

# Enabling Fair and Privacy-Preserving Applications Using Reconciliation Protocols on Ordered Sets

Daniel A. Mayer\*, Georg Neugebauer†, Ulrike Meyer†, and Susanne Wetzel\*

\* Stevens Institute of Technology, Hoboken, New Jersey 07030

{mayer, swetzel}@stevens.edu

† UMIC Research Center, RWTH Aachen University, Germany

{neugebauer, meyer}@umic.rwth-aachen.de

**Abstract**—Fair and privacy-preserving reconciliation protocols on ordered sets have been introduced recently. Despite the fact that these protocols promise to have a great impact in a variety of applications, so far their practical use has been explored to a limited extent only. This paper addresses this gap. As main contributions, this paper identifies e-voting, auctions, event scheduling, and policy reconciliation as four far-reaching areas of application and shows how fair and privacy-preserving reconciliation protocols can be used effectively in these contexts.

## I. INTRODUCTION

Recently, multi-party protocols for enabling fair and privacy-preserving reconciliation of ordered sets have been introduced [1], [2]. In this context, *privacy-preserving* refers to the fact that the parties' input sets remain private during and after protocol execution and only the result of the reconciliation is made known to all legitimate parties involved in this process. The reconciliation is carried out in a distributed fashion without the involvement of a trusted third party. The concept of *fairness* ensures that the computations are performed in a way that recognizes the parties' individual preferences on their input sets. In particular, the reconciliation is optimized according to some notion of fairness which was agreed upon by all parties prior to the reconciliation process. Examples of such notions of fairness were introduced in [3] and include the *sum of ranks* and the *minimum of ranks*. In case of the former, the sum of the preferences of all parties is optimized while the latter maximizes the lowest of the preferences of all parties involved.

In practice, there is a wealth of applications which share the requirements for privacy-preserving execution and fair treatment of all parties involved. Nevertheless, prior work in the context of fair and privacy-preserving reconciliation by-and-large has not yet explored the suitability of these reconciliation protocols for such applications.

This paper addresses this gap. In particular, this paper identifies four high impact areas for fair and privacy-preserving reconciliation: *event scheduling*, *e-voting*, *auctions*, and *policy reconciliation*. In the context of event scheduling, involved parties have various incentives to keep their free time slots and their respective preference (w.r.t. an event time) private. Yet, the result of the scheduling should be as suitable as possible for all parties involved. For electronic auctions it is crucial that the identity of the bidders and their bids remain private. Only

the winning bid and the respective winner should be generally known to everyone at the end of the auction. Fairness in this context ensures that each bid is taken into consideration and that the true winner is determined. In electronic voting, the secrecy of the ballot must be ensured. In addition, the tallying, i.e., the determining of the winner should be performed under fair consideration of all votes. Policy reconciliation enables the negotiation of agreements between parties even when these parties do not necessarily trust one another. In order to allow for a fair negotiation, it is a must that the choices each party is willing to accept remain private at all times. Yet, the goal of the reconciliation process is to yield an agreement that is fair to all parties involved.

This paper demonstrates how various applications in these four high-impact areas can in fact be *reduced* to fair and privacy-preserving reconciliation protocols on ordered sets (PROS). I.e., this paper shows how the inputs for these four applications are mapped to input sets, how the PROS are applied, and how the results of the PROS are translated back into the domain-specific realm of the applications.

*Outline:* In Section II we give an intuition for the approach taken in this paper. We also review ordered sets and the respective set operations that will be used in the remainder of the paper. Our main contributions are presented in Sections III to V, which detail how applications can be mapped to multi-party privacy-preserving reconciliation of ordered sets.

## II. OUR APPROACH

In [4], Kissner and Song present privacy-preserving set operations for multisets<sup>1</sup>. The operations include union, intersection, as well as element reduction. In their work, Kissner and Song also define the privacy-preserving composition of these basic operations. Independently, two-party, fair and privacy-preserving reconciliation protocols for ordered sets were introduced in [3], [1]. The work by Neugebauer et al. [2] leverages both lines of work introducing multi-party (i.e., for two or more parties) privacy-preserving reconciliation on ordered sets (MPROS) crafting specific multisets.

In this paper, we show how the inputs, for instance in event scheduling, can be mapped into suitable multisets. Then,

<sup>1</sup>Due to space limitations we do not include an exhaustive review of related work on privacy-preserving operations in general. The interested reader is referred to [1], [2] for more details.

we demonstrate how the purpose of the application and its intended outcome can be expressed by crafting a specific composition of union, intersection, and element reduction operations on these multisets. Finally, we present how the obtained result on multisets can be translated back into the context of the respective application. In the following, we will refrain from focusing on how the resulting protocols are actually implemented. Instead, we will use the compositions defined in [4], [2] as building blocks. A more detailed description on how these building blocks are implemented can be found in the appendix of this paper.

Our proposed constructions involve  $n$  parties  $P_1, \dots, P_n$  holding input multisets  $S_1, \dots, S_n$ . As first described in [2], the preference (or rank) of an element  $e$  is encoded as the multiplicity  $m$  of an element in a multiset denoted as  $\{e^m\}$ . When computing the *union* of two multisets  $S' = S_i \cup S_j$ , an element which has multiplicity  $m_i$  in  $S_i$  and  $m_j$  in  $S_j$  has multiplicity  $m' = m_i + m_j$  in  $S'$ . Similarly, when computing the *intersection*  $S' = S_i \cap S_j$ , the multiplicity of an element in  $S'$  is  $m' = \min(m_i, m_j)$ . Furthermore, applying the *element reduction* operator  $Rd_t$  with a reduction value  $t$  to a set  $S_i$  reduces the multiplicity of an element in the resulting set to  $\max(m_i - t, 0)$ .

In the remainder of this paper we will show how the different applications can be implemented using the MPROS building blocks  $Rd_t(\cup_i S_i)$  and  $Rd_t\left(\left(\cup_i S_i\right) \cap \left(\cap_j S_j\right)\right)$  by crafting specific multisets  $S_i, S_j$ . In [4], [2] these building blocks were proven to be privacy-preserving against semi-honest adversaries, i.e., adversaries which follow the protocol but attempt to learn as much from the output and the computation as possible. These privacy guarantees only hold if all parties  $P_i$  use input multisets  $S_i$  of equal size chosen from a common domain  $M$ . For some of the applications considered in this paper this is not always naturally the case. To address this problem, we apply a padding scheme in which the multisets are extended with a sufficient number of dummy elements chosen from a common domain  $D$ . To enable the distinguishing of real input values from dummy elements it is required that  $D \cap M = \emptyset$ .

In the following, for each application we first define the desired output as it would be computed if a trusted third party was used. We then show how the same output can be computed in the semi-honest model by constructing a fair solution based on MPROS building blocks such that no party learns anything but this desired output.

### III. POLICY RECONCILIATION AND SCHEDULING

As first described in [3], PROS enables the process of fair and privacy-preserving policy reconciliation (PPPR). Policy reconciliation assumes  $n$  parties with their individual, in general different, set of rules which define parameters for collaboration or information exchange with others. Reconciliation is used to enable collaboration by determining a common set of parameters. Recently, PPPR was extended to the realm of privacy policies in social networks [5].

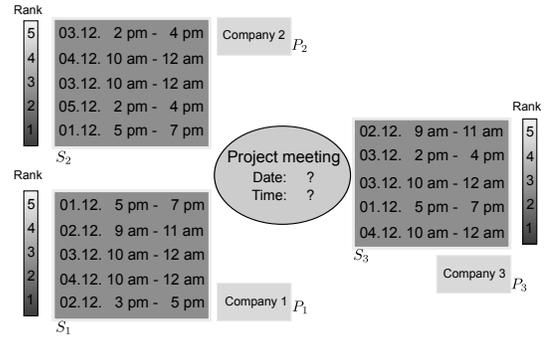


Fig. 1. Scheduling example with three parties each holding five time slots with assigned preferences.

A related concept for collaboration is the scheduling of an appointment. Based on the two-party PROS introduced in [3], [1] it is possible to implement the privacy-preserving and fair scheduling of a meeting time between two entities [6].<sup>2</sup> These results would, e.g., allow the implementation of a privacy-preserving version of *ZocDoc* [7], a service for scheduling doctor's appointments online. Furthermore, the multi-party extension presented in [2] enables an even broader spectrum of applications including a fair and privacy-preserving variant of *Doodle* [8].

#### A. Fair and Privacy-Preserving Doodle

*Doodle* currently allows multiple parties to schedule an event using a central website on which every participant can mark the times at which she is available. One cannot, however, order one's free time slots by personal preference. Also, all participants or at least the initiator can see the free and busy times for all participants. This is necessary since *Doodle* does not offer an automated reconciliation mechanism to determine the *best* time slot.

Using MPROS it is possible to automatically determine the optimal time slot while simultaneously taking the preferences of all parties into account. In addition, all inputs, i.e., time slots and preferences, of all participants remain private. A reasonable choice to ensure fairness is the sum of ranks fairness function which optimizes the sum of the preferences for common time slots of all parties.

In the example illustrated in Figure 1 three parties attempt to find a time for a joint project meeting. The optimal solution according to the sum of ranks fairness function is the common time slot *03.12. 10-12 am*, which is most-preferred among the three companies. The sum of ranks value corresponding to this time slot is nine.

Let each party  $P_i$  have  $k$  time slots  $s_{i1}, \dots, s_{ik}$  ranked with  $\text{rank}_{P_i}(s_{ij}) = k - j + 1$  for  $j = 1, \dots, k$ .

1) *Output*: Let  $S^\cap := S_1 \cap \dots \cap S_n$  contain all common time slots. Then the desired output is defined by

$$\arg \max_{x_l \in S^\cap} \sum_{i=1}^n \text{rank}_{P_i}(x_l)$$

i.e., as all  $x_l$  which maximize the sum of ranks over all parties.

<sup>2</sup>See also <http://www.pfairappl.info/>.

2) *Encoding*: Each party  $P_i$  defines its input as multiset  $S_i = \{s_{i1}^k, s_{i2}^{k-1}, \dots, s_{ik}\}$  with  $|S_i| = \frac{k \cdot (k+1)}{2}$ . That is, each  $S_i$  encodes all of  $P_i$ 's free time slots with the preferences represented in the multiplicity of the set elements. In addition, each party is required to construct  $S'_i = \{s_{i1}^{n-k}, s_{i2}^{n-k}, \dots, s_{ik}^{n-k}\}$  with  $|S'_i| = n \cdot k^2$  to enable the reconciliation.

3) *Fairness function*: In order to obtain the desired output in a privacy-preserving manner, the  $n$  parties compute  $W_t = Rd_t((S_1 \cup \dots \cup S_n) \cap S'_1 \cap \dots \cap S'_n)$  using MPROS building blocks. First, the union of the multisets  $S_i$  is determined which is equivalent to calculate the sum of ranks for all possible time slots  $s_{ij} \in S_i$  ( $\forall i, j$ ). Next, the multisets  $S'_i$  are used to eliminate all  $s_{ij} \notin S^\cap$  while preserving the sums of the preferences encoded in the union of the  $S_i$ .

The maximum of the sum of ranks is computed through an iterative process starting with a reduction by  $t = nk - 1$ . Note that if all parties assign the highest rank to the same time slot, it will appear  $nk$  times in  $(S_1 \cup \dots \cup S_n) \cap S'_1 \cap \dots \cap S'_n$ . This time slot would then be the only member of the multiset after reduction by  $t = nk - 1$ . If no such time slot exists, an empty set is obtained.  $t$  is then iteratively reduced until for some  $t = t'$  a non-empty set  $W_{t'}$  is obtained for the first time.  $W_{t'}$  only contains the optimal solution, i.e., all time slots  $s_{ij}$  maximizing the sum of ranks. Thus the privacy of the remaining elements in the multisets  $S_i, S'_i$  is guaranteed.

#### IV. AUCTION THEORY

In the following we show that MPROS building blocks prove suitable for solving certain challenges in the area of auction theory which is an applied branch of game theory [9], [10], [11]. The goal of an auction is to sell a single or multiple items to the bidder with the highest bid (standard auction) such that the outcome of the auction does not depend on the identity of the bidders. Auctions are universal in the sense that any item can be sold. The outcome of the auction is the highest bid and the identity of the corresponding bidder. All other bids and identities are kept private. We introduce reductions to MPROS building blocks for two types of auctions: First-Price Sealed-Bid Auctions and Vickrey Auctions.

##### A. First-Price Sealed-Bid Auctions

In First-Price Sealed-Bid Auctions [12] the bidders keep their bid private by submitting it in a sealed envelope to the auctioneer. The auctioneer acts as a trusted third party and determines the winner of the auction, i.e., the bidder with the highest bid, without revealing the identities and bids of the other bidders. More formally, there are  $n$  parties  $P_1, \dots, P_n$  with identities  $id_{P_1}, \dots, id_{P_n}$  and bids  $bid_{P_1}, \dots, bid_{P_n}$ .

1) *Output*: The desired result of the auction is given by the pair

$$(\{P_l\}, max_{i=1}^n bid_{P_i}) \quad \text{with } l \in arg max_{i=1}^n bid_{P_i}$$

which contains the set of winning bidders<sup>3</sup> and the highest bid.

<sup>3</sup>Note that the output can contain multiple winners. Obtaining the final winner of the auction in such a case is a general problem in auction theory. The same holds for electronic voting. A solution is out of scope of this paper.

2) *Encoding*: Our construction requires an arbitrary but fixed maximum amount  $bid_{max}$  which any party is allowed to bid. Each party  $P_i$  chooses its input set  $S_i$  such that it only contains its identity  $id_{P_i}$  where  $P_i$ 's bid is encoded as the multiplicity of this element. Since this encoding would result in multisets  $S_i$  of different size, these sets must be padded such that  $|S_i| = bid_{max} + n$  holds  $\forall i$ . Specifically, corresponding to  $bid_{P_i}$ , party  $P_i$  creates a set  $D_{bid_{max}-bid_{P_i}}$  with  $bid_{max} - bid_{P_i}$  elements chosen from the dummy domain resulting in  $S_i = \{id_{P_i}^{bid_{P_i}+n}\} \cup D_{bid_{max}-bid_{P_i}}$ .

3) *Fairness function*: MPROS building blocks are used to calculate the output of the function  $W_t = Rd_t(S_1 \cup \dots \cup S_n)$ . The union ensures that all bidders are taken into account and the reduction is used to determine the identity of the bidder with the highest bid as well as the highest bid itself. Due to the definition of the dummy sets  $D_{bid_{max}-bid_{P_i}}$ , each dummy element appears at most  $n$  times in the multiset  $S_1 \cup \dots \cup S_n$ . The bid of each party is shifted by  $n$  to ensure that the multiplicity of each bid is greater than that of any dummy element. The reduction therefore starts with  $t = bid_{max} + n - 1$  and then proceeds with decreasing values of  $t$  as long as the reduction results in the empty set. Let  $W_{t'}$  be the first non-empty set implying that the multiplicity of the winning identity was  $t' + 1$ . Thus, the winning bid is given by  $t' + 1 - n$  where  $n$  is the adjustment for the shift introduced earlier. It is important to note that the identities and bids of non-winning bidders are not revealed. Due to our construction,  $W_{t'}$  does not contain any dummies. This is important since a dummy value in the output could potentially leak information about the bid values.

A variation of First-Price Sealed-Bid Auctions are All-Pay Auctions where everybody pays their bid in the end. A solution can be obtained by computing  $W_t = S_1 \cup \dots \cup S_n$  using the same encoding as above. The output contains all identities  $id_{P_1}, \dots, id_{P_n}$  and their corresponding bids  $bid_{P_1}, \dots, bid_{P_n}$ . Since all values are revealed, MPROS is used to only ensure fairness and not to preserve privacy in this case.

##### B. Vickrey Auction

Vickrey Auctions [13] follow a procedure which is similar to First-Price Sealed-Bid Auctions except that the winner only pays the second-highest bid.

1) *Output*: The desired output is defined as the 2-tuple

$$(\{P_l\}, max_{i=1, \dots, n; i \neq l} bid_{P_i}) \quad \text{with } l \in arg max_{i=1}^n bid_{P_i}$$

where the first component is the set of bidders with the highest bid and the second component is the second-highest bid.

2) *Encoding*: Each party constructs two sets: set  $S_i$  to determine the second highest bid without leaking the identity of any bidder and set  $S'_i$  to determine the identity of the winner. For this, each party  $P_i$  chooses a random value  $r_{P_i} \in R$  from some agreed upon domain  $R$  with  $R \cap D = \emptyset$ . The domain  $R$  has to be large enough such that the probability of two parties choosing the same random number is negligible. As in First-Price Sealed-Bid Auctions the bid of each party is encoded in the multiplicity of the set element and the input multiset is padded with

dummy elements such that  $S_i = \{r_{P_i}^{bid_{P_i}+n}\} \cup D_{bid_{max}-bid_{P_i}}$  with  $|S_i| = bid_{max} + n, \forall i$ . Similarly, each party constructs  $S'_i = \{id_{P_i}^{bid_{P_i}+n}\} \cup D_{bid_{max}-bid_{P_i}}$  with  $|S'_i| = bid_{max} + n$  where the set element is party  $P_i$ 's identifier  $id_{P_i}$  and its corresponding bid  $bid_{P_i}$  is encoded in the multiplicity of the set element.

3) *Fairness function*: First, all parties use MPROS to compute  $W_t = Rd_t(S_1 \cup \dots \cup S_n)$ . The union guarantees that all bids are taken into account and the reduction is used to determine the highest and second highest bids without revealing the identities of any of the bidders. This is ensured by the fact that the multisets in this step contain only random numbers  $r_{P_i}$ . As above, the reduction starts with  $t = bid_{max} + n - 1$  and  $t$  is iteratively reduced. Let  $W_{t'}$  be the first non-empty set and  $W_{t^*}$  be the first set with cardinality greater or equal to two (i.e.,  $t^* \leq t'$ ). Then, as above, the second-highest bid is given by  $t^* + 1 - n$  and the highest bid is given by  $t' + 1 - n$ . The threshold  $t'$  is used in a second run of MPROS to compute  $W_{t'} = Rd_{t'}(S'_1 \cup \dots \cup S'_n)$ . Since  $t' + 1 - n$  is the highest bid, the reduction by  $t'$  returns a set  $W_{t'}$  which only contains the identity of the winner. Note that, for the same reasons discussed above, the set  $W_{t'}$  cannot contain any dummy elements and will thus not violate any privacy guarantees.

### C. Limitations

A further common auction type is Open Ascending-Bid Auctions. These are also called English Auctions and are open-end auctions. In this type of auctions the price is steadily raised typically until only one bidder is left. Due to the continuous process of the auction it is not clear how those types of auctions can be efficiently mapped to MPROS building blocks. The same argument holds for Open Descending-Bid Auctions, so-called Dutch Auctions, and Reverse Auctions.

Combinatorial Auctions [14], [15] are another well-studied type of auctions. In Combinatorial Auctions the bidders can place bids on combinations of items, so-called packages. The goal is to find an allocation of items to bidders which maximizes the auctioneer's income. It is allowed that the auctioneer retains items and that the bid for a package is higher than the sum of bids for the individual items. Due to the complexity of the reconciliation process, a generic mapping to MPROS building blocks does not appear to be possible.

## V. ELECTRONIC VOTING

The area of voting is another very interesting area of application for MPROS. Many voting systems, satisfying a range of different properties, have been proposed in the literature [16]. In this paper, we focus on *Approval Voting*, *Borda Count*, *Nanson's Method*, and *Weighted Voting*. Recently, a variety of cryptographic solutions to the voting problem have been proposed. A complete discussion is beyond the scope of this paper (for further details, see [17]). Prominent implementations of electronic voting systems are the public-audit *Helios* online voting system [18] and *Civitas* [19].

In voting, the secrecy of the ballot is an important requirement which allows voters to cast their vote independently and limits the possibility of bribery and other forms of coercion. In the following, we show how some common voting systems can be reduced to operations on ordered sets thus allowing the use of MPROS. It should be noted that for most practical scenarios MPROS building blocks either have to be secured against malicious adversaries in order to ensure correct and fair execution or only have to be applied in settings which do not provide incentive for cheating.

### A. Approval Voting

One prominent criterion for a voting system is the *Condorcet Criterion*. Informally, a voting system satisfies this criterion if it elects a *Condorcet Winner*, i.e., a candidate which beats all other candidates in pairwise comparison [20], [21], [22]. Approval Voting [23], [20, p. 190] is a non-ranked voting system which—in the presence of dichotomous preferences—always elects a Condorcet winner [20, p. 207]. In an approval voting election the ballot consists of a list of candidates  $c_1, \dots, c_k$ . A voter  $P_i$  can vote for any subset of candidates but cannot vote more than once per candidate, i.e.,  $vote_{P_i}(c_j) \in \{0, 1\}$ .

1) *Output*: The desired winner is then given as

$$\{c_l\} \text{ with } l \in \arg \max_{j=1}^k \sum_{i=1}^n vote_{P_i}(c_j)$$

2) *Encoding*: Since each party is allowed to vote for any number of candidates, the input multisets must be padded up to a common cardinality  $|S_i| = k(n + 1)$ . As above, the actual vote is shifted by  $n$  to prevent the output of dummy values. Let  $S'_i = \{c_j^{n+1} \mid vote_{P_i}(c_j) = 1\}$  be the set containing all candidates party  $P_i$  votes for with multiplicity  $n + 1$ . Then  $S_i = S'_i \cup D_{k(n+1)-|S'_i|}$  defines  $P_i$ 's input multiset.

3) *Fairness function*: To determine the winner, the voters jointly compute the set  $W_t = Rd_t(S_1 \cup \dots \cup S_n)$  using MPROS building blocks. The union effectively sums up all the votes per candidate and the maximum of the votes is then determined by means of the reduction operator  $Rd_t$ . The reduction begins with  $t = n(n + 1) - 1$ , since any candidate can receive at most  $n$  votes. Since each vote adds  $n + 1$  to the multiplicity of that candidate,  $t$  is iteratively reduced by  $n + 1$  until for some  $t = t'$  a non-empty set  $W_{t'}$  is obtained. The winners are given by all  $c_j \in W_{t'}$  with each  $c_j$  having received  $\frac{t'+1}{n+1}$  votes. Up to this point the computation does not reveal any information on the votes for non-winning candidates. If, however, this information is desired, the protocol can continue decreasing  $t$  until the votes for the remaining candidates are recovered also.

### B. Borda Count

The Borda count is a preferential voting system which uses a ranked ballot. Each voter  $P_i$  ranks (sorts) the candidates  $c_j$  ( $j = 1, \dots, k$ ) according to her preference. Based on its rank, candidate  $c_j$  is associated with a unique number of *points*  $points_{P_i}(c_j) = rank_{P_i}(c_j) \in \{1, \dots, k\}$ , such that the most

preferred candidate gets  $k$  points and the least preferred gets 1 point.

1) *Output*: The desired winner is the candidate which, in sum, receives the most points, i.e.,

$$\{c_l\} \text{ with } l \in \arg \max_{j=1}^k \sum_{i=1}^n \text{points}_{P_i}(c_j)$$

2) *Encoding*: The input multiset for party  $P_i$  is given by  $S_i = \{c_j^{\text{points}_{P_i}(c_j)} \mid j = 1, \dots, k\}$  with  $|S_i| = \frac{k(k+1)}{2}$ , i.e., all candidates are contained in the multiset and the multiplicity represents the points according to  $P_i$ 's ranking.

3) *Fairness function*: The fairness function is identical to the one for Approval Voting, since in both systems the candidate who receives the maximum of the sum of the votes wins. The multiset representation implicitly allows for more than one vote (points) per candidate. The winning set is then given by  $W_t = \text{Rd}_t(S_1 \cup \dots \cup S_n)$  and the reduction begins with  $t = nk - 1$  since the winner can have received at most  $k$  points from  $n$  voters. Again, the reduction may stop as soon as the winner was determined, or continue to obtain a ranking of all candidates in decreasing order of points.

### C. Nanson's and Baldwin's Method

The Borda count does not meet the Condorcet criterion and therefore might elect a candidate which is not preferred by the majority of the voters. *Nanson* and *Baldwin* (NB) proposed related elimination variants of the Borda count which are, in fact, Condorcet methods [20], [24].

The NB method creates and tallies ballots the same way as Borda count. After tallying, all candidates who received the smallest number of points are eliminated. These candidates are removed from the ballots, the results are re-tallied as if those candidates were not existent, and the procedure is repeated iteratively.

1) *Output*: Let  $v_l(c_j) = \sum_{i=1}^n \text{vote}_{l,P_i}(c_j)$  be the cumulated votes for candidate  $c_j$  in iteration  $l$ . The procedure starts with a base set  $A_1 = \{c_1, \dots, c_k\}$  and in each iteration

$$A_{l+1} = A_l \setminus \{c_j \in A_l \mid \forall x \in A_l : v_l(c_j) \leq v_l(x) \wedge \exists y \in A_l : v_l(c_j) < v_l(y)\}$$

is computed, i.e., the candidates with the least points in iteration  $l$  are eliminated if such candidates exist. Then, the winner is given by  $\lim_{l \rightarrow \infty} A_l$  [20].

2) *Encoding*: The encoding is identical to the one used for Borda Count.

3) *Fairness function*: The final result will be obtained through multiple rounds of MPROS. In each round  $W_t = \text{Rd}_t(S_1 \cup \dots \cup S_n)$  is computed. In contrast to the application described previously, these sets  $W_t$  do contain more than one element and set membership has to be tested without leaking the multiplicity, i.e., the number of votes for the candidates. By applying oblivious polynomial evaluation [25], [3] it is possible to test set membership without leaking information about the multiplicity.

The reduction determines the candidate with the minimum number of points and therefore begins with

$t = n \leq n \cdot \min(\text{points}_{P_i}(c_j))$  (since each candidate received at least  $n$  points) and  $t$  is iteratively increased until one candidate  $c_j \notin W_t$ . In the next round the parties repeat the computation of  $W_t$  using only those candidates  $c_j \in W_t$ . To achieve this, each party will rank the candidates in the same order as in the previous round but will skip all  $c_j \notin W_t$ . The computation stops as soon as  $|W_t| = 0$  for some  $t = t^*$  and the winners are given as all  $c_j \in W_{t^*-1}$ .

There exists a variant of this method which in each step eliminates all those candidates which received less than the average Borda score. Using a construction analogous to the one above, this variant can be reduced to MPROS as well.

It should be noted that membership testing in  $W_t$  requires the knowledge of all possible elements in the set. Since the list of candidates is public from the beginning, this is not a violation of privacy. In particular, the ballots of the individual parties remain private throughout the computation.

### D. Weighted Voting Systems

While in many scenarios the votes of all parties should count equally, weighted voting systems are also widely spread. These systems are used for motions, but they can also be applied to ranked systems such as, e.g., the ones based on Borda Count. A typical application of weighted voting is in contexts where representatives have differently sized constituencies [20, p. 184] such as in delegated voting, at shareholder meetings or the United States Electoral College.

In a weighted voting system each party  $P_i$  has a weight  $1 \leq w_i \leq w_{max}$  which defines party  $P_i$ 's impact on the final outcome of the election. For ranked voting systems each party  $P_i$  first ranks the candidates  $c_j$  as she would for the regular system. Then,  $P_i$  determines a new, effective,  $\text{rank}_{P_i}^{\text{eff}} = w_i \cdot \text{rank}_{P_i}(c_j)$  ( $j = 1, \dots, k$ ) and runs the protocol as if  $\text{rank}_{P_i}^{\text{eff}}(c_j)$  was the rank originally assigned to  $c_j$ .

1) *Output*: The desired output is then given by

$$\{c_l\} \text{ with } l \in \arg \max_{j=1}^k \sum_{i=1}^n \text{rank}_{P_i}^{\text{eff}}(c_j)$$

2) *Encoding*: Since all  $P_i$  can have different  $w_i \leq w_{max}$ , an encoding analogous to the Borda Count would result in differently sized sets for each party and therefore a padded encoding with  $|S_{max}| = \frac{k(k+1)}{2} w_{max} + nk$  is used:  $S_i = S'_i \cup D_{|S_{max}| - |S'_i|}$  with  $S'_i = \{c_j^{\text{rank}_{P_i}^{\text{eff}}(c_j) + n} \mid j = 1, \dots, k\}$ .

3) *Fairness function*: The fairness function is given by  $W_t = \text{Rd}_t(S_1 \cup \dots \cup S_n)$  and the winner is determined using the standard techniques described above.

### E. Limitations

Above we have shown how MPROS can be used to tally ballots and determine winners based on some fairness functions. There are many voting systems, however, which are based on techniques for which an efficient reduction to ordered sets seems difficult. Examples include graph-based algorithms (e.g., Schultze's beatpath [26], ranked pairs [27]), or algorithms which create all possible permutations of candidate

sequences (e.g., Kemeny-Young [28]). While for many of these methods it is possible to compute the tally in a privacy-preserving fashion, the winner needs to be determined outside of MPROS.

## VI. CONCLUSION

In this paper we have shown how privacy-preserving reconciliation of ordered sets can be applied to enable event scheduling, policy reconciliation, e-voting, and auctions. Specifically, we have proposed solutions for fair and privacy-preserving Doodle, First-Price Sealed-Bid Auctions, Vickrey Auctions, Approval Voting, Borda Count, Nanson's and Baldwin's Method, and Weighted Voting Systems. For each application, we have presented an encoding of the private inputs into multisets and have defined fairness functions which enable the computation of the desired outputs utilizing MPROS building blocks.

As part of future work we will design and implement a secure multi-party computation framework to make the functionality of MPROS protocols generally accessible.

## ACKNOWLEDGMENTS

In part, this work was supported by NSF Award CCF 1018616, the DFG, and the UMIC Research Center at RWTH Aachen University.

## REFERENCES

- [1] U. Meyer, S. Wetzel, and S. Ioannidis, "New Advances on Privacy-Preserving Policy Reconciliation." in *Cryptology ePrint Archive, Report 2010/64*, 2010, <http://eprint.iacr.org/2010/064>.
- [2] G. Neugebauer, U. Meyer, and S. Wetzel, "Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets," in *13th Information Security Conference (ISC) 2010*, ser. LNCS, 2010.
- [3] U. Meyer, S. Wetzel, and S. Ioannidis, "Distributed Privacy-Preserving Policy Reconciliation," in *IEEE International Conference on Communications, 2007. ICC'07*. IEEE, 2007, pp. 1342–1349.
- [4] L. Kissner and D. Song, "Privacy-Preserving Set Operations," in *Advances in Cryptology—CRYPTO 2005*, ser. LNCS, vol. 3621. Springer, 2005, pp. 241–257.
- [5] D. Ferrest, "Privacy-Preserving Policies Reconciliation in Social Networks," Diploma Thesis, RWTH Aachen University, 2010.
- [6] D. A. Mayer, D. Teubert, S. Wetzel, and U. Meyer, "Implementation and Performance Evaluation of Privacy-Preserving Fair Reconciliation Protocols on Ordered Sets," in *First ACM Conference on Data and Application Security and Privacy (CODASPY'11)*, 2011.
- [7] "ZocDoc - Find Dentists or Doctors and Book Online Instantly," <http://www.zocdoc.com/>.
- [8] "Doodle: Easy Scheduling," <http://www.doodle.com/>.
- [9] V. Krishna, *Auction Theory*. MIT Press, 2010.
- [10] R. Cassady, *Auctions and Auctioneering*. MIT Press, 1967.
- [11] P. Klemperer, *The Economic Theory Of Auctions*. Edward Elgar Pub, 2000.
- [12] J. H. Kagel and L. D., "Independent Private Value Auctions: Bidder Behavior in First, Second and Third Price Auctions with Varying Numbers of Bidders," *Economic Journal*, vol. 103, no. 419, pp. 868–879, 1993.
- [13] W. Vickrey, "Counterspeculation, Auctions and Competitive Sealed Tenders," *Journal of Finance*, pp. 8–37, 1961.
- [14] S. J. Rassenti, V. L. Smith, and R. L. Bulfin, "A Combinatorial Auction Mechanism for Airport Time Slot Allocation," *The Bell Journal of Economics*, vol. 13, no. 2, pp. 402–417, 1982.
- [15] P. Cramton, Y. Shoham, and R. Steinber, *Combinatorial Auctions*. MIT Press, 2006.
- [16] W. Smith, "Descriptions of Single-Winner Voting Systems," 2006, <http://math.temple.edu/~wds/homepage/votedesc.pdf>.

- [17] T. Moran and M. Naor, "Receipt-Free Universally-Verifiable Voting With Everlasting Privacy," in *Advances in Cryptology—CRYPTO 2006*, ser. LNCS, vol. 4117. Springer, 2006, pp. 373–392.
- [18] B. Adida, "Helios: Web-Based Open-Audit Voting," in *Proceedings of the 17th conference on Security symposium*. USENIX Association, 2008, pp. 335–348.
- [19] M. Clarkson, S. Chong, and A. Myers, "Civitas: Toward a Secure Voting System," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 354–368.
- [20] S. J. Brams and P. C. Fishburn, "Voting Procedures," in *Handbook of Social Choice and Welfare*, K. J. Arrow, A. K. Sen, and K. Suzumura, Eds. Elsevier, April 2002, vol. 1, ch. 4, pp. 173–236.
- [21] M. Condorcet, *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. L'imprimerie royale, 1785.
- [22] P. Fishburn, "Condorcet Social Choice Functions," *SIAM Journal on Applied Mathematics*, vol. 33, no. 3, pp. 469–489, 1977.
- [23] S. Brams and P. Fishburn, "Approval Voting," *The American Political Science Review*, vol. 72, no. 3, pp. 831–847, 1978.
- [24] E. J. Nanson, "Methods of Election," in *Transactions and Proceedings of the Royal Society of Victoria*, vol. 18, no. 954. The Royal Society of Victoria, 1882, pp. 197–240.
- [25] M. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," in *Advances in Cryptology - EUROCRYPT 2004*, ser. LNCS, vol. 3027. Springer, 2004, pp. 1–19.
- [26] M. Schulze, "A New Monotonic and Clone-Independent Single-Winner Election Method," *Voting matters*, vol. 17, pp. 9–19, 2003.
- [27] T. Tideman, "Independence of Clones As a Criterion for Voting Rules," *Social Choice and Welfare*, vol. 4, no. 3, pp. 185–206, 1987.
- [28] J. Kemeny, "Mathematics Without Numbers," *Daedalus*, vol. 88, no. 4, pp. 577–591, 1959.

## APPENDIX

In the following, we briefly review the protocols introduced in [4], [2] which operate on multisets, i.e., sets in which elements may occur more than once.

The work in [4] specifies algorithms for privacy-preserving intersection, union, and element reduction of multisets. In [2], these constructions are leveraged for developing multi-party privacy-preserving reconciliation protocols on ordered sets (MPROS) which allow the incorporating of the parties' preferences on their inputs. The main contribution of [2] is the design of a suitable encoding of the preferences. Based on the encoding, the MPROS protocols determine the maximum of the preferences according to a common fairness function (e.g., the sum of ranks or the minimum of ranks [3]) in a privacy-preserving manner, requiring only intersection ( $\cap$ ), union ( $\cup$ ), and reduction ( $Rd_t$ ) operations.

In MPROS protocols, multisets are represented as polynomials where a set element appearing  $m$  times is expressed as an  $m$ -fold root. All computations in the protocols are performed on encrypted polynomials utilizing the properties of an asymmetric, semantically secure, additively homomorphic threshold cryptosystem. In [2], the correctness and security of the MPROS protocols in the semi-honest model are shown assuming  $c < n$  colluding parties.

The performance of the MPROS protocols in [2] for the sum of ranks ( $Rd_t \left( \left( \bigcup_i S_i \right) \cap \left( \bigcap_j S_j \right) \right)$ ) and minimum of ranks ( $Rd_t \left( \bigcap_i S_i \right)$ ) composition scheme is polynomial-time bounded with respect to the number of parties and input elements. Similar results can be obtained for the applications presented in this paper.