Multiparty Proximity Testing with Dishonest Majority from Equality Testing

Ran Gelles
gelles@cs.ucla.edu

Rafail Ostrovsky
rafail@cs.ucla.edu

Kina Winoto
kwinoto@ucla.edu

July 4, 2012

Abstract

Motivated by the recent widespread emergence of location-based services (LBS) over mobile devices, we explore efficient protocols for *proximity-testing*. Such protocols allow a group of friends to discover if they are all close to each other in some physical location, without revealing their individual locations to each other. We focus on hand-held devices and aim at protocols with very small communication complexity and a small number of rounds.

The proximity-testing problem can be reduced to the *private equality testing* (PET) problem, in which parties find out whether or not they hold the same input (drawn from a low-entropy distribution) without revealing any other information about their inputs to each other. While previous works analyze the 2-party PET special case (and its LBS application), in this work we consider highly-efficient schemes for the *multiparty* case with no honest majority. We provide schemes for both a direct-communication setting and a setting with a honest-but-curious mediating server that does not learn the users' inputs. Our most efficient scheme takes 2 rounds, where in each round each user sends only a couple of ElGamal ciphertexts.

1 Introduction

The ubiquity of mobile devices has led to the rise of *Location-Based Services* (LBS), services that depend on one's current location [Axe05]. An interesting location-based service is *proximity testing*, in which users determine if they are in proximity to one another. This has multiple applications. For instance, assume you and your friends visit a mall, identified by a GROUPON®-like company who sends you a group-coupon for, say, one of the mall's restaurants. While several products and applications based on proximity have recently appeared (such as magnetU®, etc.), a great deal of concern arises from privacy issues [LSHG08, GG03, ZGH07, KGMP07, ŠTŠ+09]. A recent work by Narayanan, Thiagarajan, Lakhani, Hamburg, and Boneh [NTL+11] provides a private scheme for proximity testing by reducing the problem to private equality testing (PET) in which each of the users A and B holds a private input out of a finite set $\mathcal{D}$ of possible inputs (X_A and X_B respectively), and the aim is to jointly perform a private computation of the predicate $X_A \stackrel{?}{=} X_B$, without revealing the private inputs. Reducing proximity-testing to PET allows us to focus on the latter and extend the known result to the multiparty case.

The PET task is a special (and more simple) case of the private set intersection problem (which is, in turn, a special case of secure multiparty computation (MPC)), in which each user holds a set of private inputs, and the protocol outputs all the elements shared between the users (in some variants, the protocol only outputs the size of the intersection). A PET scheme

can always be realized using a private set-intersection scheme if each user's set includes only his private input X_i. Since we focus on the special case of PET rather than the general set-intersection, we are able to achieve secure protocols with improved efficiency, specifically, less communication rounds.

Our Contributions. In this work we provide efficient, constant round protocols for n-party PET, secure against a coalition of up to $n-1$ malicious adversaries.

We begin by extending the 2-party schemes of Narayanan et al. [NTL+11] and Lipmaa [Lip03] to an arbitrary number of users, obtaining a 2-round n-party PET scheme that uses a mediating server, as well as a 2-round n-party PET scheme that does not rely on a mediating server.

We also consider security against *arbitrary* computationally-bounded adversaries. While it is relatively simple to design a scheme resistant to semi-honest adversaries, this is not the case with malicious adversaries. In addition to privacy issues, malicious adversaries sometimes jeopardize the soundness of the protocol. We show schemes that are secure against a malicious adversary (with abort). To the best of our knowledge, the schemes we present here are the first n-party PETs proven secure against malicious adversaries with no honest majority.[1] General purpose MPC for malicious majority also considers the notion of fairness [IOS12]. In this work we do not consider fairness: first, we allow aborts; further, only one party is supposed to learn the protocol's output (while other parties learn nothing).

Another contribution we describe in this paper regards the efficiency of 2-party proximity testing realized using PET. We follow a method introduced by Narayanan et al. [NTL+11] and later used by Tonicelli et al. [TDdMA11], which performs a quantization of one's locations (say, GPS position) into centers of hexagonal cells, and then runs several PET instances using the quantized locations (see discussion in [NTL+11]). By making more efficient choices of the quantization we are able to reduce the amount of PET instances performed during a single proximity testing. This improves both the computation and communication complexity by 13% to 44%, depending on the underlying scheme. Since proximity-testing applications are widely used in mobile devices, any decrease in the number of operations and the required bandwidth is important.

Overview of Techniques. Our first n-party PET scheme (based on [NTL+11]) assumes an honest-but-curious server, with the additional requirement that the server does not learn the users' inputs. Each party sends its input to the server, masked with a random one-time pad (OTP), known to all users except the server. Using the linearity of OTP, and using randomness shared with some of the users, the server is able to compute a message which is fixed if all the parties share the same private input, and uniformly distributed otherwise. The main insight is that if all private inputs are the same, then their sum is exactly n times the value of a single input. The server can add up the (masked) inputs, deduct n times Alice's (masked) input, and multiply the result with a random number. That way, the result is fixed (as a function of the mask) if all users had the same input, and random otherwise.

Our second scheme (based on ideas from [Lip03, FNP04, NTL+11]) does not require a trusted server. The users replace the role of the server by "summing" the inputs themselves. In order to keep the inputs private yet still be able to manipulate them, we replace the OTP with homomorphic encryptions. We also add a layer of secret sharing [BGW88] to prevent intermediate results from leaking information. Each private input is masked with a random share, however

[1] Excluding general-purpose MPC schemes.

the sum of all the random shares is 0 (or other value known to the parties), and thus does not affect the sum of the inputs. This scheme so far can only deal with semi-honest adversaries.

In order to obtain privacy against a malicious-adversary, it is common to use zero-knowledge proof-of-knowledge (ZKPoK) in which a prover convinces a verifier that he knows some secret, without leaking any information about the secret. When considering the multiparty case, the parties either conduct a 2-party ZKPoK sequentially, so the number of rounds becomes linear in the number of parties, or they perform multiple instances of a 2-party ZKPoK in parallel, which requires proving its security for parallel composition. To this end we use a variant which allows a single prover to perform a ZKPoK scheme with n verifiers, namely, a 1-Prover n-Verifier ZKPoK scheme. On top of being zero-knowledge, the prover must convince an honest verifier that he knows some secret, even if the rest of the verifiers collude with the prover to fool the honest verifier. This task can be performed in a constant number of rounds; in addition, our variant also reduces the communication—the prover broadcasts a single message to all the verifiers rather than engaging in n instances of the same protocol. We show how to construct a 1-Prover n-Verifier ZKPoK from any 2-party ZKPoK (a Σ-protocol) and a trapdoor commitment scheme. For concreteness, our PET scheme uses a 1-Prover n-Verifier ZKPoK for discrete log based on a Σ-Protocol by Schnorr [Sch91] and a trapdoor-commitment based on Pedersen's commitment [Ped92].

Our alternative PET scheme utilizes a different underlying primitive, namely, a password-authenticated key-exchange (PAKE) [KOY01, KOY09]. Specifically, we show how to realize a 2-round PET secure against malicious adversaries, using oracle access to a PAKE subroutine. Thus, with the existence of constant-round UC-secure PAKE schemes such as [CHK+05, ACCP08, HL11] we obtain a constant-round private PET. The idea is to use the PAKE in order to establish an encryption key, separately between each pair of users. If two users use the same password as an input to the PAKE protocol, they will end up sharing the same session key (otherwise their session keys will mismatch). The parties then use the keys to send random shares to each other, such that the sum of the shares is, say, 0. All these shares are then sent to Alice, who adds them up—the sum will be 0 if every pair of users used matching session keys (otherwise, the sum will be random).

Related Work. 2-party PET was inspired by Yao's *socialist millionaire*'s problem [Yao82], and examined for "real-life" problems by Fagin, Naor and Winkler [FNW96]. The work of Boudot, Schoenmakers and Traoré [BST01] gives a rigorous security proof for this problem in the *random oracle model*. The work of Naor and Pinkas [NP99] draws a connection between PET and Oblivious Transfer (OT). This connection is extended by Lipmaa [Lip03], providing private OT protocols based on homomorphic encryption, and showing how to realize an efficient (2-party) PET scheme using the same methods. Lipmaa also provides a security proof for PET in the semi-honest model. Tonicelli, Machado David and de Morais Alves [TDdMA11] present an efficient UC-secure scheme for a 2-party PET.

Damgård, Fitzi, Kiltz, Nielsen and Toft [DFK+06] present an unconditionally secure scheme (in the secret sharing model) for multiparty PET with $O(1)$ rounds (this translates to 7 rounds in the plain model), however their result requires an honest majority. Nishide and Ohta [NO07] extend this result and achieve a 5-round scheme (in the plain model), again, assuming honest majority.

2 Preliminaries

Let us begin by describing several notations and cryptographic primitives we use throughout. We let κ be the security parameter, and assume that any function, set size or running time implicitly depends on this parameter (especially when we write neg to describe a negligible function in κ, i.e., $neg < 1/poly(\kappa)$ for large enough κ). We say that two ensembles of distributions $\{X_\kappa\}_{\kappa\in\mathbb{N}}$, $\{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are *computationally indistinguishable* and write $\{X_\kappa\} \stackrel{c}{\equiv} \{Y_\kappa\}$ if for any probabilistic polynomial-time (PPT) algorithm C, for large enough κ,

$$|\Pr[C(1^\kappa, X_\kappa) = 1] - \Pr[C(1^\kappa, Y_\kappa) = 1]| < neg.$$

Semantically Secure Public-Key Encryption Schemes. A public-key encryption scheme consists of three PPT algorithms (GEN, ENC, DEC) such that $GEN(1^\kappa) = (pk, sk)$ generates a pair of a public and secret key, $ENC(m) = c$ encrypts a message m and $DEC(c') = m'$ decrypts a ciphertext c'.

An encryption scheme is said to be *semantically secure*, if for any PPT adversary Adv, and any two messages m_0, m_1 of equal length chosen by the adversary,

$$\Pr[\mathsf{Adv}(1^\kappa, pk, ENC_{pk}(m_b)) = b] < \frac{1}{2} + neg,$$

over the coin tosses of the algorithms and the uniform choice of $b \in \{0, 1\}$, where pk is the public key generated by $GEN(1^\kappa)$. See [Gol04] for the analog definition for semantically secure *private-key* encryptions systems.

Homomorphic Encryption. Informally, an encryption scheme $ENC : (m, r, k) \to ENC_k(m, r)$ that takes message m, randomness r and key k is *additively homomorphic* if

$$ENC_k(m_1, r_1) \cdot ENC_k(m_2, r_2) = ENC_k(m_1 + m_2, r_1 \circ r_2)$$

for some deterministic binary operation $\circ$. See e.g. [OS07] for a formal definition. We focus on an homomorphic variant of the ElGamal scheme [Elg85], which consists of a cyclic group G of order q in which the Decisional Diffie-Hellman (DDH) problem is difficult [Bon98], and a generator $g \in G$. A random number $x \in \mathbb{Z}_q^*$ is chosen as the private key, and the public key is $(g, h) = (g, g^x)$. To encrypt a message m with the above public key, one randomly picks $r \in \mathbb{Z}_q$ and computes $ENC_{(g,h)}(m, r) = (g^r, h^r \cdot h^m)$. The decryption of a ciphertext $c = (a, b)$ is given by $DEC_x(c) = b/a^x = h^m$. It is easy to verify that this variant is additively homomorphic. Note that the decryption outputs h^m rather than m, however, for a low-entropy $\mathcal{D}$, m can be found by searching the entire domain. The semantic security of the ElGamal scheme follows from the hardness of DDH in G.

Another homomorphic encryption system is the *Paillier Encryption system* [Pai99], whose security relies on the decisional composite residuosity assumption. Although our schemes are specified using the ElGamal system, they can be easily modified using variants of the Paillier encryption, or using any other semantically-secure additively homomorphic encryption.

Chosen-Plaintext Unforgeable (private-key) Encryption Schemes.
Let (GEN, ENC, DEC) be a private-key encryption scheme. We say that an encryption scheme is *chosen-plaintext unforgeable* [KY01] if, for any x, an adversary cannot forge a valid encryption

of x with non-negligible probability, even with oracle access to encryptions of any $x' \neq x$, Formally, we say that an encryption scheme is ε-chosen-plaintext unforgeable, if, for sk generated by $GEN(1^\kappa)$ and for any PPT adversary Adv,

$$\Pr[\mathsf{Adv}^{ENC_{sk}(\cdot)}(1^\kappa) = (x, y) \text{ s.t. } DEC_{sk}(y) = x] \leq \varepsilon,$$

over the coin tosses of all the algorithms. Note that Adv cannot query the oracle on x.

3 Model and Privacy Notions

We now define the ideal functionality of an asymmetric Private Equality Testing (PET) scheme. An n-party asymmetric-PET scheme is a distributed protocol between a specific party A (which we call Alice at times for convenience) and another $n-1$ parties, denoted $B_1, B_2, \ldots, B_{n-1}$. Each one of the users holds a private input $X_A, X_{B_1}, X_{B_2}, \ldots, X_{B_{n-1}}$, respectively. Sometimes, for ease of notation, we denote the private inputs as $\bar{X} = (X_A, X_1, X_2, \ldots, X_{n-1})$. At the end of the computation, Alice obtains the value

$$\mathcal{F}_{ideal}(X_A, X_1, \ldots, X_{n-1}) = \begin{cases} 1 & \text{if } X_A = X_1 = \cdots = X_{n-1} \\ 0 & \text{otherwise} \end{cases}$$

and the rest of the parties obtain $\perp$. That is, the protocol is asymmetric: Alice learns whether or not all the private inputs are equal, while the other parties learn nothing. From this point and on we refer to asymmetric-PET simply as PET.

We say that some realization π of a PET scheme is *complete* if it answers correctly for all positive instances, $\Pr[\pi(\bar{X}) = \mathcal{F}_{ideal}(\bar{X}) = 1] = 1$. Similarly, we say that a scheme is *ε-sound* if for any distribution over $\bar{X}$, the protocol answers affirmatively with probability at most ε for all negative instances, $\max_{\bar{X}} \Pr[\pi(\bar{X}) = 1 \wedge \mathcal{F}_{ideal}(\bar{X}) = 0] \leq \varepsilon$, over the randomness of the parties.

Adversaries. In this work we consider both semi-honest (also known as honest-but-curious) and malicious adversaries, using standard notions (see for instance [Gol04]). While in the *semi-honest adversary model* parties are assumed to follow the protocol, in the *malicious adversary model* the corrupted parties might deviate from the protocol as desired, as well as intercept and block messages from other users (excluding broadcast messages). The corruption model is static.

The trusted server is always honest (even in the malicious-adversary model) and never colludes with other parties. However, differently from the standard notion of trusted server we assume that the server is curious (passive adversary) and require that it cannot learn the honest users' inputs.

Privacy. The privacy model we consider here is a stand alone security against a coalition of up to $n-1$ adversaries (using simulation-based definition). Informally, for any distribution on $\bar{X}$, we would like the private inputs $\bar{X}$ to remain hidden, regardless of the adversary's actions. For instance, even if $n-1$ parties collude together, they should have no advantage in finding the input of the nth player, other than just guessing it (given the PET's output and the a-priory probability of that location). Before we formalize our notion of privacy, we distinguish two types of attacks: *on-line* and *off-line* [GLNS93]. In an *on-line* attack, the adversary actively participates in a PET scheme, while in an off-line attack, the adversary tries to extract data

from a PET instance's transcript, possibly by checking for inconsistencies between a given PET transcript and each one of the values in the (low-entropy) dictionary $\mathcal{D}$. Throughout this paper we assume $|\mathcal{D}|$ is polynomial in the security parameter.

We formalize the notion of privacy using standard notion of *simulatability* in the *real/ideal paradigm*. Namely, we require that for any PPT adversary Adv running π with any set of honest parties, there exists a PPT simulator Sim such that for any private inputs, Sim's output and Adv's view are computationally indistinguishable, yet Sim only has access to the ideal functionality $\mathcal{F}_{ideal}$,[2] (rewindable) black-box access to Adv, and the trusted setup parameters of the protocol.

Assume $P_1, \ldots, P_m$ are the honest parties $m \leq n$, where each P_i has a unique role $\mathsf{role}_i \in \{A, B_1, \ldots, B_n\}$ and a private input X_{P_i}. Denote by $View_{\bar{X}}(\mathsf{Adv} \Leftrightarrow \pi)$ the view of the adversary running π with $P_1, \ldots, P_m$, with inputs $\bar{X}$. The view contains all the messages and randomness of the corrupt parties. Denote by $Output_{\bar{X}}(\mathsf{Sim}^{\mathsf{Adv}} \Leftrightarrow \mathcal{F}_{ideal})$ the output of a simulator which has rewindable black-box access to Adv, running $\mathcal{F}_{ideal}$ with the honest parties, with inputs $\bar{X}$. Note that both the adversary and the simulator are given the inputs of the corrupt parties $X_{P_{m+1}}, \ldots, X_{P_n}$, but are oblivious to the inputs of the honest parties $X_{P_1}, \ldots, X_{P_m}$.

Definition 1. *We say that an Equality Testing protocol π is **private** if for any set of $m \leq n$ honest parties $P_1, \ldots, P_m$ acting as $\mathsf{role}_1, \ldots, \mathsf{role}_m$, with unique $\mathsf{role}_i \in \{A, B_1, \ldots, B_{n-1}\}$, and for any PPT Adv, there exists a PPT simulator Sim such that for any set of inputs $\bar{X}$*

$$View_{\bar{X}}(\mathsf{Adv} \Leftrightarrow \pi) \stackrel{c}{\equiv} Output_{\bar{X}}(\mathsf{Sim}^{\mathsf{Adv}} \Leftrightarrow \mathcal{F}_{ideal}) \qquad \text{if } A \text{ is corrupt, and}$$

$$View_{\bar{X}}(\mathsf{Adv} \Leftrightarrow \pi) \stackrel{c}{\equiv} Output_{\bar{X}}(\mathsf{Sim}^{\mathsf{Adv}}) \qquad \text{otherwise.}$$

Recall that all of our definitions use an implicit security parameter κ. Specifically, the above distributions are in fact the ensembles $\{View_{\bar{X}}(\mathsf{Adv} \Leftrightarrow \pi)\}_\kappa$ etc., and all the PPT machines run in polynomial time in κ.

In order to achieve a private PET, some assumptions are necessary. The following claim suggests that a minimal requirement for the honest parties is to resist impersonation attacks.

Claim 1. *Assuming messages can be blocked (by the adversary), a private PET scheme allows each user to identify the sender of a received message.*

To see this, assume a private PET with $m \geq 2$ honest parties in which an honest party, (wlog) P_1, cannot identify the origin of a received message. Let Adv run the PET scheme with P_1, acting as the other $n-1$ users (blocking the messages of the other honest users). Since P_1 cannot validate the origin of his received messages, he proceeds using Adv's messages as the legitimate messages. Then, Adv is able to guess X_{P_1} via a trivial guessing attack: he picks a value $x \in \mathcal{D}$ and runs the PET scheme, setting all the inputs to x. If the PET scheme returns 1, Adv outputs x; otherwise Adv outputs a uniform guess from $\mathcal{D} - \{x\}$. This attack succeeds with probability at least $2/|\mathcal{D}|$ (assuming a uniform distribution on $\mathcal{D}$), regardless of the inputs of the other honest parties, which contradicts the privacy of the scheme.

From this point and on, we assume the users communicate using an authenticated channel.

[2] Note that due to the asymmetric nature of the protocol, Sim has access to $\mathcal{F}_{ideal}$ only when it simulates A.

The f-Hybrid Model. Assume f is some arbitrary subroutine used as part of the protocol π. In order to focus on the analysis of π yet "hide" the details of f's implementation, we use the *f-hybrid model*. In this model we assume the function $f : (x_1, \ldots, x_n) \to (y_1, \ldots, y_n)$ is implemented via an ideal trusted party who gets the input x_i from user i and outputs y_i to the same user. Let $View^f$ and $Output^f$ denote the same distribution as above, in the f-hybrid model. The notion of privacy (Definition 1) can be extended to the f-hybrid model in a straightforward way.

The f-hybrid model is interesting due to the following fact. If some realization g of f is UC-secure [Can01], then proving the security of π in the f-hybrid model implies that an equivalent protocol that uses g to implement the functionality f is secure as well.

Lemma 2 (Universal Composability [Can01])**.** *Let g be an n-party protocol that (UC-)securely realizes an ideal functionality f, and let π be an n-party protocol in the f-hybrid model. Let π^g be the protocol obtained from π by replacing each call of f with the subroutine g.*

If π securely realizes some ideal functionality $\mathcal{F}$ in the f-hybrid model, then π^g securely realizes $\mathcal{F}$ from scratch.

We stress that we do not prove our schemes to be UC-secure, however we will use UC-secure subprotocols as part of our schemes.

4 n-Party Private Equality Testing with A Trusted Server

We now provide a private equality testing protocol for an arbitrary number of users, via a trusted server. Note that although the server is 'trusted', we require that it learns nothing out of the messages it processes. The main idea is that user A performs a generalization of the 2-party scheme [NTL+11] with each one of the users (B_1 to B_{n-1}), however, replies are sent to the server S, which incorporates them all into a single reply that is sent to A. This simple extension is interesting as a first step towards obtaining a private scheme without a trusted server.

For this scheme we assume a secure channel between each party and the (semi-)trusted server. Also, we assume p is a large prime known to all parties, such that $1/p$ is negligible in the security parameter κ. All the parameters of the protocol should be considered as numbers in $\mathbb{Z}_p$. The scheme is given in Protocol 1.

Protocol 1 A private n-party PET with a trusted server S

(trusted) setup : Let p be a prime number, known to all parties (p depends on the security parameter). All the values below are over $\mathbb{Z}_p$.
Let k_A be a key known to all the users, excluding the server. In addition, for every $i \in \{1, \ldots, n-1\}$, A and B_i share $k_{A,i}$. Each B_i shares randomness $r_{S,i}$ with the server. Each user has a secure private channel with the server.

1. Alice sends $m_A = X_A + k_A$ to the server.
2. Each B_i sends $m_i = r_{S,i}(X_i + k_A) + k_{A,i}$ to the server.
3. The server computes $\mathsf{res} = \sum_i (r_{S,i} m_A - m_i)$, and sends res to party A.
4. Alice's output is the boolean predicate $\mathsf{res} \stackrel{?}{=} \sum_i -k_{A,i}$.

Proposition 3. *The trusted server scheme (Protocol 1) is complete and $1/p$-sound.*

Proof. If all parties possess the same input, $\forall i\ X_A = X_i \equiv x$, we get

$$\mathsf{res} = \sum_i r_{S,i}[(x + k_A) - (x + k_A)] - k_{A,i} = \sum_i -k_{A,i},$$

and the scheme is complete. On the other hand, if party B_j has a different input than X_A, then

$$\mathsf{res} = \sum_{i \neq j} r_{S,i}(X_A - X_i) + r_{S,j}(X_A - X_j) - \sum_i k_{A,i}.$$

The probability that res equals $\sum_i -k_{A,i}$ is given by

$$\Pr_{r_{S,1} \ldots r_{S,n-1}} \Big[\sum_{i \neq j} r_{S,i}(X_A - X_i) = -r_{S,j}(X_A - X_j) \Big] =$$

$$\Pr_{r_{S,1} \ldots r_{S,n-1}} \Big[r_{S,j} = \sum_{i \neq j} r_{S,i}(X_A - X_i)(X_j - X_A)^{-1} \Big] = \frac{1}{p}\ .$$

□

Privacy. Using the above, we see that the protocol is private: the B_is receive no information at all, while the value obtained by party A is uniformly distributed in the case that some party B_j has input $X_j \neq X_A$, and fixed otherwise.

Theorem 4. *Protocol 1 is private against a semi-honest server and any coalition of semi-honest adversaries (excluding the server).*

Proof. Constructing a simulator for the (only non-trivial[3]) case where user A is corrupt (and possibly, several of the B_is), is as follows. The simulator runs Adv to obtain the messages sent by the corrupt parties to the server. In order to simulate the server's reply, the simulator uses the inputs $X_{P_{m+1}}, \ldots, X_{P_n}$, known to him by definition, and queries $\mathcal{F}_{ideal}$. If the output is 1, the simulator outputs $\sum_i -k_{A_i}$; otherwise, it outputs a random value. It is easy to verify that for any given input $\bar{X}$, the simulator's output is distributed in the same way as a transcript Protocol 1. □

Malicious Adversaries. The simulator described above extends to the case of malicious adversaries. As above, we consider only the non-trivial case in which A is corrupt. Note that the simulator is given the trusted setup parameters, specifically, it learns k_A and $k_{A,i}, r_{S,i}$ for any corrupt B_i. The value m_A sent by A can always be interpreted as $X'_A + k_A$ for some X'_A. Similarly, each message m_i uniquely defines a value $X'_i = (m_i - k_{A,i})/r_{S,i} - k_A$, which can be extracted from each corrupt party's message to the server. The simulator then queries $\mathcal{F}_{ideal}$ and continues as above.

However, in the malicious adversary model, one should keep in mind that the soundness of a protocol may be compromised. Assume the worst case scenario in which all the B_is are corrupted. In order to force $\mathsf{res} = -\sum k_{A,i}$, it must be that $\sum_i (m_i - k_{A,i}) = m_A \sum_i r_{S,i}$. If the malicious adversary can choose the randomness $r_{S,i}$ then he can use $\sum_i r_{S,i} = 0$ which allows him to control the value of the servers reply by choosing the values of m_i. However,

[3] In the other cases, the view is only the outgoing messages of the corrupt parties, which is trivial to simulate with black-box access to the adversary.

if the randomness is out of the adversary's control (that is, chosen by a trusted third party), the probability that $\mathsf{res} = -\sum k_{A,i}$ equals the probability of guessing the value of m_A, i.e., the probability to guess X_A.

Theorem 5. *In the malicious adversary model (with semi-honest server), the trusted server scheme (Protocol 1) is private, complete and $(\varepsilon + neg)$-sound, where ε is the probability of guessing X_A.*

We stress that in contrast to the above, the scheme is not private against a coalition that includes the server. When the server colludes with any of the B_is, the key k_A is revealed to the server which can learn X_A from A's message. In a similar way, if the server colludes with A, they can learn the inputs of each one of the B_is. Another drawback of the above protocol is the fact that each run requires fresh pre-shared keys and randomness, which reduces the efficiency.[4] In the following sections we present schemes that overcome these issues. The following schemes do not assume the existence of a trusted server, and are secure against any such malicious server that might be involved when implementing the schemes. [Yet, the above scheme is more efficient if we assume a semi-trusted parties, which is reasonable in some practical scenarios.]

5 Homomorphic PET Schemes

We now show multiparty PET schemes that do not use a trusted server. Generalizing a 2-party PET scheme into the multiparty case is not straightforward: if Alice performs a 2-party PET separately with each one of the B_is, she can learn which of the B_is are located next to her, and which are not, thus this trivial extension is not private.

Informally, in order to achieve privacy in the multiparty case, each user[5] i performs a secret sharing of a random value s_i, which is used to mask the 2-party PET result between the user and Alice. The users sum all the shares they have received from all the other users and obtain a secret sharing of $\sum_i s_i$. Then, the users send all the shares to Alice, who can reconstruct $\sum_i s_i$. Since the value of any individual s_i remains secret, Alice has no use of the separate 2-party PET results, and she must "sum" them up in order to deduct the sum of the secret shares. This prevents Alice from learning any information about the users separately, and guarantees privacy.

We assume secure channels between any two users, and let G be a cyclic group with prime order q, in which DDH is hard. We let g be a fixed generator of G. The scheme is described in Protocol 2. Note that the protocol requires only 2 rounds of communication, where a single round means simultaneous mutual communication between all the parties.

Completeness and Soundness. Each B_i sends Alice the value $\hat{s}_i$ along with an ElGamal encryption of the encoding of $r_i(X_i - X_A)$ masked with s_i,

$$(a_i, b_i) = (g^{r_i r_A + t_i}, h^{r_i(X_A - X_i) + s_i + (r_i r_A + t_i)}).$$

[4] This issue was discussed in previous works, and several solutions were suggested, such as the PRF construction used by [NTL+11].

[5] We slightly abuse the notations here, so that $i, j \in \{A, B_1, \ldots, B_{n-1}\}$. We further abuse the notations and refer to a general user (except Alice) as B_i.

Protocol 2 A private n-party homomorphic PET against semi-honest coalitions

1. Each party $i = A, B_1, \ldots, B_{n-1}$, randomly picks $s_i \in \mathbb{Z}_q$ and computes an (n,n)-secret-sharing of s_i, obtaining $s_{iA}, s_{iB_1}, \ldots, s_{iB_{n-1}}$ such that $\sum_j s_{ij} = s_i$. The share s_{ij} is sent to user j over a secure channel.
2. Alice randomly picks (a private key) $x \in \mathbb{Z}_q^*$, and publishes her public key $(g, h) = (g, g^x)$. Alice encodes her input as h^{X_A} and broadcasts (or sends to each party over the secure channel) an ElGamal encryption of her encoded input, that is, she sends $(g^{r_A}, h^{X_A + r_A})$ with a random $r_A \in \mathbb{Z}_q$.
3. Each party $i = B_1, \ldots, B_{n-1}$ receives the message (a_A, b_A) and performs the following. The user randomly picks two numbers r_i, t_i and computes $a_i = a_A^{r_i} \cdot g^{t_i}$ and $b_i = b_A^{r_i} \cdot h^{t_i - r_i X_i + s_i}$.
4. Each user (including Alice) computes the sum of all the shares he has (including his own, s_{ii}), $\hat{s}_i = \sum_j s_{ji}$, and sends Alice the message $m_i = (a_i, b_i, \hat{s}_i)$ over the secure channel.
5. Alice receives a message $(u_1, u_2, u_3)_i$ from each of the other parties. She decrypts each message using her private key so that for every user i she gets $\mathsf{res}_i = (u_2)_i / (u_1)_i^x = h^{r_i(X_A - X_i) + s_i}$. Alice also computes $\sigma = \sum_i (u_3)_i + \hat{s}_A$.
6. Alice's output is the boolean predicate $h^\sigma \stackrel{?}{=} h^{s_A} \prod_i \mathsf{res}_i$.

By decoding and multiplying all these messages in Step 6, Alice obtains the sum of the differences, $h^{\sum_{i \neq A}[r_i(X_A - X_i) + s_i]}$, which equals $h^{0 + \sum_{i \neq A} s_i}$ if all the users have the same input. Note that

$$\sigma = \sum_i \hat{s}_i = \sum_i \sum_j s_{ij} = \sum_i s_i,$$

thus

$$h^\sigma = h^{s_A} \prod_{i \neq A} \mathsf{res}_i = h^{s_A} h^{\sum_{i \neq A} s_i}.$$

On the other hand, if there exists a user j such that $X_A \neq X_j$, then the value res_j is a random power of h, which is independent of σ. The probability that Alice outputs 1 in this case is $O(1/q)$, which is negligible.

We note that Step 5 can be performed in a more efficient way by decrypting the product of all the values $(u_2)_i$, instead of decrypting each one individually.

Privacy. We show now that Protocol 2 is private in the semi-honest model. The case of an honest Alice is trivial, due to the semantic security of the encryption system. For the other case, we assume that there are $2 \leq m \leq n$ honest players, without loss of generality, $B_1, B_2, \ldots, B_m$. Note that the adversary cannot learn the secret shares $\{s_{ij}\}_{i,j \in \{B_1, \ldots, B_m\}}$. It is a well known fact that the shares s_{ij} of the value s_i are $(n-1)$-wise independent. Also, the values $\hat{s}_i$ (Step 4) are merely an (n, n)-secret sharing of $\sum_i s_i$ which are $(n-1)$-wise independent as well.All that the adversary learns is the value of the sum $\sum_i s_i$. Moreover, note that if s_i is unknown, the message $(u_1, u_2, u_3)_i$ is an encryption of a random number, and thus random.

Theorem 6. *Protocol 2 is private against a semi-honest adversary.*

Proof. If Alice is honest then the information the other parties see is nothing but an ElGamal encryption of Alice's input (encrypted with Alice's public key) and random shares. By the semantic security of the ElGamal scheme, their view is indistinguishable from an encryption of a random value (along with random shares), and it is clear that such a view is simulatable.

For the case where Alice is corrupt, for every PPT adversary Adv, we construct a PPT simulator Sim in the following way. The simulator runs Adv for the corrupt parties, and simulates the activity of the honest B_1 to B_m according to the protocol until Step 3 (randomly picking the secret values s_i and performing secret sharing, etc.).

In order to simulate the messages m_i of the honest parties, the simulator queries $\mathcal{F}_{ideal}$, with all the corrupt users' inputs set to X_A (obtained as part of the input $\bar{X}$). If $\mathcal{F}_{ideal} = 0$ then for each honest user's message m_i, Sim outputs an ElGamal encryption of a random value r_i, along with $\hat{s}_i$ according to the protocol, that is, the message $m_i = (g^{t_i}, h^{t_i+r_i}, \hat{s}_i)$, with a random t_i. If $\mathcal{F}_{ideal} = 1$ then for each honest m_i, Sim outputs an ElGamal encryption of the secret share s_i, that is, it outputs the message $m_i = (g^{t_i}, h^{s_i+t_i}, \hat{s}_i)$. The rest of the transcript follows the protocol.

Let us analyze the distribution of the adversary's view running the protocol with the honest parties. Assume there are $m \geq 2$ honest parties. Consider the values $r_i(X_A - X_i) + s_i$ which are sent (encrypted) by the honest parties. The secret sharing guarantees us that these values are $(m-1)$-independent. Although their joint distribution is not uniform, it is easy to verify that the sum of the values $r_i(X_A - X_i) + s_i$ is $\sum_i s_i$ when all the honest parties have the same input as A, and uniform otherwise. Note that this proves the theorem, since the corresponding (plaintext) values of the simulator are distributed exactly the same.

We note that in the case where $m = 1$ the secret sharing has no meaning (s_1 is known to the adversary due to knowing $\sum_i s_i$ and $\sum_{i \neq B_1} s_i$), however, this case is still secure due to the privacy of the 2-party scheme in [Lip03]. □

5.1 Private Homomorphic PET Against Malicious Adversaries

The above proof does not hold when the adversary is malicious, mainly due to the following two issues. First, a malicious A can send different messages to different parties (e.g., different values of X_A with different B_is); yet, this can easily be resolved if A uses a broadcast channel. The other issue is that a malicious A might ignore the input X_A, and the simulator might not be able to know the value X_A actually used. Moreover, instead of choosing a secret key x and publishing a public-key pair (g, g^x), the adversary might publish and use a public key for which he does not know the secret key, so there is no hope for the simulator to learn X_A from A's encrypted message (Step 2).

A standard technique to overcome this issue is to use a zero-knowledge proof of knowledge (ZKPoK) scheme, in which party A proves she knows the secret key x corresponding to the public key (g, g^x), yet without revealing any information about x to the other parties. This also allows the simulator to extract the secret x, which in turn lets it extract the value X_A from A's message.

Party A is required to convince *all* the B_is that she knows the secret x, which requires performing the ZKPoK scheme $n-1$ times, separately with each B_i. If this is done sequentially, the round complexity increases from constant to linear in n. On the other hand, if $n-1$ instances are to be performed in parallel, the ZKPoK must be secure under parallel composition. While the round complexity remains the same for such composition, the communication complexity multiplies by $n-1$. Although there is no hope to obtain sub-linear communication for parallel composition, the fact that A plays the same role in all the instances allows us to "compose" the separate instances so that all the B_is collaborate to act as a "single" verifier. This way, A performs a single ZKPoK instance over a broadcast channel, and we save a factor of $n-1$ in A's outgoing communication. Such a composed scheme is denoted a 1-Prover n-Verifier ZKPoK.

Definition 2 (1PnV-ZKPoK protocol[6])**.** *Let R be some binary relation and let $\kappa()$ be a function from bitstrings to the interval $[0..1]$. An interactive protocol between a single prover $P = P(x, w)$ and n verifiers $V_1(x), \ldots, V_n(x)$ is called a* 1-Prover n-Verifier Zero-Knowledge Proof of Knowledge *(1PnV-ZKPoK) protocol for a relation R with knowledge error κ, if for any $(x, w) \in R$ the following holds:*

1. *(Completeness) The honest Prover on input $(x, w) \in R$ interacting with honest verifiers, causes* all *the verifiers to accept with probability 1.*

2. *(Zero-knowledge) For any PPT verifiers $V'_1, \ldots, V'_n$ there exists a PPT simulator that has rewindable black-box access to $V'_1, \ldots, V'_n$ and produces a transcript T that is indistinguishable from a transcript of $V'_1, \ldots, V'_n$ with the real prover P.*

3. *(Extended extractability) There exists a PPT extractor* Ext *such that the following holds. For any prover P' and any verifiers $V'_1, \ldots, V'_{j-1}, V'_{j+1}, \ldots, V'_n$ (that might collude with P') for some $1 \leq j \leq n$, let $\epsilon(x)$ be the probability (the honest) V_j accepts x. There exists a constant c such that whenever $\epsilon(x) > \kappa(x)$,* Ext *outputs a correct w in expected time at most*
$$\frac{|x|^c}{\epsilon(x) - \kappa(x)}.$$

Note that while the zero-knowledge property is standard, when considering the proof-of-knowledge (extractability) property, the prover is allowed to collude with several of the verifiers, in order to fool an honest verifier. An extractor in this case is given black-box access to all the corrupt parties, possibly, including other verifiers.

In Appendix A we show how to construct a 1PnV-ZKPoK from a Σ-Protocol and a (perfectly-hiding) trapdoor commitment. Specifically, we show the following:

Proposition 7. *There exists a 5-move 1PnV-ZKPoK scheme for the discrete logarithm relation with negligible knowledge error.*

With Proposition 7 we construct a PET protocol that is private against malicious adversaries, by changing Protocol 2 in the following way. After publishing her public keys, party A performs a 1P(n-1)V-ZKPoK scheme for discrete log with all of the B_is. Each of the B_is acts as a verifier, and continues with Protocol 2 only if A succeeds in the ZKPoK (likewise, A aborts if any check defined in the scheme fails).[7] The complete scheme is given in Protocol 3. The scheme takes 9 moves, however moves 1–2 and 6–8 can be done at the same time, which leads to 6 rounds of mutual communication.

Theorem 8. *Protocol 3 is private against any coalition of malicious adversaries.*

Proof. We prove the privacy of Protocol 3 by constructing a simulator. Note that the only difference from Protocol 2 is the 1PnV-ZKPoK part of the scheme, and we now explain how to change the simulator of Protocol 2 accordingly.

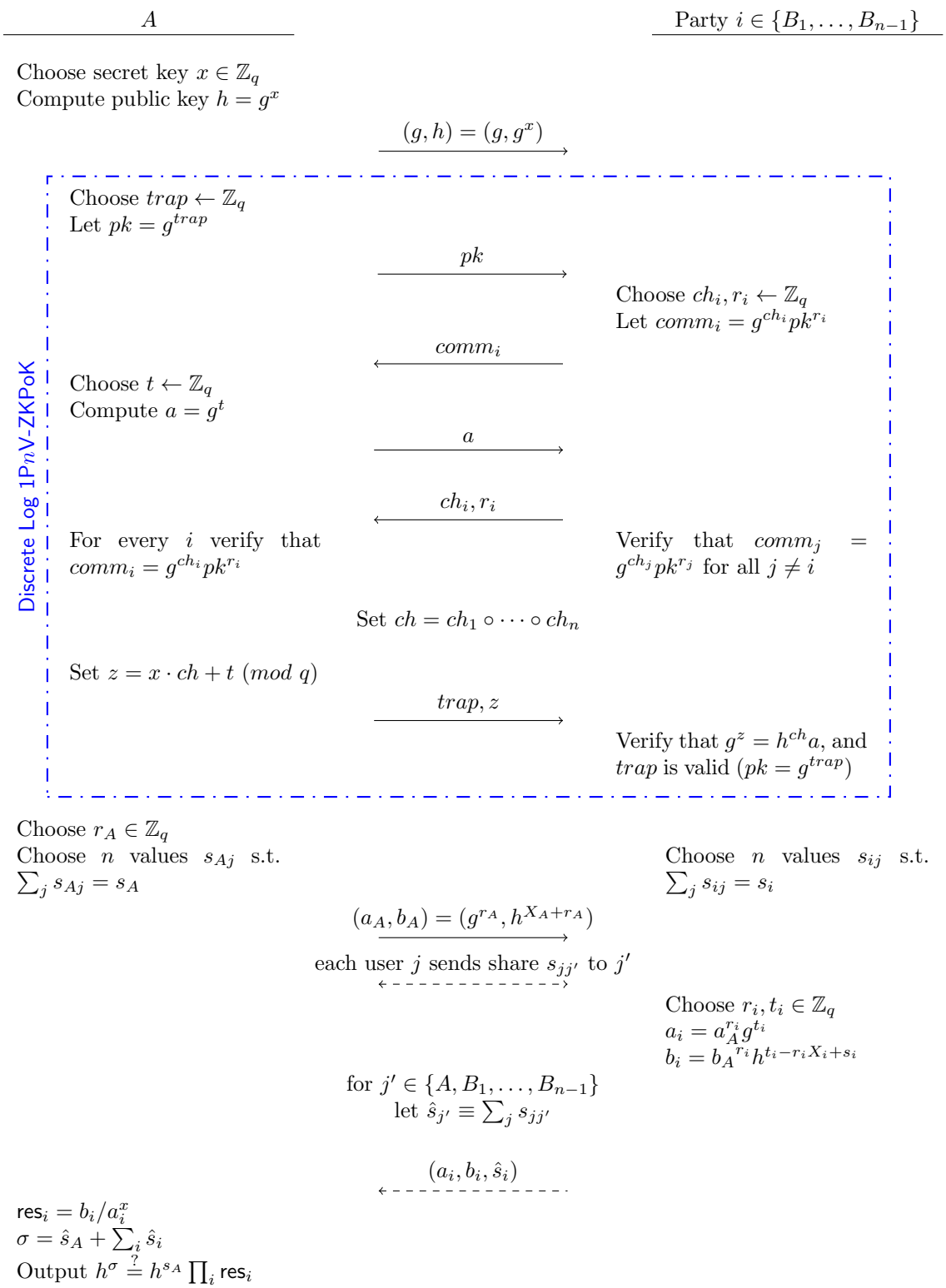
First consider the simple case in which A is honest. The simulator randomly chooses inputs $X_{P_1}, \ldots, X_{P_m}$ for the honest parties, and simulates their part according to the protocol. In particular, Sim chooses a secret key x, and performs the 1PnV-ZKPoK as the prover (which is possible since Sim knows x). It is clear that the generated transcript of the 1PnV-ZKPoK is

[6] We consider only PPT parties, thus this is, in fact, a ZK *argument*.

[7] We can assume that if any party aborts, this fact is broadcasted and the other parties abort as well.

Protocol 3 n-party PET with Homomorphic Encryption and ZKPoK

(trusted) setup : A group G of prime order q (which depends on n and the security parameter κ), a generator g. For two numbers $a, b \in Z_q$, define $a \circ b$ to be the concatenation of the least $\lfloor \log q/n \rfloor$ significant bits of a with the least $\lfloor \log q/n \rfloor$ significant bits of b. Dashed arrows mean private communication, while solid arrows mean broadcast.



distributed the same. The rest of the transcript is distributed correctly, as argued above, due to the semantic security of the ElGamal encryption.

In the case where A is corrupt, the simulator performs as follows. First, it extracts the value of x used by A via the extended extractability property of the 1PnV-ZKPoK, (if the extraction fails, Sim outputs an aborting transcript). With the knowledge of the private key x, Sim decrypts the message (a_A, b_A) sent on the 7th move, and obtains h^{X_A}. Next, the simulator exhaustively checks any value in $\mathcal{D}$. If for some value $d \in \mathcal{D}$, $h^d = h^{X_A}$, the simulator uses d as the "guessed" input of party A. Otherwise, the simulator sets $d = \perp$. This step is polynomial in the size of the dictionary. Next the simulator queries $\mathcal{F}_{ideal}$ using d as the input of each of the corrupt parties.[8] The rest of the simulation is exactly as in the proof of Protocol 2: the (plaintext of the) messages sent by the m honest users are $(m-1)$-independent, and their sum is random unless $d = X_i$ for all the honest parties. Hence, if $\mathcal{F}_{ideal} = 0$, the simulator replies on the 9th move for every honest i with (a_i, b_i) being encryptions of random values, and if $\mathcal{F}_{ideal} = 1$, Sim lets (a_i, b_i) be the encryption of the value s_i. □

6 Realizing PET via PAKE

In this section we construct a PET scheme based on a password-authenticated key-exchange (PAKE) primitive (see for instance [GL06, KOY01, CHK+05] and references therein). PAKE is performed between two users, A and B, where each user $i \in \{A, B\}$ possesses a password pw_i drawn from a low-entropy dictionary $\mathcal{D}$. The distributed computation $\mathsf{PAKE}(pw_A, pw_B)$ outputs a key k_i for each user i, such that k_i is uniformly distributed and $k_A = k_B$ if $pw_A = pw_B$.

On its surface, there seems to be some connection between PET and PAKE protocols, or any other password-authenticated primitives: users want to compute some functionality $\mathcal{F}$ whose output depends on whether or not the users share the same password. An adversary that holds a different password should not be able to learn $\mathcal{F}$'s outcome, and moreover, should learn nothing about the passwords of the users. On the other hand, there are several differences: PAKE seems to be a much stronger primitive, as the users begin with nothing but a (small-entropy) password and end up with a random session key, while in PET they might pre-share some set of keys, and end up with (only) a binary output.

Surprisingly, while a secure PAKE can be performed using a *single* communication round in the CRS model [BPR00, KV11], a secure PET requires at least 2 rounds to avoid off-line dictionary attacks. Indeed, if we assume a single-round 2-party PET, after receiving a message from B, it is possible for user A to simulate the PET scheme for any possible value in $X_A \in \mathcal{D}$, off-line, using the received message. The simulated-PET will output 1 only for the input $X_A = X_B$, leaking its value to Alice. One reason for this discrepancy is that when the users do not share the same password, they end up with different keys when the PAKE protocol terminates, yet *they are not aware of this fact*. In order to learn whether their keys match or not (which they trivially learn during PET), an additional round is required.[9]

We show that an n-party PET scheme can be realized using a chosen-plaintext unforgeable semantically-secure encryption scheme along with a secure 2-party PAKE, using one's private input as one's password. Our construction is defined in a hybrid model, assuming a secure two-party PAKE as a sub-protocol. Specifically, Let $\text{2PAKE}^{\mathcal{D},N} : (pw_1, pw_2) \to (k_1, k_2)$ be the

[8] We extend the definition of $\mathcal{F}_{ideal}$ to return 0 if any of its inputs is $\perp$.

[9] See also [CHK+05] for a discussion about mutual authentication in PAKE.

ideal functionality that accepts two passwords and outputs two numbers $k_1, k_2 \in Z_N$ such that k_1, k_2 are uniformly chosen and $k_1 = k_2$ if $pw_1 = pw_2$ and $pw_1, pw_2 \in \mathcal{D}$.

The main idea is to use 2PAKE$^{\mathcal{D},N}$ in order to form secure channels between each pair of users, using their private input X_i as the 2PAKE$^{\mathcal{D},N}$ password. Unless the users share the same input, they would use different keys to encrypt and decrypt messages, and "randomize" the transmitted messages. The scheme works as follows. Each user performs an (n, n)-secret sharing of the value 0 and sends the shares to the rest of the $n-1$ users, encrypted with a chosen-plaintext unforgeable semantically-secure encryption, using the PAKE generated key. Next, each user adds all the shares he has received, so the users still jointly hold a secret-sharing of 0. Finally, all the parties send Alice their accumulated shares.

In the case that all users hold the same password (and thus use matching encryption keys), Alice reconstructs the value 0 and learns that all the parties have the same private input. Otherwise, shares sent over channels with mis-matching keys are decrypted to a random value. This prevents Alice from reconstructing the value 0, and she learns that at least one party has a different private input. The scheme is described in Protocol 4.

Protocol 4 A realization of an n-party PET via 2PAKE$^{\mathcal{D},N}$

(trusted) setup : Assume $\mathbb{Z}_N$ is some fixed finite field, where N depends on κ. Assume a symmetric-key semantically secure $1/N$-chosen-plaintext unforgeable encryption scheme (GEN, ENC, DEC), and oracle access to 2PAKE$^{\mathcal{D},N}$.

1. Each user $i \in (A, B_1, B_2, \ldots, B_{n-1})$ runs 2PAKE$^{\mathcal{D},N}$ with any other user $j \neq i$, using his private input X_i as his PAKE password. Let 2PAKE$^{\mathcal{D},N}(X_i, X_j) = (sk_{ij}, sk_{ji})$.
2. Each user $i \in (A, B_1, B_2, \ldots, B_{n-1})$, randomly picks n values $s_{ij} \in \mathbb{Z}_N$ such that $\sum_j s_{ij} = 0$ and sends $ENC_{sk_{ij}}(s_{ij})$ to user $j \in (A, B_1, B_2, \ldots, B_{n-1})$.
3. User i, upon receiving the message m_j from user j, computes $\tilde{s}_{ji} = DEC_{sk_{ij}}(m_j)$. Next, each user $i \neq A$ sends the value $ENC_{sk_{iA}}(s_{ii} + \sum_{j \neq i} \tilde{s}_{ji})$ to Alice.
4. Alice decrypts the message from user i using the key sk_{Ai}, to obtain $\tilde{\tilde{s}}_i$. Alice computes $\mathsf{res} = s_{AA} + \sum_{i \neq A}(\tilde{\tilde{s}}_i + \tilde{s}_{iA})$, the sum of all the shares she has received (including her own).
5. Alice's output is the value of the boolean predicate $\mathsf{res} \stackrel{?}{=} 0$.

Theorem 9. *In the 2PAKE$^{\mathcal{D},N}$-hybrid model, Protocol 4 is private against any coalition of malicious adversaries.*

Proof. For the case where A is honest, we note that all the information that the adversary receives is (at most) $n-1$ shares of an (n, n)-secret sharing scheme, thus even if the adversary is capable of correctly decrypting the received messages, his view is uniformly distributed.

Otherwise, let $B_1, \ldots, B_m$ be honest. First we assume that there exists at least two honest parties B_i, B_j such that $X_i \neq X_j$. It follows that any honest user is incapable of decrypting *all* the messages he receives during Step 2, and hence the sum obtained in Step 3 in unexpected. Moreover, since the encryption scheme is $1/N$-chosen-plaintext unforgeable, the decrypted value is uniformly distributed, and so is the view of the adversary (even if he succeeds in guessing all $X_1, \ldots, X_m$). Next assume that all the honest parties share the same private input $X_1 = \cdots = X_m$. In this case as well, unless the adversary successfully guesses this private input and runs the PAKE scheme to obtain matching keys with all the honest parties, all that the adversary sees is random shares.

Constructing a simulator is now straightforward, and we give here only a sketch. The simulator runs Adv and learns, for each corrupt party i, the passwords pw_{ij} used as input

for 2PAKE$^{\mathcal{D},N}$. If the same password is used with all the honest parties, $pw_{iB_1} = \cdots = pw_{iB_m}$, then the simulator sets X_i as this password, and otherwise it sets $X_i = \perp$. Next, the simulator queries $\mathcal{F}_{ideal}$ using the private inputs X_i obtained in the previous step. If $\mathcal{F}_{ideal} = 1$ then the private inputs of the honest parties are the same as Adv's inputs, and the simulator outputs a transcript following the protocol and simulating all the honest parties with the same private input as Adv. Otherwise, the simulator picks random X_i values for the honest parties (more accurately, he can choose any set of inputs such that not all of them are equal) and completes the protocol using Adv and simulating each honest party's behavior with the random inputs. Using the fact that the adversary learns nothing about the keys used for encryption and the semantic security of the encryption scheme, the privacy follows. □

By Lemma 2, the existence of an efficient, constant round, UC-secure 2-party PAKE [CHK+05, ACCP08, HL11, KV11] implies the privacy of an equivalent protocol in the standard (non-hybrid) model:

Corollary 10. *Protocol 4 is private against any coalition of malicious adversaries (in the standard model).*

We are left to show that the scheme is sound when considering a malicious adversary.

Theorem 11. *Protocol 4 is $(\varepsilon + neg)$-sound in the malicious adversary model, where ε is the probability of guessing the password of party A.*

Proof. Assume the worst-case scenario in which all the B_is are corrupted. In order to cause Alice to output 1, they must send (2 sets of) messages to Alice such that the sum of their messages is a fixed[10] value c. However, for at least one of the secure channels between an adversarial B_i and Alice, the keys mismatch (except with small probability ε where the adversary successfully guesses the private input X_A). In the case where the key Alice uses is unknown, the adversary succeeds in generating a valid encryption of some value controlled by him with only negligible probability, due to the $1/N$-unforgeability of the encryption system. Thus, any message sent to Alice using a mismatching key is decrypted into a random value (or otherwise violates the $1/N$-unforgeability of the encryption scheme). The shares Alice receives sum up to c with only negligible probability, which completes the proof. □

Due to rushing, the adversary can wait until he learns Alice's message before sending a message to Alice. However, the adversary is required to send Alice *two* messages (one in Step 2 and the other in Step 3), while receiving from Alice *only one* message. Therefore no matter what the adversary does, in order to cheat successfully he must send Alice at least one message that decrypts to some given value, while other messages previously received by the adversary decrypt to different values with high probability. The adversary cannot create a successful cheating reply, even given previous received messages, due to the unforgeability of the encryption scheme.

7 Optimizing the Efficiency of 2-party Proximity Testing

As recently presented by Narayanan et al. [NTL+11], proximity testing can be realized via equality testing of quantized locations. To this end, the entire area is partitioned, using a fixed

[10] We can assume that we strengthen the adversary so that he knows this fixed value.

grid, into several regions, and the party's location is quantized to a center of such a region (or its index number, etc.). This same method is applicable to the multi-party case as well.

Using a grid, however, presents an inherent problem, where two parties might be arbitrarily close to each other, yet belong to different regions. A suggested remedy [ŠTŠ+09, NTL+11] is to overlay grids atop each other. The solution of Narayanan et al. uses three hexagonal grids, which is the minimal necessary to avoid the above problem. Combining such quantization with PET yields a proximity testing scheme with constants δ, γ such that if the parties are distanced at most δ from each other, the scheme outputs 1, and if they are at distance at least $\gamma\delta$, the output is guaranteed to be 0. For the hexagonal triple-grid, $\gamma = 4/\sqrt{3}$, and δ is controlled by the hexagon's size. See [NTL+11] for full details and the exact definition of the parameters γ, δ.

One of the main goals considered by Narayanan et al. is efficiency, and indeed their schemes are highly efficient: in their synchronous scheme Alice performs only three modular exponentiations, and Bob only two, and in their trusted-server (asynchronous) scheme the parties perform several additions and one modular multiplication each.[11] In addition, their schemes are bandwidth-economical: only a single message is sent by each party. Our multiparty extensions are also very efficient: Protocol 2 requires five exponentiations from Alice and two from Bob, but also $O(n)$ multiplications; Protocol 1 requires just one multiplication and several additions from each of the users, while the server is required to perform $O(n)$ multiplications and additions. In the homomorphic schemes, each message contains 2 elements of the group (or field), and thus takes $2\log|G|$ bits, while in the trusted-server scheme each message takes only $\log|G|$ bits.[12]

This efficiency makes the protocol extremely suitable for mobile devices. Unfortunately, using three fixed grids triples the amount of operations needed, as the protocol runs a PET instance separately for each grid.

7.1 Improved Efficiency for 2-Party Proximity Testing

We suggest an improvement for the 2-party case, which decreases both the communication complexity and the number of operations. Assume that t fixed grids are used. Our approach is that rather than repeating the scheme t times, Alice performs the scheme using a single grid—the one in which she is closest to the center.

For the synchronous scheme, Alice just sends the number of the grid she uses along with her message, and Bob continues the scheme using the specific grid chosen by Alice. The communication complexity decreases from $t \cdot (4\log|G|)$ bits in the original scheme to $4\log|G| + \log t$. The amount of operations also decreases by a factor of t (e.g., for the scheme in [NTL+11] we reduce the number of modular exponentiations from 15 to 5). Further, to keep the specific grid in use hidden from Bob (since it leaks some information on Alice's location), we do the following: first we make sure that cells of different grids have different numbers. Bob performs the scheme with all possible t grids (duplicating Alice's message n times), and sends all the replies back to Alice. Alice picks the reply according to the grid she uses using 1-out-of-n Oblivious Transfer, and completes the scheme.

For the trusted server model, we can reduce the number of operations as well. Bob still performs the scheme with all possible t grids, however, the server uses only one grid chosen by Alice. The parties can agree on a permutation π on $\{1, 2, \ldots, t\}$ so that the ith entry sent to the server matches the $\pi(i)$-th grid. This keeps the server oblivious to the specific grid in use.

[11] Omitting constant operations and encryption/decryption needed for transmitting data over a secure channel.

[12] The traffic also includes a single public key message and constant overhead for a secure communication. We neglect this overhead as well.

For the scheme of [NTL+11] with $t = 3$ hexagonal grids, the above reduces the total traffic from $3 \cdot (3 \log |G|)$ bits to $5 \log(|G|) + 2$ bits, assuming a field of size $|G|$, and neglecting overhead as before. The amount of modular multiplications performed by A and B decreases from 6 to 5.

The change in the quantization also changes the constants δ, γ: for the triple-hexagonal grid, δ becomes $1/\sqrt{12}$ times the side of the hexagon, and $\gamma = \sqrt{28}$.

8 Conclusions

We have presented several PET schemes, based on various primitives and assumptions, and proved their privacy in the malicious adversary model. Some schemes are implemented via ElGamal encryption, however, equivalent schemes can be realized using any semantically-secure homomorphic encryption.

See Table 1 for a summary of our results. The presented schemes are highly efficient, requiring only 2-rounds of communication and a total communication complexity of $O(n \log |G|)$ bits, with small constants. This makes these schemes suitable for various applications on limited-resource devices, such as smart-phones.

Since our scheme is asymmetric we do not consider fairness. In the presence of $n-1$ dishonest parties, such a task would require stronger primitives, see for instance [GIM+10].

Protocol	Adversary	Soundness ε	Rounds
Server-based (Protocol 1)	Malicious	Pr[Guess X_A] (negligible if semi-honest)	2
Homomorphic Enc. (Protocol 2)	Semi-Honest	negligible	2
Homomorphic Enc. +ZKPoK (Protocol 3)	Malicious	—	6
PAKE-based (Protocol 4)	Malicious	Pr[Guess X_A] (negligible if semi-honest)	3

Table 1: Summary of our protocols and their properties.

Acknowledgments. We thank Alan Roytman, Sanjam Garg and Akshay Wadia for useful discussions, and thank Serge Fehr for pointing out trapdoor commitment based ZKPoK schemes to us. We also thank the anonymous reviewers for comments and suggestions.

RG is supported in part by DARPA and NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174. RO is supported in part by NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[ACCP08] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient two-party password-based key exchange protocols in the uc framework. In *Proceedings of the 2008 The Cryptopgraphers' Track at the RSA conference on Topics in cryptology*, CT-RSA'08, pages 335–351. Springer Berlin, Heidelberg, 2008.

[Axe05] K. Axel. *Location-Based Services: Fundamentals and Operation*. John Wiley & Sons, Hoboken, NJ, USA, 2005.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 1–10, New York, NY, USA, 1988. ACM.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In Joe Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin / Heidelberg, 1998.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer Berlin / Heidelberg, 2000.

[BST01] Fabrice Boudot, Berry Schoenmakers, and Jacques Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1-2):23 – 36, 2001.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136 – 145, oct. 2001.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Phil MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 557–557. Springer Berlin / Heidelberg, 2005.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430. Springer Berlin / Heidelberg, 2000.

[Dam10] Ivan Damgård. On Σ-protocols, 2010. http://www.daimi.au.dk/~ivan/Sigma.pdf.

[DFK+06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304. Springer Berlin / Heidelberg, 2006.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469 – 472, July 1985.

[Fis01] M. Fischlin. *Trapdoor commitment schemes and their applications*. PhD thesis, Johann Wolfgang Goethe-Universität, Frankfurt am Main., 2001.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer Berlin / Heidelberg, 2004.

[FNW96] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Commun. ACM*, 39:77–85, May 1996.

[GG03] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, MobiSys '03, pages 31–42, New York, NY, USA, 2003. ACM.

[GIM+10] Dov Gordon, Yuval Ishai, Tal Moran, Rafail Ostrovsky, and Amit Sahai. On complete primitives for fairness. In Daniele Micciancio, editor, *Theory of Cryptography*, volume 5978 of *Lecture Notes in Computer Science*, pages 91–108. Springer Berlin / Heidelberg, 2010.

[GL06] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. *Journal of Cryptology*, 19:241–340, 2006.

[GLNS93] L. Gong, M.A. Lomas, R.M. Needham, and J.H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *Selected Areas in Communications, IEEE Journal on*, 11(5):648 –656, June 1993.

[Gol04] Oded Goldreich. *Foundations of cryptography. Vol II: Basic applications*. Cambridge University Press, New York, 2004.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[HL11] Xuexian Hu and Wenfen Liu. Efficient password-based authenticated key exchange protocol in the uc framework. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology*, volume 6151 of *Lecture Notes in Computer Science*, pages 144–153. Springer Berlin / Heidelberg, 2011.

[IOS12] Yuval Ishai, Rafail Ostrovsky, and Hakan Seyalioglu. Identifying cheaters without an honest majority. In Ronald Cramer, editor, *Theory of Cryptography*, volume 7194 of *Lecture Notes in Computer Science*, pages 21–38. Springer Berlin / Heidelberg, 2012.

[KGMP07] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *Knowledge and Data Engineering, IEEE Transactions on*, 19(12):1719 –1733, 2007.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer Berlin / Heidelberg, 2001.

[KOY09] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient and secure authenticated key exchange using weak

passwords. *J. ACM*, 57(1):3:1–3:39, November 2009.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer Berlin / Heidelberg, 2011.

[KY01] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin / Heidelberg, 2001.

[Lip03] Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 416–433. Springer Berlin / Heidelberg, 2003.

[LSHG08] Kevin A. Li, Timothy Y. Sohn, Steven Huang, and William G. Griswold. Peopletones: a system for the detection and notification of buddy proximity on mobile phones. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 160–173, New York, NY, USA, 2008. ACM.

[NO07] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 343–360. Springer Berlin / Heidelberg, 2007.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC '99, pages 245–254, New York, NY, USA, 1999. ACM.

[NTL+11] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location Privacy via Private Proximity Testing, 2011. The 18th Annual Network & Distributed System Security Symposium, NDSS 2011.

[OS07] Rafail Ostrovsky and William Skeith. A survey of single-database private information retrieval: Techniques and applications. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411. Springer Berlin / Heidelberg, 2007.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin / Heidelberg, 1999.

[Ped92] Torben Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Berlin / Heidelberg, 1992.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[ŠTŠ+09] Laurynas Šikšnys, Jeppe Thomsen, Simonas Šaltenis, Man Yiu, and Ove Andersen. A location privacy aware friend locator. In Nikos Mamoulis, Thomas Seidl, Torben Pedersen, Kristian Torp, and Ira Assent, editors, *Advances in Spatial and Temporal Databases*, volume 5644 of *Lecture Notes in Computer Science*, pages 405–410. Springer Berlin / Heidelberg, 2009.

[TDdMA11] Rafael Tonicelli, Bernardo David, and Vinı cius de Morais Alves. Universally composable private proximity testing. In Xavier Boyen and Xiaofeng Chen, editors, *Provable Security*, volume 6980 of *Lecture Notes in Computer Science*, pages 222–239. Springer Berlin / Heidelberg, 2011.

[Yao82] A.C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[ZGH07] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin / Heidelberg, 2007.

Appendix

A 1-Prover n-Verifier Zero-Knowledge Proof of Knowledge

In this section we show a construction of a 1PnV-ZKPoK protocol from a Σ-Protocol and a trapdoor commitment. The construction is an extension of the 2-party case (see, for instance, [Dam00, HL10]).

A.1 Preliminaries

Σ-Protocol Let R denote a binary relation, and let P_1, P_2, and $\mathsf{ZK\text{-}Ver}$ denote PPT algorithms. A protocol between a prover $P := P(x, w)$ and a verifier $V := V(x)$ is called a Σ-Protocol with

challenge space C if the following holds:

1. (3-move form) The protocol is of the following form:
   (i) P sets $(a, state) \leftarrow P_1(x, w)$, sends '$a$' to V and keeps '$state$' secret.
   (ii) V sends a random challenge $ch \leftarrow C$ to P.
   (iii) P computes $z \leftarrow P_2(x, w, state, ch)$ and sends z to V; the latter accepts if $\mathsf{ZK\text{-}Ver}(x, a, ch, z) =$ Accept or otherwise rejects.

2. (Completeness) If $(x, w) \in R$ then an honest P always causes an honest V to accept.

$$\forall (x,w) \in R, \quad \Pr\Big[(a, state) \leftarrow P_1(x,w); ch \leftarrow C; z \leftarrow P_2(x,w,state,ch); \\ \mathsf{ZK\text{-}Ver}(x,a,ch,z) = \text{Accept}\Big] = 1.$$

3. (Special honest-verifier zero-knowledge) There is a PPT simulator which, on input (x, ch), outputs a tuple (a, ch, z) and the distribution of its output is indistinguishable from the distribution of the transcript of P and V.

4. (Special soundness) Let (a, ch, z) and (a, ch', z') be two conversations with $ch \neq ch'$, that are accepting for some given x. There exists a PPT simulator (extractor) that on input x and those two conversations, computes w such that $(x, w) \in R$.

Trapdoor commitments A perfectly-hiding trapdoor commitment scheme on a message space M is a tuple of probabilistic polynomial-time algorithms (Commit, Verify, Setup, $\mathsf{TrapCom}$, $\mathsf{TrapDecom}$) such that the following holds.

1. (Completeness) For any $m \in M$,

$$\Pr\Big[(pk, trap) \leftarrow \mathsf{Setup}(1^\kappa); \\ (comm, decom) \leftarrow \mathsf{Commit}(m, pk) : \mathsf{Verify}(comm, decom, m, pk) = \text{Accept}\Big] = 1.$$

2. (Perfect Hiding) for every $m, m' \in M$, and for any pk generated by Setup, for $(comm_m, decom_m) \leftarrow \mathsf{Commit}(m, pk)$ and $(comm_{m'}, decom_{m'}) \leftarrow \mathsf{Commit}(m', pk)$, the commitments are identically distributed,

$$comm_m \equiv comm_{m'}.$$

3. (Binding) For every probabilistic polynomial time adversary Adv,

$$\Pr\Big[(pk, trap) \leftarrow \mathsf{Setup}(1^\kappa); (comm, decom_{m_1}, decom_{m_2}, m_1, m_2) \leftarrow \mathsf{Adv}(pk) : \\ \mathsf{Verify}(comm, decom_{m_1}, pk, m_1) = \text{Accept} \wedge \\ \mathsf{Verify}(comm, decom_{m_2}, pk, m_2) = \text{Accept} \ \wedge \ m_1 \neq m_2 \Big] < neg.$$

4. (Trapdoor Property) The knowledge of the trapdoor allows generating commitment $comm$, distributed in a similar way to a standard commitment, which can be opened as any $m \in M$. Formally, for any $(pk, trap) \leftarrow \mathsf{Setup}(1^k)$ and $comm \leftarrow \mathsf{TrapCom}(pk, trap)$ it holds that

(a) for every $m \in M$, and $(comm_m, decom_m) \leftarrow \mathsf{Commit}(m, pk)$,

$$comm \equiv comm_m.$$

(b) for every $m \in M$ and $decom \leftarrow \mathsf{TrapDecom}(comm, m, pk, trap)$

$$\Pr[\mathsf{Verify}(comm, decom, m, pk) = \text{Accept}] = 1$$

We require that $(pk, trap)$ is easy to verify as a valid output of *Setup*; assume that for a pair $(pk', trap') \notin \mathsf{Setup}(1^\kappa)$ for some κ, it holds that $\perp \leftarrow \mathsf{TrapCom}(pk', trap')$. In that case we say that $(pk', trap')$ is *invalid*.

See [Fis01] for various constructions of trapdoor commitment schemes.

A.2 Construction

Proposition 12. *Let* $(P_1, P_2, \mathsf{ZK\text{-}Ver})$ *be a* Σ*-Protocol with challenge space* C *for a relation* R*, and* $(\mathsf{Commit}, \mathsf{Verify}, \mathsf{Setup}, \mathsf{TrapCom}, \mathsf{TrapDecom})$ *be a trapdoor commitment scheme on a message space* M *such that* M^n *is isomorphic to* $C' \subseteq C$ *and* $\mathfrak{M} : M^n \to C'$ *be an isomorphism. The scheme given in Figure 1 is a 1PnV-ZKPoK protocol for* R *with knowledge error* $1/|M|$.

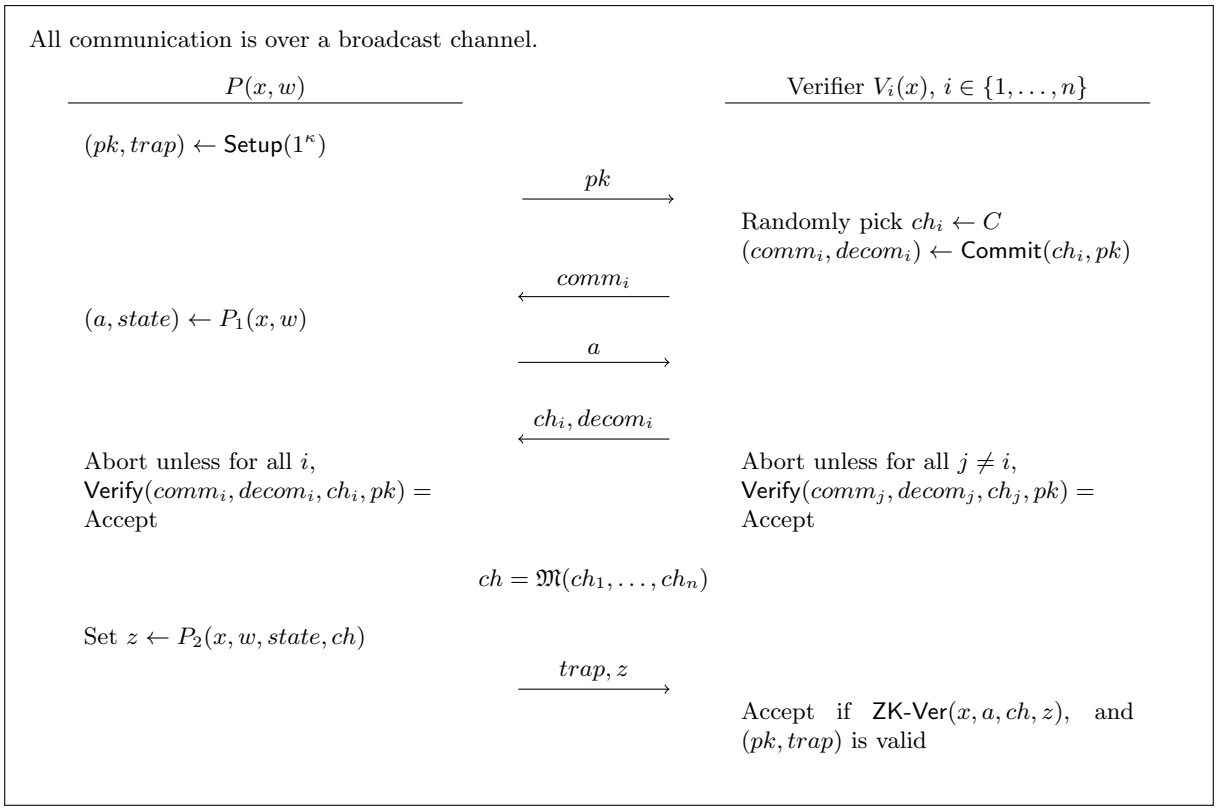


Figure 1: 1PnV-ZKPoK from a Σ-Protocol and a trapdoor commitment scheme.

Proof. The completeness trivially follows from the completeness of the Σ-Protocol and of the trapdoor commitment scheme.

Zero-knowledge: For any set of verifiers $V'_1, \ldots, V'_n$ the simulator works as follows. The simulator runs the scheme until it learns the challenges ch_i at the 4th move. Then, the simulator runs the simulator of the ZKPoK Σ-Protocol guaranteed by the special honest-verifier zero-knowledge property of the Σ-Protocol on input (x, ch) to obtain (a, ch, z). The simulator then rewinds the verifiers to the 3rd move, and sends a. Due to the binding property of the commitment scheme, the verifiers open the same challenges ch_i as they did on the first run (except with negligible probability), and the simulator completes the scheme as expected. Note that by forcing the verifiers to commit to ch_i before seeing a, their challenge is independent of a, and the honest-verifiers zero-knowledge property implies the composed scheme is zero-knowledge.

Extended extractability: Assume that for some $1 \leq j \leq n$, V_j is honest. For any prover P' and verifiers $V'_1, \ldots, V'_{j-1}, V'_{j+1}, \ldots, V'_n$, assume that V_j accepts x with probability $\epsilon(x)$. We show that there exists a constant c and a PPT extractor Ext that outputs a correct w in expected time $|x|^c/(\epsilon(x) - 1/|M|)$.

First we note that if an extractor Ext obtains two accepting transcripts, then it can extract w with polynomial time using the extractor guaranteed by the special soundness of the Σ-Protocol. This property implies the extended extractability property, in a similar way as the 2-party case (see, for instance [Dam00, Dam10, HL10]). The main difference is that the extractor cannot determine the challenge ch used by the prover but rather only a part of it ch_j. Due to rushing, it is possible that the challenges $ch_1, \ldots, ch_{j-1}, ch_{j+1}, \ldots, ch_n$ are determined adversarially after learning the value of ch_j. However the isomorphism guarantees that, for any strategy the corrupt V'_is take, the final challenge obtained from two different values $ch'_j \neq ch_j$, must be different. That is, given $ch_j \neq ch'_j$, then for any $(ch_1, \ldots, ch_{j-1}, ch_{j+1}, \ldots, ch_n)$ and $(ch'_1, \ldots, ch'_{j-1}, ch'_{j+1}, \ldots, ch'_n)$

$$ch := \mathfrak{M}(ch_1, \ldots, ch_n) \quad \neq \quad ch' := \mathfrak{M}(ch'_1, \ldots, ch'_n).$$

The proof follows from [Dam10, HL10] with straightforward adaptations, and we omit the details. □

A 1PnV-ZKPoK protocol for the discrete log relation (Proposition 7) can be constructed using Proposition 12 by composing a perfectly-hiding trapdoor commitment scheme based on Pedersen's (non-trapdoor) commitment scheme [Ped92] (see [Fis01] for construction and discussion) with Schnorr's Σ-Protocol for discrete log [Sch91]. The composed scheme is described inside the dashed frame in Protocol 3.