Verifiable Private Equality Test: Enabling Unbiased 2-Party Reconciliation on Ordered Sets in the Malicious Model

Daniel A. Mayer Susanne Wetzel
Stevens Institute of Technology
Castle Point on Hudson
Hoboken, New Jersey, USA
{mayer, swetzel}@cs.stevens.edu

ABSTRACT

In this paper we introduce the novel notion called *Verifiable Private Equality Test* (VPET) and propose an efficient 2-party protocol for its implementation. VPET enables two parties to securely perform an arbitrary number of comparisons on a fixed collection of (key, value) pairs and thus it is more generic than existing techniques such as *Private Equality Test* and *Private Set Intersection*.

In addition, we demonstrate how higher-level protocols such as *Privacy-Preserving Reconciliation on Ordered Sets* (PROS) can be implemented using VPET.

Using simulation-based techniques, our new protocols are proven secure in the malicious model. Furthermore, we present a theoretical complexity analysis as well as a thorough experimental performance evaluation of the C++ implementation of our new VPET and PROS protocols.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

General Terms

Security, Theory, Performance

Keywords

Private Equality Test, Multi-Party Computation, Malicious Model, Cryptographic Protocol, Reconciliation

1. INTRODUCTION

Privacy-preserving comparison techniques such as *Private Equality Test* (PET) [21, 43, 7, 45, 32] and *Private Set Intersection* (PSI) [23, 27, 29, 30, 17, 20, 19] are prominent techniques which are frequently used in the greater context of cryptographic protocols and specifically in privacy-preserving data mining [42, 2, 1]. Recently, many real-world applications with the need for privacy-preserving solutions have emerged. One prominent example is cloud computing which encourages the delegation of computation or data storage to third parties. In addition, the privacy awareness of the general public is increasing [41, 47]. Given these developments, the demand for privacy-preserving protocols is expected to continue to rise.

PET enables the comparison of two values while PSI computes the intersection of two sets in a privacy-preserving manner. In this context, privacy-preserving refers to the fact that the parties learn nothing but the intended result. In particular, the individual inputs of the parties remain private. It is important to note that PET may be considered a special case of PSI for sets of size one. Various protocols for both techniques offering different security guarantees have been proposed in the literature. However, all solutions to date only consider a one-time execution of the protocol, i.e., the parties involved either learn the result of a single comparison or the entire intersection of their input sets.

This paper has three main contributions. First, we introduce the novel notion and an efficient protocol for *Verifiable Private Equality Test* (VPET) which provides a middle ground between PET and PSI. VPET allows for improved control and flexibility while guaranteeing security in the presence of arbitrary adversaries. Specifically, VPET enables two parties to perform an arbitrary number of comparisons on an immutable set of input values. The main challenge in designing VPET is to enable both parties to perform such comparisons while preventing any change to the parties' original inputs in between comparisons. In particular, the straightforward approach of repeatedly executing PETs does not constitute a solution since it does not prevent parties from changing their inputs in between comparisons. To overcome this shortcoming of existing solutions, our protocol first requires both parties to commit to their input values. Given this commitment, VPET then allows both parties to verify that the correct input values were used in every comparison.

Second, we show how our new VPET protocol can be used to construct higher-level protocols with an example of *Privacy-Preserving Reconciliation of Ordered Sets* (PROS) [38, 39]. The PROS technique enables applications such as voting and scheduling [35]. Our newly constructed PROS protocol is the first to provide security in the presence of arbitrary adversaries.

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

Third, we conduct a thorough performance analysis of the newly designed VPET and PROS protocols. We present a theoretical complexity analysis as well as experimental performance evaluations based on our C++ implementation.

Outline: Section 2 introduces security models and cryptographic tools. In Section 3 we present our novel VPET protocol and show that it is secure in the malicious model. In Section 4 we then use VPET as an integral building block in constructing a PROS protocol with security in the malicious model. In addition, we present our theoretical and experimental performance evaluation in Section 5. Finally, in Section 6 we review related work.

2. PRELIMINARIES

2.1 Security Model

In this paper we assume an authenticated channel between parties and consider the presence of an active adversary which is allowed to behave maliciously by performing arbitrary actions. In particular, the adversary may not follow the protocol, i.e., it may not perform the required operations correctly. This is contrasted by the weaker notion of a passive adversary, i.e., an adversary which is honest but curious (also referred to as semi-honest) and has the goal of learning as much as possible from the protocol execution. This can informally be described as a party which follows the protocol but may perform additional polynomial time computations on the outcome as well as on all intermediate results. Formal definitions for both types of adversaries are given in [25].

Furthermore, we will leverage the *Random Oracle Model* to construct efficient non-interactive zero knowledge proofs [5].

2.2 Secure Multi-Party Computation

In secure *Multi-Party Computation* (MPC) two or more parties wish to securely compute a function which is known to all participants. This is commonly formalized as $f(x_1, x_2, \ldots, x_n, r) = (y_1, y_2, \ldots, y_n)$ where r is some randomness used during the computation and x_i and y_i are the input and output of party i $(1 \leq i \leq n)$ [25, 16]. In this context, *secure* refers to an execution which is performed correctly and in such a way that all parties learn nothing beyond the intended output of the function. In this paper we focus on secure 2-party computation.

The concept of MPC was first introduced by Yao [50] and remained a purely theoretical framework for many years. However, due to protocol improvements as well as progress in computation and communication network speeds, MPC has become practical in recent years [6, 48]. A comprehensive discussion on MPC can be found in [25].

2.2.1 Simulation-Based Security in MPC

In this work we follow the common security definition based on the simulation paradigm which compares the real-world execution to an ideal-world model [40, 26, 4, 13, 25]. The ideal model defines what a protocol is intended to do and which party receives which output. This can be understood as giving all inputs x_i to an incorruptible trusted third party (TTP) which computes the functionality $\mathcal{F}_f$ correctly and returns the correct outputs y_i to each party i $(1 \leq i \leq n)$.

However, in the real world a protocol π_f is executed to compute the function f and in addition to the outputs y_i the adversary learns all exchanged messages as well as all randomness used during the computation of f.

In order for the protocol π_f to securely implement $\mathcal{F}_f$, the real execution of π_f must not disclose any more information than what can be learned from the ideal execution. To prove this, one shows that it is possible to construct a simulator which, given the ideal output, can generate a transcript which is identical to that of the real protocol execution. If such a transcript can be generated using only the knowledge of the ideal execution, this implies that the transcript cannot contain more information than the ideal output and the protocol is called *perfectly secure* [13, 16, 25].

It should be noted that the following commonly known limitations are inherent to MPC and can not be prevented: Parties may refuse to participate in the protocol, parties may substitute their own input, and parties may abort the protocol before all parties have received the output [25].

2.2.2 Sequential Modular Composition

One crucial tool in constructing complex secure protocols is *modular composition* [40]. Given a protocol which securely computes a certain functionality, modular composition allows the use of that protocol as a secure building block (as if it were computed by a TTP) when constructing higher-level protocols. This can be formalized by introducing a *hybrid model* in which both parties communicate (as in the real world) but have trusted help for some computations (as in the ideal world) [28]. In this work we will limit ourselves to *sequential modular composition* which requires that the execution of the secure sub-protocol is finished before the execution of the main protocol continues. In [14], Canetti generalized this notion by introducing *universal composability* which allows protocols to be interleaved arbitrarily.

2.2.3 PET and PSI

PET is a two-party protocol in which both parties supply a single input value. At the conclusion of the protocol one party learns whether the two values are equal and the other party learns nothing [21].

PSI is a two-party protocol in which both parties supply a private input set. At the conclusion of the protocol one party learns the intersection of the two sets and the other party learns nothing [23].

2.3 Notation

Throughout this paper we assume the following setup. Let p, q be large primes such that $q \mid p-1$ and let $\mathbb{Z}_q^*$ be the cyclic subgroup of $\mathbb{Z}_p^*$ of order q with independently chosen, random generators g, h. The public parameters are g, h, p, q.

2.4 Zero-Knowledge Signatures of Knowledge

In the following we use signatures of knowledge to efficiently prove statements about discrete logarithms. Feige et al. introduced the notion of interactive *Zero-Knowledge Proofs Of Knowledge* (ZKPOK) [22] which enable one party to convince another party that the former knows some secret. In [10, 11], Camenisch et al. proposed *signatures of knowledge* which combine the notion of ZKPOK with a technique similar to Schnorr signatures [49]. In signatures of knowledge, the random challenge used in interactive proof

systems is replaced by a call to a random oracle $\mathcal{H}$, thus making the scheme non-interactive and more efficient.

2.4.1 Cryptographic Hardness Assumptions

Discrete Logarithm (Log) (DL) Assumption.

Let g be a generator of the cyclic group $\mathbb{Z}_q^*$ of order q and $d \leftarrow_r \mathbb{Z}_q^*$. Then for any PPT algorithm $\mathcal{A}$ it holds that $Pr\left[\mathcal{A}\left(p, g, d\right) = x \mid d = g^x \mod q\right] < \frac{1}{n^a}$ for fixed $a > 0$ and sufficiently large n [37].

The Representation (REP) Problem.

The representation problem is a generalization of the discrete log problem. Let $(g_1, \ldots, g_m)$ with $g_i \neq g_j$ for $i \neq j$ be a vector of random generators of a group $\mathbb{Z}_q^*$ of prime order q. Then for any $y \in \mathbb{Z}_q^*$ and any PPT algorithm $\mathcal{A}$ it holds that $Pr\left[\mathcal{A}\left(q, (g_1, \ldots, g_m), y\right) = (x_1, \ldots, x_m) \mid y = \prod_{i=1}^{m} g_i^{x_i}\right] < \frac{1}{n^a}$ for fixed $a > 0$ and sufficiently large n [8, 9].

2.4.2 Signatures of Knowledge

In this paper three types of signature of knowledge schemes will be used: knowledge of discrete logarithm, knowledge of equality of discrete logarithms, and knowledge of a representation.

Signature of Knowledge of Discrete Log.

Notation: $ZK_{DL}\left(e \mid y = g^e\right)$.

Computation: To prove the knowledge of $e = \log_g y$ the prover chooses $t \leftarrow_r \mathbb{Z}_q^*$ and computes $c = \mathcal{H}\left(y, g, g^t\right)$ and $r = t - c \cdot e$ [49].

Verification: Given c and r the verifier can check whether $c \stackrel{?}{=} \mathcal{H}\left(y, g, g^r y^c\right)$.

Signature of Knowledge of Equality of Discrete Logs.

Notation: $ZK_{EDL}\left(e \mid y_1 = g^e \wedge y_2 = h^e\right)$.

Computation[1] **:** To prove the knowledge of $e = \log_g y_1 = \log_h y_2$ the prover chooses $t \leftarrow_r \mathbb{Z}_q^*$ and computes $c = \mathcal{H}\left(y_1, y_2, g, h, g^t, h^t\right)$ and $r = t - c \cdot e$ [15].

Verification: Given c and r the verifier checks whether $c \stackrel{?}{=} \mathcal{H}\left(y_1, y_2, g, h, g^r {y_1}^c, n^r {y_2}^c\right)$.

Signature of Knowledge of a Representation.

Notation[2]**:** $ZK_{REP}\left((e_1, e_2) \mid y_1 = g^{e_1} \wedge y_2 = g^{e_2} \cdot h^{e_1}\right)$.

Computation: To prove the knowledge of (e_1, e_2) s.t. $y_1 = g^{e_1} \wedge y_2 = g^{e_2} \cdot h^{e_1}$ the prover chooses $t_i \leftarrow_r \mathbb{Z}_q^*$ for $i = 1, 2$ and computes $c = \mathcal{H}\left(y_1, y_2, g, h, g^{t_1}, g^{t_2} \cdot h^{t_1}\right)$ and $r_i = t_i - c \cdot e_i$ for $i = 1, 2$ [11].

Verification: Given c and r_1, r_2 the verifier checks whether $c \stackrel{?}{=} \mathcal{H}\left(y_1, y_2, g, h, y_1^c \cdot g^{r_1}, y_2 \cdot g^{r_2} \cdot h^{r_1}\right)$.

[1]Note: This scheme can be extended to simultaneously prove the equality of more than two discrete logs in a straight-forward fashion.

[2]In this paper only this specific proof of a representation is needed.

The Functionality Maintains a Private State:

- Two commitment maps $comm_0, comm_1$ containing the committed (key, value) pairs of P_0, P_1 respectively.
- A binary variable *commit* initialized to `true`.

Commit Functionality:

- On message $(\texttt{commit}, (k_{0,j_0}, v_{0,j_0}))$ from party P_0:
  - If $commit = \texttt{true}$: add (k_{0,j_0}, v_{0,j_0}) to $comm_0$ else: `abort`.
- On message $(\texttt{commit}, (k_{1,j_1}, v_{1,j_1})$ from party P_1:
  - If $commit = \texttt{true}$: add (k_{1,j_1}, v_{1,j_1}) to $comm_1$ else: `abort`.

Comparison Functionality:

- On message $(\texttt{compare}, (v_{0,j_0}, k_{0,j_0}, k_{1,j_1}, \sigma))$ from P_0 and message $\left(\texttt{compare}, (v'_{1,j_1}, k'_{1,j_1}, k'_{0,j_0}, \sigma')\right)$ from P_1:
  - Set $commit = \texttt{false}$.
  - If $k_{0,j_0} \neq k'_{0,j_0} \vee k_{1,j_1} \neq k'_{1,j_1}$: `abort`.
  - If $\sigma \neq \sigma' \vee \sigma \notin \{0,1\} \vee \sigma' \notin \{0,1\}$: `abort`.
  - If $comm_0[k_{0,j_0}]$ is not a commitment to v_{0,j_0} or $comm_1[k'_{1,j_1}]$ is not a commitment to v'_{1,j_1}: `abort`. else: return $\left(v_{0,j_0} \stackrel{?}{=} v'_{1,j_1}\right)$ to P_σ and $(\perp)$ to $P_{\overline{\sigma}}$.

Figure 1: Ideal Functionality $\mathcal{F}_{VPET}$ for VPET.

2.5 Pedersen Commitment Scheme

The commitment scheme by Pedersen is perfectly hiding and computationally binding under the DL assumption [46].

Commitment to $x \in \mathbb{Z}_q^* : r \leftarrow_r \mathbb{Z}_q^*$, $C_x = g^x \cdot h^r$

Decommitment: Reveal x, r.

3. VERIFIABLE PRIVATE EQUALITY TEST (VPET)

In the following we consider two parties P_0 and P_1 which each hold a map, i.e., a collection of (key, value) pairs, as input. We denote these maps as $M_0 = ((k_{0,0}, v_{0,0}), \ldots, (k_{0,n_0}, v_{0,n_0}))$ for P_0 and as $M_1 = ((k_{1,0}, v_{1,0}), \ldots, (k_{1,n_1}, v_{1,n_1}))$ for P_1 where n_0, n_1 are the largest indices in the respective maps (i.e., the maps have size $n_0 + 1$ and $n_1 + 1$). The values v_{δ,j_δ} ($\delta \in \{0,1\}$ and $0 \leq j_\delta \leq n_\delta$) in the maps are confidential. All keys k_{δ,j_δ} are public and are required to be unique.

In this setting, our goal is to enable both parties to perform an arbitrary number of consecutive *comparisons* (i.e., equality tests) of any arbitrary pair of values v_{0,j_0}, v_{1,j_1} while guaranteeing security in the presence of malicious adversaries. In addition, we aim for a solution which is *symmetric*, i.e., for each comparison both parties can jointly select one party (P_0 or P_1) which learns the result of the comparison. As we will see in Section 4, this in particular allows for the construction of higher-level protocols which are *mutual*, i.e., in which both parties learn the result.

Existing solutions for PET can be used to securely test two values for equality in a privacy-preserving manner. However, when considering multiple, consecutive comparisons of values in fixed maps, securing the individual comparisons alone

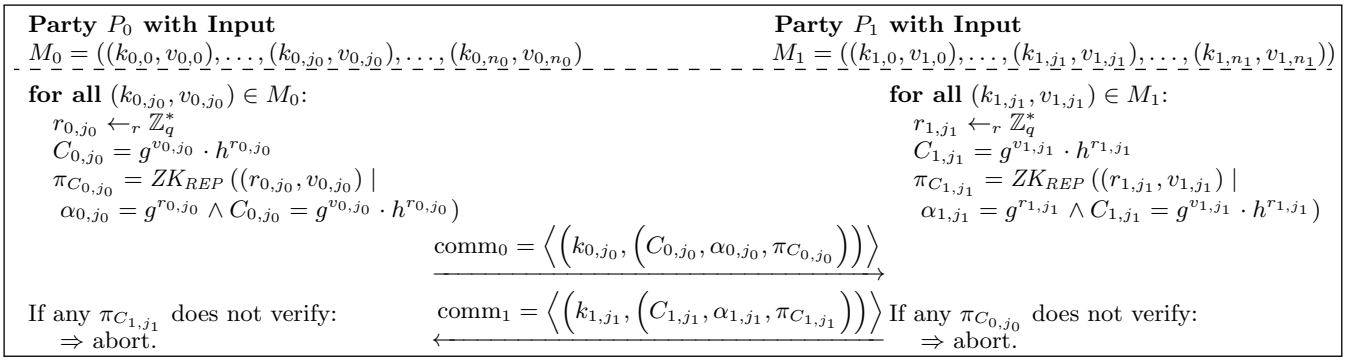

Figure 2: VPET: Commitment Phase

is not sufficient. In particular, malicious adversaries may use incorrect values for a specific comparison (e.g., values which correspond to a different key or values which are not in the map altogether).

We address this problem by introducing the novel notion called *Verifiable Private Equality Test* (VPET) which allows for comparisons on committed data. In particular, by forcing P_0 and P_1 to commit to all their (key, value) pairs, VPET ensures that both input maps can no longer be modified once the first comparison has taken place. Furthermore, for each comparison both parties specify a key k_{0,j_0} (k_{1,j_1}) to a value v_{0,j_0} (v_{1,j_1}) in the other party's map which enables both parties to verify that the comparison was indeed carried out on v_{0,j_0} and v_{1,j_1}.

3.1 Intuition

Our newly designed VPET protocol has two main phases. First, in the *commitment phase*, both parties commit to their respective input maps. This is completed before the first comparison may take place. Second, in the *comparison phase*, the parties may perform an arbitrary number of comparisons. Our construction uses Pedersen commitments (see Section 2.5) and each comparison is carried out as an equality test on blinded values. Furthermore, we employ light-weight zero-knowledge signatures of knowledge (see Section 2.4) to prove correctness of the computation as well as usage of the correct values in each comparison.

Below, we first describe the ideal functionality for VPET (Section 3.2). In Section 3.3 we then detail the actual protocol. The proofs of protocol correctness and security are given in Sections 3.4 and 3.5.

3.2 Ideal Functionality for VPET

Figure 1 shows the ideal functionality $\mathcal{F}_{VPET}$ for VPET. During the commitment phase, both parties P_0 and P_1 commit to their (key, value) pairs by sending a message $(\texttt{commit}, (k, v))$ for each (k, v) in their input map. The ideal functionality accepts commitments as long as no comparison has taken place yet, i.e., while *commit* = `true`.

In order to request the comparison of v_{0,j_0} with v_{1,j_1} where $(k_{0,j_0}, v_{0,j_0}) \in M_0$ and $(k_{1,j_1}, v_{1,j_1}) \in M_1$ such that $P_\sigma \in \{P_0, P_1\}$ learns the result of the comparison, P_0 sends $(\texttt{compare}, (v_{0,j_0}, k_{0,j_0}, k_{1,j_1}, \sigma))$ to the ideal functionality. Similarly, to request the comparison of v'_{1,j_1} with v'_{0,j_0} where $(k'_{0,j_0}, v'_{0,j_0}) \in M_0$ and $(k'_{1,j_1}, v'_{1,j_1}) \in M_1$ such that $P_{\sigma'} \in \{P_0, P_1\}$ learns the result of the comparison, P_1 sends $\left(\texttt{compare}, \left(v'_{1,j_1}, k'_{1,j_1}, k'_{0,j_0}, \sigma'\right)\right)$ to the ideal functionality.

After receiving a `compare` message from both parties, the functionality first marks the commitment phase as finished by setting *commit* = `false`. It then checks whether both parties agree on the elements which are to be compared, i.e., whether $k_{0,j_0} = k'_{0,j_0}$ and $k_{1,j_1} = k'_{1,j_1}$. Similarly, the functionality checks whether both parties agree on the recipient of the result, i.e., whether $\sigma = \sigma'$. Finally, it checks whether the values v_{0,j_0}, v'_{1,j_1} are consistent, i.e., whether the commitments correspond to the indicated keys k_{0,j_0}, k'_{1,j_1}. If all checks pass, P_σ learns one bit output $out \in \{0, 1\}$ indicating whether $v_{0,j_0} = v'_{1,j_1}$ and $P_{\overline{\sigma}}$ learns nothing ($\perp$).

3.3 Protocol for VPET

Analogous to the ideal functionality, the actual VPET protocol is implemented in two phases (see Figures 2 and 3).

Commitment Phase.

Party P_0 (resp. P_1) commits to all its (key, value) pairs $(k_{0,j_0}, v_{0,j_0}) \in M_0$ (resp. $(k_{1,j_1}, v_{1,j_1}) \in M_1$) by computing Pedersen commitments C_{0,j_0} (C_{1,j_1}) and representations α_{0,j_0} (α_{1,j_1}). By constructing a zero-knowledge proof of a representation, party P_0 (P_1) proves that it knows the pair (v_{0,j_0}, r_{0,j_0}) $((v_{1,j_1}, r_{1,j_1}))$, that the commitment was computed correctly, and that $\alpha_{0,j_0} = g^{r_{0,j_0}}$ ($\alpha_{1,j_1} = g^{r_{1,j_1}}$). A vector containing all the keys k_{0,j_0} (k_{1,j_1}) paired with a corresponding tuple of commitments and proofs is sent to the other party. If the validation of any proof fails, the respective party aborts the protocol. If all proofs pass, by soundness of the zero-knowledge proof system the commitments were computed correctly and the exponents of α_{0,j_0} (α_{1,j_1}) correspond to the exponents of the h-parts of the commitments.

Comparison Phase.

The protocol given in Figure 3 implements the comparison (i.e., equality test) of values v_{σ,j_σ} and $v_{\overline{\sigma},j_{\overline{\sigma}}}$ (corresponding to keys $k_{\sigma,j_\sigma}, k_{\overline{\sigma},j_{\overline{\sigma}}}$) in which party $P_\sigma \in \{P_0, P_1\}$ learns the result. Specifically, the bit σ determines which party learns the result. I.e., in case of $\sigma = 0 \wedge \overline{\sigma} = 1$, P_0 learns the result and in case of $\sigma = 1 \wedge \overline{\sigma} = 0$, P_1 learns the result.

For each comparison, the input for party P_σ (party $P_{\overline{\sigma}}$) is the 4-tuple $(v_{\sigma,j_\sigma}, C_{\sigma,j_\sigma}, r_{\sigma,j_\sigma}, \text{comm}_{\overline{\sigma}}[k_{\overline{\sigma},j_{\overline{\sigma}}}])$ $((v_{\overline{\sigma},j_{\overline{\sigma}}}, C_{\overline{\sigma},j_{\overline{\sigma}}}, r_{\overline{\sigma},j_{\overline{\sigma}}}, \text{comm}_\sigma[k_{\sigma,j_\sigma}]))$. The first three elements refer to the party's set element, the corresponding commitment, and the randomness used when committing to v_{σ,j_σ} ($v_{\overline{\sigma},j_{\overline{\sigma}}}$). The last element is the commitment corresponding to key $k_{\overline{\sigma},j_{\overline{\sigma}}}$ (k_{σ,j_σ}) which has been received during the commitment phase. The information about $v_{\overline{\sigma},j_{\overline{\sigma}}}$ (v_{σ,j_σ}) contained in $\text{comm}_{\overline{\sigma}}[k_{\overline{\sigma},j_{\overline{\sigma}}}]$ ($\text{comm}_\sigma[k_{\sigma,j_\sigma}]$)

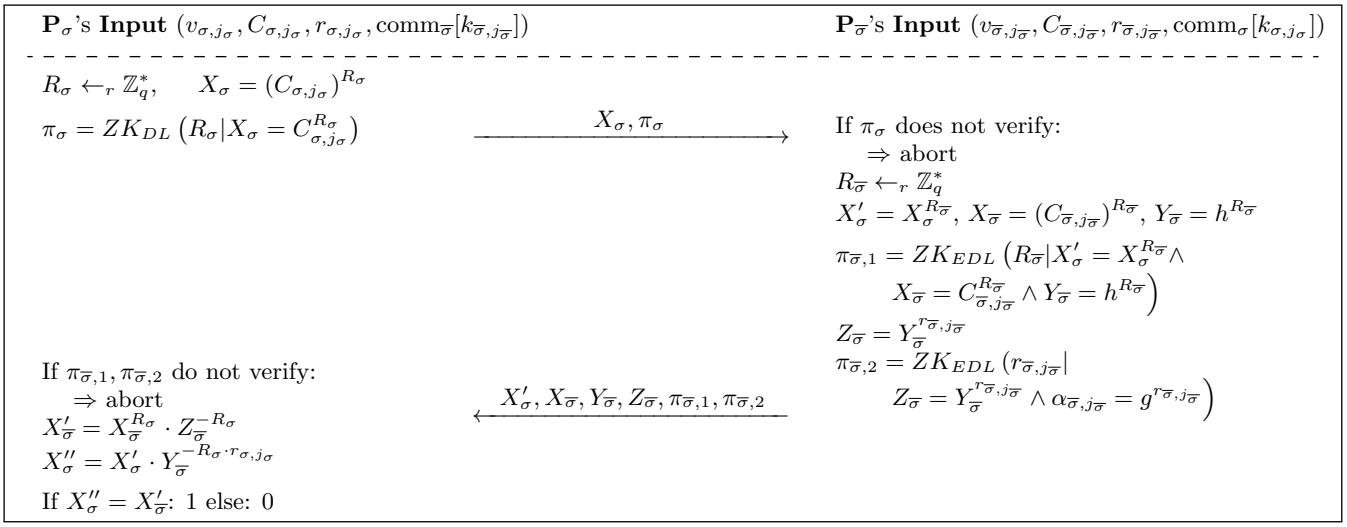

Figure 3: VPET: Comparison Phase for values with keys $k_{\sigma,j_\sigma}, k_{\overline{\sigma},j_{\overline{\sigma}}}$.

enables P_σ ($P_{\overline{\sigma}}$) to verify that $v_{\overline{\sigma},j_{\overline{\sigma}}}$ (v_{σ,j_σ}) was used in the comparison.

As part of the comparison, P_σ first blinds C_{σ,j_σ} with a random element R_σ, creates a proof π_σ, and sends the resulting value X_σ and proof π_σ to $P_{\overline{\sigma}}$. The guarantees of π_σ are two-fold. First, it ensures that the blinding was computed correctly. Second, it enables $P_{\overline{\sigma}}$ to verify that P_σ used v_{σ,j_σ}, since $P_{\overline{\sigma}}$ knows the correct C_{σ,j_σ} from the commitment phase.

If the proof verifies, $P_{\overline{\sigma}}$ chooses random $R_{\overline{\sigma}}$ and blinds both its own value $C_{\overline{\sigma},j_{\overline{\sigma}}}$ and the received X_σ using $R_{\overline{\sigma}}$. $P_{\overline{\sigma}}$ then computes $Y_{\overline{\sigma}}$ and constructs a proof $\pi_{\overline{\sigma},1}$ which ensures that the discrete logarithms of $X'_\sigma, X_{\overline{\sigma}}, Y_{\overline{\sigma}}$ with respect to the used bases are equal to $R_{\overline{\sigma}}$. Finally, $P_{\overline{\sigma}}$ computes $Z_{\overline{\sigma}}$ and constructs proof $\pi_{\overline{\sigma},2}$ which shows that the used $r_{\overline{\sigma},j_{\overline{\sigma}}}$ is equal to the randomness used during the Pedersen commitment. It then sends all values and the proofs to P_σ.

If all proofs verify, P_σ adds its own blinding R_σ to $X_{\overline{\sigma}}$. It then removes the h-part of $X_{\overline{\sigma}}^{R_\sigma}$ and X'_σ by multiplying $X_{\overline{\sigma}}^{R_\sigma}$ by $Z_{\overline{\sigma}}^{-R_\sigma}$ and X'_σ by $Y_{\overline{\sigma}}^{-R_\sigma \cdot r_{\sigma,j_\sigma}}$. This yields $X'_{\overline{\sigma}} = g^{v_{\overline{\sigma},j_{\overline{\sigma}}} \cdot R_\sigma \cdot R_{\overline{\sigma}}}$ and $X''_\sigma = g^{v_{\sigma,j_\sigma} \cdot R_\sigma \cdot R_{\overline{\sigma}}}$, i.e., both $X'_{\overline{\sigma}}$ and X''_σ contain the respective values to be compared blinded with both R_σ and $R_{\overline{\sigma}}$. Thus, equality holds if v_{σ,j_σ} equals $v_{\overline{\sigma},j_{\overline{\sigma}}}$.

3.4 Protocol Correctness

In the following we show that if the zero-knowledge proof systems are sound, the messages received by P_σ have been computed correctly and the values $X''_\sigma, X'_{\overline{\sigma}}$ allow the determining of whether v_{σ,j_σ} and $v_{\overline{\sigma},j_{\overline{\sigma}}}$ are equal.

THEOREM 1. $v_{\sigma,j_\sigma} = v_{\overline{\sigma},j_{\overline{\sigma}}} \Rightarrow X''_\sigma = X'_{\overline{\sigma}}$.

PROOF. The values X''_σ and $X'_{\overline{\sigma}}$ yield:

$$\begin{aligned} X''_\sigma &= X'_\sigma \cdot Y_{\overline{\sigma}}^{-R_\sigma \cdot r_{\sigma,j_\sigma}} = C_{\sigma,j_\sigma}^{R_\sigma \cdot R_{\overline{\sigma}}} \cdot h^{-R_{\overline{\sigma}} \cdot R_\sigma \cdot r_{\sigma,j_\sigma}} \qquad (1) \\ &= g^{v_{\sigma,j_\sigma} \cdot R_\sigma \cdot R_{\overline{\sigma}}} \cdot h^{r_{\sigma,j_\sigma} \cdot R_\sigma \cdot R_{\overline{\sigma}}} \cdot h^{-R_{\overline{\sigma}} \cdot R_\sigma \cdot r_{\sigma,j_\sigma}} \\ &= g^{v_{\sigma,j_\sigma} \cdot R_\sigma \cdot R_{\overline{\sigma}}} \\ X'_{\overline{\sigma}} &= X_{\overline{\sigma}}^{R_\sigma} \cdot Z^{-R_\sigma} = C_{\overline{\sigma},j_{\overline{\sigma}}}^{R_\sigma \cdot R_{\overline{\sigma}}} \cdot h^{-R_\sigma \cdot R_{\overline{\sigma}} \cdot r_{\overline{\sigma},j_{\overline{\sigma}}}} \qquad (2) \\ &= g^{v_{\overline{\sigma},j_{\overline{\sigma}}} \cdot R_\sigma \cdot R_{\overline{\sigma}}} \cdot h^{r_{\overline{\sigma},j_{\overline{\sigma}}} \cdot R_\sigma \cdot R_{\overline{\sigma}}} \cdot h^{-R_\sigma \cdot R_{\overline{\sigma}} \cdot r_{\overline{\sigma},j_{\overline{\sigma}}}} \\ &= g^{v_{\overline{\sigma},j_{\overline{\sigma}}} \cdot R_\sigma \cdot R_{\overline{\sigma}}} \quad \square \end{aligned}$$

THEOREM 2. *Let Γ be the event that $X''_\sigma = X'_{\overline{\sigma}}$ but $v_{\sigma,j_\sigma} \neq v_{\overline{\sigma},j_{\overline{\sigma}}}$. The event Γ occurs with negligible probability.*

PROOF. Let $v_{\sigma,j_\sigma} \neq v_{\overline{\sigma},j_{\overline{\sigma}}}$. Since R_σ and $R_{\overline{\sigma}}$ are chosen uniformly at random from $\mathbb{Z}_q^*$, both X''_σ and $X'_{\overline{\sigma}}$ are distributed uniformly at random in $\mathbb{Z}_q^*$. Let $|q| = \log_2 q$. Then the probability of event Γ occurring is $prob(\Gamma) \approx 2^{-|q|}$ which is negligible for sufficiently large q. $\square$

It should be noted that the probability $prob(\Gamma)$ remains negligible even if a polynomial number of comparisons are carried out.

3.5 Security Proof for VPET

In order to prove our VPET protocol secure, we follow the ideal-/real-world simulation paradigm which was reviewed in Section 2.2.1. In the following, we describe the construction of two ideal-world simulators SIM_0 and SIM_1, i.e., one simulator for each party. For the simulation of the comparison phase there are two choices: $\sigma = 0$ or $\sigma = 1$. However, except for the switched roles, the simulation for both choices is identical. Therefore, without loss of generality we fix $\sigma = 0$.

SIM_0 interacts with a potentially corrupted P_0^* as P_1 and acts as ideal world P_0 towards a TTP ($P_0^* \rightleftarrows SIM_0 \rightleftarrows TTP \rightleftarrows P_1$). SIM_1 interacts with a potentially corrupted P_1^* as P_0 and acts as ideal world P_1 towards TTP ($P_0 \rightleftarrows TTP \rightleftarrows SIM_1 \rightleftarrows P_1^*$).

3.5.1 Construction of Ideal-World Simulator SIM_0

Commitment Phase.

1. SIM_0 receives $\left\langle \left(k_{0,j_0}, \left(C_{0,j_0}, \alpha_{0,j_0}, \pi_{C_{0,j_0}}\right)\right)\right\rangle$ from P_0^*.
2. SIM_0 acts as verifier for proofs $\pi_{C_{0,j_0}}$. If any proof $\pi_{C_{0,j_0}}$ does not verify, abort. Else, extract (r_{0,j_0}, v_{0,j_0}) from the interaction with P_0^*.
3. SIM_0 chooses random $C_{1,j_1}, \alpha_{1,j_1}$, simulates proofs $\pi_{C_{1,j_1}}$ and sends $\left\langle \left(k_{1,j_1}, \left(C_{1,j_1}, \alpha_{1,j_1}, \pi_{C_{1,j_1}}\right)\right)\right\rangle$ to P_0^*.
4. SIM_0 interacts with the TTP and commits to the extracted inputs (k_{0,j_0}, v_{0,j_0}): $(\texttt{commit}, (k_{0,j_0}, v_{0,j_0})) \rightarrow \mathcal{F}_{VPET}$ $\forall j_0$.

Comparison Phase for Values v_{0,j_0} and v_{1,j_1}.

1. SIM_0 interacts with the TTP to perform the comparison: $out \leftarrow (\texttt{compare}, (v_{0,j_0}, k_{0,j_0}, k_{1,j_1}, P_0))$
   - If $out = 1$, the values compared are equal and SIM_0's input has to be fixed as follows: Rewind P_0^* (using the same random tape) to the point just before SIM_0 sent its commitment vector. SIM_0 then chooses $r_{1,j_1} \leftarrow_r \mathbb{Z}_q^*$ and fixes $C_{1,j_1} = g^{v_{0,j_0}} \cdot h^{r_{1,j_1}}$. SIM_0 remembers that the value corresponding to k_{1,j_1} has been fixed and uses it.
   - If $out = 0$, the values compared are not equal and SIM_0 continues using the random dummy input chosen in step 3 of the commitment phase.
2. SIM_0 receives X_0 and interacts with P_0^* as verifier in proof π_0.
3. If π_0 does not verify, abort. Else, extract R_0 from the interaction with P_0^*.
4. SIM_0 then follows the remainder of the protocol specification and finally outputs $\perp$.

Note that rewinding is required at most $n_1 + 1$ times, since the initial input map requires at most n_1+1 fixes. The simulator therefore runs in expected polynomial time. Since the distribution of all messages exchanged during the interaction with the simulator is identical to the distribution in the real protocol, the steps above describe a perfect simulation.

3.5.2 Construction of Ideal-World Simulator SIM_1

Commitment Phase.

1. SIM_1 chooses random $C_{0,j_0}, \alpha_{0,j_0}$, simulates proofs $\pi_{C_{0,j_0}}$ and sends $\left\langle \left(k_{0,j_0}, \left(C_{0,j_0}, \alpha_{0,j_0}, \pi_{C_{0,j_0}}\right)\right)\right\rangle$ to P_1^*.
2. SIM_1 receives $\left\langle \left(k_{1,j_1}, \left(C_{1,j_1}, \alpha_{1,j_1}, \pi_{C_{1,j_1}}\right)\right)\right\rangle$ from P_1^*.
3. SIM_1 acts as verifier for proofs $\pi_{C_{1,j_1}}$. If any proof $\pi_{C_{1,j_1}}$ does not verify, abort. Else, extract (r_{1,j_1}, v_{1,j_1}) from the interaction with P_1^*.
4. SIM_1 interacts with the TTP and commits to the extracted inputs (k_{1,j_1}, v_{1,j_1}): $(\texttt{commit}, (k_{1,j_1}, v_{1,j_1})) \rightarrow \mathcal{F}_{VPET}\ \forall j_1$.

Comparison Phase for Values v_{0,j_0} and v_{1,j_1}.

1. SIM_1 interacts with the TTP to perform the comparison: $out \leftarrow (\texttt{compare}, (v_{1,j_1}, k_{1,j_1}, k_{0,j_0}, P_0))$
2. Since $out = \perp$, P_1^* will receive no output and no rewinding is required to fix SIM_1's input.
3. SIM_1 picks random $R_0 \leftarrow_r \mathbb{Z}_q^*$, computes X_0 according to the protocol and sends it together with a simulated proof π_0 to P_1^*.
4. SIM_1 receives $X_1', X_1, Y_1, Z_1, \pi_{1,1}, \pi_{1,2}$.

Since the distribution of all messages exchanged during the interaction with the simulator is identical to the distribution in the real protocol, the steps above describe a perfect simulation.

4. PROS PROTOCOL WITH SECURITY IN THE MALICIOUS MODEL

In this section we demonstrate how VPET can be used as a building block to construct higher-level protocols. In particular, we introduce a new protocol for PROS that exhibits security in the malicious model.

4.1 Related Work on PROS

In [38], Meyer et al. extend on the concept of PSI by allowing parties to associate preferences with their set elements. In the following, such sets will be referred to as ordered sets and the position of an element within the ordered set is its *preference* or *rank*. Based on the additional information represented by an element's preference, Meyer et al. introduced protocols for *privacy-preserving reconciliation of ordered sets* (PROS) in the semi-honest model. PROS enables both parties to jointly compute one common set element which maximizes the parties' combined preferences in a privacy-preserving and unbiased manner. In this context, *unbiasedness* means that each party's preferences are taken into consideration when computing the result.

PROS protocols can be used in the context of various applications, including voting, auctions, and scheduling [35]. However, PROS protocols known to date (both in the two-party and in the multi-party setting) only provide weak security guarantees, i.e., are secure in the presence of semi-honest adversaries [38, 39, 44]. While this is sufficient in scenarios where the parties have some degree of trust, arbitrary settings pose the challenge that parties may behave maliciously.

4.1.1 Preference Order Composition Schemes

In the following we assume party P_0 (resp. P_1) holding an ordered set $S_0 = \langle v_{0,1}, \ldots, v_{0,n}\rangle$ ($S_1 = \langle v_{1,1}, \ldots, v_{1,n}\rangle$) with $|S_0| = |S_1| = n$ as its input. The set elements are assumed to be sorted in decreasing order of the party's preference. Given S_0 and S_1, PROS allows the parties to determine one common set element such that it maximizes a *combined preference order* $\leq_{0,1}$ under a *Preference Order Composition Scheme* (POCS). Meyer et al. introduce two POCS: *Sum of Ranks* and *Minimum of Ranks* [38, 39].

Definition 1. *Preference Order Composition Schemes*
Sum of Ranks: $\forall x, y \in S_0 \cap S_1 : \quad x \leq_{0,1} y :\Leftrightarrow rank_0(x) + rank_1(x) \leq rank_0(y) + rank_1(y)$

Minimum of Ranks: $\forall x, y \in S_0 \cap S_1 : \quad x \leq_{0,1} y :\Leftrightarrow min(rank_0(x), rank_1(x)) \leq min(rank_0(y), rank_1(y))$

where $rank_0(x)$ and $rank_0(y)$ ($rank_1(x)$ and $rank_1(y)$) correspond to the preference of x and y in set S_0 (S_1).

Given a POCS, the PROS protocols in [38] proceed by pairwise comparing elements in decreasing order as dictated by the POCS. Each comparison is carried out using Freedman's PSI protocol [23]. When two elements are found to be equal, the protocol terminates and both parties learn the resulting set element.

Notation: To denote a specific pair of elements $(v_{0,i}, v_{1,j})$ to be compared in step k of a specific POCS C, we define a function which returns the corresponding pair of indices (i, j) for a given step $1 \leq k \leq n^2$: $(i, j) \leftarrow \texttt{get\_pocs\_pair}_C(k)$. This function allows the devising of PROS protocols independently of a specific POCS.

Party P_0 with Input: $S_0 = \langle v_{0,1}, \ldots, v_{0,n} \rangle$
Party P_1 with Input: $S_1 = \langle v_{1,1}, \ldots, v_{1,n} \rangle$

Commitment to Ordered Input Sets

for all $v_{0,i} \in S_0$:
$(\texttt{commit}, (i, v_{0,i})) \rightarrow \mathcal{F}_{\text{VPET}}$
for all $v_{1,i} \in S_1$:
$(\texttt{commit}, (i, v_{1,i})) \rightarrow \mathcal{F}_{\text{VPET}}$

Comparison Phase

for $1 \leq k \leq n^2$:

1. $(i, j) \leftarrow \texttt{get\_pocs\_pair}_C(k)$
2. P_0 sends $(\texttt{compare}, (v_{0,i}, i, j, 0)) \rightarrow \mathcal{F}_{\text{VPET}}$ and P_1 sends $(\texttt{compare}, (v_{1,j}, j, i, 0)) \rightarrow \mathcal{F}_{\text{VPET}}$. P_0 receives $out_0 \leftarrow \mathcal{F}_{\text{VPET}}$
3. P_0 sends $(\texttt{compare}, (v_{0,i}, i, j, 1)) \rightarrow \mathcal{F}_{\text{VPET}}$ and P_1 sends $(\texttt{compare}, (v_{1,j}, j, i, 1)) \rightarrow \mathcal{F}_{\text{VPET}}$. P_1 receives $out_1 \leftarrow \mathcal{F}_{\text{VPET}}$
4. - If $out_0 = out_1 = 1$, then P_0 outputs $v_{0,i}$, P_1 outputs $v_{1,j}$, and both terminate the protocol.
   - Else, both parties continue with the next iteration of the loop.

Figure 4: PROS with security in the malicious model.

4.2 Intuition for Our New Protocol

In principle, our new PROS protocol operates very similar to the protocol by Meyer et al. [38]. Specifically, the protocol successively compares two set elements, one from each ordered set, according to the POCS C. In PROS both parties are supposed to learn the result of the computation, i.e., the protocol for element comparison must be mutual. The PROS protocol introduced in [38] used PSI to compare elements. In order to achieve that PROS is mutual in this case, the PSI is carried out twice with the roles switched for the second execution. This approach only provides for security in the semi-honest model as there is no guarantee that the parties use the same input sets in both executions. However, in order to achieve stronger security guarantees, i.e., security in the malicious model, the technique used to compare two elements has to be modified significantly.

First, each party is required to commit to their ordered input set such that set elements and their order can not be changed in the course of the protocol. Second, whenever the protocol requires the comparison of two specific set elements, both parties need to prove that they use the correct set elements according to the POCS. Third, both parties need to prove that the actual comparison is performed correctly.

In the following we show how these challenges can be met by using the VPET protocol as a building block.

4.3 New Protocol for PROS

The new PROS protocol is detailed in Figure 4. In our protocol description we leverage the hybrid model (see Section 2.2.2) to allow for calls to the ideal VPET functionality. First, both parties commit to their respective ordered input sets by sending $(\texttt{commit}, (i, v_{0,i}))$ $\forall v_{0,i} \in S_0$ ($(\texttt{commit}, (i, v_{1,i}))$ $\forall v_{1,i} \in S_1$) to $\mathcal{F}_{VPET}$. Second, analogous to the protocol in [38], our protocol operates in multiple steps to compute the optimal result.

In each step, two set elements are compared through the comparison functionality provided by $\mathcal{F}_{VPET}$. Let $(v_{0,i}, v_{1,j})$ with $(i, j) \leftarrow \texttt{get\_pocs\_pair}_C(k)$ be the set elements which are compared in step k. In order for both parties to learn the result of the comparison, $\mathcal{F}_{VPET}$ is invoked twice: once with P_0 and once with P_1 learning the result. For this, first P_0 sends $(\texttt{compare}, (v_{0,i}, i, j, 0))$ and P_1 sends $(\texttt{compare}, (v_{1,j}, j, i, 0))$ to $\mathcal{F}_{VPET}$. Subsequently, P_0 sends $(\texttt{compare}, (v_{0,i}, i, j, 1))$ and P_1 sends $(\texttt{compare}, (v_{1,j}, j, i, 1))$. The second invocation compares the same elements as the first, however P_1 instead of P_0 learns the result. Note that $\mathcal{F}_{VPET}$ guarantees that the correct elements are used by both parties and, in addition, that the same elements are used in both comparisons.

If two elements are found to be equal, both parties output the resulting element and halt. Otherwise, the protocol continues with the next step, comparing the next pair of elements based on the POCS.

Signature	Computation	Verification
ZK_{DL}	1	2
ZK_{EDL}	2	4
ZK_{3EDL} [3]	3	6
ZK_{REP}	3	5

[3] Compares the discrete log of three elements simultaneously.

Table 1: Modular exponentiations required for computation and verification of ZK signatures of knowledge.

4.4 Security Proof

Since our novel PROS protocol consists exclusively of calls to the ideal VPET functionality, security in the hybrid model directly follows from the sequential modular composition theorem (see Section 2.2.2). When the calls to $\mathcal{F}_{VPET}$ are replaced with the actual VPET protocol, the PROS protocol remains secure in the presence of malicious adversaries.

4.5 Limitations and Improvements

In order for the POCS introduced in Section 4.1.1 to guarantee unbiasedness, the input sets of both parties must be of equal size. In our protocol this can be enforced by requiring that the respective commitment vectors are of equal size.

One limitation of our PROS protocol is that one can not enforce that a party's ordered input set contains unique elements, i.e., that the set is well-formed. This assurance could be achieved by introducing additional costly zero-knowledge proofs.

In order to improve the balance and unbiasedness of the protocol, it is possible to first flip a coin at the beginning of each step to determine which party will learn the output first.

5. PROTOCOL PERFORMANCE

In this section we present a detailed performance analysis of our new VPET and PROS protocols. In Section 5.1, we first analyze the theoretical complexity for the VPET

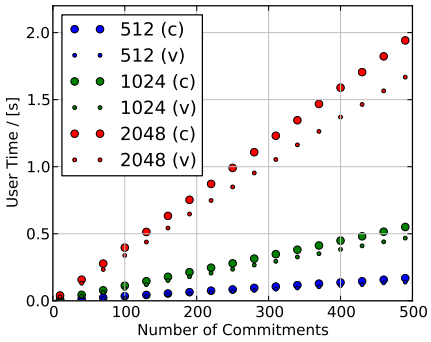

Figure 5: VPET user time for computation (c), verification (v) as a function of the number of commitments for $\log_2 p = 512, 1024, 2048$; $\log_2 q = 224$.

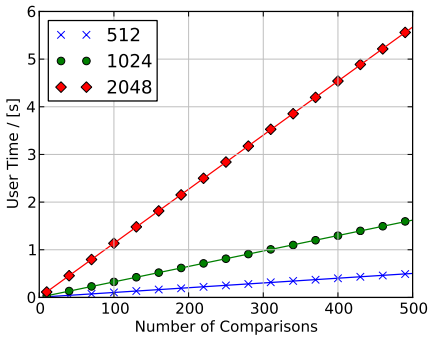

Figure 6: VPET user time for party P_σ as a function of the number of comparisons using $\log_2 p = 512, 1024, 2048$ and $\log_2 q = 224$.

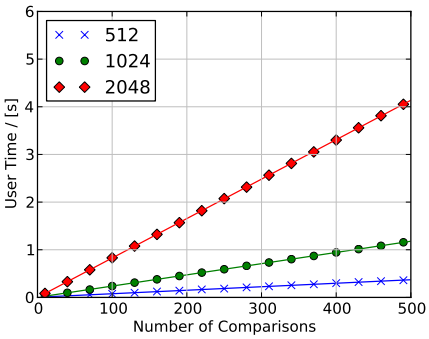

Figure 7: VPET user time for $P_{\overline{\sigma}}$ as a function of the number of comparisons using $\log_2 p = 512, 1024, 2048$ and $\log_2 q = 224$.

and PROS protocols. We then provide the results of our experimental performance evaluation in Section 5.2.

5.1 Theoretical Performance Analysis

5.1.1 ZK Signatures of Knowledge

A summary of the number of modular exponentiations required to compute and verify the respective signatures of knowledge is provided in Table 1.

5.1.2 VPET Protocol

Commitment Phase.

For each value in their respective input sets, parties P_0 and P_1 compute three exponentiations for the commitments and three exponentiations for ZK_{REP}. To verify the other party's ZK_{REP} an additional five exponentiations are required for each set element. Overall the commitment to n values requires a total of $11n$ modular exponentiations.

Comparison of Elements.

P_σ performs one modular exponentiation to compute X_σ, one exponentiation to compute ZK_{DL}, six to verify ZK_{3EDL}, four to verify ZK_{EDL}, two to compute $X'_{\overline{\sigma}}$, two to compute X''_σ, for an overall total of 16 modular exponentiations.

$P_{\overline{\sigma}}$ computes two exponentiations to verify ZK_{DL}, four in total to compute $X'_\sigma, X_{\overline{\sigma}}, Y_{\overline{\sigma}}, Z_{\overline{\sigma}}$, three to compute ZK_{3EDL}, two to compute ZK_{EDL}, for an overall total of 11 modular exponentiations.

5.1.3 Worst-Case Performance of PROS

Both parties commit to their respective input sets which requires $11n$ exponentiations (see above). In the worst case, both POCS introduced in [38] (i.e., Sum of Ranks and Minimum of Ranks) perform n^2 comparisons. In each step k of the POCS, two comparisons are carried out (one in which $P_\sigma = P_0$ and $P_{\overline{\sigma}} = P_1$ and vice versa). Therefore, in each step both P_0 and P_1 each perform $16 + 11 = 27$ modular exponentiations. Thus, the overall worst-case complexity of our new protocol is $O(n^2) = O(27n^2 + 11n)$ exponentiations.

5.2 Experimental Performance Analysis

5.2.1 Implementation Details

Our new VPET and PROS protocols as well as all subprotocols such as zero-knowledge proofs and commitment schemes were implemented in C++. Furthermore, our implementation leverages the well-known and efficient GNU Multiple Precision Arithmetic (GMP) Library 5.0.2 to allow for multi-precision integer computations as they are required for reasonable key sizes [3].

The network interface uses UNIX TCP sockets for which Nagle's algorithm [33, pp. 815-816] was disabled to avoid performance degradation due to transmission delays.

5.2.2 Test Environment

All performance tests were performed on a number of servers with identical hardware configurations. Each server has two Intel Xeon E5440 CPUs at 2.83GHz, 8GB main memory and runs a 64-bit Linux. The code for both parties of the protocols was executed on the same physical machine. Since all of our test systems are equipped with two CPUs, the interference of the two processes is assumed to be negligible.

5.2.3 VPET Performance

The size of all values in the maps was fixed to 32 bits.

Commitment and Comparison Phases.

The commitment and comparison phases of the VPET protocol were analyzed independently. Figure 5 shows the performance results for the commitment phase using a modulus of size $\log_2 p = 512, 1024, 2048$ and $\log_2 q = 224$. The map size was varied from 10 to 500 in steps of 10. For each map size the user time, i.e., the time the process was actually allocated to the CPU, for computing and verifying a commitment was recorded separately and averaged over 100 randomly generated maps. As expected, the time for computation and verification of commitments increases linearly with the map size. The average time for computing a single commitment (resp. verification) is 0.4 ms (0.3 ms) for 512-bit, 1.1 ms (1.0 ms) for 1024-bit, and 4.0 ms (3.4 ms) for 2048-bit moduli. The smaller times for the verification are

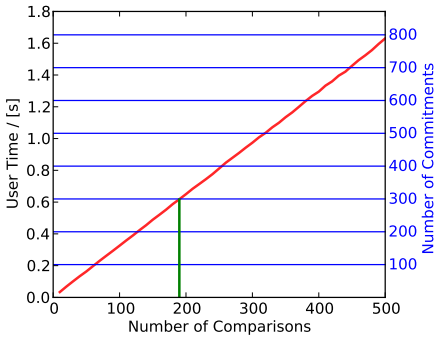

Figure 8: The horizontal lines mark the user time required for 100, . . . , 800 commitments. The graph indicates the user time as a function of comparisons.

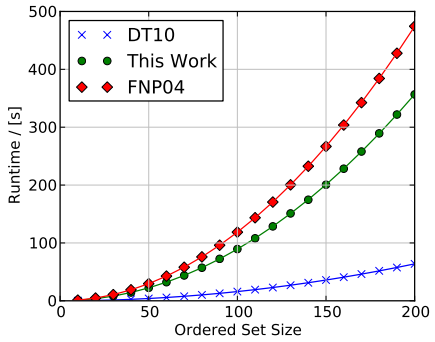

Figure 9: PROS worst-case runtime as a function of ordered set size using 1024-bit moduli. Quadratic behavior for all three protocols.

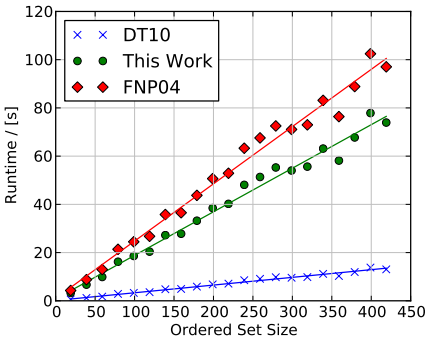

Figure 10: PROS average-case runtime as a function of ordered set size using 1024-bit moduli and fraction of 0.05 common elements in the ordered sets.

in agreement with the theoretical analysis in Section 5.1.

Using the same parameters as before, Figures 6 and 7 show the results for the comparison phase of VPET for P_σ and $P_{\overline{\sigma}}$ respectively. As expected, the user time for both parties linearly increases with the number of comparisons. On average, the time for a single comparison in case of P_σ (resp. $P_{\overline{\sigma}}$) is 1.0 ms (0.7 ms) for 512-bit, 3.3 ms (2.4 ms) for 1024-bit, and 11.3 ms (8.3 ms) for 2048-bit moduli. Again, the smaller times for $P_{\overline{\sigma}}$ are in agreement with the theoretical analysis in Section 5.1.

Commitment Amortization.

Figure 8 directly relates the complexity of commitments to the complexity of comparisons. The horizontal lines indicate the user time (as marked on the left axis) required for 100, . . . , 800 commitments (as marked on the right axis). The graph itself corresponds to the user time for a varied number of comparisons (10 to 500 in steps of 10). Note that for two input maps of size n there are n^2 distinct pairs of values which could be compared. Figure 8 illustrates the break-even point for which the time spent for commitments equals that for comparisons. For example, for 300 commitments less than 200 comparisons are required to break even. Moreover, 300 commitments allow for 90,000 distinct comparisons and the overhead for the commitment phase therefore only constitutes a small fraction of the overall runtime of VPET.

5.2.4 PROS Performance

For all tests, the size of all set elements was 32 bits.

Worst-Case.

In the previous section we have shown that the VPET protocol exhibits a practical behavior that matches the theoretical prediction. Using our VPET implementation as a building block, we constructed the PROS protocol as described in Section 4. In our experiments we compare the performance of our new PROS protocol to the PROS protocols with security in the semi-honest model as introduced in [39, 36]. For comparing two set elements, these semi-honest protocols utilize the PSIs introduced in [23] (FNP04) and [20] (DT10).

Figure 9 shows the worst-case performance results for all three PROS protocol variants using the Sum of Ranks POCS and 1024-bit moduli. The cardinality of the ordered sets was varied from 10 to 200 in steps of 10 such that the input sets for both parties were disjoint (i.e., to force worst-case behavior). For each set size, the runtime was averaged over 10 randomly generated sets.

As predicted and shown in [38, 36], all protocols exhibit a quadratic behavior in the worst-case. It is clear that the efficient DT10 PSI enables the construction of an efficient PROS protocol in the semi-honest model. Compared to DT10, our protocol with security in the malicious model exhibits an approximate overhead factor of 5.6. It is important to note that the requirement to do each comparison twice to make our new protocol mutual already amounts to an overhead factor of two. In conclusion, VPET enables the efficient construction of mutual higher-level protocols in the malicious model.

Average-Case.

For our performance tests we defined the average-case input as both parties having a fraction of set elements in common. These common elements may be at random positions within the ordered sets. The ordered sets for both parties were generated such that both have exactly 5% of their elements in common. To obtain the results in Figure 10, the set size was varied from 20 to 420 in steps of 20. For each set size, the runtime was averaged over 100 randomly generated sets. Similar to the results in [36], the average case shows a linear complexity. Compared to the worst-case, the runtime for all three PROS protocols is significantly reduced. As expected, the overhead factor of our construction compared to the DT10-based variant remains at approximately 5.6.

6. RELATED WORK

The first solution for PSI was proposed by Freedman et al. based on oblivious polynomial evaluation and homomorphic encryption [23]. Freedman's protocol provides security in the presence of semi-honest adversaries. It was later improved with security against malicious adversaries by several authors [12, 17, 29].

A different approach, which resulted in protocols that are

more efficient than the ones based on oblivious polynomial evaluation, is *oblivious pseudo-random function* (OPRF) evaluation. The applicability of OPRF to the PSI problem was first suggested in [24]. Hazay and Lindell were the first to introduce efficient protocols, one with security in the presence of covert adversaries and one with security against malicious servers and semi-honest clients [27]. More recently, the OPRF approach was extended to exhibit active security in the common reference string model [30]. Finally, a similar idea based on an unpredictable function was proposed in [31].

Recently, De Cristofaro et al. presented a novel approach for PSI based on the comparison of blinded values [20, 19]. Their solution has linear complexity and the authors propose protocols with security in the semi-honest as well as in the malicious model. The blinding technique applied in [19] is similar to the one used in our construction of the VPET protocol. However, their approach does not involve commitments and it therefore does not enable proofs about the correct usage of inputs.

Similarly, a variety of protocols providing different security guarantees have been proposed for PET. Fagin et al. have surveyed multiple techniques which can be used to determine whether two people are thinking about the same person [21]. In [43], Naor et al. present a solution for PET using oblivious polynomial evaluation which is based on oblivious transfer. Boudot et al. approached the problem with the goal of designing a fair, i.e., mutual, solution. Their protocol provides security under the discrete logarithm assumption [7]. More recently, Damgard et al. proposed a protocol based on a bit-decomposition scheme [18] which was later improved on by Nishide et al. [45]. In [32], Murat et al. utilize threshold homomorphic encryption and zero knowledge proofs to construct a PET protocol with security in the malicious model.

Finally, Lipmaa et al. propose a verifiable homomorphic oblivious transfer protocol which is sender-private [34]. Their protocol enables a sender to commit to all elements in its database. Subsequently, these values then can be related to other values used in the computation.

7. FUTURE WORK

Future work includes the enhancement of VPET to enable arbitrary comparisons which are not limited to equality tests. In addition, we plan to extend our new VPET and PROS protocols to the multi-party setting.

Acknowledgment

This work was supported by NSF Award CCF 1018616.

8. REFERENCES

[1] C. Aggarwal and P. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Advances in Database Systems. Springer, 2008.

[2] R. Agrawal, A. Evfimievski, and R. Srikant. Information Sharing Across Private Databases. In *ACM Management of Data (SIGMOD)*, pages 86–97, New York, NY, USA, 2003. ACM.

[3] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. NIST Special Publication 800-57: Recommendation for Key Management - Part 1. Technical report, National Institute of Standards and Technology, 05 2011.

[4] D. Beaver. Foundations of Secure Interactive Computing. In *Advances in Cryptology (CRYPTO)*, volume 576 of *LNCS*, pages 377–391. Springer Berlin / Heidelberg, 1992.

[5] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security (CCS)*, pages 62–73, New York, NY, USA, 1993. ACM.

[6] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, et al. Secure Multiparty Computation Goes Live. In *Financial Cryptography and Data Security (FC)*, volume 5628 of *LNCS*, pages 325–343. Springer, 2009.

[7] F. Boudot, B. Schoenmakers, and J. Traore. A Fair and Efficient Solution to the Socialist Millionaires Problem. *Discrete Applied Mathematics*, 111(1-2):23 – 36, 2001.

[8] S. Brands. An Effcient Off-line Electronic Cash System Based On The Representation Problem. CWI Technical Report CS-R9323, 1993.

[9] S. Brands. Untraceable Off-line Cash in Wallet with Observers. In *Advances in Cryptology (CRYPTO)*, volume 773 of *LNCS*, pages 302–318. Springer Berlin / Heidelberg, 1994.

[10] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology (CRYPTO)*, volume 1294 of *LNCS*, pages 410–424. Springer Berlin / Heidelberg, 1997.

[11] J. Camenisch and M. Stadler. Proof Systems for General Statements About Discrete Logarithms. Technical report, 1997.

[12] J. Camenisch and G. Zaverucha. Private Intersection of Certified Sets. In *Financial Cryptography and Data Security (FC)*, volume 5628 of *LNCS*, pages 108–127. Springer Berlin / Heidelberg, 2009.

[13] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13:143–202, 2000.

[14] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[15] D. Chaum and T. Pedersen. Wallet Databases with Observers. In *Advances in Cryptology (CRYPTO)*, volume 740 of *LNCS*, pages 89–105. Springer Berlin / Heidelberg, 1993.

[16] R. Cramer, I. Damgård, and J. Nielsen. Multiparty Computation, an Introduction, 2009.

[17] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Efficient Robust Private Set Intersection. In *Applied Cryptography and Network Security (ACNS)*, volume 5536 of *LNCS*, pages 125–142. Springer, 2009.

[18] I. Damgard, M. Fitzi, E. Kiltz, J. Nielsen, and T. Toft. Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation. In *Theory of Cryptography (TCC)*, volume 3876 of *LNCS*, pages 285–304. Springer Berlin / Heidelberg, 2006.

[19] E. De Cristofaro, J. Kim, and G. Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *Advances in Cryptology (ASIACRYPT)*, volume 6477 of *LNCS*, pages 213–231. Springer Berlin / Heidelberg, 2010.

[20] E. De Cristofaro and G. Tsudik. Practical Private Set Intersection Protocols with Linear Complexity. In *Financial Cryptography and Data Security (FC)*, volume 6052 of *LNCS*, pages 143–159. Springer, 2010.

[21] R. Fagin, M. Naor, and P. Winkler. Comparing Iinformation Without Leaking It. *Communications of the ACM*, 39:77–85, May 1996.

[22] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge Proofs of Identity. *Journal of Cryptology*, 1:77–94, 1988.

[23] M. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Advances in Cryptology (EUROCRYPT)*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.

[24] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword Search and Oblivious Pseudorandom Functions. In *Theory of Cryptography (TCC)*, volume 3378 of *LNCS*, pages 303–324. Springer Berlin / Heidelberg, 2005.

[25] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge University Press, 2009.

[26] S. Goldwasser and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. In *Advances in Cryptology (CRYPTO)*, volume 537 of *LNCS*, pages 77–93. Springer Berlin / Heidelberg, 1991.

[27] C. Hazay and Y. Lindell. Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. *Journal of Cryptology*, 23(3):422–456, 2008.

[28] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer, 2010.

[29] C. Hazay and K. Nissim. Efficient Set Operations in the Presence of Malicious Adversaries. In *Public Key Cryptography (PKC)*, volume 6056 of *LNCS*, pages 312–331. Springer, 2010.

[30] S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *Theory of Cryptography (TCC)*, volume 5444 of *LNCS*, pages 577–594. Springer, 2009.

[31] S. Jarecki and X. Liu. Fast Secure Computation of Set Intersection. In *Security and Cryptography for Networks (SCN)*, volume 6280 of *LNCS*, pages 418–435. Springer Berlin / Heidelberg, 2010.

[32] M. Kantarcioglu and O. Kardes. Privacy-Preserving Data Mining in the Malicious Model. *International Journal of Information and Computer Security*, 2(4):353 – 375, 2008.

[33] C. M. Kozierok. *The TCP/IP Guide*. No Starch Press, 2005.

[34] H. Lipmaa. Verifiable Homomorphic Oblivious Transfer and Private Equality Test. In *Advances in Cryptology (ASIACRYPT)*, volume 2894 of *LNCS*, pages 416–433. Springer Berlin / Heidelberg, 2003.

[35] D. A. Mayer, G. Neugebauer, U. Meyer, and S. Wetzel. Enabling Fair and Privacy-Preserving Applications Using Reconciliation Protocols on Ordered Sets. In *Sarnoff Symposium*. IEEE, 2011.

[36] D. A. Mayer, D. Teubert, S. Wetzel, and U. Meyer. Implementation and Performance Evaluation of Privacy-Preserving Fair Reconciliation Protocols on Ordered Sets. In *Conference on Data and Application Security and Privacy (CODASPY)*, pages 109–120. ACM, 2011.

[37] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[38] U. Meyer, S. Wetzel, and S. Ioannidis. Distributed Privacy-Preserving Policy Reconciliation. In *IEEE International Conference on Communications (ICC)*, pages 1342–1349. IEEE, 2007.

[39] U. Meyer, S. Wetzel, and S. Ioannidis. New Advances on Privacy-Preserving Policy Reconciliation. In *Cryptology ePrint Archive, Report 2010/64*, 2010. http://eprint.iacr.org/2010/064.

[40] S. Micali and P. Rogaway. Secure Computation (Abstract). In *Advances in Cryptology (CRYPTO)*, pages 392–404, London, UK, 1992. Springer-Verlag.

[41] M. Milian. Two Lawsuits Target Apple, App Makers Over Privacy Concerns. `http://cnn.com/2010/TECH/mobile/12/28/apple.app.lawsuits/index.html`, December 2010.

[42] A. Miyaji and M. Rahman. Privacy-Preserving Data Mining in Presence of Covert Adversaries. In *Advanced Data Mining and Applications (ADMA)*, volume 6440 of *LNCS*, pages 429–440. Springer, 2010.

[43] M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Symposium on Theory of Computing (STOC)*, pages 245–254, New York, NY, USA, 1999. ACM.

[44] G. Neugebauer, U. Meyer, and S. Wetzel. Fair and Privacy-Preserving Multi-Party Protocols for Reconciling Ordered Input Sets. In *Information Security Conference (ISC)*, volume 6531 of *LNCS*, pages 136–151. Springer, 2010.

[45] T. Nishide and K. Ohta. Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In *Public Key Cryptography (PKC)*, volume 4450 of *LNCS*, pages 343–360. Springer Berlin / Heidelberg, 2007.

[46] T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology (CRYPTO)*, volume 576 of *LNCS*, pages 129–140. Springer Berlin / Heidelberg, 1992.

[47] J. Pepitone. Facebook Settles FTC Charges Over 2009 Privacy Breaches, 2011.

[48] B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure Two-Party Computation is Practical. In *Advances in Cryptology (ASIACRYPT)*, volume 5912 of *LNCS*, pages 250–267. Springer, 2009.

[49] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4:161–174, 1991.

[50] A. Yao. Protocols for Secure Computations. In *Foundations of Computer Science*, volume 23, pages 160–164. IEEE, 1982.