Design and Implementation of Privacy-Preserving Reconciliation Protocols

Georg Neugebauer, Lucas Brutschy and Ulrike Meyer
Department of Computer Science
RWTH Aachen University

Susanne Wetzel
Department of Computer Science
Stevens Institute of Technology

ABSTRACT

Privacy-preserving reconciliation protocols on ordered sets are protocols that solve a particular subproblem of secure multiparty computation. Here, each party holds a private input set of equal size in which the elements are ordered according to the party's preferences. The goal of a reconciliation protocol on these ordered sets is then to find all common elements in the parties' input sets that maximize the joint preferences of the parties. In this paper, we present two main contributions that improve on the current state of the art. First, we propose two new protocols for privacy-preserving reconciliation and prove their correctness and security properties. We implement and evaluate our protocols as well as two previously published multi-party reconciliation protocols. Our implementation is the first practical solution to reconciliation problems in the multi-party setting. Our comparison shows that our new protocols outperform the original protocols. The basic optimization idea is to reduce the highest degree polynomial in the protocol design. Second, we generalize privacy-preserving reconciliation protocols, i. e., relaxing the input constraint from totally ordered input sets of equal size to pre-ordered input sets of arbitrary size.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Cryptographic controls

General Terms

Security, Privacy, Secure Multi-Party Computation

1. INTRODUCTION

The problem of secure multiparty computation was first introduced by Yao [24]. It solves the problem of jointly computing a function on private input sets of multiple parties without involving a third party and without revealing the private inputs of one party to any of the other parties. Privacy-preserving reconciliation protocols use ordered sets as private inputs. The goal of a reconciliation protocol on these ordered sets is then to find all common elements in the parties' input sets that maximize the joint preferences of the parties. A reconciliation protocol is privacy-preserving, if it does not reveal anything about the private inputs of a party to any other party except from what can be deduced from the desired output of the protocol.

Two-party protocols that solve the reconciliation problem for totally ordered input sets of equal size have first been proposed in [16, 17]. The performance of these two-party protocols was studied in [15]. These two-party protocols make use of a privacy-preserving set intersection protocol such as [4]. In [20, 21] the first protocols were proposed that address the multi-party case. These protocols are based on privacy-preserving operations on multisets, i.e., sets in which elements may occur more than once. In particular the protocols use set intersection, set union, and set reduction operations. Privacy-preserving protocols for these three operations were first introduced in [12]. Recently, further protocols for multi-party set intersection [2, 13, 19, 23] and set union [5, 9] have been proposed. An overview on a variety of applications of privacy-preserving reconciliation protocols including scheduling applications, electronic voting, and online auctions is provided in [14].

While the correctness and security of the multi-party reconciliation protocols has been proven in [20, 21] and their complexity has been theoretically studied, a practical performance evaluation of these protocols has not yet been conducted. We close this gap and show that in general the practical worst-case performance confirms the theoretical expectations. As a main contribution of this paper, we propose two new protocols for privacy-preserving reconciliation and prove their correctness and security properties. We evaluate the practical performance of these new protocols and compare them to the original protocols [20, 21]. This evaluation shows that the new protocols clearly outperform the previously developed protocols in practice. The main optimization is due to a reduced maximal polynomial degree in a protocol run, compare Section 3.1.

As a second contribution, we generalize the original protocols to arbitrary pre-ordered input sets. Note that the two-party protocols proposed in [16, 17] and the multi-party protocols proposed in [20, 21] were based on totally ordered input sets of equal size k. Our generalization allows parties to order their inputs less strictly and equally prefer some of their input elements. This allows for a broader application of the protocols. The main idea is to generalize the multiset intersection operation [12] to input sets of arbitrary sizes.

EDBT/ICDT '13, March 18 - 22 2013, Genoa, Italy

The rest of the paper is structured as follows: In Section 2, we review basic concepts used in our paper. In Section 3, we describe our two new reconciliation protocols. Section 4 presents the results of the practical evaluation of the two newly proposed and the two original reconciliation protocols. In Section 5, we present the generalization of reconciliation protocols to arbitrary pre-ordered input sets.

2. PRELIMINARIES

Unless stated otherwise, in this paper, we consider n parties $P_1, ..., P_n$ with input sets $S_1, ..., S_n$ with *exactly* k elements each chosen from a common domain D. Each party has certain preferences associated with its input set which allow a party to define a total order on its inputs.

2.1 Ordered Sets and Composition Schemes

We define the input of each party as follows:

DEFINITION 2.1 (ORDERED SETS & RANKING). *Let D be a set called the* domain. *$(S, <) \in 2^D \times 2^{D \times D}$ is called* ordered set *if $<$ is a strict total order on S. We write $\{x_1 > ... > x_k\}$ for the ordered set $(\{x_1, ..., x_k\}, \{(x_j, x_i) \,|1 \leq i < j \leq k\})$. The ranking function $rank_S : S \to \mathbb{N}$ is defined by $rank_S(x_i) = k - i + 1$.*

To define the goal of reconciliation protocols, we first define the composition of multiple ordered sets into a single pre-ordered set:

DEFINITION 2.2 (COMPOSITION SCHEMES). *Let $(S_1, <_1), ..., (S_n, <_n)$ be ordered sets. The pre-ordered set $\left(S_1 \cap ... \cap S_n, \lesssim_{\{S_1,..,S_n\}}\right)$ is called the* combined pre-ordered set of $(S_1, <_1), ..., (S_n, <_n)$ w.r.t. *f if $\lesssim_{\{S_1,..,S_n\}}$ is the pre-order induced by the function $f : (S_1 \cap ... \cap S_n) \to \mathbb{N}$. The function f is called the* composition scheme.

We consider two composition schemes first described in [16], that are reasonable choices, in the sense that they are unbiased with respect to the preferences, i.e., the orders of any of the parties. The first composition scheme is called the *minimum of ranks* scheme and determines the rank of an element in the intersection of all sets based on the minimal rank assigned to it by any party.

DEFINITION 2.3 (MINIMUM OF RANKS). *The* minimum of ranks composition scheme (MR) *is defined by the function*

$$f(x) = min\{rank_{S_1}(x), ..., rank_{S_n}(x)\}$$

The second composition scheme determines the rank of an element of the intersection of the ordered sets to be combined by the sum of the ranks assigned to it by each party.

DEFINITION 2.4 (SUM OF RANKS). *The* sum of ranks composition scheme (SR) *is defined by the function*

$$f(x) = rank_{S_1}(x) + ... + rank_{S_n}(x)$$

2.2 Reconciliation on Ordered Sets

Next, we define a reconciliation protocol, compare [20]:

DEFINITION 2.5 (RECONCILIATION ON ORDERED SETS). *A privacy-preserving, preference-maximizing protocol for a composition scheme f is a multi-party protocol between n parties $P_1, ..., P_n$ each with an ordered input set $(S_i, <_i)$ drawn from the same domain D. Upon completion of the protocol, each party learns (X, t):*

$$X = arg\ max_{x \in (S_1 \cap ... \cap S_n)} f(x)$$

$$t = max_{x \in (S_1 \cap ... \cap S_n)} f(x)$$

$$where\ arg\ max_{x \in D} f(x) = \{x | \forall y \in D : f(y) \leq f(x)\}$$

In addition, each party P_i learns nothing about the inputs and preferences of the other parties, except what can be deduced from X, t and its private input set $(S_i, <_i)$.

The result of the protocol is the set of common elements with maximum rank in the combined preference order and the corresponding maximum rank. In the following, we will refer to the multi-party reconciliation problem of ordered sets as MPROS and the variants for minimum of ranks and sum of ranks as MPROS^{MR} and MPROS^{SR}, respectively.

2.3 Additively Homomorphic Cryptosystem

Our protocols require a threshold version of a semantically secure, additively homomorphic, asymmetric cryptosystem. A cryptographic system which has *additively homomorphic* properties, provides an operation in the ciphertext domain to compute the encrypted sum of two plaintexts given only the corresponding ciphertexts. More formally, let $E_{pk}(\cdot)$ be the encryption function with public key pk. In an additively homomorphic cryptosystem, there is an operation $+_h$ such that $E_{pk}(a+b) = E_{pk}(a) +_h E_{pk}(b)$ which can be computed efficiently given only $E_{pk}(a), E_{pk}(b)$, and pk. It is also possible to perform efficient scalar multiplication with a scalar value s by repetitive application of the $+_h$ operation. We denote the iterative homomorphic sum by $\tilde{\sum}$, i. e., all additions are homomorphic additions $+_h$. In addition, we require the cryptosystem to provide semantic security [8]. That is, given two ciphertexts, it is infeasible for an attacker to find any meaningful relationship between the plaintexts based only on the ciphertexts.

In a (v, n)-threshold version of an additively homomorphic cryptosystem, the private key k_{priv} is shared among the n parties with each party P_i holding a private share s_i $(1 \leq i \leq n)$. Using the private key share s_i, a party P_i can compute a *partial decryption* of a ciphertext. To successfully decrypt a given ciphertext, v of the n key shares are required to compute the plaintext by combining v partial decryptions of the ciphertext. In our complexity considerations as well as in our implementation we use a threshold version of the Paillier cryptosystem [22, 3] which provides semantic security and is additively homomorphic. The plaintext space is $\mathbb{Z}_N$ where N is an RSA modulus. The ciphertext domain is $\mathbb{Z}_{N^2}$. The homomorphic addition $+_h$ is a multiplication and the scalar operation $\times_h$ a modular exponentiation in the ciphertext domain $\mathbb{Z}_{N^2}$.

2.4 Privacy-Preserving Multiset Operations

Our protocols make use of the privacy-preserving multiset operations introduced by Kissner et al. [12]. In their work, the authors give the basic building blocks for computing any function on multisets expressed by the following grammar:

$$\Upsilon ::= S_i | Rd_t(\Upsilon) | \Upsilon \cap \Upsilon | \Upsilon \cup S_i$$

Note that the above grammar prohibits the union of two multisets that are both the results of multiset operations. In addition, each private multiset intersection requires input multisets of the same size k [12]. As one of the contributions of this paper, we generalize Kissner's multiset intersection to input multisets of arbitrary size by a generalization of Kissner's Theorem 3, see Section 5. The formal definition of all multiset operations $\cap, \cup, Rd_t$ is given below.

Representing Multisets as Polynomials.

The essential idea of the operations proposed by Kissner et al. is to encode the elements of multisets as the roots of an encrypted polynomial and compute the result of the set operations using an additively homomorphic cryptosystem. A private input multiset $S_i = \{s_{i,1}, ..., s_{i,k}\}$ of party P_i is encoded by defining a polynomial $f_i(x) = \prod_{j=1}^{k}(x - s_{i,j})$ and then encrypting the coefficients of the resulting polynomial. The result of multiset operations can now be computed solely by manipulating encrypted polynomials. The sum of two polynomials can be computed by homomorphic addition of their coefficients. We denote this operation by $\phi +_h \gamma$, where ϕ and γ are encrypted polynomials.

Furthermore, the encrypted result π of a polynomial multiplication of an encrypted polynomial ϕ and a plaintext polynomial g can be computed by homomorphic addition and scalar multiplication:

$$\pi_i = \tilde{\sum}_{j=0}^{i} \phi_j \times_h g_{i-j}, \quad 0 \leq i \leq deg(\phi) + deg(g)$$

We denote this operation by $\phi \times_h g$. Finally, we can compute the derivative ϕ' of an encrypted polynomial ϕ using only homomorphic operations. A coefficient ϕ'_i of the derivative of ϕ is given by:

$$\phi_i^{'} = (i+1) \times_h \phi_{i+1}, \quad 0 \leq i \leq deg(\phi) - 1$$

Using polynomial addition, multiplication, and derivation, privacy-preserving set operations can be achieved as follows.

Union: For an encrypted polynomial ϕ and a plaintext polynomial g, representing the two sets F and G of arbitrary size, a polynomial representation of the union $F \cup G$ can be computed by homomorphic polynomial multiplication:

$$\phi \times_h g$$

The product contains all roots of f and g in the corresponding summed multiplicity. Furthermore, from the decryption of $\phi \times_h g$, one cannot learn more information than from $F \cup G$, as proven in Theorem 1 of [12].

Intersection: For two encrypted polynomials ϕ, γ representing two sets F and G of equal size, the intersection can be computed by the following term:

$$\phi \times_h s +_h \gamma \times_h r$$

Here, s and r are random polynomials of degree $deg(\phi)$. The roots of the result polynomial are those common to ϕ and γ (with minimum multiplicity) and thus represent the elements of $F \cap G$. Again, from the decryption of the resulting polynomial, one cannot learn more than from $F \cap G$. This is proven in Theorem 3 of [12] for same cardinality sets.

Set Reduction: For an encrypted polynomial γ representing a multiset G, we can compute the element reduction $Rd_t(G)$ by the following term:

$$\tilde{\sum}_{i=0}^{t} \gamma^{(i)} \times_h F_i \times_h r_i$$

Here, each r_i is chosen uniformly and independently from the set of polynomials of degree $deg(\gamma^{(i)})$ and each F_i is a fixed polynomial of degree i with the additional requirement that all fixed polynomials do not share roots, i.e., $gcd(F_0, ..., F_t) = 1$. The result $Rd_t(G)$ contains all elements $a \in G$ with multiplicity $max\{b - t, 0\}$, if an element a has multiplicity b in G. Again, see [12] for an in-depth discussion of correctness and security.

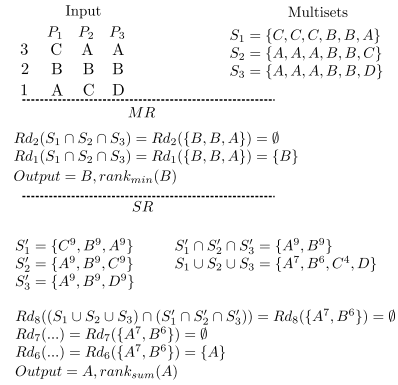

Figure 1: Example for MPROS^{MR} and MPROS^{SR}

2.5 Adversary Model

In this paper, we consider the honest-but-curious adversary model, which is also referred to as the *semi-honest model* [7]. In the semi-honest model all parties act according to the prescribed actions in the protocols. They may, however, try to infer as much information as possible from all results obtained during the execution of the protocol.

2.6 Prior MPROS Protocols

Neugebauer et al. propose protocols secure in the semi-honest model for solving MPROS^{MR} and MPROS^{SR} using a sequence of Kissner's set operations [20, 21]. Both protocols run in polynomial time for multiple parties. Each party holds *exactly* k inputs. We briefly introduce the main idea of the protocols; see [20, 21] for an in-depth description and security proofs.

The main idea of the protocols is to represent the ordered input sets of the parties as multisets, by encoding the rank of an element in the multiplicity of the element in the multiset. For example, an ordered set $\{A > B > C\}$ is encoded as the multiset $\{A, A, A, B, B, C\}$. We call this the *rank encoding*, in the following:

$$renc(S_i) = \left\{ s^{rank_i(s)} | s \in S_i \right\} \quad (1)$$

By carefully designing a sequence of multiset operations that are performed on the multiset, the authors in [20, 21] show that we can solve the reconciliation problem for the minimum of ranks as well as the sum of ranks composition scheme. In the protocols MPROS^{MR} and MPROS^{SR} each party P_i $(i = 1, ..., n)$ holds a private input set S_i which contains its k distinct inputs $s_{1i}, ..., s_{ik}$. The MPROS^{MR} protocol computes

$$Rd_t(renc(S_1) \cap ... \cap renc(S_n)) \quad (2)$$

for $t = k-1, k-2, ..., 0$ until the resulting set is non-empty for the first time. All elements in this non-empty set then maximize the minimum of ranks. The non-empty check is done by a threshold decryption of the result set and a computation of the roots of the decrypted polynomial. Similarly, the protocol MPROS^{SR} computes

$$Rd_t((renc(S_1) \cup ... \cup renc(S_n)) \cap (menc(S_1) \cap ... \cap menc(S_n))) \quad (3)$$

using the so-called multiplicity encoding:

$$menc(S_i) = \left\{ s^{n \cdot k} | s \in S_i \right\} \quad (4)$$

The protocol continues for $t = kn-1, kn-2, ..., n-1$ until the resulting set is non-empty for the first time. All elements in

this non-empty set then maximize the sum of ranks assigned by each party. The auxiliary sets $menc(S_i)$ ensure that only inputs that are common to all parties are in the resulting set. Figure 1 illustrates the operation of both protocols for three parties with three ordered inputs. The computational complexity of $MPROS^{MR}$ is $O(k^6 + n \cdot k^4)$ and $O(n^4 \cdot k^6)$ for $MPROS^{SR}$. The authors in [20, 21] show that regarding communication, we have $O(n \cdot k^3)$, respectively $O(n^3 \cdot k^3)$ where n is the number of parties and k the number of inputs. The protocols are proven secure with up to $c < n$ colluding attackers in the semi-honest model [20].

3. OPTIMIZED MPROS PROTOCOLS

In this section, we propose two new MPROS protocols. One for the minimum of ranks (MPROS^{MR}) and one for the sum of ranks composition scheme (MPROS^{SR}). The idea of the approach proposed in the section is to improve the current approach based on reduced degrees of the polynomials involved a protocol run. In the following we first describe our optimization approach in more detail and then describe the two new protocols. In Section 4 we explore the performance gains of our new protocols over the ones proposed in [20].

3.1 Optimization Approach

The MPROS protocols proposed in [20] are based on a combination of privacy-preserving intersections, unions, and reductions of multisets that encode the ordered input sets of the participating parties. The privacy-preserving multiset operations in turn are based on homomorphic additions and scalar multiplications using, e.g., the Paillier cryptosystem. Let b be the bitsize of the public key, i.e., $b = log_2(N)$, where N^2 is the modulo of the Paillier cryptosystem. Then the efficiency of the homomorphic operations can be determined as follows:

1. Homomorphic addition $+_h$: Requires a multiplication of two $2b$-bit integers. Using the Karatsuba method [11] the complexity of this multipication is $O\left(b^{1.585}\right)$.

2. Homomorphic scalar multiplication $\times_h$: Requires an exponentiation of a $2b$-bit integer with a b-bit integer. Using exponentiation by squaring with Montgomery reduction [18] leads to a complexity of $O\left(b^{2.585}\right)$.

Note that the efficiency of the set operations (intersection, union, and reduction) mostly depends on the number of homomorphic scalar multiplications used. In the original protocols, almost all such exponentiations are performed when an encrypted polynomial is multiplied by an unencrypted polynomial. The number of homomorphic multiplications per polynomial multiplication is *quadratic* in the degree of the polynomials. We can therefore conclude that the efficiency of the protocol depends largely on the *highest degree polynomial* encountered in the protocol execution. Note that the size of the highest degree polynomial encountered in the protocols depends on the encoding of the ordered input sets of the users as well as the number and type of operations performed during protocol execution, compare Table 1.

E. g., the highest degree polynomial in the original protocol for MPROS^{MR} is constructed by computing an element reduction of an intersection of ranked encoded sets of all parties. We obtain the following maximal polynomial degree:

$$deg\left(Rd_{k-1}\left(\bigcap_{i=1}^{n} renc\left(S_i\right)\right)\right) = k\cdot(k+1)+(k-1) = k\cdot(k+2)-1$$

Similarly we obtain the following maximal degree for the original protocol for MPROS^{SR}:

$$\begin{aligned} & deg\left(Rd_{n\cdot k-1}\left(\bigcup_{i=1}^{n} renc\left(S_i\right) \cap \bigcap_{i=1}^{n} menc\left(n, S_i\right)\right)\right) \\ = \;& 4\cdot n\cdot k^2 + n\cdot k - 1 \end{aligned}$$

In the following, we construct two new protocols following the general idea to reduce the maximal polynomial degree.

3.2 Optimized Protocol for MPROS^{MR}

Our new protocol is based on the following observation: Each minimum of ranks problem can be defined as a composition of several subproblems. We define extensions on ordered sets and then prove a statement about the extension of minimum of ranks problems.

DEFINITION 3.1. *An ordered set $\{s_1 \geq ... \geq s_{q+p}\}$ is called* p-extension *of a set $\{r_1 \geq .... \geq r_q\}$ iff $\forall i \in \{1...q\}$: $s_i = r_i$. A tuple of ordered sets $(S_1, ..., S_n)$ is a p-extension of another tuple of ordered sets $(R_1, ..., R_n)$ iff for all $1 \leq i \leq n$, S_i is a p-extension of R_i.*

In other words, a p-extension adds low-ranked elements to the ordered set, while keeping the q highest-ranked elements the same. The rank of each element from the original ordered set is therefore increased by p.

We can observe that if a minimum of ranks problem has a non-empty solution, this solution is correct for all extensions, as expressed by the following theorem.

THEOREM 3.1. *For every minimum of ranks problem with ordered input sets $S_1, ..., S_n$ with a non-empty solution $R \subseteq \left(\bigcap_{i=1}^{n} S_i\right)$, R is the correct solution to the minimum of ranks problems for all extensions of $(S_1, ..., S_n)$.*

PROOF. Consider an 1-extension $(E_1, ..., E_n)$ (the general case for other values of p follows by induction). Let x be an element of the domain and R be the correct minimum of ranks solution for $(S_1, ..., S_n)$, as defined by

$$R = \arg\ \max_{x\in(S_1\cap...\cap S_n)} \{\min_{1\leq i\leq n} \text{rank}_{S_i}(x)\}$$

and let R' be the solution for $(E_1, ..., E_n)$, defined accordingly. We show this property by proving that each element of R is also in R', and each value not in R is also not in R'. If $x \in R$, then

$$\begin{aligned} & x \in (S_1 \cap ... \cap S_n) \text{ and } \min_{1\leq i\leq n} \text{rank}_{S_i}(x) > 0 \\ \Rightarrow\; & x \in (E_1 \cap ... \cap E_n) \text{ and } \min_{1\leq i\leq n} \text{rank}_{E_i}(x) > 1 \end{aligned}$$

and thus $x \in R'$. If $x \notin R$, then

$$\begin{aligned} & \exists y \in (S_1 \cap ... \cap S_n) \text{ such that} \\ & \min_{1\leq i\leq n} \text{rank}_{S_i}(y) > \min_{1\leq i\leq n} \text{rank}_{S_i}(x) \\ \Rightarrow\; & \exists y \in (E_1 \cap ... \cap E_n) \text{ such that} \\ & \min_{1\leq i\leq n} \text{rank}_{E_i}(y) > \min_{1\leq i\leq n} \text{rank}_{E_i}(x) \end{aligned}$$

and therefore $x \notin R'$. As a consequence, the results R and R' contain exactly the same elements. □

Furthermore, we can observe that a p-element extension of a minimum of ranks problem with an *empty* solution also has an empty solution or a solution with a minimum of ranks of less or equal p, as expressed by the following theorem.

Size of different encodings where S is of cardinality k

Type	Symbol	Result Degree	Example
Simple	S	k	$\{A, B\}$ $\rightarrow f(x) = (x - d_A)(x - d_B)$
Rank	$renc((S, \leq))$	$\frac{k(k+1)}{2}$	$renc(\{A > B\})$ $\rightarrow f(x) = (x - d_A)^2 (x - d_B)$
Multiplicity	$menc(n, S)$	$n \cdot k^2$	$menc(2, \{A, B\})$ $\rightarrow f(x) = (x - d_A)^4 (x - d_B)^4$

Effect of set operations on the size of polynomials where $f, f_1, ..., f_n$ are polynomials

Type	Symbol	Result Degree
Element Reduction	$Rd_t(f)$	$deg(f) + t$
Set Intersection	$f_1 \cap ... \cap f_n$	$2 \cdot max\{deg(f_1), ..., deg(f_n)\}$
Set Union	$f_1 \cup ... \cup f_n$	$\sum_{i=1}^{n} deg(f_i)$

Table 1: Encodings and operations and their effect on the polynomial degree

THEOREM 3.2. *For every minimum of ranks problem with ordered input sets $(S_1, ..., S_n)$ with an empty solution, each p-extension $(E_1, ..., E_n)$ of $(S_1, ..., S_n)$ has an empty solution or a solution with a maximum minimum of ranks of less or equal p.*

PROOF. It is sufficient to show the case $p = 1$; the general case follows by induction. Consider an element x of the domain D and let $R \subseteq D$ be the minimum of ranks solution of $(E_1, ..., E_n)$. Note that $(S_1 \cap ... \cap S_n)$ is empty. If $x \notin (E_1 \cap ... \cap E_n)$, then $x \notin R$. If $x \in (E_1 \cap ... \cap E_n)$, then we have $\min_{1 \leq i \leq n} \text{rank}_{E_i}(x) = 1$, since $x \notin (S_1 \cap ... \cap S_n)$, and $(E_1, ..., E_n)$ is a 1-extension of $(S_1 \cap ... \cap S_n)$. Therefore, $x \in R$ and the maximum minimal rank of the solution is 1. □

Based on these observations, we can construct a new MPROS protocol for the minimum of ranks composition scheme that works as follows:

Let $S_i = \{s_{i1} > ... > s_{ik}\}$ be the input set of party P_i. Then, the protocol operates in rounds. In round $1 \leq l \leq k$ of the protocol, the parties compute

$$\cap_{i=1}^{n} \{s_{i1}, \ldots, s_{il}\}$$

If the resulting set is empty the parties continue with round $l+1$. If the resulting set is non-empty, the resulting set contains the common elements of all parties with the maximal minimum of ranks value $k - l + 1$.

Note that, in the worst case, this protocol requires k rounds. In each round, the protocol performs a set intersection of n sets with a maximum of k elements. Therefore, the largest polynomial degree possible that occurs during protocol execution is $2k$. The original protocol produces larger polynomials of degree $k(k + 2) - 1$. We now briefly explain the formal description of MPROS^{MR} given in Algorithm 3.1. For ease of presentation, we show the complete protocol for $c = n - 1$ which is listed in Algorithm 3.1. A secure protocol for $c < n$ can be constructed similar to the constructions in [20, 21]. In the initialization in Step 1, each party encrypts all input elements representing the highest ranked input elements, and broadcasts the encryption. Steps 2 and 3 compute the set intersection of the polynomials $\phi_{i,k-t}$ and decrypt the result. In Step 4, the result is tested for emptiness. Based on the outcome, one of two actions are performed: If the result is not empty, we have found the correct result and we terminate the protocol. If the result is empty, we consider a larger subproblem and repeat the set intersection. For this purpose, each party adds the next highest ranked elements $d_{i,k-t+1}$ to its current polynomial $\phi_{i,k-t}$ using the scalar multiplication, resulting in the polynomial $\phi_{i,k-t+1}$ broadcast for the next round. Then, the protocol returns to Step 2 and decrements t.

Correctness.

The correctness is given by Theorem 3.1 and 3.2.

Security & Privacy.

According to Definition 2.5 we have to prove that in our new protocol none of the parties learns more than what can be deduced from the optimal solution and its minimum of ranks value. The protocol consists of several rounds of set intersection operations. As shown by Kissner et al. [12] the set intersection operations in each round are privacy-preserving in the semi-honest model in the presence of at most $n - 1$ attackers. Therefore the parties learn nothing else but what can be inferred from the result of the set intersection in each round. Note that this result is the empty set in all but the last round of the protocol in which the result is the optimal solution. In addition, each party learns the number of rounds the protocol takes to terminate. In particular, in the first round, each party learns that if there is an optimal solution, then its minimum of ranks is larger than $k-1$. In the l-th round, each party learns that if there is an optimal solution, then its minimum of ranks is larger than $k - l$ etc. This, however, is equivalent to knowing the minimum of ranks value of the optimal solution. According to Definition 2.5 this is part of the desired output of a privacy-preserving reconciliation protocol. Note that this Definition in particular means that in the special case of $n = 2$ each party learns the exact preference value assigned to the optimal solution by the other party.

Complexity.

Step 1 of the algorithm is an initialization step of negligible complexity. Therefore, we can focus the analysis on Steps 2-4 which iterate up to k times. The messages broadcast in each round are γ_i and the partial decryption share of π, both of which are of size $deg(\gamma_i) \cdot b$. As the degree of γ_i is $2(k - t)$, the amount of data sent by a party is given by:

$$O\big((n-1) \cdot \sum_{j=1}^{k} 2j \cdot b\big) = O\left(n \cdot k^2 \cdot b\right)$$

Similarly, the main computations performed in each round are n polynomial multiplications of an encrypted and an unencrypted polynomial both of degree $k - t$. As each multiplication of coefficients is performed homomorphic, it cor-

Algorithm 3.1 Optimized protocol for MPROS^{MR} secure in the semi-honest model

Setting Parties $P_1, ..., P_n$ with ordered input sets $(S_i, <_i)$ for each $i \in \{1, ..., n\}$ chosen from common domain D. Keys for a (n, n) threshold version of an additively homomorphic cryptosystem have been distributed and secure channels between each pair of the n parties established.

1. Input Encryption
   (a) Each party P_i $(i = 1, ..., n)$ encrypts its highest ranked input $\phi_{i,1} = E(x - d_{i,1})$ and broadcasts the encryption.
2. Set Intersection (Initially $t = k - 1$)
   (a) Each party P_i $(i = 1, ..., n)$
      i. Chooses random polynomials $r_{i,j}$ of degree $k - t$
      ii. Calculates and broadcasts $\gamma_i = \tilde{\sum}_{j=0}^{n} (\phi_{j,k-t} \times_h r_{i,j})$, which is of degree $2 \cdot (k - t)$
   (b) Each party P_i $(i = 1, ..., n)$ calculates $\pi = \tilde{\sum}_{l=1}^{n} \gamma_i$
3. Threshold Decryption All parties together perform a threshold decryption of π
4. Result Check
   (a) Each party P_i $(i = 1, ..., n)$
      i. Calculates the set of elements of S_i which are roots of the decrypted polynomial π^*: $R = \{d \,|d \in S_i, (X - d)|\pi^*\}$
      ii. If $R \neq \emptyset$ terminates the protocol with result $(R, t + 1)$
      iii. If $R = \emptyset$ and $t = 0$ terminates the protocol with $(\emptyset, 0)$
   (b) Computes $\phi_{i,k-t+1} = \phi_{i,k-t} \times_h (x - d_{i,k-t+1})$, broadcasts the result and proceeds with Step 2 and $t = t - 1$

Algorithm 3.2 Optimized Semi-honest model protocol for $MPROS^{SR}$, part 1

Setting Parties $P_1, ..., P_n$ with ordered input sets $(S_i, <_i)$ for each $i \in \{1, ..., n\}$ chosen from common domain D.

1. Input Generation & Set Union
   (a) Each party P_i for $i = 1, ..., n$ calculates the input polynomial $f_i(x) = \prod_{d_{i,j} \in S_i} (x - d_{i,j})^{rank_{S_i}(d_{i,j})}$
   (b) P_1 sends an encryption ϕ_1 of f_1 to P_2
   (c) For each $i = 2...n$, party P_i
      i. Receives ϕ_{i-1} from P_{i-1} and computes $\phi_i = \phi_{i-1} \times_h f_i$
      ii. If $i < n$, sends ϕ_i to P_{i+1}, otherwise sends ϕ_n with $deg(\phi_n) = n \cdot \frac{k \cdot (k+1)}{2}$ to $P_1, ..., P_{n-1}$
2. Set Intersection
   (a) Each party P_i broadcasts the encryption λ_i of its input using the *enc* encoding $g_i(x) = \prod_{d_{i,j} \in S_i} (x - d_{i,j})$
   (b) Each party P_i $(i = 1, ..., n)$
      i. Chooses random polynomials $r_{i,j}$ of degree k
      ii. Calculates and broadcasts $\gamma_i = \tilde{\sum}_{j=0}^{n} (\lambda_j \times_h r_{i,j})$, which is of degree $2 \cdot k$
   (c) Each party P_i $(i = 1, ..., n)$ calculates $\pi = \tilde{\sum}_{l=1}^{n} \gamma_i$
3. Set Reduction (Initially $t = n \cdot k - 1$)
   (a) Each party P_i $(i = 1, ..., n)$
      i. Calculates the $1...t$-th derivatives of encrypted polynomial ϕ denoted by $\phi^{(1)}, ..., \phi^{(t)}$.
      ii. Chooses $t + 1$ random polynomials $q_{i,0}, ..., q_{i,t}$ of degree $0, .., t$
      iii. Calculates the encrypted polynomial $\phi_i^* = \tilde{\sum}_{l=0}^{t} \phi^{(l)} \times_h F_l \times_h q_{i,l}$ and broadcasts ϕ_i^*
   (b) Each party P_i $(i = 1, ..., n)$ computes $\Phi = \tilde{\sum}_{j=1}^{n} \phi_j^*$

Algorithm 3.3 Optimized Semi-honest model protocol for $MPROS^{SR}$, part 2

4. Set Intersection All parties jointly compute the set intersection on Φ and π and obtain the encrypted result μ
5. Threshold Decryption All parties jointly perform a threshold decryption of μ
6. Result Check Each party P_i $(i = 1, ..., n)$ calculates the set of elements of S_i which are roots of μ: $R = \{d \,|d \in S_i, (X - d)|\mu\}$
   (a) If $R = \emptyset$ and $t > 0$, proceed with Step 3 and $t = t - 1$
   (b) If $R = \emptyset$ and $t = 0$, the result of the protocol is $(\emptyset, 0)$
   (c) Otherwise the result of the protocol is $(R, t + 1)$

responds to exponentiation in $O\left(b^{2.585}\right)$. In conclusion the protocol has the following computational complexity:

$$O\left(n \cdot \sum_{j=1}^{k} j^2 \cdot b^{2.585}\right) = O\left(n \cdot k^3 \cdot b^{2.585}\right)$$

The new protocol therefore improves upon the previous protocol [20, 21], reducing the computation by a factor of $O\left(k^3\right)$ and the communication by a factor of $O\left(k\right)$.

3.3 Optimized Protocol for MPROS^{SR}

The proposed subproblem relation does not hold for MPROS^{SR}. Nevertheless, we can reduce the maximal polynomial size in the original MPROS^{SR} protocol. We accomplish this by changing the sequence of set operations. Recall that, in the original protocol, the sequence of set operations was computed as given in Equation 3. Here, the complete term inside the reduction operation can be computed at once. The only iterative operation for each t is the element reduction. Note that in the reduction step in which the solution is found, the maximal multiplicity of the set elements is 1. Otherwise the protocol would have terminated a step earlier. Furthermore, note that the only purpose of the multiplicity encodings is to remove those elements from the result, which are not contained in every input set and at the same time preserving the summed up ranks. Based on these observations, we can construct an optimized variant of the protocol by computing:

$$Rd_t\Big(renc\left(S_1\right) \cup ... \cup renc\left(S_n\right)\Big) \cap \left(S_1 \cap ... \cap S_n\right) \qquad (5)$$

The formal protocol description is given in Algorithms 3.2 and 3.3. First, in Step 1 all parties calculate the set union on their input multisets $S_1, \ldots, S_n$ using the rank encoding *renc*. This union encodes not only all the input elements that are held by the parties but it also represents the sum of the preferences for each element across all parties. In Step 2, all parties calculate the set intersection operation on the input sets. This step is necessary in order to ensure that those elements are eliminated from $S_1 \cup ... \cup S_n$ which are held by some but not all parties. All parties iteratively participate in the element reduction by t (Step 3) with $t = nk-1, ..., n-1$ on the result of the union step. The purpose of this step is to maximize the preference order. Next (Step 4), the result is intersected with the common elements obtained in Step 2. Finally, all parties participate in the threshold decryption of the result and check whether at least one of its input elements is a root of the polynomial. If this is the case, the common elements with the maximal sum of ranks are found. If this is not the case, the parties repeat the reduction and intersection step with a value t decreased by one.

Correctness.

The correctness of the protocol is due to the fact that the left side of the outermost intersection (Equation 5) will still return an empty set if no result exists for t, or a set where each element appears at most once. Therefore, it is sufficient to intersect this result of the element reduction with the set of elements that appear in all input sets, using the simple encoding instead of the multiplicity encoding (*menc*).

Security & Privacy.

According to Definition 2.5 we have to prove that in our new protocol none of the parties learns more than what can be deduced from the optimal solution and its sum of ranks value. Our protocol consists of several rounds of compositions of set intersections, set unions and set reductions. Each of these operations as well as their composition is privacy-preserving according to Kissner et al. [12]. I.e. in each round, the parties learn nothing but the result of the composition of operations carried out. Note that this result is empty in all but the last round of the protocol, in which the result is the optimal solution according to the sum of ranks scheme. In each round the reduction operation is carried out with a different value of t decremented by one. Thus, in addition to the round results, each party learns the number of rounds required to obtain the solution. In particular, in the first round the parties learn that there is no solution with a sum of ranks of nk etc. This is equivalent to learning the sum of ranks of the solution. According to Definition 2.5 this is part of the desired output of a privacy-preserving reconciliation protocol.

Complexity.

The asymptotic complexity for the optimized and the original protocol is the same. However, the constants in the complexity are smaller in the optimized protocols as the maximal polynomial degree occuring during protocol execution decreases. Note that the term inside the element reduction and the right side of the intersection can be computed once (Equation 5). In each round, the parties have to compute an element reduction and a set intersection. Although we add an additional operation, the maximal polynomial degree involved in these operations becomes smaller:

$$2 \cdot max\left\{n \cdot \frac{k(k+1)}{2} + n \cdot k - 1, 2 \cdot k\right\} = n \cdot k^2 + 3 \cdot n \cdot k - 1$$

As detailed in Section 3.1 the maximal polynomial size in the original protocol is $4 \cdot n \cdot k^2 + n \cdot k - 1$. Based on our tight coupling of polynomial degrees to computation cost, we expect that this encoding leads to a protocol with the same asymptotical complexity but lower constants. In Section 4,

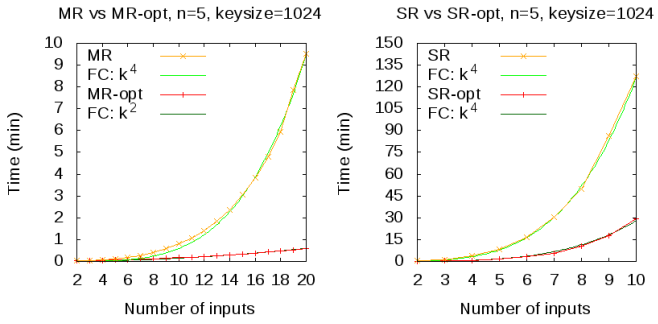

Figure 2: Runtime comparison in case $n = 5$

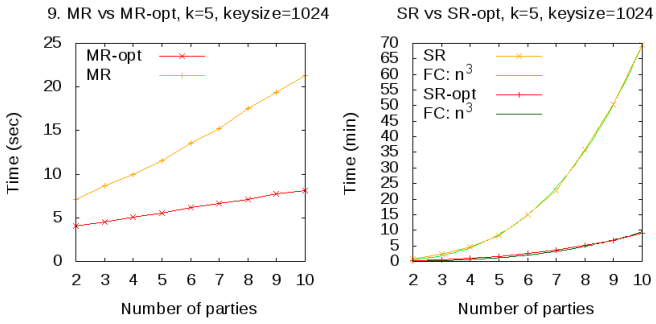

Figure 3: Runtime comparison in case $k = 5$

we provide empirical results of the speed-up achieved by the optimized protocol.

4. IMPLEMENTATION & EVALUATION

Our core implementation is written in Java. We used the GNU Multiple Precision Arithmetic Library (GMP) [6] to efficiently compute expensive arithmetic operations such as modular exponentiation. We wrote a native C++ library and used it with the Java Native Interface (JNI) [10]. GMP implements the Karatsuba and Schönhage and Strassen algorithm for multiplication of b-bit integers. For modular exponentiation, exponentiation by squaring with Montgomery reduction is implemented, compare [6]. We used the Java HotSpot(TM) VM, version 1.7.0 and GMP, version 5.0.5. We implemented the Paillier cryptosystem as the additively homomorphic cryptosystem used for secure computation. Secure channels are established via SSL, threshold key shares are predistributed, and communication is asynchronous. We implemented both MPROS protocols presented in [20, 21] as well as our newly-developed optimized variants of MPROS^{MR} and MPROS^{SR}. As an optimization, we compute all set operations in parallel, whenever a set operation is iteratively repeated for a decreasing k value (set intersection for $MPROS^{MR}$, and set reduction for $MPROS^{SR}$, compare Section 3). We also applied this optimization to the previously published protocols in [20, 21].

We evaluate the performance of all four MPROS protocols denoted as MR (Section 4.2 [20]), MR-opt (Section 3.2), SR (Section 4.2 [20]), and SR-opt (Section 3.3) with respect to computation and communication overhead. We measure the worst case complexity which means that the unbiased solution is the least preferred common input among all parties. We provide a description of our test setup in Section 4.1 and present a selection of test results in Section 4.2.

4.1 Test Environment

We chose a desktop-based test environment. The setup consists of 10 identical systems each with a 2.93 GHz i7 CPU 870 and 16 GB RAM running a 64-bit Linux with kernel version 3.2.0. All systems are connected via secure channels using TLS. Keys are distributed at start-up. Each MPROS protocol run is identified by a unique session id. The adjustable parameters for each run are the number of parties n, the number of inputs k, and the key size in number of bits b. In general, we tested all four implemented MPROS protocols with up to 10 parties and varied the number of inputs. We tested for 1024 and 2048 bit key sizes. The inputs of each party were randomly generated with one common input among all parties. In order to enforce the worst case behavior, the common input was fixed as the input with least preference for each party. The overall runtime is averaged over three independent MPROS protocol runs for each parameter set.

4.2 Test Results

We present a selection of our test results which shows the influence of the number of parties n and the number of inputs k for all four MPROS protocols. We compare the minimum of ranks protocol (MR) with our newly-developed optimized variant (MR-opt) as well as the sum of ranks protocol (SR) with our optimized variant (SR-opt). We present our results measuring the overall runtime of the four MPROS protocols for a reasonable keysize of $b = 1024$ bit varying the number of parties or the number of inputs. The impact of the keysize is roughly quadratic in the number of bits $O(b^2)$.

Figure 2 shows the results for $n = 5$. We see that, as expected in theory, the optimized variants MR-opt, and SR-opt outperform the previously developed protocols MR, and SR. We calculated the best fitting curve (FC) with the lowest asymptotic standard error for all data sets. The FC for all MPROS protocols is better than expected from theory. This is due to our optimized implementation which computes the intersection and reduction operation in parallel which nearly eliminates the iterative overhead. The performance gain of the optimized protocol MR-opt compared to the previously described protocol MR is very high. The runtime for a larger number of parties n and inputs k is in the range of seconds which is even suitable for real-time applications. The impact of the optimized SR-opt protocol is lower. We also do not obtain a better fitting curve than $O(k^4)$ in SR-opt which is the same as for SR. However, the performance of SR-opt is still better than SR and lies within the range of a few minutes rather than up to a few hours. Figure 3 shows the results for $k = 5$. Again, the optimized variants are faster than the previously described protocols. In the minimum of ranks case, the runtime is linear regarding the number of parties n. For the sum of ranks case, we obtain a runtime of $O(n^3)$. This clearly shows that both protocol types are suitable for a large number of parties n.

Table 3 shows runtime results for four different parameter sets. We see that MR-opt is the fastest protocol with a running time of seconds. The previously described MR protocol is still fast. However, the runtime lies within seconds up to minutes. The optimized sum of ranks protocol SR-opt is

Protocol	$deg(poly)_{max}$	Communication (T)	Computation (T)	Practical results
MR [20]	$k \cdot (k+2) - 1$	$O(b \cdot n \cdot k^3)$	$O(b^{2.585} \cdot (k^6 + n \cdot k^4))$	$O(n \cdot k^4)$
MR-opt	$2 \cdot k$	$O\left(b \cdot n \cdot k^2\right)$	$O\left(b^{2.585} \cdot n \cdot k^3\right)$	$O(n \cdot k^2)$
SR [20]	$4 \cdot n \cdot k^2 + n \cdot k - 1$	$O(b \cdot n^3 \cdot k^3)$	$O(b^{2.585} \cdot n^4 \cdot k^6)$	$O(n^3 \cdot k^4)$
SR-opt	$n \cdot k^2 + 3 \cdot n \cdot k - 1$	$O(b \cdot n^3 \cdot k^3)$	$O(b^{2.585} \cdot n^4 \cdot k^6)$	$O(n^3 \cdot k^4)$

Table 2: Summary of theoretical (T) and practical protocol complexities

fast compared to the old one with a running time of minutes. The slowest protocol is SR with a running time of minutes up to hours.

Table 2 summarizes the results in theory and practice. All four protocols are polynomial-time bounded with respect to the number of parties n and inputs k. All protocols are suitable for a large number of parties n. MR-opt and SR-opt even for a larger number of inputs k.

This is the first implementation solving the reconciliation problem in practice. Within our test range, the runtime varies from a few seconds up to a few hours, which seems well suited, e. g., for a distributed privacy- preserving scheduling application or an electronic voting system. However, with a large number of inputs k, e. g. $k = 100$, the protocols are not suitable for real-time applications or applications in which the result of the reconciliation is instantly needed.

5. MPROS ON PRE-ORDERED SETS

The proposed protocols enable reconciliation on ordered input sets of equal size k. This is a rather strong input restriction and not suitable for all applications. In the following, we generalize MPROS protocols to arbitrary pre-ordered input sets. First, we adjust Definition 2.1 to cope with more generic orders:

DEFINITION 5.1 (PRE-ORDERED SETS & RANKING). *Let D be a set called the* domain. *$(S, \lesssim) \in 2^D \times 2^{D \times D}$ is called* pre-ordered set *if $\lesssim$ is a total pre-order on S. We write $\{x_1 \gtrsim x_2 \gtrsim ...\}$ for the pre-ordered set $(\{x_1, x_2, ...\}, \{(x_j, x_i) \,|x_i, x_j \in S\})$. The ranking function $rank_S : S \to \mathbb{N}$ is defined for all set elements, i. e., $\forall i \in S : \exists\, rank_S(x_i)$.*

Generalizing the previously introduced reconciliation protocols from totally ordered input sets of equal size k to pre-ordered input sets of arbitrary size requires a private multiset intersection protocol that allows for the intersection of sets of different sizes. This holds even if we relax the condition on the order and still require each party to hold an input set of the same size. Unfortunately, Kissner's multiset intersection operation has only been proven to be correct and privacy-preserving for input polynomials of the same degree (see Section 2.4 and Theorem 3 & Lemma 2 of [12]). We therefore generalize the multiset intersection operation in the following.

5.1 Generalized Multiset Intersection

We generalize Kissner's Lemma 2 (page 26-27 of [12]) such that it holds for two polynomials of arbitrary degree. The general idea is, given two polynomials p_1, p_2, we choose two random polynomials r, s of the higher degree, $max\{deg(p_1), deg(p_2)\}$, to compute the intersection $p_1 \times_h r +_h p_2 \times_h s$ in a privacy-preserving manner. The generalized Lemma 2 is:

Setting	MR-opt	MR	SR-opt	SR
$n = 2, k = 10, b = 1024$	$8s$	$33s$	$3m\,24s$	$10m\,50s$
$n = 5, k = 5, b = 1024$	$3s$	$7s$	$1m\,35s$	$8m\,18s$
$n = 5, k = 5, b = 2048$	$8s$	$23s$	$7m$	$32m\,49s$
$n = 10, k = 5, b = 1024$	$9s$	$21s$	$9m$	$69m\,30s$

Table 3: Runtime for some example configurations

LEMMA 5.1. *Let f, g be polynomials in $R[x]$ where R is a ring such that no polynomial-time bounded adversary can find the size of its subfields with non-negligible probability, $deg(f) = \alpha, deg(g) = \gamma, \beta \geq \alpha \geq \gamma, gcd(f, g) = 1$, and $f[deg(f)] \in R^* \wedge g[deg(g)] \in R^*$. Let $r = \sum_{i=0}^{\beta} r[i]x^i$ and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \leq i \leq \beta}\ r[i] \leftarrow R$, $\forall_{0 \leq i \leq \beta}\ s[i] \leftarrow R$ (independently). Let $u = f * r + g * s = \sum_{i=0}^{\alpha+\beta} u[i]x^i$. Then $\forall_{0 \leq i \leq \alpha+\beta}\ u[i]$ are distributed uniformly and independently over R.*

Proof.

Our goal is to calculate the number z of r, s pairs such that $f * r + g * s = u$ for any fixed polynomials f, g, u with $gcd(f, g) = 1$. If the number of possible result polynomials u is equal to the total number of possible r, s pairs divided by z, then this implies that the coefficients of the result polynomial u are distributed uniformly if we choose the coefficients of r, s uniformly and independently from R. Let us assume there exists at least one pair $\hat{r}, \hat{s}$ for a specific u such that $f * \hat{r} + g * \hat{s} = u$. For any pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$, it holds that

$$f * \hat{r} + g * \hat{s} = f * \hat{r}' + g * \hat{s}'$$
$$f * (\hat{r} - \hat{r}') = g * (\hat{s}' - \hat{s}).$$

As $gcd(f, g) = 1$, we conclude that $g | \hat{r} - \hat{r}'$ and $f | \hat{s}' - \hat{s}$ using Kissner's Lemma 22 [12]. We may apply Lemma 22, since it is proven for polynomials of arbitrary degree. Let

$$p * g = \hat{r} - \hat{r}' \quad \wedge \quad p * f = \hat{s}' - \hat{s}. \tag{6}$$

On the one hand we have to show that there exists no pairs $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$ that are not generated by a single choice of the polynomial p of degree at most $\beta - \alpha$. On the other hand we need to show that each polynomial p, of degree at most $\beta - \alpha$, determines exactly one unique pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$. We first show that the two equations in (6) can only be valid if the degree of p is at most $\beta - \alpha$. The degree of $\hat{r} - \hat{r}'$ is β and the degree of g is γ. The product of g and p yields a polynomial of degree $\gamma + deg(p)$. Since the result $\hat{r} - \hat{r}'$ is of degree β, $deg(p)$ can be $\beta - \gamma$ at most. The degree of $\hat{s}' - \hat{s}$ is β and the degree of f is α. The product of f and p yields a polynomial of degree $\alpha + deg(p)$. Since the result $\hat{s}' - \hat{s}$ is of degree β, $deg(p)$ can at most be $\beta - \alpha$. Since it holds that $\alpha \geq \gamma$, the two equations are only valid if $deg(p)$ is at most $\beta - \alpha$.

Now we show that there exists no pairs $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$, that are not generated by some choice of one polynomial p of degree at most $\beta - \alpha$. Let $p' * g = \hat{r} - \hat{r}'$ and $p * f = \hat{s}' - \hat{s}$ be valid for any p', p. As we proved that $g | \hat{r} - \hat{r}'$ and $f | \hat{s}' - \hat{s}$, we can represent f and g as

$$f * (\hat{r} - \hat{r}') = g * (\hat{s}' - \hat{s}) \quad \wedge \quad f * (p' * g) = g * (p * f).$$

We apply Lemma 21 [12] as the leading coefficients of f and

g are members of R^*:

$$f * (p' * g) = (g * p) * f$$
$$\Rightarrow p' * g = g * p \quad \Rightarrow \quad p' = p \quad \text{(Lemma 21)}$$

We have shown our assumption since it holds that $p = p'$. Finally we show that each polynomial p exactly determines one unique pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$. It holds that $\hat{r}' = \hat{r} - g * p$, $\hat{s}' = \hat{s} + f * p$ and $f, g, \hat{r}, \hat{s}$ are fixed. Thus a choice of p determines both $\hat{r}', \hat{s}'$. The uniqueness is guaranteed due to Lemma 21 [12]. If these assignments were not unique, there would exist polynomials p, p' such that either $\hat{r}' = \hat{r} - g * p = \hat{r} - g * p'$ or $\hat{s}' = \hat{s} + f * p = \hat{s} + f * p'$ for some polynomials $p \neq p'$. As a result the number of polynomials p, of degree at most $\beta - \alpha$, is exactly equivalent to the number of r, s pairs such that $f * r + g * s = u$ and there are $z = |R|^{\beta-\alpha+1}$ such polynomials p. There are $|R|^{2\beta+2}$ r, s pairs. As $\frac{|R|^{2\beta+2}}{z} = \frac{|R|^{2\beta+2}}{|R|^{\beta-\alpha+1}} = |R|^{\alpha+\beta+1}$ which is equivalent to the $|R|^{\alpha+\beta+1}$ possible result polynomials u. □

We have shown that all $u[i]$ are distributed uniformly and independently over R if we choose both random polynomials r, s of the degree $\beta = max\{deg(f), deg(g)\}$. Thus, we have shown that the multiset intersection operation is also correct and privacy-preserving for polynomials with arbitrary degrees.

5.2 Arbitrary Pre-Ordered Input Sets

Lemma 5.1 enables MPROS protocols on pre-ordered input sets of arbitrary size. The optimized protocol variants shown in Section 3 as well as the previously described protocols in [20, 21] can be reused in the following way. Using Lemma 5.1, for each multiset intersection operation we choose random polynomials of the maximal degree of the given (encrypted) input polynomials. This however requires the parties to reveal the size of their input polynomials as well as the size of the encoded input sets to each other. In applications in which this is considered to be a privacy breach, further adjustments to the protocols with size hiding techniques (e.g. as described in [1]) are required. One possible technique is by using random dummy elements chosen as part of the ordered input to fill up the input set up to a previously agreed size.

6. CONCLUSION

We proposed two new protocols for privacy-preserving reconciliation on ordered sets. We showed that our newly developed protocols considerably improve the theoretical complexity of the state-of-the art MPROS protocols [20, 21]. We implemented the new protocols as well as the two most efficient previously published multi-party reconciliation protocols. The evaluation of our implementation confirms the theoretical expectations and shows that our newly developed protocols clearly outperform the previously proposed protocols in practice. In addition, we generalized privacy-preserving reconciliation protocols from ordered input sets of equal size k to pre-ordered input sets of arbitrary size. This enables applications in which participants are allowed to equally prefer some of their input elements. As future work, we strive to design and implement protocol variants secure in the malicious model and evaluate the generalized reconciliation protocols with arbitrary input sets.

7. ACKNOWLEDGMENTS

This work has been supported by the DFG project ME 3704/1-1 and NSF Award CCF 1018616.

8. REFERENCES

[1] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (if) size matters: size-hiding private set intersection. PKC'11, pages 156–173, Berlin, Heidelberg, 2011. Springer-Verlag.

[2] Jung Hee Cheon, Stanislaw Jarecki, and Jae Hong Seo. Multi-party privacy-preserving set intersection with quasi-linear complexity. Cryptology ePrint Archive, Report 2010/512, 2010.

[3] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Proc. of the 4th Intern. Conference on Financial Cryptography*, FC '00, pages 90–104, London, UK, 2001. Springer-Verlag.

[4] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Proceedings of EUROCRYPT'04*, 2004.

[5] Keith Frikken. Privacy-preserving set union. In *Proc. of the 5th Intern. Conf. on Applied Cryptography and Network Security*, ACNS '07, pages 237–252. Springer-Verlag, 2007.

[6] GNU Multiple Prec. Arithmetic Library. http://gmplib.org/.

[7] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *STOC '87: Proc. of the Nineteenth Annual ACM Conference on Theory of Computing*. ACM, 1987.

[8] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *ACM Symposium on Theory of Computing - STOC 1982*, pages 365–377. ACM, 1984.

[9] Jeongdae Hong, Jung Woo Kim, Jihye Kim, Kunsoo Park, and Jung Hee Cheon. Constant-round privacy preserving multiset union. *IACR Cryptology ePrint Archive*, page 138, 2011.

[10] JNI: Java Native Interface for integration of code written in other languages. http://java.sun.com/docs/books/jni/.

[11] A. Karatsuba and Yu. Ofman. Multiplication of Many-Digital Numbers by Automatic Computers. In *Proceedings of the USSR Academy of Sciences 145*, 1962.

[12] L. Kissner and D. X. Song. Privacy-Preserving Set Operations (Last modified June 2006). In *CRYPTO*, pages 241–257, 2005.

[13] R. Li and C. Wu. An unconditionally secure protocol for multi-party set intersection. In *ACNS '07*, pages 226–236. Springer-Verlag, 2007.

[14] D. Mayer, G. Neugebauer, U. Meyer, and S. Wetzel. Enabling fair and privacy-preserving applications using reconciliation protocols on ordered sets. In *34rd IEEE Sarnoff Symposium*, Princeton, 2011.

[15] 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.

[16] U. Meyer, S. Wetzel, and S. Ioannidis. Distributed privacy-preserving policy reconciliation. In *ICC*, pages 1342–1349, 2007.

[17] U. Meyer, S. Wetzel, and S. Ioannidis. New advances on privacy-preserving policy reconciliation. In *iacr eprint 2010/64*, 2010. http://eprint.iacr.org/2010/064.

[18] Peter Montgomery. Modular Multiplication Without Trial Division. In *Math. Computation, vol. 44*, 1985.

[19] G. S. Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhary, and C. P. Rangan. Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security. In *Cryptology and Network Security*, pages 21–40. Springer-Verlag, 2009.

[20] G. Neugebauer, U. Meyer, and S. Wetzel. Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets. In *Proc. of ISC 2010*, LNCS.

[21] G. Neugebauer, U. Meyer, and S. Wetzel. Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets (Extended Version). Cryptology ePrint Archive, Report 2010/512, 2011.

[22] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology*, pages 223–238. Springer-Verlag, 1999.

[23] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Selected areas in cryptography. chapter Information Theoretically Secure Multi Party Set Intersection Re-visited, pages 71–91. Springer-Verlag, Berlin, Heidelberg, 2009.

[24] A. Yao. Protocols for Secure Computation. In *Proc. of the IEEE (FOCS)'82*, 1982.