Privacy-Preserving Multi-party Reconciliation Using Fully Homomorphic Encryption

Florian Weingarten[1], Georg Neugebauer[1], Ulrike Meyer[1], and Susanne Wetzel[2]

[1] LuFG IT-Security, UMIC Research Centre
RWTH Aachen University, D-52074 Aachen, Germany
{weingarten,neugebauer,meyer}@umic.rwth-aachen.de
[2] Department of Computer Science, Stevens Institute of Technology
NJ 07030, USA
swetzel@stevens.edu

Abstract. Fully homomorphic cryptosystems allow the evaluation of arbitrary Boolean circuits on encrypted inputs and therefore have very important applications in the area of secure multi-party computation. Since every computable function can be expressed as a Boolean circuit, it is theoretically clear how to achieve function evaluation on encrypted inputs. However, the transformation to Boolean circuits is not trivial in practice. In this work, we design such a transformation for certain functions, i.e., we propose algorithms and protocols which make use of fully homomorphic encryption in order to achieve privacy-preserving multi-party reconciliation on ordered sets. Assuming a sufficiently efficient encryption scheme, our solution performs much better than existing approaches in terms of communication overhead and number of homomorphic operations.

Keywords: privacy, secure group computation, cryptographic protocols, multi-party reconciliation protocols, fully homomorphic encryption.

1 Introduction

The problem of secure multi-party computation was first introduced by Yao [1]. It is about jointly computing a function on private inputs of multiple parties without involving another trusted party and without revealing the private inputs of any party. Privacy-preserving reconciliation protocols on ordered sets are protocols that solve a particular subproblem of secure multi-party computation. Here, each party holds a private input set 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. 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 [2,3]. The performance of these two-party protocols was studied in [4]. They make use of a privacy-preserving set intersection protocol such as [5]. In [6,7] the first protocols were proposed which

J. Lopez, X. Huang, and R. Sandhu (Eds.): NSS 2013, LNCS 7873, pp. 493–506, 2013.

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 [8]. Recently, further protocols for multi-party set intersection [9,10,11,12] and set union [13,14] 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 [15].

As a main contribution of this paper, we utilize fully homomorphic encryption [16,17] to design two new protocols for multi-party reconciliation on ordered sets and analyze their security properties and efficiency. Our two variants can guarantee more privacy regarding the output of the protocol compared to [6,7]. Our evaluation shows that the new protocols outperform the previously developed protocols in terms of communication and number of homomorphic operations.

The rest of this paper is structured as follows: In Sect. 2, we briefly review basic concepts used in our paper. In Sect. 3, we describe our two new reconciliation protocols. Sect. 4 presents the comparison of our new protocols with the previously developed protocols. In Sect. 5, we draw conclusions of our results.

2 Preliminaries

In this section, we will lay down some preliminaries. In particular, we will define the setting of reconciliation problems as well as outline the multi-party solution by Neugebauer et al. [6] since their core idea is the ground work for our solution in the next section. We introduce some necessary notation and tools from the area of Fully Homomorphic Encryption (FHE).

We consider n parties $\mathcal{A}_1, ..., \mathcal{A}_n$ with private input sets $P_{\mathcal{A}_1}, ..., P_{\mathcal{A}_n} \subseteq P$, each having exactly k (pairwise distinct) elements chosen from a common input domain P. Each party has certain "preferences" associated with its input set which orders a party's elements. The preference of a rule in this ordering is called its *rank* and is identified by a bijective function $\text{rank}_{\mathcal{A}_i} : P_{\mathcal{A}_i} \to \{1, ..., k\}$.

The goal of a reconciliation protocol is to find the "best" common input elements in a fair way, i. e., taking the preferences of all parties equally into account. The two notions of fairness introduced by Meyer et al. [3], called *preference order composition schemes*, are defined by the following two functions.

Definition 1. *For a common input* $x \in P_{\mathcal{A}_1} \cap ... \cap P_{\mathcal{A}_n}$*, define the functions* $f_{SR}, f_{MR} : \bigcap_{i=1}^{n} P_{\mathcal{A}_i} \to \mathbb{N}$ *by*

$$f_{SR}(x) := \text{rank}_{\mathcal{A}_1}(x) + ... + \text{rank}_{\mathcal{A}_n}(x) \text{ and}$$
$$f_{MR}(x) := \min(\text{rank}_{\mathcal{A}_1}(x), ..., \text{rank}_{\mathcal{A}_n}(x))$$

called sum of ranks (SR) *and* minimum of ranks (MR).

With f_{SR}, maximizing the combined rank means finding one or more rules which are ranked as high as possible by all parties, i.e., the ranks of all parties count. With f_{MR}, we want to find a rule which is not ranked very low by any party,

i.e., only the smallest ranking of that rule counts. Depending on the concrete application, one of those two definitions of "fairness" might be preferred.

2.1 Reconciliation on Ordered Sets

We now give a formal definition of a reconciliation protocol.

Definition 2 (Reconciliation on Ordered Sets). *A* reconciliation protocol on ordered sets *for a preference composition scheme f is a multi-party protocol between n parties $\mathcal{A}_1, ..., \mathcal{A}_n$, each with an ordered input set $(P_{\mathcal{A}_i}, \text{rank}_{A_i})$ as described above. As output of the protocol, each party learns the maximal rank $\max_{x \in I} f(x)$ (if one exists) as well as the set of all rank-maximizing elements $\arg\max_{x \in I} f(x) = \{x \in I \mid \forall y \in I \ : \ f(y) \leq f(x)\}$ where $I := P_{\mathcal{A}_1} \cap ... \cap P_{\mathcal{A}_n}$.*

In the following, we will refer to the multi-party reconciliation problem on ordered sets as MPROS and the variants for minimum of ranks and sum of ranks as MPROS^{MR} and MPROS^{SR}, respectively.

2.2 Adversary Model

In this paper, we consider the honest-but-curious adversary model, which is also referred to as the *semi-honest model* [18]. In this model, all parties are assumed to 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.

Definition 3 (Security of Reconciliation Protocols). *A reconciliation protocol on ordered sets (as defined in Definition 2) is said to be* privacy-preserving (in the semi-honest model) *if none of the participating parties gains any additional information about the other party's private inputs except from what can be deduced from the protocol output, i.e., the maximal rank and the set of rank-maximizing elements.*

2.3 Prior Privacy-Preserving MPROS Protocols

Neugebauer et al. [6,7] proposed an approach which is based on the work of Kissner and Song [8] about privacy-preserving multiset operations. 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, which allows to perform certain operations on encrypted polynomials.

The basic idea of Neugebauer et al. is to encode the ordered inputs as multisets where the rank of each element is encoded by the element's multiplicity in the set. The preference order composition schemes f_{SR} and f_{MR} can then be modeled using operations on these multisets. Privacy is preserved by the use of a semantically secure [19] additively homomorphic cryptosystem.

Let $P_{\mathcal{A}_1}, ..., P_{\mathcal{A}_n} \subseteq P$ be the inputs of the parties and $P \subseteq M$ be the set of possible inputs, encoded as elements of the plaintext space M of some semantically secure and additively homomorphic cryptosystem. Assume the rank of each input element $p \in P_{\mathcal{A}_i}$ is encoded by its multiplicity in the set $S_{\mathcal{A}_i}$, i.e., p appears $\text{rank}_{\mathcal{A}_i}(p)$ times in $S_{\mathcal{A}_i}$. Every such set $S_{\mathcal{A}_i}$ with k distinct elements can now be identified by a polynomial f_i of degree $\sum_{j=1}^{k} j = \frac{1}{2}k(k+1)$ such that the distinct roots of f_i correspond to the distinct set elements and the multiplicity of the root corresponds to the multiplicity of the element in $S_{\mathcal{A}_i}$. When using such multiset representations, the multiset intersection operation precisely coincides with the definition of f_{MR}. The MPROS^{MR} protocol computes $\text{Rd}_t(S_{\mathcal{A}_1} \cap ... \cap S_{\mathcal{A}_n})$ for $t = k-1, k-2, ..., 0$ until the resulting set is non-empty for the first time, where $\text{Rd}_t(A)$ for some multiset A denotes the set which results from *reducing* the multiplicity of each element in A by (up to) t. All elements in this non-empty set then maximize the minimum of ranks. The emptiness check is done by a threshold decryption of the result set and a computation of the roots of the decrypted polynomial. Threshold decryption is possible in a threshold version of an additively homomorphic cryptosystem. The private key is shared among the n parties with each party $\mathcal{A}_i$ holding a private share s_i. Using s_i, a party can now compute a *partial decryption* of a ciphertext. To successfully decrypt a given ciphertext, a certain number of key shares are required to compute the plaintext by combining the partial decryptions of the ciphertext.

The construction of [8,6] for the sum of ranks preference order composition scheme f_{SR} works as follows. At first glance, multiset union seems to coincide with f_{SR} because the multiplicity of an element in the union is defined as the sum of the multiplicities of the element in the single sets. However, we have to rule out all elements which are not in the intersection because there may be elements in the union that are not shared by all parties. Neugebauer et al. describe a protocol for MPROS^{SR} which computes $\text{Rd}_t((S_{\mathcal{A}_1} \cup ... \cup S_{\mathcal{A}_n}) \cap (S'_{\mathcal{A}_1} \cap ... \cap S'_{\mathcal{A}_n}))$ where $S'_{\mathcal{A}_i}$ contains the same elements as $S_{\mathcal{A}_i}$ but every element has multiplicity $n \cdot k$. Like before, 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 $S'_{\mathcal{A}_i}$ ensure that only inputs that are common to all parties are contained in the resulting set.

2.4 Fully Homomorphic Encryption

In the following, we make use of an asymmetric *fully homomorphic encryption scheme* which operates on the binary plaintext space $M = \mathbb{F}_2 = \{0, 1\}$ and generates ciphertexts from some set C via a probabilistic polynomial-time encryption algorithm E_{pk}. There is a (deterministic) polynomial-time decryption algorithm D_{sk} such that $\mathsf{D}(\mathsf{E}(m)) = m$ for all $m \in M$. Furthermore, we have algorithms $\boxplus_{\text{pk}}$ and $\boxtimes_{\text{pk}}$ which perform homomorphic operations, i.e., $\mathsf{D}(\mathsf{E}(m_1) \boxplus \mathsf{E}(m_2)) = m_1 + m_2$ and $\mathsf{D}(\mathsf{E}(m_1) \boxtimes \mathsf{E}(m_2)) = m_1 \cdot m_2$, where $+$ and $\cdot$ denote addition and multiplication in $\mathbb{F}_2$, i.e., binary XOR and AND. We also use the notations $\mathsf{E}, \mathsf{D}, \boxplus$ and $\boxtimes$ on tuples to denote component-wise application.

We use the notation $T^{\mathsf{A}}_{\mathsf{B}}(\ell)$ to denote the number of times algorithm A calls algorithm B on inputs of bit-length ℓ, for example $T^{\mathsf{A}}_{\boxplus}(\ell)$, $T^{\mathsf{A}}_{\boxtimes}(\ell)$, $T^{\mathsf{A}}_{\mathsf{E}}(\ell)$, or $T^{\mathsf{A}}_{\mathsf{D}}(\ell)$

to denote the runtime of A in terms of number of homomorphic additions, homomorphic multiplications, encryptions, or decryptions. We will write $T^{\mathsf{A}}(\ell)$ to denote the total number of homomorphic operations (both additions and multiplications). For a probabilistic algorithm, we write $x \leftarrow \mathsf{A}(x)$ to denote that x is one possible output of algorithm A on input x.

Furthermore, we use a couple of common "tool" algorithms which operate on Boolean inputs and only use XOR and AND operations and can therefore be adapted to operate on encrypted data by using a fully homomorphic encryption scheme. We omit the description of those algorithms due to lack of space, but their implementation is straightforward and mimics the common circuit implementations of those gadgets, see e.g. [20]. In particular, for inputs $c, d \in C^\ell$ with plaintext bit-length ℓ, we use the following algorithms:

- Negation: $\mathsf{Not}(c)$ for $\ell = 1$ flips a bit, i.e., $\mathsf{D}(\mathsf{Not}(c)) = \mathsf{D}(c) + 1$. $O(1)$
- Equality: $\mathsf{Equal}(c, d)$ returns an encryption of 1 if $\mathsf{D}(c) = \mathsf{D}(d)$ and an encryption of 0 otherwise. This can be generalized to $m \geq 2$ inputs. $O(m\ell)$
- Greater than: $\mathsf{GT}(c, d)$ returns an encryption of 1 if $\mathsf{D}(c) > \mathsf{D}(d)$ and an encryption of 0 otherwise. Here, $>$ denotes the order on M^ℓ, interpreted as binary representations of natural numbers. $O(\ell)$
- If-Then-Else: $\mathsf{IFE}(b, c, d)$ for $b \in C$ returns an encryption of $\mathsf{D}(c)$ if $\mathsf{D}(b) = 1$ and an encryption of $\mathsf{D}(d)$ otherwise. $O(\ell)$
- Maximum and Minimum: $\mathsf{Max}(c, d)$ and $\mathsf{Min}(c, d)$ which return encryptions of $\max(\mathsf{D}(c), \mathsf{D}(d))$ and $\min(\mathsf{D}(c), \mathsf{D}(d))$. This can be generalized to $m \geq 2$ inputs. $O(m\ell)$
- Addition: $\mathsf{Add}(c, d)$ returns an encryption of $\mathsf{D}(c) + \mathsf{D}(d)$, where $+$ denotes addition of binary numbers with carry. The output ciphertext tuple has length $\lceil \log_2(m) \rceil + \ell$. $O(m(\log(m) + \ell))$

The asymptotic complexities are given in terms of homomorphic operations, i.e., homomorphic additions and multiplications. More details on the tool algorithms are given in [21].

3 Our Contribution

3.1 FHE-Based Algorithm

We assume that $P_{\mathcal{A}_i} \subseteq \{0,1\}^\ell \setminus \{0^\ell\}$, so all parties agree on an ℓ-bit binary encoding of the possible inputs such that 0^ℓ is not a valid input encoding. Let χ_i denote an "extended" rank function $\mathrm{rank}_{\mathcal{A}_i}$ which assigns the rank 0 to all elements which are not included in the input of party $\mathcal{A}_i$ at all:

$$\chi_i : \{0,1\}^\ell \to \{0, ..., k\},\ x \mapsto \begin{cases} \mathrm{rank}_{\mathcal{A}_i}(x) & x \in P_{\mathcal{A}_i} \\ 0 & x \notin P_{\mathcal{A}_i} \end{cases}$$

If $k := |P_{\mathcal{A}_1}| = ... = |P_{\mathcal{A}_n}| \leq |P| < 2^\ell$ is the number of inputs, we can assume that every χ_i maps to $\{0,1\}^K$ instead of $\{0, ..., k\}$ where $K = \lceil \log_2(k+1) \rceil$. In

practice, the rank function χ_i of a party $\mathcal{A}_i$ could for example be described by an ordered list of $|P|$ bit-strings each of length K, i.e., a complete truth table of all K output components. Let $X_1, ..., X_n$ be "encryptions" of the extended rank functions such that we have $X_i : \{0,1\}^\ell \to C^K$ with $\mathsf{D}(X_i(x)) = \chi_i(x)$ for all $x \in \{0,1\}^\ell$. For an input element $x \in \{0,1\}^\ell$, the value $X_i(x) \in C^K$ is an encryption of the rank of x in the input of $\mathcal{A}_i$.

Example. Let $P = \{a, b, c, d, e, f\}$ be the possible inputs and $P_{\mathcal{A}_1} = \{a, b, e\}$ the input of $\mathcal{A}_1$ with ranking $a <_{\mathcal{A}_1} e <_{\mathcal{A}_1} b$. We need at least $\ell = 3$ bits for encoding the possible inputs, for example $a = 001$, $b = 010$, $c = 011$, $d = 100$, $e = 101$, and $f = 110$. Assume that the parties agreed to use $k = 3$ input elements in their inputs, we therefore need $K = \lceil \log_2(3+1) \rceil = 2$ bits to encode each possible rank (including 0). The extended rank function of $\mathcal{A}_1$ and an encryption of it now look as follows:

Input $x \in P \subseteq \{0,1\}^\ell$	$\chi_1(x) \in \{0,1\}^K$	$X_1(x) \in C^K$
000 (invalid)	–	–
001 (a)	01 (1)	$\mathsf{E}(01)$
010 (b)	11 (3)	$\mathsf{E}(11)$
011 (c)	00 (-)	$\mathsf{E}(00)$
100 (d)	00 (-)	$\mathsf{E}(00)$
101 (e)	10 (2)	$\mathsf{E}(10)$
110 (f)	00 (-)	$\mathsf{E}(00)$

The ordered list of $|P| = 6$ ciphertext K-tuples of the third column of this table is what $\mathcal{A}_1$ sends to the other participants.

First, we look at the case of f_{MR} because this can be modeled as a multiset intersection as in [6] and as described in Sect. 2.3. We will use the tool algorithms from Sect. 2.4.

Reconciliation Algorithm for f_{MR}

Input: Encrypted $X_1, ..., X_n$ as described before and one $P_{\mathcal{A}_i}$ in plaintext.

1. Set $S := \emptyset$ and $R := \emptyset$.
2. $\forall x \in P_{\mathcal{A}_i}$, compute $y \leftarrow \mathsf{Min}(X_1(x), ..., X_n(x))$ and add (x, y) to S.
3. Set $max_y \leftarrow \mathsf{E}(0) \in C^K$.
4. For each $(x, y) \in S$, compute $max_y \leftarrow \mathsf{Max}(y, max_y)$.
5. For each $(x, y) \in S$, compute $x' \leftarrow \mathsf{E}(x) \boxtimes \mathsf{Equal}(y, max_y)$ and add it to the set R.
6. Return (max_y, R).

Output: Encrypted maximal rank max_y and a set R of encrypted elements which either have maximal rank max_y or decrypt to 0^ℓ. If there exists no maximal rank (because the inputs are disjoint), max_y will decrypt to 0 and all elements in R will decrypt to 0^ℓ.

In Step 2, party $\mathcal{A}_i$ computes the value of f_{MR} for all elements. Since $P_{\mathcal{A}_1} \cap \ldots \cap P_{\mathcal{A}_n} \subseteq P_{\mathcal{A}_i}$, the set $P_{\mathcal{A}_i}$ is an upper bound for the intersection and party $\mathcal{A}_i$ never needs to process more than $k = |P_{\mathcal{A}_i}|$ elements. For each element, the minimum of ranks is computed and the result together with its encrypted rank is added to the set S. In Step 4, we iterate over all the elements from Step 2 and compute the maximum by comparing with the "largest" element known at this point. In Step 5, we iterate again over all elements (of length ℓ) and multiply them with the result of Equal, thus effectively canceling out all elements which have rank other than max_y. Elements which are not shared by all parties are also implicitly canceled because they all have rank 0.

Computational Complexity. In Step 2, we loop $|P_{\mathcal{A}_i}| = k$ times and call Min each time on n inputs of length K. In Step 4, we call Max on inputs of length K and in Step 5, we call Equal on inputs of length K. The result is multiplied with ℓ bits. This gives a total of

$$\underbrace{\overbrace{|P_{\mathcal{A}_i}|}^{=k} \cdot T^{n\mathsf{Min}}(K)}_{\text{Step 2}} + \underbrace{\overbrace{|S|}^{=k} \cdot T^{\mathsf{Max}}(K)}_{\text{Step 4}} + \underbrace{\overbrace{|S|}^{=k} \cdot \ell \cdot T^{\mathsf{Equal}}(K)}_{\text{Step 5}}$$

$$\in k \cdot (O(nK) + O(K) + O(\ell K)) \subseteq O(nk \log(k)\ell)$$

homomorphic operations. As already mentioned in the example above, we use $K \cdot |P|$ ciphertexts to encode every X_i. We want to point out that Step 5 can easily be modified to only return *one* maximal element instead of all of them, which would reduce the complexity to $O(nk\ell)$. Also, we could emit the output of the rank max_y, which would increase privacy. We chose this variant in order to be compatible to Definition 2 and to be comparable to Neugebauer et al. [6].

We now have a look at the sum of ranks composition scheme f_{SR}. The basic idea is to compute a multiset union but omit all the elements which are not in the intersection. This is achieved by adding the ranks using the Add algorithm in combination with negated Equal calls to filter out elements which are not shared by all parties. The basic idea is the same as before, only Step 2 differs. Here we compute the value of f_{SR} instead of f_{MR}. We use Add to compute the sum of ranks. Next, we check if any of the terms used in this sum was zero. If so, the result will be multiplied by zero thus eliminating this entry from the set of possible maximal elements. The algorithm is shown in detail on the next page.

Computational Complexity. In Step 2, we use Add to compute the sum of n ciphertexts, each having length K. The result has length $K + \lceil \log_2(n) \rceil$. Next, we compute Equal on inputs of length K and negate the result. This is done n times and the product of those n bits is then multiplied with every bit of y. In Step 4, we compute Max on inputs of length $K + \lceil \log_2(n) \rceil$. Summing up, we have

$$\underbrace{k \cdot (T^{n\mathsf{Add}}(K) + n \cdot (T^{\mathsf{Equal}}(K) + T^{\mathsf{Not}}(1)))}_{\text{Step 2}}$$
$$+ \underbrace{k \cdot T^{\mathsf{Max}}(K + \lceil \log_2(n) \rceil)}_{\text{Step 4}} + \underbrace{k \cdot \ell \cdot T^{\mathsf{Equal}}(K + \lceil \log_2(n) \rceil)}_{\text{Step 5}}$$
$$\in k \cdot (O(n(\log(n) + K)) + nO(K) + \ell \cdot O(K + \log(n)))$$
$$\subseteq O(n \log(n) k \log(k) \ell)$$

homomorphic operations.

Like before, we can reduce the complexity to $O(n \log(n) k \ell)$ by only computing one maximal element instead of all. Also, we do not have to return max_y.

Reconciliation Algorithm for f_{SR}

Input: Encrypted $X_1, ..., X_n$ as described before and one $P_{\mathcal{A}_i}$ in plaintext.

1. Set $S := \emptyset$ and $R := \emptyset$.
2. For every $x \in P_{\mathcal{A}_i}$, compute
$$y \leftarrow \mathsf{Add}(X_1(x), ..., X_n(x)) \in C^{K+\lceil \log_2(n) \rceil}$$
$$y' \leftarrow y \boxtimes \bigboxtimes_{i=1}^{n} \mathsf{Not}(\mathsf{Equal}(X_i(x), \mathsf{E}(0)))$$
and add (x, y') to the set S.
3. Set $max_y \leftarrow \mathsf{E}(0) \in C^{K+\lceil \log_2(n) \rceil}$.
4. For each $(x, y) \in S$, compute $max_y \leftarrow \mathsf{Max}(y, max_y)$.
5. For each $(x, y) \in S$, compute $x' \leftarrow \mathsf{E}(x) \boxtimes \mathsf{Equal}(y, max_y)$ and add it to the set R.
6. Return (max_y, R).

Output: Encrypted maximal rank max_y and a set R of encrypted elements which either have maximal rank max_y or decrypt to 0^ℓ. If there exists no maximal rank (because the inputs are disjoint), max_y will decrypt to 0 and all elements in R will decrypt to 0^ℓ.

3.2 Reducing the Encoding Size

Our algorithm has runtime (almost) linear in the number of parties n but uses $\lceil \log_2(k+1) \rceil \cdot |P|$ ciphertexts to encode a k-element input where P is the input domain with $|P| < 2^\ell$. In certain situations, very large input domains P might be necessary and such a large number of ciphertexts may not be acceptable due to the communication overhead or bandwidth limitations.

The factor $|P|$ originates from the idea of using complete truth tables for representing the rank functions $\chi_i : \{0,1\}^\ell \to \{0,1\}^K$. We now give a smaller representation to allow our algorithm to use smaller encodings and therefore fewer ciphertexts. The price for this is a slightly higher computational complexity in terms of homomorphic operations. Let

$$X_i := \{(\underbrace{\mathrm{rank}_{\mathcal{A}_i}(x)}_{\in\{1,\ldots,k\}}, \underbrace{\mathsf{E}(x)}_{\in C^\ell}) \mid x \in P_{\mathcal{A}_i}\},$$

so for each rule, we save an encryption of the rule itself together with its (unencrypted) rank.

We will be able to reuse both variants of our algorithm (f_{SR} and f_{MR}), the only thing we change is the way $X_i(x)$, the encrypted rank of the element x in the input of party $\mathcal{A}_i$, is computed. Before, this was just a lookup from a table we have received from party $\mathcal{A}_i$, therefore, this can be done in constant time and without using any homomorphic operations. We can get that same value by computing

$$X_i(x) \leftarrow \boxplus_{(r,c)\in X_i} \left(\underbrace{\mathsf{E}(r)}_{\in C^K} \boxtimes \underbrace{\mathsf{Equal}(c, \mathsf{E}(x))}_{\in C^1} \right).$$

This basically just compares x to all inputs in X_i. If one of the inputs in X_i encrypts x, the rank of x is returned. This is correct since exactly one of the terms in this sum will encrypt a non-zero value.

Computational Complexity. Computing $X_i(x)$ this way takes $|X_i| = k$ calls to Equal on inputs of length ℓ. The result will be multiplied by $\mathsf{E}(r)$ which has length $K := \lceil \log_2(k+1) \rceil$. We then use $k-1$ calls to $\boxplus$ (in each component) for adding the results. In total, we get:

$$\begin{aligned} T_{\boxplus}(\ell) &= k \cdot T_{\boxplus}^{\mathsf{Equal}}(\ell) + (k-1) \cdot K \in O(k\ell) \\ T_{\boxtimes}(\ell) &= k \cdot (T_{\boxtimes}^{\mathsf{Equal}}(\ell) + K) \in O(k\ell) \\ T_{\mathsf{E}}(\ell) &= k \cdot (K + \ell) \in O(k\ell) \end{aligned}$$

Having a look at Step 2 again, we see that we need to compute $X_i(x)$ for every $x \in P_{\mathcal{A}_i}$ and every $1 \le i \le n$. In total, we use nk computations of the above kind, resulting in $O(nk^2\ell)$ additional homomorphic operations for Algorithm II which will lead to a complexity of $O(nk^2\ell)$ for f_{MR} and $O(n \log(n) k^2 \ell)$ for f_{SR}.

Summarizing, we can change the number of ciphertexts we need for encoding an input from $|P| \cdot \lceil \log_2(k+1) \rceil$ to $\log_2(|P|) \cdot k$ by increasing the asymptotic number of homomorphic operations by a factor of k. Which method is preferable will depend on the application parameters. For large values of k but small values of ℓ, we might still be better off with the original "truth table approach".

3.3 Putting It All Together: FHE Reconciliation Protocols

Recall that our algorithms take encrypted encodings of the parties' sets as inputs and return an encryption of an element which is maximal with respect to some

preference order composition scheme. Up until now, we did not clarify how to use those algorithms in a secure and privacy-preserving way in order to solve the reconciliation problem. Especially, how are ciphertexts exchanged and who holds the secret decryption key?

We assume a fully homomorphic encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D}, \boxplus, \boxtimes)$ with plaintext space $M = \mathbb{F}_2$ which is semantically secure. One way to implement a protocol is to assume that we have a *threshold fully homomorphic encryption scheme* such that the key generation algorithm G and the decryption algorithm D have to be jointly performed by all parties together and that no single party or collaboration of less than n parties can decrypt any ciphertext on its own but everyone can compute E, $\boxplus$ and $\boxtimes$. Recently, the authors of [22] published such a threshold version of Gentry's fully homomorphic encryption scheme.

Multi-party Reconciliation Protocol using Threshold FHE

1. All parties agree on the preference order composition scheme they want to use, on the number of inputs k in each party's set, and on an ℓ-bit binary encoding of the possible input elements such that 0^ℓ is not a valid encoding.
2. All parties $\mathcal{A}_1, ..., \mathcal{A}_n$ jointly generate a key pair $(sk, pk) \leftarrow \mathsf{G}(\lambda)$ such that everybody knows pk and only a share of sk.
3. Every party A_i generates an encrypted encoding X_i of its input and sends it to the other $n-1$ parties.
4. Every party runs the selected algorithm.
5. All parties participate in a threshold decryption of all outputs.

If a semantically secure homomorphic encryption scheme is used and no party can decrypt any ciphertexts on its own, no information about the private inputs will leak except what can be deduced from the maximal elements and their rank.

As an alternative, we present a protocol which is based on a non-threshold fully homomorphic cryptosystem. However, we need the help of an additional instance which is not one of the parties participating in the actual protocol. This additional instance is used for key generation and certain decryptions and is therefore called the *keyholder* $\mathcal{K}$. We require that all parties trust $\mathcal{K}$ not to collude with any of the other parties. If $\mathcal{K}$ plays by our rules, nobody will learn any private inputs, not even $\mathcal{K}$ itself. However, $\mathcal{K}$ is not to be seen as a trusted third party in the traditional sense. We do not have to trust $\mathcal{K}$ with the entire computation or with our secret inputs. The protocol is shown on the next page.

Like before, Step 3 will be performed by every party on its own and requires no interaction at all. Abusing $\mathcal{K}$ as a decryption oracle is not possible because $\mathcal{K}$ will wait until it receives ciphertexts from all parties and only send the decrypted results back if they are all equal. Even if $\mathcal{K}$ is compromised, not all private inputs will necessarily leak. However, if $\mathcal{K}$ colludes with some other party, this might be the case. Assuming the semi-honest model, this will not happen.

Multi-party Reconciliation Protocol using Keyholder $\mathcal{K}$

1. Keyholder $\mathcal{K}$ generates a key pair $(sk, pk) \leftarrow G(\lambda)$ and publishes the public key pk.
2. As before, each party generates an encrypted encoding of its k input elements, and sends it to the $n-1$ other parties.
3. Each party runs the selected algorithm and sends the result to $\mathcal{K}$.
4. $\mathcal{K}$ uses the secret key sk to decrypt the n results he received from the parties. If all results encrypt the same value, this (decrypted) value will be sent back to all parties.

A possible drawback of the above protocol might be that $\mathcal{K}$ learns the result of the algorithm. If this is a problem, the parties can use a blinding technique first. The results R of the algorithms have length $k\ell$. The parties agree on an $k\ell$-bit one-time pad $x \in \{0,1\}^{k\ell}$ to blind the results by computing $c' \leftarrow \mathsf{E}(x) \boxplus c$. If all parties use the same x, all c' will encrypt the same value as required in order for $\mathcal{K}$ to send back the results in Step 4. Without knowing the value of x used by the parties, the keyholder will not learn anything at all. The parties can just exchange x over some arbitrary confidential channel. After the decrypted result $\mathsf{D}(c')$ is received, all parties can obtain $\mathsf{D}(c) \in \{0,1\}^{k\ell}$ by computing $\mathsf{D}(c') + x$.

4 Comparing Results

We now compare our results using fully homomorphic encryption (FHE) with the results by Neugebauer et al. [6]. Table 1 summarizes the complexity results of our algorithms. Algorithm II denotes Algorithm I with reduced encoding size as described in Sect. 3.2. Recall that n is the number of parties, k is the number of inputs in each party's set, ℓ is the bit-length of the input elements (i.e., the logarithm of the size of the input domain).

Deriving the complexity for our *reconciliation protocol with keyholder* from Sect. 3.3 is now straightforward. We are counting the *total* number of operations, i.e., the sum of the number of operations each single party has to perform. We do not count key generation and distribution from Step 1 and we assume the parties already agreed on a fully homomorphic encryption scheme, its security parameter λ and on the one-time pad used for blinding in Step 4.

Table 1. Number of homomorphic operations

Algorithm	f	Homomorphic operations	Ciphertexts
I	f_{MR}	$O(nk\log(k)\ell)$	$\lceil\log_2(k+1)\rceil \cdot \lvert P\rvert \in O(\log(k)2^\ell)$
II	f_{MR}	$O(nk^2\ell)$	$k \cdot \log_2(\lvert P\rvert) \in O(k\ell)$
I	f_{SR}	$O(n\log(n)k\log(k)\ell)$	
II	f_{SR}	$O(n\log(n)k^2\ell)$	

Beginning with Step 2, each party has to encrypt its own input. Each input consists of k input elements and each input element has bit-length ℓ. As Table 1 shows, with Algorithm I each party has to encrypt at most $\lceil \log_2(k+1) \rceil \cdot |P|$ bits and send those bits to all other parties. With Algorithm II, each party encrypts $k \cdot \ell$ bits. In Step 3, every party runs the selected algorithm on the inputs it received from the other $n-1$ parties. This step requires no communication or further encryption operations.

Finally, in Step 4, each party blinds its result with the agreed one-time pad $x \in \{0,1\}^{k\ell}$. This requires a total of $k\ell$ encryptions and homomorphic additions for each party. Every party sends the blinded result $c \in C^{k\ell}$ to the keyholder $\mathcal{K}$ for decryption. This requires $nk\ell$ calls to D. After verifying that all results encrypt the same value, $\mathcal{K}$ sends the decrypted blinded result back to all n parties.

Table 2. Complexities of reconciliation protocols

Protocol	f	$\boxplus, \boxtimes, \odot$	E	D	#Messages
Neugebauer et al. [6]	f_{MR}	$O(k^6 + nk^4)$	$O(nk^2)$	$O(n^2k^3)$	$O(n^2k^3)$
FHE with Algo. I	f_{MR}	$O(n^2k\log(k)\ell)$	$O(n\log(k)2^\ell)$	$nk\ell$	$O(n^2\log(k)2^\ell)$
FHE with Algo. II	f_{MR}	$O(n^2k^2\ell)$	$O(nk\ell)$	$nk\ell$	$O(n^2k\ell)$
Neugebauer et al. [6]	f_{SR}	$O(n^4k^6)$	$O(nk^2)$	$O(n^4k^3)$	$O(n^4k^3)$
FHE with Algo. I	f_{SR}	$O(n^2\log(n)k\log(k)\ell)$	$O(n\log(k)2^\ell)$	$nk\ell$	$O(n^2\log(k)2^\ell)$
FHE with Algo. II	f_{SR}	$O(n^2\log(n)k^2\ell)$	$O(nk\ell)$	$nk\ell$	$O(n^2k\ell)$

Table 2 shows the number of operations which have to be performed in the different protocols as well as the number of messages which are exchanged. The analysis shows total numbers for all parties combined rather than for each single party. Recall that the new parameter ℓ in our results stands for the length of the input encodings (so 2^ℓ is the size of the input domain). In [6], the size of the input domain is tightly coupled to the security parameter of the cryptosystem whereas with our solution, ℓ is chosen by the user and the protocol will run faster if only small input domains are required (which we believe to be the case in most practical applications).

Note that interpretation of those numbers themselves is complicated without mentioning the specific cryptosystems which are used. The number of messages are counted as number of ciphertexts. This is not the same as the actual amount of data which has to be transmitted, which relies on the bit-length of the ciphertexts and will depend on the security parameter λ. Also, the number of calls to $\boxplus, \boxtimes$, or $\odot$ does not reflect the actual computational complexity because we cannot precisely state how expensive each of those calls is. We could try to be more precise by comparing an instantiation of our scheme using the Gentry scheme [23] with an instantiation of [6] using the Paillier scheme [24]. However, since Gentry's scheme is still under active research and little is known about it's practical efficiency, we do believe that such a comparison would not yield reliable insights.

However, we are confident in saying that given a fully homomorphic encryption scheme with comparable efficiency to current homomorphic encryption schemes, our protocols will outperform [6] by several orders of magnitude. Further details on our protocols and results can be found in [21].

5 Conclusion

In this paper, we developed a privacy-preserving multi-party reconciliation protocol which utilizes fully homomorphic encryption. We showed how to use ideas from circuit theory in order to compose algorithms which operate on encrypted data by utilizing small tool algorithms. Our protocol consists of an initial setup phase in which parties exchange encrypted data, followed by an offline computation phase, and a final phase for aggregating the result of the protocol.

We compare our approach to Neugebauer et al. [6] and we observe that our protocol has several advantages. As already mentioned, our computation is performed mainly offline and we only need a small and constant number of protocol rounds. Furthermore, fewer messages have to be exchanged and we require considerably fewer homomorphic operations, encryptions and decryptions. Although the exact computational complexity (for a fixed security level) cannot be made precise, we argued that our approach is likely to outperform Neugebauer et al. [6] in practice, assuming our protocol is instantiated with a sufficiently practicable fully homomorphic encryption scheme. In terms of privacy, our protocol allows for a stricter definition than Neugebauer et al. [6], namely, we can easily adopt our algorithms to output only one randomly chosen maximal element (instead of all of them) and the maximal rank does not have to be part of the output.

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

References

1. Yao, A.C.: Protocols for secure computations. In: Proceedings of the 23rd SFCS, pp. 160–164. IEEE Computer Society, Washington, DC (1982)
2. Meyer, U., Wetzel, S., Ioannidis, S.: Distributed privacy-preserving policy reconciliation. In: ICC, pp. 1342–1349 (2007)
3. Meyer, U., Wetzel, S., Ioannidis, S.: New Advances on Privacy-Preserving Policy Reconciliation. In: IACR eprint 2010/64, `http://eprint.iacr.org/2010/064`
4. Mayer, D.A., Teubert, D., Wetzel, S., Meyer, U.: Implementation and Performance Evaluation of Privacy-Preserving Fair Reconciliation Protocols on Ordered Sets. In: First ACM CODASPY (2011)
5. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
6. Neugebauer, G., Meyer, U., Wetzel, S.: Fair and Privacy-Preserving Multi-party Protocols for Reconciling Ordered Input Sets. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 136–151. Springer, Heidelberg (2011)

7. Neugebauer, G., Meyer, U., Wetzel, S.: Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets, Extended Version (2011), http://eprint.iacr.org/2011/200
8. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
9. Cheon, J.H., Jarecki, S., Seo, J.H.: Multi-party privacy-preserving set intersection with quasi-linear complexity. Cryptology ePrint Archive, Report 2010/512 (2010)
10. Li, R., Wu, C.: An unconditionally secure protocol for multi-party set intersection. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 226–236. Springer, Heidelberg (2007)
11. Sathya Narayanan, G., Aishwarya, T., Agrawal, A., Patra, A., Choudhary, A., Pandu Rangan, C.: Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 21–40. Springer, Heidelberg (2009)
12. Patra, A., Choudhary, A., Rangan, C.P.: Selected areas in cryptography, pp. 71–91. Springer, Heidelberg (2009)
13. Frikken, K.: Privacy-Preserving Set Union. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 237–252. Springer, Heidelberg (2007)
14. Hong, J., Kim, J.W., Kim, J., Park, K., Cheon, J.H.: Constant-round privacy preserving multiset union. IACR Cryptology ePrint Archive, 138 (2011)
15. Mayer, D., Neugebauer, G., Meyer, U., Wetzel, S.: Enabling fair and privacy-preserving applications using reconciliation protocols on ordered sets. In: 34th IEEE Sarnoff Symposium. IEEE, Princeton (2011)
16. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: Proceedings of the 41st STOC, pp. 169–178. ACM, New York (2009)
17. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
18. Goldreich, O., Micali, S.M., Wigderson, A.: How to play ANY mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987, pp. 218–229. ACM, New York (1987)
19. Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the 14th STOC, pp. 365–377. ACM Press, New York (1982)
20. Wegener, I.: The complexity of Boolean functions. John Wiley & Sons, Inc., New York (1987)
21. Weingarten, F.: Evaluating the Use of Fully Homomorphic Encryption in Secure Multi-Party Computation. Diploma Thesis, Research Group IT-Security, RWTH Aachen University (2011)
22. Myers, S., Sergi, M., Shelat, A.: Threshold fully homomorphic encryption and secure computation, vol. 2011 (2011)
23. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University, Stanford, CA, USA, AAI3382729 (2009)
24. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)