Privacy-Preserving Multi-Party Reconciliation Secure in the Malicious Model (Extended version)

Georg Neugebauer[1], Lucas Brutschy[1], Ulrike Meyer[1], and Susanne Wetzel[2]

[1] Department of Computer Science, RWTH Aachen University
[2] Department of Computer Science, Stevens Institute of Technology

Abstract. The problem of fair and privacy-preserving ordered set reconciliation arises in a variety of applications like auctions, e-voting, and appointment reconciliation. While several multi-party protocols have been proposed that solve this problem in the semi-honest model, there are no multi-party protocols that are secure in the malicious model so far. In this paper, we close this gap. Our newly proposed protocols are shown to be secure in the malicious model based on a variety of novel non-interactive zero-knowledge-proofs. We describe the implementation of our protocols and evaluate their performance in comparison to protocols solving the problem in the semi-honest case.

Keywords privacy-enhancing technologies, secure multi-party computation, cryptographic protocols, zero-knowledge proofs, malicious model

1 Introduction

In many applications, multiple parties need to jointly compute a function of their individual inputs, while keeping the inputs to the function private from each other. *Secure multi-party computation* solves this problem without the use of a trusted third party (TTP). Seminal work in this area [2,34] shows that any functionality that can be modeled as a Boolean or arithmetic circuit can be computed in private. As these early generic solutions exhibit a high complexity for the computation of some functionality, a second line of research focuses on developing protocols that can compute only specific functionality but in a very efficient way. Today, both approaches coexist and results arguing in favor developing special purpose protocols [9,13] as well as such arguing in favor of generic approaches have been published recently [12,20].

One such specific functionality is the *reconciliation of ordered sets* first introduced in [26,27] for the two-party case and generalized to multiple parties in [29,30]. As shown in [25], reconciliation of appointments, some types of electronic auctions, and certain e-voting schemes can be reduced to solving this problem. For example, when reconciling an appointment between multiple parties, each party can be considered to have a private input set of possible dates and order these dates according to its individual preferences. A privacy-preserving reconciliation protocol uses these ordered input sets to determine a date which is a *fair*

choice given the preferences expressed by each party, *without* revealing anything else about the ordered inputs to the other parties or even to a TTP.

The protocols proposed in [26,27,29,30] are shown to be secure only in the semi-honest model, that is, under the assumption that all parties follow the prescribed actions of the protocol while trying to extract as much information as possible from their view of the protocol run. In [24] the problem of malicious attackers, i.e. attackers that may deviate from the protocol, is solved for the two-party case. To the best of our knowledge, however, there are currently no multi-party ordered set reconciliation protocols which are secure in the malicious model. This is mainly due to the fact that a straightforward generalization of the two-party protocols is already highly inefficient for the semi-honest case [30]. The semi-honest multi-party protocols introduced in [30] therefore substantially differ in their design from the two-party protocols suggested in [26].

In this paper we propose multi-party protocols solving the reconciliation on ordered sets problem that are provably secure in the malicious model. Our constructions are based on the semantically secure, additively homomorphic Paillier cryptosystem [33] and a series of non-interactive zero-knowledge proofs to provide verifiable set operations. Here, we propose a new homomorphic linear equations proof which enables more efficient verifiable set operations than the ones previously proposed in [22]. In addition, we describe our implementation of the newly proposed protocols and evaluate their performance in comparison to the most efficient currently known semi-honest protocols proposed in [29].

2 Preliminaries

Adversary Models. In secure multi-party computation two adversary models are commonly used: the semi-honest and the malicious model. Both models assume the existence of pairwise encrypted and authenticated channels between the participating parties, such that external attackers do not have to be considered. A semi-honest adversary is an insider attacker that tries to infer as much (secret) information as possible from its view of a protocol run, but strictly follows the prescribed actions of the protocol.

A malicious adversary is an insider attacker that can almost arbitrarily deviate from the protocol. The only actions it cannot take, are the ones that cannot be prevented, namely: refusal to participate in the protocol, manipulation of own input, and protocol abortion. To prove that a protocol is privacy-preserving in the malicious model the simulation paradigm is used, which compares the real-world execution of the protocol to an ideal-world execution. In the ideal world, all parties provide their inputs to a trusted third party, which computes the correct output and provides the results to each party. In the real world, in addition to the protocol's correct output, the adversary learns all messages exchanged between the parties and all randomness generated during the execution of the protocol. One shows that it is possible to construct a simulator which, given the ideal output, can generate a transcript that is identical to that of the real protocol execution. If such a transcript can be generated using only the knowledge

of the ideal execution, the protocol is privacy-preserving in the malicious model. For a formal definition see [6,17].

Definition 1 (Ordered sets and ranking functions [29]). *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. As a shorthand, 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 $rank_S(x_i) = k - i + 1$.*

Definition 2 (Composition schemes [29]). *Let $(S_1, <_1), ..., (S_n, <_n)$ be ordered sets. The ordered set $\left(S_1 \cap ... \cap S_n, \leq_{\{S_1,..,S_n\}}\right)$ is the* combined ordered set of $(S_1, <_1), ..., (S_n, <_n)$ w.r.t. *f if $\leq_{\{S_1,..,S_n\}}$ is the order induced by the function $f : (S_1 \cap ... \cap S_n) \to \mathbb{R}$. The function f is called the* composition scheme.

Definition 3 (Privacy-preserving reconciliation on ordered sets [29]). *A multi-party reconciliation protocol on ordered sets for an order composition scheme f ($MPROS^f$) is a multi-party protocol between n parties $P_1, ..., P_n$ each with an input $(S_i, <_i)$ of size k drawn from the same domain D. Upon completion of the protocol, each party learns (X, t) with:*

$$X = \underset{x \in (S_1 \cap ... \cap S_n)}{arg\ max}\ f(x) \qquad t = \underset{x \in (S_1 \cap ... \cap S_n)}{max} f(x)$$

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

A reconciliation protocol on ordered sets is said to be privacy-preserving in the semi-honest (the malicious) model iff no semi-honest (malicious) party learns anything about the inputs and preferences of the other parties, except what can be deduced from the output (X, t) of the protocol and his own private input set.

Note that the output of a reconciliation protocol includes both the set of common elements with maximum rank in the combined preference order and the corresponding maximum order under the composition scheme f. In this paper, we consider two composition schemes proposed in [26] that take the preferences of each party equally into account.

Definition 4 (MR). *The* minimum of ranks composition scheme (MR) *is defined by the function $f(x) = min\{rank_{S_1}(x), ..., rank_{S_n}(x)\}$.*

Definition 5 (SR). *The* sum of ranks composition scheme (SR) *is defined by the function $f(x) = rank_{S_1}(x) + ... + rank_{S_n}(x)$.*

The Paillier Cryptosystem. In this paper, we make use of the probabilistic public-key cryptosystem proposed by Paillier [33]. The cryptosystem is additively homomorphic and semantically secure under computational assumptions. Let $E_{pk}(\cdot)$ be the encryption function with public key pk. In an additively homomorphic cryptosystem, there is an operation $+_h$ such that $E_{pk}(a + b) = E_{pk}(a) +_h E_{pk}(b)$ and $+_h$ can be computed efficiently given only $E_{pk}(a), E_{pk}(b)$, and pk. We denote the homomorphic summation by $\tilde{\sum}$. Furthermore, there are

several threshold variants [11,15] of the Paillier cryptosystem which make it a very popular choice in secure multi-party computation protocols. Formally, a (t, n)-threshold cryptosystem is a cryptosystem with a public key pk and n *private key shares* $sk_1, ..., sk_n$. Using a private key share, a party can compute a *partial decryption* of a ciphertext. To successfully recover the plaintext of a given ciphertext, t of the n key shares are required to compute and combine t decryption shares to obtain the plaintext.

Encryption. $E_{pk}(m) \stackrel{def}{=} g^m r^N \bmod N^2$ for any plaintext $m \in \mathbb{Z}_N^*$, where $r \in \mathbb{Z}_N^*$ is selected uniformly at random.

Homomorphic addition. For two ciphertexts $\alpha, \beta \in \mathbb{Z}_{N^2}$, the homomorphic operation $+_h$ is given by $\alpha +_h \beta \stackrel{def}{=} \alpha \cdot \beta \bmod N^2$, where $\alpha, \beta \in \mathbb{Z}_{N^2}$.

Homomorphic scalar multiplication. For a ciphertext $\alpha \in \mathbb{Z}_{N^2}$ and a scalar $s \in \mathbb{Z}_N^*$, the scalar multiplication is defined by $s \times_h \alpha \stackrel{def}{=} \alpha^s \bmod N^2$.

Threshold decryption. To perform a threshold decryption, each party P_i computes the partial decryption $c_i = c^{2\Delta sk_i} \bmod N^2$ where $\Delta = n!$ and sk_i denotes the private key share of party P_i. Given t partial decryptions, one can recover the plaintext by computing the Lagrange interpolation on the exponents [15].

Rerandomization. To mask a given ciphertext α with a new random value, we multiply it with an additional random factor. Formally, the rerandomization operation $[\cdot]_R$ is defined as $[\alpha]_R \stackrel{def}{=} \alpha \cdot R^N \bmod N^2$.

Privacy-Preserving Multiset Operations. Our protocols make use of the privacy-preserving operations on multisets, i.e., sets in which elements may occur more than once, introduced by Kissner et al. [22]. In these operations a private input multiset $S_i = \{s_{i,1}, ..., s_{i,k}\}$ of party P_i is encoded by the polynomial $f_i(x) = \prod_{j=1}^{k} (x - s_{i,j})$. The coefficients of this polynomial are encrypted with an additively homomorphic cryptosystem. There exists a recent alternative line of work for multiset operations based on secret sharing [4,10].

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 with the corresponding summed up 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 [22].

Intersection. For two encrypted polynomials ϕ, γ representing two sets F and G of equal size, the intersection can be computed by the 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 $F \cap G$ (Theorem 3 in [22]).

Set Reduction. For an encrypted polynomial γ representing a multiset G one can compute the element reduction $Rd_t(G)$ by the term $\tilde{\sum}_{i=0}^{t} \gamma^{(i)} \times_h F_i \times_h r_i$. Here, each r_i is chosen uniformly at random and independently from the set of polynomials of degree $deg(\gamma^{(i)})$ and each F_i is a fixed polynomial of degree i such that, 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 [22] for an in-depth discussion of correctness and security.

Privacy-Preserving Reconciliation. The multi-party protocols for privacy-preserving reconciliation on ordered sets proposed in [29] are based on the privacy-preserving set operations introduced above.

In particular, the protocol for the minimum of ranks composition scheme works as follows. Let $S_i = \{s_{i,1} > ... > s_{i,k}\}$ 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 $\bigcap_{i=1}^{n} \{s_{i,1}, \ldots, s_{i,l}\}$. 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 maximum minimum of ranks value $k - l + 1$. The protocol for the sum of ranks composition scheme computes

$$Rd_t\Big(renc(S_1) \cup ... \cup renc(S_n)\Big) \cap \big(S_1 \cap ... \cap S_n\big).$$

Here, $renc(S_i)$ denotes an encoding of the multiset S_i in which each element occurs with the multitude indicated by its rank. I.e., the highest ranked element s_{ik} occurs k times while the lowest ranked element s_1 occurs only once. The details of the protocols as well as the proof of correctness and security in the semi-honest model can be found in [29]. We detail how our newly proposed protocols differ from these protocols in Section 4.

Zero-Knowledge Proofs of Knowledge. Suppose a prover P has knowledge of an x such that $(y, x) \in R$ for some relation R and a public value y. He wants to convince a verifier V of this knowledge without revealing anything but this fact. A protocol that realizes this functionality is called a zero-knowledge proof of knowledge (ZKPK) protocol. Any ZKPK protocol must satisfy three properties: First, it must be *correct,* i.e., if the prover knows x, then the prover can convince the verifier that he knows x. Second, it must be *sound,* i.e., without knowledge of x, a prover can not convince the verifier. Third, it must satisfy the property of *zero-knowledgeness,* i.e., the verifier learns nothing but the fact that P knows an x such that $(y, x) \in R$. For a more formal definition of these properties, see [17].

A well known form of ZKPK protocols are Σ-Protocols, which are interactive two-party protocols in which the verifier generates a random challenge. These protocols can be generalized to multiple parties by executing them with each verifier separately, requiring the prover to react to the multiple challenges generated by the verifiers, which is somewhat inefficient. The so-called *Fiat-Shamir heuristic* [14] solves this problem by replacing the challenge generated by the verifier by the result of a hash function. The resulting protocols are known as

non-interactive ZKPK protocols. The security of Fiat-Shamir-based proof protocols is given in the random oracle model [1].

Proof of Plaintext Knowledge. In a proof of plaintext knowledge for the Paillier cryptosystem, a prover tries to prove to the verifier that he knows m, r such that $y = g^m \cdot r^N \bmod N^2$ for a known ciphertext y. Interactive variants of the plaintext knowledge proof for the Paillier cryptosystem were proposed in [7,8].

Proof of Correct Multiplication. Suppose two parties know the three ciphertexts α, β, γ. In the proof of correct multiplication, the prover shows that he knows the plaintext m of γ and that $m *_h \alpha = \beta$. Interactive variants of the correct multiplication proof for the Paillier cryptosystem were proposed in [7,8].

Proof of a Subset Relation Using Verifiable Shuffles. Consider a set of plaintext values D. In our setting, the prover selects a subset S of k distinct elements from D in an unknown order and sends encryptions in a verifiable manner to the other parties. More formally, we need a *verifiable shuffle protocol* [18,19,28,32]. In our setting, we use a protocol proposed by Nguyen et al. [32], since it can be applied directly in the Paillier cryptosystem, can be made *non-interactive* using the Fiat-Shamir heuristic and runs in linear time in the size of domain D.

Verifiable Threshold Decryption. In order to prove that a party correctly computed the partial decryption in a threshold decryption scheme for the Paillier cryptosystem, we use an adaption of techniques by Fouque et al. [15].

3 Novel ZK-Proofs

We provide novel non-interactive ZK-Proofs that allow us to specify new protocols for verifiable set union, intersection, and reduction operations. In particular, we convert the interactive proof for plaintext knowledge and correct multiplication [7,8] into non-interactive ZKPK. We present a new ZKPK for homomorphic linear equations and show how this can be used to construct ZKPK for verifiable set operations. We show that these new ZKPKs are more efficient than the ones previously proposed in [22].

Proof of Plaintext Knowledge. Algorithm 4 in the Appendix shows how to turn the interactive ZKPK proposed in [8] into a *non-interactive* proof using the *strong* variant of the Fiat-Shamir heuristic. I. e., we hash the commitment t *and* the statement y to be proven [3] and not only the commitment t. We also show correctness, special soundness, and special honest-verifier zero-knowledge for the given proof in the Appendix.

Proof of Correct Multiplication. Algorithm 5 in the Appendix shows how to turn the interactive ZKPK proposed in [8] into a *non-interactive* proof. Commitment t_2 and responses m'' and r_2'' are constructed in the same way as in Algorithm 4. The detailed proofs for correctness, special soundness, and special honest-verifier zero-knowledge are given in the Appendix.

Proof of a Homomorphic Linear Equation. Assume the following linear equation, where $\alpha_1, ..., \alpha_p, \beta$ are Paillier ciphertexts and $m_1 ... m_p$ are Paillier plaintexts $m_1 *_h \alpha_1 +_h ... +_h m_p *_h \alpha_p = \beta$. It is possible to construct a proof for the correctness of a homomorphic linear equation, where the scalar factors are only known to the verifier in encrypted form. For an equation with p linear factors we perform p plaintext knowledge proofs in parallel, together with an additional constraint that the given equation holds, similar to the construction proving correct multiplication.

Algorithm 1 lists the steps required to construct and verify the corresponding proof. α_i range over the ciphertext factors involved in the computation, β is the result of the equation, m_i are the secret scalar factors used in the multiplications, r_i the secret randomization factors used for encrypting those to γ_i, and pid_{P_i} is a random value assigned to each party. We use a single challenge, but $p+1$ commitments and $2p+1$ responses in our proof. The construction and reconstruction of the commitments follow the same principle as given in the plaintext knowledge and correct multiplication proof.

Correctness: We show that the commitments $t'_R, t'_i, i \in \{1, ..., p\}$ match the original commitments t'_R, t_i, and thus $h(pid_P, g, \alpha_1, ..., \alpha_p, \beta, t'_R, t'_1, ..., t'_p) = c$. The plaintext knowledge commitments are constructed in the same way as in Algorithm 4, and thus the correctness argument for those commitments also holds in this protocol. In addition, the equation commitment t'_R is equal to t_R :

$$\begin{aligned} t'_R &= \beta^c \left[\alpha_1 *_h m'_1 +_h ... +_h \alpha_p *_h m'_p\right]_{R''} \\ &= \left(\prod_{i=1}^{p} \alpha_i^{m_i} \cdot R^N\right)^c \cdot \prod_{i=1}^{p} \alpha_i^{m'_i - c m_i} \cdot \left(R' \cdot R^{-c}\right)^N = \left(\prod_{i=1}^{p} \alpha_i^{m'_i}\right) \cdot R'^N = t_R \end{aligned}$$

Thus, the reconstructed challenge is equal to the transmitted c, and the verification succeeds.

Soundness: If we can extract the private values $m_i, r_i, i \in \{1, ..., p\}$, and R from two protocol transcripts with the same commitment but different challenges, then it follows that m_i, r_i, and R must have been used to generate the two responses, and thus, the prover must know m_i, r_i, and R, compare [7].

We can construct an extractor in the following way: Given two transcripts $\left(c_1, m''_{11}, ..., m''_{1p}, r''_{11}, ..., r''_{1p}, R''_1\right)$ and $\left(c_2, m''_{21}, ..., m''_{2p}, r''_{21}, ..., r''_{2p}, R''_2\right)$ for the same commitments, we can extract the private values as follows. Let σ, s be defined by $\sigma\left(c_2 - c_1\right) = 1 + sN$. σ and s exist under the assumption that $c_2 - c_1$ is coprime to N (which is true if $c_2 - c_1$ is less than p and q). Extract m_i, r_i by

Algorithm 1 Construction and verification of a linear equation proof

Specification:

$$ZKPK\left\{ \begin{matrix} m_1, ..., m_p \\ r_1, ..., r_p, R \end{matrix} \middle| \begin{matrix} [\alpha_1 *_h m_1 +_h ... +_h \alpha_p *_h m_p]_R = \beta \\ \wedge \qquad \bigwedge_{i=1}^{p} \gamma_i = E(m_i, r_i) \end{matrix} \right\}$$

Construction:

(1) Select random $m'_i, r'_i \in \mathbb{Z}_N^*$ for $i \in \{1, .., p\}$ and $R' \in \mathbb{Z}_N^*$
(2) Compute equation commitment $t_R = \left[\alpha_1 *_h m'_1 +_h ... +_h \alpha_p *_h m'_p\right]_{R'}$
(3) Compute plaintext knowledge commitments $t_i = E(m'_i, r'_i)$
(4) Get a challenge $c = h(pid_P, g, \alpha_1, ..., \alpha_p, \gamma_1, ..., \gamma_p, \beta, t_R, t_1, ..., t_p)$
(5) For $i \in \{1, ..., p\}$, compute responses $m''_i = m'_i - cm_i \mod N$ and $r''_i = r'_i \cdot r_i^{-c} \mod N$
(6) Compute $R'' = R' \cdot R^{-c}$
(7) Send $(c, m''_1, ..., m''_p, r''_1, ..., r''_p, R'')$

Verification:

(1) Reconstruct equation commitment $t'_R = \beta^c \left[\alpha_1 *_h m''_1 +_h ... +_h \alpha_p *_h m''_p\right]_{R''}$
(2) For $i \in \{1, ..., p\}$, reconstruct plaintext commitments $t'_i = g^{m''_i} r''^N_i \gamma_i^c$
(3) Verify $h(pid_P, g, \alpha_1, ..., \alpha_p, \beta, t'_R, t'_1, ..., t'_p) = c$

---

computing $m_i = (m''_{1i} - m''_{2i})\sigma$ and $r_i = \left(\frac{r''_{1i}}{r''_{2i}}\right)^\sigma y^{-s} \mod N^2$. We can see that m_i, r_i are extracted correctly by following the proof idea of [7,8]. Since both transcripts are valid, and the commitments $t_i, t_R, i \in \{1, ..., p\}$ in both cases are the same, we also have equal reconstructed commitments t'_i: $g^{m''_{1i}} r''^N_{1i} y^{c_1} = g^{m''_{2i}} r''^N_{2i} y^{c_2} \pmod{N^2}$. By transformation and exponentiation by σ, we obtain

$$g^{(m''_{1i} - m''_{2i})\sigma} \left(\frac{r''_{1i}}{r''_{2i}}\right)^{\sigma N} = y^{\sigma(c_2 - c_1)} \pmod{N^2}.$$

The right side is by definition of σ simply y^{1+sN}. Using the form of y we get

$$g^{(m''_{1i} - m''_{2i})\sigma} \left(\left(\frac{r''_{1i}}{r''_{2i}}\right)^\sigma \cdot y^{-s}\right)^N = g^m r^N \pmod{N^2}.$$

Thus, we have $m_i = (m''_{1i} - m''_{2i})\,\sigma$ and $r_i = \left(\frac{r''_{1i}}{r''_{2i}}\right)^\sigma \cdot y^{-s} \mod N^2$.

Similarly, we extract R from two responses R''_1, R''_2 with the same randomness and different challenges c_1, c_2 by finding σ, s such that $\sigma\,(c_2 - c_1) = 1 + sN$, and computing

$$R = \left(\frac{R''_1}{R''_2}\right)^\sigma \cdot \beta^{-s} \mod N^2$$

Protocol	Challenges	Commitments	Responses
Several multiplication proofs *Kissner et al. [22]*	$(k+1)^2$	$2k^2+4k+2$	$3k^2+6k+3$
Several linear equation proofs *Our new approach*	$2k+1$	k^2+4k+1	$2k^2+6k+3$

Table 1. Number of challenges, commitments, and responses for a verifiable polynomial multiplication where each input set consists of k elements

Honest-Verifier Zero-Knowledge: Here, the transcript of this proof protocol is simply the proof $\left(c, m_1'', ..., m_p'', r_1'', ..., r_p'', R''\right)$. A simulator can generate an accepting transcript in the following way. Again, select all values randomly $(c, m_1'', ..., m_p'', r_1'', ..., r_p'', R'')$, and compute the corresponding commitments $t_i' = g^{m_i''} r_i''^N \gamma_i^c$ and $t_R' = \beta^c \left[\alpha_1 *_h m_1' +_h ... +_h \alpha_p *_h m_p'\right]_{R''}$. Then, let the random oracle return c for the input $pid_P, g, \alpha_1, ..., \alpha_p, \gamma_1, ..., \gamma_p, \beta, t_R, t_1, ..., t_p$.

Complexity: The proof consists of a hash, p random Paillier plaintexts (with random components) and an additional random value of size b. For p linear factors, we need to perform p binary exponentiations with a Paillier modulus N of size b, thus we have the computation complexities $O\left(p \cdot b^3\right)$.

Next, we show how to construct proofs for computations on polynomials. Basically, the following proofs are based on parallel execution of several linear homomorphic equation proofs, but using a common challenge for all protocols. This is commonly referred to as And-Composition of proofs [5].

Proof of Correct Polynomial Operations. We start with the construction of proofs for the correct multiplication of polynomials. For a polynomial f let $E(f)$ denote the coefficient-wise encryption of f and let the corresponding Greek letter ϕ denote the tuple of the encrypted coefficients. Assume we want to prove that $\psi = f *_h \gamma$ for some f and that $\phi = E(f)$. To construct a corresponding proof, we consider the homomorphic polynomial multiplication using the standard long multiplication of polynomials with homomorphic operations. In this expanded form, we get a set of $deg(\psi) + 1 = deg(f) + deg(\gamma) + 1$ homomorphic linear constraints. We can denote this proof in general as listed below.

$$ZKPK\left\{ \left(f_0, ..., f_{deg(f)}\right) \;\middle|\; \begin{array}{c} \bigwedge_{i=0}^{deg(\psi)} \left(\tilde{\sum}_{j=0}^{i} \gamma_j *_h f_{i-j}\right) = \psi_i \\ \wedge \quad \bigwedge_{i=0}^{deg(\phi)} \phi_i = E\left(f_i\right) \end{array} \right\}$$

Here, a coefficient f_j of a polynomial f is considered to be zero $if\ j > deg(f)\ or\ j < 0$. This enables verifiable set union. The security and correctness of the proof directly follows from the correctness and security of the used sub-protocols for plaintext knowledge, correct multiplication and homomorphic linear equations. This approach can be extended to arbitrary linear expressions of polynomials:

$$\phi_1 *_h f_1 +_h ... +_h \phi_s *_h f_s = \psi$$

The construction is completely analogous to the construction above, only that we have s as many multiplications in each linear homomorphic constraint.

Verifiable Set Operations. We can construct verifiable set intersection, union, and reduction operations based on verifiable polynomial multiplication. Note that the efficiency of each of these operations depends on the efficiency of the verifiable polynomial multiplication. Table 1 compares our new verifiable polynomial multiplication to the approach previously proposed by Kissner et al. [22].

This previous approach is based on proving the correctness of all involved homomorphic multiplications. This requires the prover to sent all intermediate results to the verifier and provide one proof per homomorphic multiplication. Our generalization of the multiplication proof to arbitrary linear expressions enables more efficient ZKPK on polynomials.

4 MPROS Secure in the Malicious Model

In this section, we propose two new protocols for $MPROS^{MR}$ and $MPROS^{SR}$ and prove their security in the malicious model. Previous multi-party protocols [29,30,31] only provide security in the semi-honest model.

4.1 A Malicious Model Protocol for $MPROS^{MR}$

We first present a malicious model protocol for $MPROS^{MR}$. We use the following core techniques to inhibit malicious behavior: Encryptions of all chosen random polynomials, the secret input sets, and all intermediate computation results are broadcasted to all other parties together with zero-knowledge proofs proving the correctness of the computations involving those secret values.

Protocol Description: The formal protocol description is shown in Algorithm 2. Encrypted values are denoted by lower case Greek letters, e.g., $\delta_{i,j}$ denotes the encrypted value of $d_{i,j}$. The protocol starts with the distribution of the input sets. Each party computes a shuffle of the domain, such that the first k elements represent its input set and proves the correctness of the shuffle. When all encrypted shuffles have been distributed, the parties verify the proofs of all other parties. Whenever a proof verification fails, the protocol is aborted.

In Step 2 and 3, all parties compute the set intersection of the polynomials $\phi_{i,k-t}$, verify the corresponding proofs $\Pi_{\text{INTERSECT},i}$ and decrypt the result π. In Step 4, the result is tested for emptiness. Based on the outcome, one of two actions is performed: If the result is non-empty, we have found the correct result and terminate the protocol. If the result is empty, we repeat the set intersection with a decreased threshold value t. For this purpose, each party adds the next highest ranked element $d_{i,k-t}$ to its current polynomial $\phi_{i,k-t}$ using a simple set union operation, resulting in the polynomial $\phi_{i,k-t+1}$ for the next round. After the verification of the set union operation, the protocol returns to Step 2.

Algorithm 2 Malicious model protocol for $MPROS^{MR}$

Setting: Parties $P_1, ..., P_n$ with ordered input sets, chosen from common domain D, $S_i = \{d_{i,1} > ... > d_{i,k}\}$, $i \in \{1, ..., n\}$. Each party P_i holds a key share for a (n, n)-threshold decryption scheme.

1. Initial Polynomial
   (a) Each party P_i $(i = 1, ..., n)$
      i. Computes an encrypted shuffle $(\delta_{i,1}, ..., \delta_{i,k}, ...)$ of the domain D where the first k elements denote the input set elements.
      ii. Broadcasts the shuffle and correctness proof $\Pi_{SHUFFLE,i}$ (see Section 2)
   (b) Each party P_i $(i \in \{1, ..., n\})$ for $j \in \{1, .., n\}$
      i. If $j \neq i$, verifies $\Pi_{SHUFFLE,j}$
      ii. Chooses random polynomial $r_{i,j,1}$ of degree 1
      iii. Computes and commits to $\rho_{i,j,1} = E_1(r_{i,j,1})$
2. Set Intersection (Initially $t = k - 1$. Let $\phi_{i,1} = \left(E(1), \delta_{i,1}^{-1}\right)$)
   (a) Each party P_i $(i = 1, ..., n)$.
      i. Opens the commitment to $\rho_{i,j,k-t}$
      ii. Computes and broadcasts $\gamma_i = \left[\tilde{\sum}_{j=0}^{n} (\phi_{j,k-t} *_h r_{i,j,k-t})\right]_r$
      iii. Broadcasts a proof $\Pi_{\text{INTERSECT},i}$ that γ_i is correctly computed
   (b) Each party P_i $(i = 1, ..., n)$
      i. For $j \in \{1, .., n\} \setminus \{i\}$ verifies $\Pi_{\text{INTERSECT},j}$
      ii. Calculates $\pi = \sum_{i=1}^{n} \gamma_i$
3. Decryption : All parties perform a malicious model threshold decryption of π and obtain the result polynomial p.
4. Emptiness Test / Set Union
   (a) Each party P_i $(i = 1, ..., n)$
      i. Computes the set of elements of S_i which are roots of p: $R = \{d \in S_i : (X - d) | p.\}$
      ii. If $R \neq \emptyset$, terminates the protocol with result $(R, t + 1)$
      iii. If $R = \emptyset$ and $t = 0$, terminates the protocol with $(\emptyset, 0)$
      iv. Computes and broadcasts $\phi_{i,k-t+1} = [\phi_{i,k-t} *_h (x - d_{i,k-t})]_r$
      v. Broadcasts a proof $\Pi_{\text{UNION},i}$ that $\phi_{i,k-t+1}$ is correctly computed
   (b) Each party P_i $(i \in \{1, ..., n\})$ for $j \in \{1, .., n\}$
      i. If $j \neq i$, verifies $\Pi_{UNION,j}$
      ii. Chooses random polynomial $r_{i,j,k-t+1}$ of degree $k - t + 1$ and commit to $\rho_{i,j,k-t+1} = E_1(r_{i,j,k-t+1})$
   (c) Proceed with Step 2 using $t - 1$

$$\Pi_{SHUFFLE,i} = ZKPK\left\{ d_1, .., d_k \,\middle|\, \{d_1, .., d_k\} \subseteq D \wedge \bigwedge_{i=1}^{k} \delta_i = E\left(d_i\right) \right\}$$

$$\Pi_{INTERSECT,i} = ZKPK\left\{ r_{i,1}, ..., r_{i,n}, R \,\middle|\, \begin{array}{c} \gamma_i = \left[\tilde{\sum}_{j=0}^{n} \left(\phi_j *_h r_{i,j}\right)\right]_R \\ \wedge \quad \bigwedge_{l=0}^{n} \rho_{i,j} = E_1(r_{ij}) \end{array} \right\}$$

$$\Pi_{UNION,i,t} = ZKPK\left\{ d_{i,k-t}, \phi_{i,k-t}, R \,\middle|\, \begin{array}{c} \phi_{i,k-t+1} = \left[\phi_{i,k-t} *_h \left(x - d_{i,k-t}\right)\right]_R \\ \wedge \quad \delta_{i,k-t} = E\left(d_{i,k-t}\right) \end{array} \right\}$$

Table 2. The proofs used in Algorithm 2

Correctness: We compute the function $Rd_t(S_1 \cap ... \cap S_n)$ as the semi-honest variants discussed in [29,30,31]. Assuming that the zero-knowledge proofs of knowledge are difficult to forge, each party is forced to perform the same computations as in the semi-honest variant of the protocol. Therefore the correctness results from [29] also apply to our malicious model variant.

Security in the Malicious Model

Setting: The ZK proof protocols are based on Σ-Protocols that have been converted into non-interactive protocols using the Fiat-Shamir heuristic. Since the verifier does not interact in the proof generation, it is sufficient to show special honest-verifier zero-knowledge for these protocols, see [17] for more details. We show the security of the protocol against at most $n-1$ attackers. This is achieved with the help of a broadcast of the ZKPK proofs to all $n-1$ other parties.

ZK-Proofs: Our protocol uses several types of proofs, all of which are listed in Table 2. The proofs for proving the correct set intersection $\Pi_{INTERSECT,i}$ and the proof for correct set union $\Pi_{UNION,i,t}$ directly follow from the proofs outlined in Section 3. Note that each union proof $\Pi_{UNION,i,t}$ requires a successful proof verification $\Pi_{UNION,i,t+1}$ with the previously used threshold value $t+1$. Furthermore, we require each party to prove that its chosen subset is part of the domain using a verifiable shuffle and compute a verifiable decryption as described in Section 2.

Solving the 0-Polynomial Problem: Malicious attackers can manipulate the protocol by inserting 0-polynomials in the protocol, i.e., polynomials where all coefficients are set to zero. Other parties can not detect these polynomials, as they only receive encrypted versions and by the semantic security of the cryptosystem it is infeasible to check if it encrypts a zero or not. We solve the problem in the following manner: The first coefficient of polynomials that are chosen by a party is always assumed to be a known encryption of 1 (E_1), compare Step 1.b.iii. and 4.b.ii. in Algorithm 2. Since all such computations are reblinded before they are sent to the other parties, this does not reduce the security of the protocol.

Algorithm 3 Simulation Algorithm for $MPROS^{MR}$

1. For each simulated honest party $P_i \in \Phi$
   (a) Generate an ordered set of random values R_i of size k
   (b) Follow Step 1a) according to the protocol, using R_i as input
2. For each malicious party $P_i \in \Gamma$, extract from the received proof $\Pi_{SHUFFLE,i}$ the private ordered set S_i
3. Send the extracted ordered sets $\{S_i | P_i \in \Gamma\}$ to TTP
   Each honest party $P_i \in \Phi$ sends its set S_i to TTP
4. TTP computes and sends the following results to SIM and the honest players

$$A = \underset{x \in (S_1 \cap ... \cap S_n)}{\operatorname{argmax}} \left\{ \min_{1 \le i \le n} \operatorname{rank}_i(x) \right\}$$

$$m = \max_{x \in (S_1 \cap ... \cap S_n)} \left\{ \min_{1 \le i \le n} \operatorname{rank}_i(x) \right\}$$

5. For $t = k-1, ..., m$ follow the protocol (Steps 1b - 4) for each simulated honest party $P_i \in \Phi$ with input R_i and each malicious party $P_i \in \Gamma$ with input S_i
6. For $t = m-1$ and each simulated honest party $P_i \in \Phi$ and every $P_j \in (\Phi \cup \Gamma)$
   (a) Select random polynomial s of size $(k - (m-1) - |A|)$
   (b) Compute polynomial $p = \prod_{a \in A}(x - a) \cdot s$
   (c) Select the remaining polynomials $r_{i,j}$ (of the honest parties) such that $\sum_{i=1}^{n} f_{i,m-1}\left(\sum_{j=1}^{n} r_{i,j,m-1}\right) = p$. See [22], Lemma 2 for a proof that these $r_{i,j}$ exist. Commit to the random polynomials $r_{i,j,m-1}$.
7. Follow and complete the protocol for each party

Simulation Proof

Theorem 4.11 *Assuming that the additively homomorphic, threshold cryptosystem $E(\cdot)$ is semantically secure and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the $MPROS^{MR}$ protocol in Algorithm 2, for any coalition Γ of at most $n-1$ colluding players, there is a player (or group of players) SIM operating in the ideal model, such that the views of the players in the ideal model are computationally indistinguishable from the views of the honest players and Γ in the real model.*

We give the algorithm for a simulator *SIM* in the ideal world that represents one or more honest participants and executes the above protocol with the set of potentially malicious and colluding parties $P_1, ..., P_l$. In addition, the simulator performs the ideal world protocol with the trusted third party *TTP*. The simulator *SIM* acts as a translator between the real world protocol and the ideal world protocol and acts as the honest parties $P_{l+1}, ..., P_n$ in the protocol with the malicious parties. The intuition of the simulation proof is as follows:

If we can generate all protocol messages from only the interaction with the trusted third party, which, in the ideal world, does not leak any information

about the private inputs of parties $P_{l+1}, ..., P_n$, then the exchanged protocol messages can not contain more information, than the information provided in the ideal world, i. e., the output of an MPROS protocol, compare Definition 3. We give the simulator the power to extract values from zero-knowledge proofs which is a common approach in malicious model security proofs [6,17].

The algorithm for the simulator is given in Algorithm 3. The simulator starts by constructing *random* inputs to the real-world protocol, i.e., selecting a random ordered set of size k and constructing and sending the corresponding encryptions and proofs (Step 1). The random inputs are used in place of the real inputs of the honest parties, which ensures that no information is leaked in the first step.

After receiving the encryptions and the proofs from the malicious parties, the simulator then uses the extractors given in the soundness proofs of the zero-knowledge proof protocols to extract the private sets from the provided proofs (Step 2). This makes it possible to perform the ideal-world protocol with the TTP and the honest parties (Steps 3 and 4). After receiving the result of the protocol from the TTP, the simulator proceeds with the protocol execution in the real model until threshold value $t = m - 1$ (Step 5). Then, it inserts a polynomial representation of the result into the real-world protocol execution by choosing the random polynomials $r_{i,j,m-1}$ accordingly (Step 6).

Under the assumption of a semantically secure threshold cryptosystem, the views of the players in the ideal model given by the simulator are computationally indistinguishable from the views in the real model. The protocol output is given by Definition 3 and is the same for the ideal and the real model (Step 6). □

Complexity Analysis. Let n denote the number of parties, k the number of inputs, b the bit size of the modulus, and D the domain of inputs. For the protocol, we have the computation complexity $O\left(\left(|D| + k^3 \cdot n\right) \cdot n \cdot b^3\right)$ for each party. The computations in the malicious model protocol are $O(n)$ more complex compared to [29], because we have to verify $n - 1$ proofs in each step.

The shuffle proof verification (which depends on the size of the domain D), increases the complexity by a factor of $O\left(|D| \cdot n \cdot b^3\right)$ for $n-1$ such verifications. The communication of the protocol is tightly coupled with the computation, since each computation needs to be proven by a corresponding proof sent over the network. The size of all messages exchanged over the network is therefore bound by $O\left(\left(|D| + k^3 \cdot n\right) \cdot n \cdot b\right)$. Compared to the semi-honest variants [29], the communication complexity is increased due to the additional transmission of ZK-proofs.

4.2 A Malicious Model Protocol for $MPROS^{SR}$

Similar to the minimum of ranks protocol, a malicious model protocol for ordered set reconciliation with the *sum of ranks* composition scheme can be constructed. The most important difference between the construction of the sum of ranks protocol and the minimum of ranks protocol is that the former also uses set reduction. To compute a set reduction, we need a sum of polynomial multiplications which can be proven secure using the usual zero-knowledge proof

Problem	Model	Comp./Comm. Complexity
$MPROS^{MR}$	Semi-honest, standard model, [29]	$O\left(k^3 \cdot n \cdot b^3\right)$ $O\left(k^2 \cdot n \cdot b\right)$
	Malicious, random oracle model, Section 4	$O\left(\left(\lvert D\rvert + k^3 \cdot n\right) \cdot n \cdot b^3\right)$ $O\left(\left(\lvert D\rvert + k^3 \cdot n\right) \cdot n \cdot b\right)$
$MPROS^{SR}$	Semi-honest, standard model, [29]	$O\left(k^6 \cdot n^4 \cdot b^3\right)$ $O\left(k^3 \cdot n^3 \cdot b\right)$
	Malicious, random oracle model, Section 4	$O\left(\left(\lvert D\rvert + k^5 \cdot n^4\right) \cdot k \cdot n \cdot b^3\right)$ $O\left(\left(\lvert D\rvert + k^5 \cdot n^4\right) \cdot k \cdot n \cdot b\right)$

Table 3. Overview of MPROS protocols proposed in this paper or previous work

construction described in Section 3. Otherwise, the protocol is similar to the semi-honest constructions of Neugebauer et al. [29,30,31].

Using the techniques from this paper, a protocol for the sum of ranks composition scheme with computation complexity $O\left(\left(\lvert D\rvert + n^4 \cdot k^5\right) \cdot n \cdot k \cdot b^3\right)$ and $O\left(\left(\lvert D\rvert + n^4 \cdot k^5\right) \cdot n \cdot k \cdot b\right)$ communication complexity can be constructed. We omit the formal protocol description due to space restrictions. However, we implemented and evaluated both malicious model variants, i. e., $MPROS^{MR}$ and $MPROS^{SR}$. Table 3 summarizes the results in theory. All four protocols are polynomial-time bounded with respect to the number of parties n and inputs k.

5 Implementation & Evaluation

We implemented and evaluated our newly proposed protocols as well as the two protocols proposed in [29]. In the following we describe our implementation, provide a description of our test setup, and then present a selection of test results.

Implementation. Our core implementation is written in Java version 1.7.0. We used the GNU Multiple Precision Arithmetic Library (GMP) version 5.0.5 [16] to efficiently compute expensive arithmetic operations such as modular exponentiation. Therefore, we wrote a native C++ library and used it with the Java Native Interface (JNI) [21].

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 our ZK-framework for the Paillier cryptosystem, both MPROS protocols presented in [29] as well as our malicious model variants of $MPROS^{MR}$ and $MPROS^{SR}$, see Section 4. We evaluate the performance of all four MPROS protocols denoted as MR, MR-zk, SR, and SR-zk with respect to computation and communication overhead. Therefore, we measure the runtime and count the number of bytes transmitted for each party.

Whenever possible, we used parallelization by simultaneous computation in threads — depending on the number of available CPU cores. Our protocols

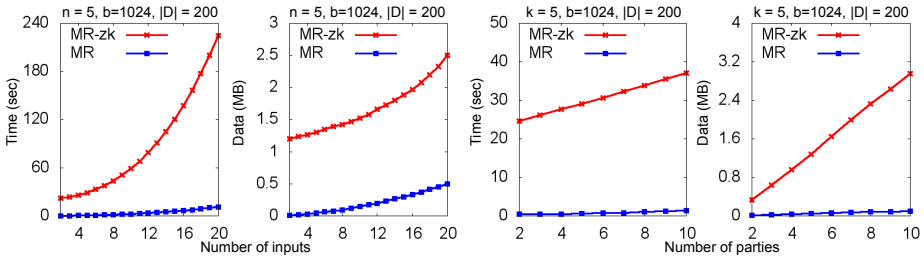

Fig. 1. Comparison of MR-zk and MR for $n = 5$ and $k = 5$

require each party to verify $n-1$ proofs in each step of the protocol. Each of the $n-1$ proofs is of the same size and requires the same amount of computation to be verified. This allows linear speed-up of the computation by parallelized proof verification. Due to our choice to use non-interactive zero-knowledge proofs, we can even precompute zero-knowledge proofs of upcoming protocol steps. These optimizations significantly improve the practical efficiency.

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. The adjustable parameters for each run are the number of parties n, the number of inputs k, the size of the input domain $|D|$, and the keysize in number of bits b. We tested all four implemented MPROS protocols with up to 10 parties and varied the number of inputs. We tested for a keysize of 1024 and 2048 bit. 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 the least preference for each party. The overall runtime and the number of bytes transmitted are averaged over three independent MPROS protocol runs.

Test Results. In Figure 1 and Figure 2, we present our results for a reasonable keysize of $b = 1024$ bit varying the number of parties or the number of inputs for $n = 5$ and $k = 5$. The impact of the keysize is roughly quadratic in the number of bits $O(b^2)$ and the impact of the domain is linear in the size of the domain $O(|D|)$. As expected from theory, the runtime for the malicious model variants MR-zk, and SR-zk is higher than for the semi-honest variants MR, and SR. For an input domain D of 200 elements and up to 20 inputs k, we have a runtime of up to 4 minutes for MR-zk and up to 12 seconds for MR. Also, the amount of data transmitted is higher for the malicious model variants with, e. g., 2.5 MB for MR-zk compared to 0.5 MB for MR in case of 20 inputs. Both protocols MR and MR-zk show linear behavior with respect to the number of parties n due to the parallelized proof verification. Figure 2 shows the results for the sum of ranks composition scheme. We see that we obtain similar results as for the minimum

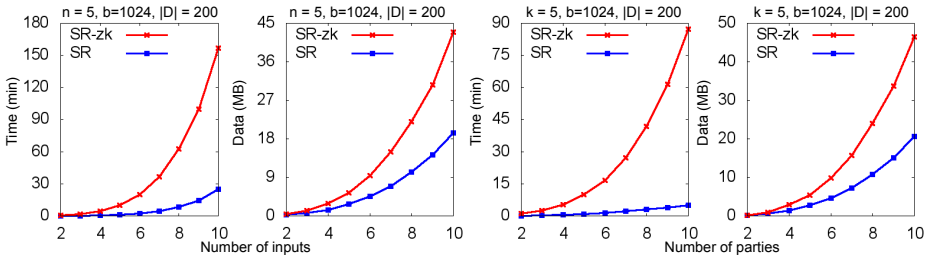

Fig. 2. Comparison for SR-zk and SR for $n = 5$ and $k = 5$

of ranks composition scheme. However, the runtime and data transmitted is in general higher. E. g., the computation time for SR-zk is in the range of 150 minutes for ten inputs with five parties compared to 25 minutes for SR. As a conclusion, the notion of a stronger security model comes at the price of slower protocols and the need to transmit more data. The protocol MR-zk (*SR-zk*) is roughly twenty (*ten*) times slower than MR (*SR*) and the transmitted data is ten (*two*) times higher than in case of MR (*SR*).

6 Conclusion

We designed and implemented the first multi-party protocols for ordered set reconciliation which are provably secure in the malicious model. Our security proofs are based on a novel framework for secure computation on Paillier-encrypted polynomials which is resistant against malicious attackers. Our theoretical analysis of the asymptotic complexity of our new protocols is confirmed by the practical evaluation of our implementation. As part of future work, we will investigate if we can further increase the performance of our protocols by using the recently developed set operations techniques [4,10].

References

1. M. Bellare and P. Rogaway. Random oracles are practical. In *Computer and Communications Security - CCS 1993*, pages 62–73. ACM, 1993.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *ACM Symposium on Theory of Computing - STOC 1988*, pages 1–10. ACM, 1988.
3. D. Bernhard, O. Pereira, and B. Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Advances in Cryptology – ASIACRYPT 2012*, LNCS, pages 626–643. Springer, 2012.
4. Marina Blanton and Everaldo Aguiar. Private and oblivious set and multiset operations. In *ASIACCS*, pages 40–41, 2012.
5. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical report, ETH Zürich, 1997.
6. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, pages 143–202, September.

7. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. *Cryptology ePrint Archive*, 2000/055, 2000.
8. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT 2001*, pages 280–300. Springer, 2001.
9. Emiliano De Cristofaro and Gene Tsudik. On the performance of certain private set intersection protocols (And some remarks on the recent paper by Huang et al. in NDSS'12). Cryptology ePrint Archive, Report 2012/054, 2012. http://eprint.iacr.org/.
10. Many D., Burkhart M., and Dimitropoulos X. Fast Private Set Operations with SEPIA. Technical report, Communication Systems Group, ETH Zurich, 2012.
11. I. Damgård and M. J. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography - PKC 2001*, pages 119–136. Springer, 2001.
12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.
13. Emiliano De Cristofaro and Gene Tsudik. Experimenting with fast private set intersection. In Stefan Katzenbeisser, Edgar Weippl, L.Jean Camp, Melanie Volkamer, Mike Reiter, and Xinwen Zhang, editors, *Trust and Trustworthy Computing*, volume 7344 of *Lecture Notes in Computer Science*, pages 55–73. Springer Berlin Heidelberg, 2012.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, pages 186–194. Springer, 1986.
15. P. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*, pages 351–368. Springer, 2001.
16. GNU Multiple Precision Arithmetic Library, 2012. http://gmplib.org/.
17. O. Goldreich. *Foundations of cryptography: Basic applications*, volume 2. Cambridge University Press, 2004.
18. J. Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, 2002.
19. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT 2008*, pages 379–396. Springer, 2008.
20. Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols. In *Network and Distributed System Security Symposium*, 2012.
21. JNI: Java Native Interface for integration of code written in other languages, 2012. http://java.sun.com/docs/books/jni/.
22. L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005, LNCS*, pages 241–257. Springer-Verlag, 2005.
23. J. Loftus, A. May, N. P. Smart, and F. Vercauteren. On cca-secure somewhat homomorphic encryption. In *Proceedings of the 18th international conference on Selected Areas in Cryptography*, SAC'11, pages 55–72, Berlin. Springer.
24. D. Mayer and S. Wetzel. Verifiable Private Equality Test: Enabling Unbiased 2-party Reconciliation on Ordered Sets in the Malicious Model. In *ASIACCS, 7th Symposium on Information, Computer and Communications Security*. ACM, 2012.
25. D. A. Mayer, G. Neugebauer, U. Meyer, and S. Wetzel. Enabling fair and privacy-preserving applications using reconciliation protocols on ordered sets. In *IEEE Sarnoff Symposium 2011*, pages 1 – 6. IEEE, 2011.
26. U. Meyer and S. Wetzel. Distributed privacy-preserving policy reconciliation. In *ICC 2007*, pages 1342–1349. IEEE, 2007.

27. U. Meyer, S. Wetzel, and S. Ioannidis. New advances on privacy-preserving policy reconciliation. *Cryptology ePrint Archive*, 2010/064, 2010.
28. C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Computer and Communications Security - CCS 2001*, pages 116–125. ACM, 2001.
29. G. Neugebauer, L. Brutschy, U. Meyer, and S. Wetzel. Design and Implementation of Privacy-Preserving Reconciliation Protocols. In *The 6th International Workshop on Privacy and Anonymity in the Information Society (PAIS)*, ACM, 2013. To appear, preprint at http://itsec.rwth-aachen.de/publications/pais_preprint.pdf.
30. G. Neugebauer, U. Meyer, and S. Wetzel. Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets. In *ISC 2010*, LNCS, 2010.
31. 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.
32. L. Nguyen, R. Safavi-Naini, and K. Kurosawa. Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In *Applied Cryptography and Network Security - ACNS 2004*, pages 61–75. Springer, 2004.
33. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT 1999*, pages 223–238. Springer, 1999.
34. A. C. Yao. Protocols for secure computations. In *Symposium on Foundations of Computer Science - SFCS 1982*, pages 160–164. IEEE, 1982.

A Appendix

A.1 Proof of Plaintext Knowledge

Algorithm 4 illustrates the ZKPK. The proof generation consists of computing the commitments, the challenge, and the responses, and the proof verification consists of reconstructing the commitment t from the received challenge and the responses, and finally verifying that the challenge was correctly generated for this particular commitment. The protocols assume that each party P_i has a random pid_{P_i} assigned to it, that is used as a hash argument to prevent the reuse of the proof by other parties.

Correctness: We can see that the reconstructed commitment t' matches the original commitment t, and therefore $h(pid_{P_i}, g, y, t') = c$:

$$\begin{aligned} t' &= g^{m''} r''^N y^c = g^{m''} \cdot \left(r' \cdot r^{-c}\right)^N \cdot \left(g^m r^N\right)^c \\ &= g^{m''+cm} \cdot r'^N \cdot r^{-cN+cN} = g^{m'} r'^N = t \qquad (\text{mod } N^2) \end{aligned}$$

Special Soundness: We show special soundness for the *interactive* proof, i. e., in Step (3) of the protocol the challenge is computed by the verifier as a randomly chosen value $c \in \mathbb{Z}_n^*$. Special soundness for the *non-interactive* variant is achieved by instantiation of the strong Fiat-Shamir heuristic similar to the constructions in [23].

Given two transcripts (c_1, m_1'', r_1'') and (c_2, m_2'', r_2'') for the same commitment, we can extract the values of m and r as follows. Let σ, s be defined by $\sigma\,(c_2 - c_1) = 1 + sN$. σ and s exist under the assumption that $c_2 - c_1$ is coprime

Algorithm 4 Construction and verification of a plaintext knowledge proof

Specification: $ZKPK\{(m,r)|y = E(m,r)\}$
Construction:

(1) Select random $m', r' \in \mathbb{Z}_N^*$
(2) Compute commitment $t = g^{m'} r'^N \bmod N^2$
(3) Obtain a challenge $c = h(pid_{P_i}, g, y, t)$
(4) Compute responses $m'' = m' - cm \bmod N$ and $r'' = r' \cdot r^{-c} \bmod N^2$
(5) Send (c, m'', r'')

Verification:

(1) Compute $t' = g^{m''} r''^N y^c \bmod N^2$
(2) Verify $h(pid_{P_i}, g, y, t') \stackrel{?}{=} c$

---

to N (which is true if $c_2 - c_1$ is less than p and q). Extract m, r by computing $m = (m_1'' - m_2'')\sigma$ and $r = \left(\frac{r_1''}{r_2''}\right)^{\sigma} y^{-s} \bmod N^2$. We can see that m, r are extracted correctly by following the proof idea of [7,8]. Since both transcripts are valid, and the commitment t in both cases is the same, we also have equal reconstructed commitments t_1', t_2': $g^{m_1''} r_1''^N y^{c_1} = g^{m_2''} r_2''^N y^{c_2} \pmod{N^2}$. By transformation and exponentiation by σ, we obtain

$$g^{(m_1'' - m_2'')\sigma} \left(\frac{r_1''}{r_2''}\right)^{\sigma N} = y^{\sigma(c_2 - c_1)} \pmod{N^2}.$$

The right side is by definition of σ simply y^{1+sN}. Using the form of y we get

$$g^{(m_1'' - m_2'')\sigma} \left(\left(\frac{r_1''}{r_2''}\right)^{\sigma} \cdot y^{-s}\right)^N = g^m r^N \pmod{N^2}.$$

Thus, we have $m = (m_1'' - m_2'')\,\sigma$ and $r = \left(\frac{r_1''}{r_2''}\right)^{\sigma} \cdot y^{-s} \bmod N^2$.

Special Honest-Verifier Zero-Knowledge: The verifier learns the following transcript (c, m'', r''). It is sufficient to show *special honest-verifier zero-knowledge* as the verifier V can not manipulate the challenge c. A simulator can easily generate such an accepting transcript in the following fashion:

1. Select random $m'' \in \mathbb{Z}_N^*$, $r'' \in \mathbb{Z}_{N^2}^*$, and random challenge c.
2. Compute $t = g^{m''} r''^N y^c \bmod N^2$.
3. Make the random oracle output c for input pid_P, g, y, t.

The transcript (c, m'', r'') is accepting and randomly distributed, and thus it is computationally indistinguishable from the transcript of a real protocol run. Note that the simulator can control the output of the hash function since we are proving the properties in the random oracle model.

Algorithm 5 Construction and verification of multiplication proof

Specification: $ZKPK\{(m, r_1, r_2) | \alpha^m r_1^N = \beta \wedge \gamma = E(m, r_2)\}$
Construction:

(1) Select random $m', r_1', r_2' \in \mathbb{Z}_{N^2}^*$
Compute commitments $t_1 = \alpha^{m'} r_1'^N \bmod N^2$ and $t_2 = E(m', r_2')$
(2) Obtain a challenge $c = h(pid_{P_i}, g, \alpha, \beta, \gamma, t_1, t_2)$
(3) Compute responses $m'' = m' - cm \bmod N$,
as well as $r_2'' = r_2' \cdot r_2^{-c} \bmod N$ and $r_1'' = r_1' \cdot r_1^{-c}$
(4) Send (c, m'', r_1'', r_2'')

Verification:

(1) Compute $t_1' = \beta^c \alpha^{m''} r_1''^N \bmod N^2$ and $t_2' = g^{m''} r_2''^N \gamma^c \bmod N^2$
(2) Verify $h(pid_{P_i}, g, \alpha, \beta, \gamma, t_1', t_2') = c$

A.2 Proof of Correct Multiplication

Algorithm 5 illustrates the ZKPK. The additional commitment t_1 and response r_1'' are used to prove knowledge of the randomization factor r_1 which is used to compute $\beta = m \cdot \alpha$.

Correctness: We have $t_2 = t_2'$, following the completeness argument for Algorithm 4. For t_1 and t_1', the same can be verified by inspection:

$$\begin{aligned} t_1' &= \beta^c \alpha^{m_1''} r_1'^N \\ &= \left(\alpha^m r_1^N\right)^c \alpha^{m''} \left(r_1' \cdot r_1^{-c}\right)^N \\ &= \alpha^{m''+cm} r_1^{cN-cN} r_1'^N \\ &= \alpha^{m'} r_1'^N \\ &= t_1 \qquad (\bmod N^2) \end{aligned}$$

Thus, the hash function returns the original challenge c and the verification succeeds.

Special Soundness: First, we extract m, r_2 in the same way as in the plaintext knowledge proof. Similarly, we can extract r_1 from two responses $r_{1,1}'', r_{1,2}''$ with the same commitment and different challenges c_1, c_2 by finding σ, s such that $\sigma(c_2 - c_1) = 1 + sN$, and computing

$$r_1 = \left(\frac{r_{1,1}''}{r_{1,2}''}\right)^\sigma \cdot \beta^{-s} \bmod N^2$$

with the same reasoning as given for the plaintext knowledge proof.

Special Honest-Verifier Zero-Knowledge: The verifier learns the following transcript (c, m'', r_1'', r_2''). A simulator can generate an accepting transcript in the following way:

1. Select random m'', r_1'', r_2'' and challenge c
2. Compute $t_1 = \beta^c \alpha^{m''} r_1''^N \bmod N^2$ and $t_2 = g^{m''} r_2''^N \gamma^c \bmod N^2$
3. Make the random oracle output c for $pid_P, g, \alpha, \beta, \gamma, t_1, t_2$

Both transcripts are computationally indistinguishable (based on the semantic security of the cryptosystem), therefore he does not learn anything secret.