduce the task of secure comparison to that of secure multiplication, instantiating them with a suitable secure multiplication protocol in the commodity based cryptography model [4] yield unconditionally secure comparisons.

Many cryptographic protocols were designed in the commodity-based model for functionalities such as commitments [42, 9, 35], oblivious transfer [3, 2], secure linear algebra [26, 24], verifiable secret sharing [36, 27], proximity testing [44] and oblivious polynomial evaluation [45]. For recent and fundamental results on the power of pre-distributed correlated randomness, see [32]. A solution for efficient privacy preserving linear regression over distributed datasets in the commodity-based model was recently proposed by de Cock et al. [18].

1.2 Our Contributions

In this paper we first propose a new two-party unconditionally secure comparison protocol in the commodity based cryptography model. Our protocol uses only ℓ invocations of the underlying secure multiplication protocol (which in our case is implemented using Beaver's solution for secure multiplication in the commodity-based model [4]). While our solution uses a linear number of instances of the multiplication protocol as in previous solutions, our constants are better: the previous best solution due to Toft [43] uses $13\ell+6\sqrt{\ell}$ instances of the underlying multiplication protocol. The caveats are that our protocol needs $\lceil \log(\ell+1) \rceil$ rounds and the output is not 0/1 represented in $\mathbb{Z}_q$, but instead either 0 or uniformly distributed in $\mathbb{Z}_q^*$.

For the typical values of ℓ used in machine learning classifiers, the $\lceil \log(\ell+1) \rceil$ rounds will be very close to the 6 rounds of Toft's protocol. In our applications, the only implication of the output not being 0/1 is in the possibilities to build secure distributed argmax protocols. Considering an argmax protocol for k input values $v_1, \ldots, v_k$, if the underlying comparison protocol has 0/1 output there are two approaches for building the argmax protocol: (1) compare each pair of values and then multiply the results appropriately to obtain the argmax. This approach executes $k(k-1)$ instances of the comparison protocol and $k(k-2)$ instances of the multiplication protocol, and has round complexity equal to the round complexity of the comparison protocol plus $\lceil \log(k-2) \rceil$; (2) use a tree-based approach in which the values are placed in the leaves and then the internal nodes are computed upwards by using the comparison protocol to keep track of largest value in its descendants (and the corresponding argument) until one reaches the root and its associated argument is revealed. This approach executes $k-1$ instances of the comparison protocol and $(k-1)(2\ell + 2\lceil \log k \rceil)$ instances of the multiplication protocol, and has round complexity equal to the round complexity of the comparison protocol times $\lceil \log(k) \rceil$. On the other hand, for comparison protocols with output like ours, one can only use the first approach for building the argmax protocol. Nevertheless, putting all costs together, using

the bit-decomposition of the inputs; while our protocol already considers the input as being bit-wise secret sharings.

our underlying comparison protocol will be the best option for most typical values of ℓ and k in our applications. In particular, for typical ℓ, this is the case if k is small; or, if the round complexity is the main implementation bottleneck, even for big k. On the remaining cases one can use Toft's protocol [43] as the underlying building block.

We then proceed to propose new, efficient and unconditionally secure protocols for private machine learning classification [12, 11] based on our secure comparison protocol. Here, we show that, in the commodity-based model, we can obtain very efficient protocols for two general classifiers: (i) hyperplane decision and (ii) Naïve Bayes classifiers. Our protocols are computationally very efficient (requiring solely additions and multiplications of integers) and also very efficient in terms of the amount of data that has to be pre-distributed. On the other hand, previous results [12, 11, 31] require computationally expensive operations (*e.g.* modular exponentiations) and incur in higher communication overhead.

We also point out that the trusted initializer can be replaced by a secure two-party computation protocol (though at the cost of unconditional security, since such protocols require computational assumptions). Most of the correlated randomness needed in our protocols is used by the underlying multiplicaiton protocol [4] and consists of multiplicative triples, which can be generated through known protocols [25, 40]. Notice that the techniques used for computing multiplicative triples can also be directly used for computing correlated randomness required by the vector inner product protocol used in the classifiers [26]. The remaining correlated data is of the form $z \in \mathbb{Z}_q^*, r \in \mathbb{Z}_q, z_{\mathsf{A}} = r, z_{\mathsf{B}} = z - r$, which can be trivially computed with an additively homomorphic encryption scheme, *e.g.* Paillier [39]. We outline the techniques for substituting the trusted initializer in our protocols but leave a full description, security proof and performance analysis as a future work.

Security Model: Our proposed protocols are secure in the framework of universal composability (UC) [16] with honest-but-curious corruptions (i.e., the corrupted parties follow the protocol instructions, but try to learn additional information).

2 Commodity Based Cryptography

In the commodity based cryptography [3, 2] model a trusted initializer (TI) distributes values (i.e., correlated randomness which are the commodities) to the parties before the start of the protocol execution, possibly long before the inputs are known. This pre-distributed data can be obtained in many different settings: (1) it can pre-distributed by a single trusted center that is active during a setup phase. Alice and Bob contact the trusted center during a setup phase, receive their correlated data and no further communication is necessary between Alice and Bob and the center. Note that the center does not engage in the execution of the protocol itself nor is aware of Alice and Bob's input; (2) it can be pre-distributed by many centers that do not interact with each other. A secure protocol is possible if a majority of the servers is honest [3, 5]; (3) it can be precomputed by players that do communicate through private channels with each

other and emulate a trusted center by running a general MPC protocol (given that a majority of them is honest and a broadcast channel is available).

The TI has no access to the parties' secret inputs and does not communicate with the parties except for delivering the pre-distributed values during the setup. One main advantage of this model is the already mentioned high computational efficiency that arises from the fact that the parties only need to derandomize the pre-computed instances to match their own inputs. Another advantage is the fact that since the computations are pre-distributed by a trusted initializer, most protocols yield perfect security. In this work we model the trusted initializer as an ideal functionality that generates the correlated randomness that is distributed to the parties.

Beaver [4] proposed a very efficient protocol for secure distributed (modular) multiplication in the commodity-based model which will be used as a building block in this work. The protocol description is omitted due to the lack of space, see the references for details.

Eliminating the Trusted Initializer In this multiplication protocol, the trusted initializer can be substituted by a two-party protocol that computes the required multiplicative triples $(t, a_{\mathsf{A}}, b_{\mathsf{A}})$ for A and $(a_{\mathsf{B}}, b_{\mathsf{B}}, I = (a_{\mathsf{A}}b_{\mathsf{B}} + a_{\mathsf{B}}b_{\mathsf{A}} - t))$ for B. Since generating this specific correlated data finds several applications, it is a well studied problem. Well known protocols for general purpose multiparty computation [23, 22] introduced a pre-processing phase protocol where multiplication triples are generated using additively homomorphic encryption schemes (*e.g.* Paillier [39]), which can be adapted to the two-party case as described in [40]. A different approach builds on oblivious transfer to perform the necessary secure multiplications [30]. Recent implementation results [25, 40] show that both techniques achieve good performance. In particular, [25] shows that an OT based protocol requires only tens of milliseconds per triple. However, we remark that using such protocols for computing correlated randomness implies the loss of unconditional security guarantees, since they are based on computational assumptions. Comprehensive surveys of different methods for generating multiplicative triples and their concrete efficiency can be found in [25, 40].

3 Secure Distributed Comparison Protocol

Using the protocol for secure distributed multiplication presented in [4] as a subprotocol we build our secure distributed comparison protocol. It implements the distributed comparison functionality $\mathcal{F}_{DC}^{\mathcal{R}}$, which takes the ℓ-bit integers x, y to be compared in form of shares, reconstruct them from the shares, performs the comparison and then distributes shares of zero (in the field $\mathbb{Z}_q$) if $y > x$ and shares of an uniformly random $w \in \mathbb{Z}_q^*$ if $y \leq x$. More specifically, upon receiving A's shares of the inputs, $x_{\mathsf{A}}, y_{\mathsf{A}}$, and B's shares of the inputs, $x_{\mathsf{B}}, y_{\mathsf{B}}$, $\mathcal{F}_{DC}^{\mathcal{R}}$ proceeds as follows: (1) runs the algorithm $\mathcal{R}$ to reconstruct the inputs x and y from the shares $x_{\mathsf{A}}, y_{\mathsf{A}}, x_{\mathsf{B}}, y_{\mathsf{B}}$; (2) picks a random $r \in \mathbb{Z}_q$ and sends it to A; (3) if $y > x$, sends $-r \mod q$ to B; (4) if $y \leq x$, picks a random $w \in \mathbb{Z}_q^*$ and sends $w - r \mod q$ to B.

The protocol that implements it follows the lines of Damgård, Geisler and Krøigaard [20], which is one of the most efficient known solutions for the secure comparison problem. But due to our usage of pre-distributed, correlated randomness it is possible to eliminate the use of (homomorphic) encryption and the computation of the XOR of the shared inputs bits, which are both computationally intensive steps.

Let the ℓ-bit integers being compared be $x = (x_\ell, \ldots, x_1)$ and $y = (y_\ell, \ldots, y_1)$. Let q be a prime such that $q > 2^{\ell+2}$ (all operations are modulo q). The parties A and B have additive shares of each bit of x and y. A has $x_\mathsf{A} = (x_{\ell\mathsf{A}}, \ldots, x_{1\mathsf{A}})$ and $y_\mathsf{A} = (y_{\ell\mathsf{A}}, \ldots, y_{1\mathsf{A}})$ and B has $x_\mathsf{B} = (x_{\ell\mathsf{B}}, \ldots, x_{1\mathsf{B}})$ and $y_\mathsf{B} = (y_{\ell\mathsf{B}}, \ldots, y_{1\mathsf{B}})$ such that $x_{i\mathsf{A}}, x_{i\mathsf{B}}, y_{i\mathsf{A}}, y_{i\mathsf{B}} \in \mathbb{Z}_q$, $x_i, y_i \in \{0,1\}$, $x_i = x_{i\mathsf{A}} + x_{i\mathsf{B}} \mod q$ and $y_i = y_{i\mathsf{A}} + y_{i\mathsf{B}} \mod q$. We write $[y_i]$ to denote the shares of y_i, i.e., $y_{i\mathsf{A}}$ and $y_{i\mathsf{B}}$ such that $y_i = y_{i\mathsf{A}} + y_{i\mathsf{B}} \mod q$. The distributed comparison protocol uses these shares and proceeds as follows:

1. The trusted initializer chooses uniformly random $z \in \mathbb{Z}_q^*$ and $r \in \mathbb{Z}_q$ and distributes the shares $z_\mathsf{A} = r$ to A and $z_\mathsf{B} = z - r$ to B. It also pre-distributes the correlated randomness necessary for the execution of ℓ instances of the distributed multiplication protocol.
2. For $i = 1, \ldots, \ell$, A and B locally compute shares $[d_i]$ where $d_i = x_i - y_i$, i.e., A computes $d_{i\mathsf{A}} = x_{i\mathsf{A}} - y_{i\mathsf{A}} \mod q$ and B computes $d_{i\mathsf{B}} = x_{i\mathsf{B}} - y_{i\mathsf{B}} \mod q$. Note that $d_i \in \{0, 1, -1\}$.
3. For $i = 1, \ldots, \ell$, A and B locally compute shares $[c_i]$ where $c_i = x_i - y_i + 1 + \sum_{j=i+1}^{\ell} d_j 2^{\ell-j+2}$ (the shares of 1 are fixed a priori, lets say A's share is 1 and B's is 0). Let the shares of c_i that A has be $c_\mathsf{A} = (c_{\ell\mathsf{A}}, \ldots, c_{1\mathsf{A}})$ and those of B be $c_\mathsf{B} = (c_{\ell\mathsf{B}}, \ldots, c_{1\mathsf{B}})$.
4. A and B use ℓ times the distributed multiplication protocol in order to compute shares of $w = z \prod_{i=1}^{\ell} c_i$. Let the final shares be w_A and w_B, such that $w = (w_\mathsf{A} + w_\mathsf{B}) \mod q$. A outputs w_A and Bob outputs w_B.

Note that if one of the parties, lets say A, is supposed to learn the result of the comparison, this can be easily accomplished by having B sending his share w_B to A, who can then reconstruct w and test whether $w = 0 \mod q$ and so $y > x$ or $w \neq 0 \mod q$ and so $y \leq x$.

Theorem 1 *Let q be a prime such that $q > 2^{\ell+2}$ and let $\mathcal{R}$ be the recovering algorithm that takes the additives shares modulo q of each bit x_i of x and y_i of y and returns x and y. The distributed comparison protocol is correct and securely implements the distributed comparison functionality $\mathcal{F}_{DC}^{\mathcal{R}}$ against honest but curious adversaries in the commodity based model.*

Proof. **Correctness:** We have that $y > x$ if and only if there exists i such that all the bits $(x_\ell, \ldots, x_{i+1})$ are identical to the bits $(y_\ell, \ldots, y_{i+1})$ and $x_i - y_i + 1 = 0$. Equivalently, $y > x$ if and only if there exists i such that all $d_\ell = \ldots = d_{i+1} = 0$ and $d_i + 1 = 0$. We first prove the following useful lemmas.

Lemma 2 *Let $S_i = \sum_{j=i+1}^{\ell} d_j 2^{\ell-j+2}$, where $d_j \in \{0, 1, -1\}$. If $d_\ell = \ldots = d_{i+1} = 0$, then $S_i = 0$; otherwise $S_i \notin \{-3, -2, -1, 0, 1, 2, 3\}$ modulo q.*

Proof. If all $d_\ell = \ldots = d_{i+1} = 0$ then it follows trivially that $S_i = 0$. Now we show that if $S_i \in \{-3, -2, -1, 0, 1, 2, 3\}$ modulo q then all $d_\ell = \ldots = d_{i+1} = 0$. Suppose that there are some d_j which are not 0 and let $k \in \{i+1, \ldots, \ell\}$ be smallest value such that $d_k \in \{1, -1\}$. Since the operations are modulo $q > 2^{\ell+2}$, the sum of the powers $\sum_{j=k+1}^{\ell} 2^{\ell-j+2} \leq 2^{\ell-k+2} - 4$ and the power of 2 associated with d_k is $2^{\ell-k+2} \leq 2^{\ell+1}$, it follows that $S_i \notin \{-3, -2, -1, 0, 1, 2, 3\}$ modulo q if d_k is non-zero. Therefore all $d_\ell = \ldots = d_{i+1} = 0$ if $S_i \in \{-3, -2, -1, 0, 1, 2, 3\}$ modulo q.

Lemma 3 *For any $i \in \{1, \ldots, \ell\}$, $c_i = 0$ if and only if $d_\ell = \ldots = d_{i+1} = 0$ and $d_i + 1 = 0$.*

Proof. It is trivial that if $d_\ell = \ldots = d_{i+1} = 0$ and $d_i + 1 = 0$, then $c_i = 0$, so lets prove the other direction. Since $d_i \in \{0, 1, -1\}$, it implies that $d_i + 1 \in \{0, 1, 2\}$. But by Lemma 2, $S_i = \sum_{j=i+1}^{\ell} d_j 2^{\ell-j+2}$ is such that $S_i \in \{-3, -2, -1, 0, 1, 2, 3\}$ modulo q only if $d_\ell = \ldots = d_{i+1} = 0$, which imply that $c_i = d_i + 1 + S_i$ will be different from zero if $d_i + 1 \neq 0$ or any $d_\ell, \ldots, d_{i+1}$ is different from 0.

Hence we have that $y > x$ if and only if there exists i such that $c_i = 0$. Since z is uniformly random in $\mathbb{Z}_q^*$ and $w = z \prod_{i=1}^{\ell} c_i \mod q$, we have that $w = 0$ if and only if there exists i such that $c_i = 0$; otherwise it is uniformly distributed in $\mathbb{Z}_q^*$. And thus the correctness of the protocol is proved.

Security: Note that the first and second steps of the protocol do not require message exchanging (the shares of the constant 1 in the second step can be fixed a priori). So the only messages exchanged are those for the execution of the distributed multiplication protocol and therefore the real protocol transcript can be perfectly simulated as explained in [4].

Note that even if one of the parties, lets say A, is supposed to learn the result of the comparison and so B sends w_B to A as a last step, this does not affect the security of the protocol since this last message can also be perfectly simulated by a simulator that only learns A's inputs and outputs. This is due to the fact that either $y > x$ and so some $c_i = 0$, $w = 0 \mod q$ and $w_\mathsf{B} = -w_\mathsf{A} \mod q$ or $y \leq x$ and so all $c_i \neq 0 \mod q$ and w is uniformly random in $\mathbb{Z}_q^*$ since z is uniformly distributed in $\mathbb{Z}_q^*$.

4 argmin and argmax

Suppose that the parties A and B have shares of a tuple of values $(v_1, \ldots, v_k)$ and they want to learn the value $m \in \{1, \ldots, k\}$ such that $v_m \leq v_i$ for all $i \in \{1, \ldots, k\}$, but no party should learn any v_i or the relative order between the elements. I.e., the parties just want to learn $m = \text{argmin}_{i \in \{1,\ldots,k\}} v_i$. Using our protocol for secure distributed comparison it is possible to give a practical, secure solution for this problem. The main idea is that for such m we have that $v_m \leq v_i$ for all other v_i. Hence if we compare v_m with each v_i using our distributed comparison protocol with v_m playing the role of y and v_i in the role of x, then all the output values that are shared between A and B will be

uniformly random in $\mathbb{Z}_q^*$, so the product of them all is also uniformly random in $\mathbb{Z}_q^*$. On the other hand, for any j, if there is some i such that $v_j > v_i$, then the result of comparing v_j playing the role of y with v_i in the role of x will be 0, and so the product of the outputs is also 0.

The argmin functionality $\mathcal{F}^{\mathcal{R}}_{\text{argmin}}$ is parametrized by the size q of the field in which the operations are done, the bit-length ℓ of the values being compared and the algorithm $\mathcal{R}$ that reconstructs the inputs from the shares. Upon receiving A's inputs shares, $(v_{1\mathsf{A}}, \ldots, v_{k\mathsf{A}})$, and B's inputs shares, $(v_{1\mathsf{B}}, \ldots, v_{k\mathsf{B}})$, $\mathcal{F}^{\mathcal{R}}_{\text{argmin}}$ proceeds as follows: (1) runs $\mathcal{R}$ to reconstruct the inputs $(v_1, \ldots, v_k)$ from the shares; (2) computes $m = \text{argmin}_{i \in \{1,\ldots,k\}} v_i$; (3) sends m to A.

The argmin protocol works as follows. Let ℓ be the bit length of the values v_i and let q be a prime such that $q > 2^{\ell+2}$. All operations are modulo q. The trusted initializer pre-distributes all the correlated randomness necessary for the execution of the instances of the distributed multiplication protocol and of the distributed comparison protocol (with the same field size q). A has shares of the inputs $(v_{1\mathsf{A}}, \ldots, v_{k\mathsf{A}})$ and B also has shares of the inputs $(v_{1\mathsf{B}}, \ldots, v_{k\mathsf{B}})$. For $j = 1, \ldots, k$, the protocol then proceeds as follows:

1. For all $i \in \{1, \ldots, k\} \backslash j$, A and B execute the distributed comparison protocol with inputs v_j in the role of y and v_i in the role of x. Let w_i denote the shared output obtained by A and B.
2. A and B use $k-2$ times the distributed multiplication protocol in order to compute shares of $w = \prod_{i \in \{1,\ldots,k\} \backslash j} w_i$. Let the final shares be w_{A} and w_{B}. B sends his share w_{B} to A.
3. A recovers $w = (w_{\mathsf{A}} + w_{\mathsf{B}}) \mod q$. If $w \neq 0 \mod q$, append j to the value to be output by A in the end.

Theorem 4 *Let q be a prime such that $q > 2^{\ell+2}$ and let $\mathcal{R}$ be the recovering algorithm that takes the additives shares modulo q of each bit of v_i and returns v_i. The* argmin *protocol is correct and securely implements the* argmin *functionality $\mathcal{F}^{\mathcal{R}}_{\text{argmin}}$ against honest but curious adversaries in the commodity based model.*

Proof. **Correctness:** Let m be a value such that $v_m \leq v_i$ for all $i \in \{1, \ldots, k\} \setminus m$. Each comparison of v_m playing the role of y and v_i ($i \in \{1, \ldots, k\} \setminus m$) in the role of x will result in a shared output value w_i which is uniformly random in $\mathbb{Z}_q^*$, and so $w = \prod_{i \in \{1,\ldots,k\} \backslash j} w_i$ is also uniformly random in $\mathbb{Z}_q^*$ and m is appended to A's output. On the other hand, for any j such that there is some i with $v_j > v_i$, the result of comparing v_j playing the role of y with v_i in the role of x will be 0, and so the product of the outputs is also 0 and and j is not appended to A's output.

Security: As explained in the previous sections the messages involved in the distributed comparison and the distributed multiplication protocols can be straightforwardly and perfectly simulated by the simulator. The share sent in the third step only reveal 1 bit of information, either $w = 0$ if the value is not the argmin or w is completely random in $\mathbb{Z}_q^*$ if the value is the argmin. This can be trivially simulated given the output of the argmin functionality.

The complementary problem is to compute the value $m = \text{argmax}_{i \in \{1,\ldots,k\}} v_i$. From the secure argmin protocol it is trivial to obtain a secure argmax protocol by simply running the argmin protocol with inputs $(2^\ell - 1 - v_1, \ldots, 2^\ell - 1 - v_k)$.

5 Applications

Practical secure protocols for distributively computing the comparison have many applications such as: auctions, private machine learning classifiers, benchmarking and secure extraction of statistical data from databases. In this section, we show that our previously proposed protocol for comparison can be used directly to obtain secure auctions and that our protocol for secure computing the argmax function can be used to obtain private machine learning classifiers.

5.1 Auctions

As already mentioned by Damgård et al. [20] a secure comparison protocol can be used to execute an auction protocol in which no party learns the value of the highest bid until the auction is finished. Lets say that A is the auction house and B is the accounting company. A and B hold shares of the biggest bid x done so far, x_A and x_B respectively. Now if a new participant wants to bid y, he creates shares y_A and y_B of his bid and send them to A and B respectively. Then A and B can compare the bids without discovering their values. This is particularly useful in the scenario of online auctions where the participants can submit a maximum bid to the system, which will them automatically bid for the participant until it wins the bid with the minimum possible bid or its maximum bid is achieved. Notice that the knowledge of the maximum bid in such case should be kept confidential from the other participants and from the auction house since they could otherwise exploit such knowledge. Using the secure comparison protocol A and B can keep track of both the highest bid so far, x, and the maximum bid y that was still not achieved (there is either none or one such value) without either A or B learning x or y. Then when a new bid z arrives, they can securely compare z to y and to x in order to update the values of the highest bid and of the maximum bid not yet achieved.

5.2 Private Machine Learning

In machine learning the supervised learning algorithms should given an input v output a guess of what class $c \in \{1, \ldots, k\}$ it belongs to. They are divided in two phases: the training and the classification. In the training phase, given labeled samples (i.e., inputs and the corresponding classes) the algorithm learns a model w. Then in the classification phase, the algorithm takes the model w and an input v and should output the guess c of which class v belongs to. Here we focus on obtaining privacy-preserving classifiers.

Imagine for instance the scenario where a health care provider has a model w to predict the occurrence of certain diseases which was built using the medical

profile of several persons and a patient wants to obtain the prediction $C(w, v)$ of what diseases he is likely to have given his medical profile v. On one hand, the model w can leak sensitive information about the medical profile of the patients whose medical profiles were used as labeled samples, and therefore should be kept confidential (this can even be required by law depending on the jurisdiction). On the other hand, the patient does not want do reveal his sensitive medical profile v. Hence the patient and the health care provider should ideally be able to interact in a protocol in which the patient learns $C(w, v)$ in the end, but nothing else; while the health care provider does not learn anything.

As pointed out by Bost et al. [11], the core building blocks for designing many private classifiers are secure comparison, argmax and inner product protocols. In this paper we developed practical, unconditionally secure solutions for computing the comparison and the argmax with a trusted initializer. A practical and unconditionally secure inner product protocol with a trusted initializer was already presented by Dowsley et al. [26], so combining these protocols (and making the necessary conversions to the bit-wise shares) we are able to obtain practical private classifiers. One example of a classifier which is used in many learning algorithms is the hyperplane decision classifier. In this classifier the model w consists of k vectors $w_1, \ldots, w_k$ and the classification of the input v is done by computing

$$C(w, v) = \arg \max_{i \in \{1,\ldots,k\}} \langle v, w_i \rangle.$$

This can be straightforwardly implemented using the inner product protocol from Dowsley et al. [26] (which shares the result between the parties), a conversion to bit-wise shares (techniques from [47] or [43] can be used) and our argmax protocol. Another classifier that can be implemented in a privacy-preserving way using these building blocks is the Naïve Bayes classifier. In this classifier the model consists of the probability that each class c_i, $\{p(C = c_i)\}_{i=1}^{k}$, happens and the probability that each element v_j of the input v happens in a certain class c_i. The classification using a maximum a posteriori decision rule is computed as

$$C(w, v) = \arg \max_{i \in \{1,\ldots,k\}} \left\{ \log p(C = c_i) + \sum_j \log p(V_j = v_j | C = c_i) \right\}.$$

Note that our protocols work with integers, so we first need to convert the log of the probabilities to integers by multiplying by a large number T. For details about this issue please refer to [46, 11]. The converted value of $\log p(C = c_i)$ can be shared by the model owner. To produce shares of $\log p(V_j = v_j | C = c_i)$ without revealing v_j to the model owner, the parties, for each possible value v_k in the domain of V_j, compute a distributed multiplication of the converted value of $\log p(V_j = v_k | C = c_i)$ (with the shares distributed by the model owner) and either 0 if $v_k \neq v_j$ or 1 if $v_k = v_j$ (the shares of this second number to be multiplied are distributed by the user). The shared results of these multiplications are then added to produce the shares of the converted value of $\log p(V_j = v_j | C = c_i) = 1 \log p(V_j = v_j | C = c_i) + \sum_{k \neq j} 0 p(V_j = v_k | C = c_i)$

without revealing v_j. From there the remaining additions can be done locally by the parties and then the argmax protocol is used to generate the output.

Eliminating the Trusted Initializer In order to substitute the trusted initializer in our private classifiers, we need to construct secure two party protocols that generate the correlated randomness for both the vector inner product protocol [26] and the comparison protocol of Section 3. As we have pointed out in Section 2, there exist a number of highly efficient methods [25, 40] for generating the correlated data used by the underlying multiplication protocol [4]. The same techniques can be applied in a straightforward way to generate the correlated randomness used in the vector inner product protocol [26].

The only remaining data that must be generated is $z \in \mathbb{Z}_q^*, r \in \mathbb{Z}_q, z_{\mathsf{A}} = r, z_{\mathsf{B}} = z - r$, which is used by the comparison protocol. These values can be trivially computed using an additively homomorphic cryptosystem such as the one by Paillier [39]. Let such cryptosystem be described by the following algorithms: Key Generation $\mathsf{Gen}(1^\lambda) \rightarrow (sk, pk)$, Encryption $Enc(pk, m) \rightarrow c$ and Decryption $Dec(sk, c) \rightarrow m$, with $m_1 + m_2 = Dec(sk, c_1 \cdot c_2)$ for $c_1 = Enc(pk, m_1)$ and $c_2 = Enc(pk, m_2)$. To generate $z_{\mathsf{A}}, z_{\mathsf{B}}$, B runs $\mathsf{Gen}(1^\lambda) \rightarrow (sk, pk)$, uniformly samples $z_1 \in \mathbb{Z}_q^*$ and sends $(pk, \hat{z}_1 = Enc(pk, z_1))$ to A. A uniformly samples $z_2 \in \mathbb{Z}_q^*, r \in \mathbb{Z}_q$, computes $\hat{z}_2 = Enc(pk, z_2), \hat{r} = Enc(pk, r), \hat{z} = \hat{z}_1 \cdot \hat{z}_2$, sends $\hat{z}_{\mathsf{B}} = \hat{z} \cdot \hat{r}^{-1}$ to B and sets $z_{\mathsf{A}} = r$. B decrypts $\hat{z}_{\mathsf{B}}$ and sets $z_{\mathsf{B}} = Dec(sk, \hat{z}_{\mathsf{B}})$. This protocol's security follows from the IND-CPA security of Paillier's cryptosystem.

Notice that the efficiency of both the multiplication protocol of [4] and the protocols introduced in this paper is independent from the efficiency of the trusted initializer, since the correlated randomness provided by it can be precomputed independently from our protocol execution. In fact, the trusted initializer (or a substitute two-party protocol) can compute the necessary correlated randomness at any given time before the actual protocol inputs are known. Hence, a large amount of correlated randomness can be pre-computed and stored for future protocol executions.

As we remarked before, such two-party protocols for generating correlated randomness require computational assumptions. In fact, it is well known that any two-party protocol that computes multiplications cannot achieve unconditional security in the plain model (*i.e.* without setup assumptions) [6, 17, 41]. Hence, substituting the trusted initializer implies the loss of unconditional security guarantees. We leave a full description, security proof and performance analysis of protocols for substituting the trusted initializer as a future work.

6 Conclusion

In this paper we proposed an unconditionally secure protocol for secure comparison of integers. We also proposed protocols for computing the *argmin* and *argmax* functions and show how they can be used to obtain generic private machine learning algorithms, namely the hyperplane based and Naïve Bayes classifiers. We proved that our protocols are secure against honest-but-curious

adversaries. Our protocols are very efficient from a storage and computational point of view. In many real-world scenarios it does make sense to assume the existence of trusted initializers. Therefore, we see commodity-based cryptography as a realistic model for obtaining efficient secure computation protocols. On the other hand, we also outline alternatives to the trusted initializer for situations where its existence might be considered an issue.

References

1. Gagan Aggarwal, Nina Mishra, and Benny Pinkas. Secure computation of the k th-ranked element. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 40–55. Springer, 2004.
2. Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer,1995.
3. Donald Beaver. Commodity-based cryptography (extended abstract). In *29th ACM STOC*, pages 446–455. ACM Press, 1997.
4. Donald Beaver. One-time tables for two-party computation. In *Computing and Combinatorics*, pages 361–370. Springer, 1998.
5. Donald Beaver. Server-assisted cryptography. In NSPW '98, pages 92–106, New York, NY, USA, 1998. ACM.
6. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, 1988.
7. Ian F. Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 515–529. Springer, 2004.
8. Ian F. Blake and Vladimir Kolesnikov. Conditional encrypted mapping and comparing encrypted numbers. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 206–220. Springer, 2006.
9. C. Blundo, B. Masucci, D. R. Stinson, and R. Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Des. Codes Cryptography*, 26(1-3):97–110, 2002.
10. Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 325–343. Springer, 2009.
11. Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. Cryptology ePrint Archive, Report 2014/331, 2014. `http://eprint.iacr.org/2014/331`.
12. Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *ISOC Network and Distributed System Security Symposium – NDSS 2015*, San Diego, California, USA, February 8–11, 2015. The Internet Society.
13. Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 236–252. Springer, 2005.
14. Justin Brickell and Vitaly Shmatikov. Privacy-preserving classifier learning. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 128–147. Springer, 2009.

15. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
16. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, 2001.
17. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, 1988.
18. Martine de Cock, Rafael Dowsley, Anderson C.A. Nascimento, and Stacey C. Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, AISec '15, pages 3–14, New York, NY, USA, 2015. ACM.
19. Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 285–304. Springer, 2006.
20. Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Homomorphic encryption and secure comparison. *IJACT*, 1(1):22–31, 2008.
21. Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. A correction to 'efficient and secure comparison for on-line auctions'. *IJACT*, 1(4):323–324, 2009.
22. Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013*, volume 8134 of *LNCS*, pages 1–18. Springer, 2013.
23. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, 2012.
24. Bernardo David, Rafael Dowsley, Jeroen van de Graaf, Davidson Marques, Anderson C. A. Nascimento, and Adriana C. B. Pinto. Unconditionally secure, universally composable privacy preserving linear algebra. *Information Forensics and Security, IEEE Transactions on*, 11(1):59–73, Jan 2016.
25. Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In NDSS 2015. The Internet Society, 2015.
26. Rafael Dowsley, Jeroen van de Graaf, Davidson Marques, and Anderson C. A. Nascimento. A two-party protocol with trusted initializer for computing the inner product. In Yongwha Chung and Moti Yung, editors, *WISA 10*, volume 6513 of *LNCS*, pages 337–350. Springer, 2011.
27. Rafael Dowsley, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, Hideki Imai, and Anderson C. A. Nascimento. Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. *IEICE Transactions*, 94-A(2):725–734, 2011.
28. Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 457–472. Springer, 2001.
29. Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 330–342. Springer, 2007.
30. Niv Gilboa. Two party RSA key generation. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 116–129. Springer, 1999.

31. Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC 12*, volume 7839 of *LNCS*, pages 1–21. Springer, 2013.
32. Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 600–620. Springer, 2013.
33. Rajendra S. Katti and Cristinel Ababei. Secure comparison without explicit XOR. *CoRR*, abs/1204.2854, 2012.
34. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *1st ACM Conference on Electronic Commerce*, pages 129–139, New York, NY, USA, 1999.
35. Anderson C. A. Nascimento, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, and Hideki Imai. Unconditionally secure homomorphic pre-distributed bit commitment and secure two-party computations. In Colin Boyd and Wenbo Mao, editors, *ISC 2003*, volume 2851 of *LNCS*, pages 151–164. Springer, 2003.
36. Anderson C. A. Nascimento, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, and Hideki Imai. Unconditionally non-interactive verifiable secret sharing secure against faulty majorities in the commodity based model. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04*, volume 3089 of *LNCS*, pages 355–368. Springer, 2004.
37. Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE Computer Society Press, 2013.
38. Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 343–360, Beijing, China, April 16–20, 2007. Springer, Heidelberg, Germany.
39. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.
40. Pille Pullonen. Actively secure two-party computation: Efficient beaver triple generation. Master's thesis, University of Tartu, 2013.
41. Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, 1989.
42. Ronald L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Preprint available at http://people.csail.mit.edu/rivest/Rivest- commitment.pdf, 1999.
43. Tomas Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 357–371. Springer, 2009.
44. Rafael Tonicelli, Bernardo Machado David, and Vinícius de Morais Alves. Universally composable private proximity testing. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec 2011*, volume 6980 of *LNCS*, pages 222–239. Springer, 2011.
45. Rafael Tonicelli, Anderson C.A. Nascimento, Rafael Dowsley, Jörn Müller-Quade, Hideki Imai, Goichiro Hanaoka, and Akira Otsuka. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *International*

Journal of Information Security, 14(1), pages 73–84, 2015.
46. Sebastian Tschiatschek, Peter Reinprecht, Manfred Mücke, and Franz Pernkopf. Bayesian network classifiers with reduced precision parameters. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, ECML/PKDD (1) 2012, volume 7523 of LNCS, pages 74–89. Springer, 2012.
47. Thijs Veugen. Linear Round Bit-Decomposition of Secret-Shared Values. *IEEE Transactions on Information Forensics and Security*, 10(3): pages 498–506, 2015.
48. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, 1982.