Honest-Verifier Private Disjointness Testing without Random Oracles

Susan Hohenberger* and Stephen A. Weis**

Massachusetts Institute of Technology
Cambridge, MA, USA
{srhohen,sweis}@mit.edu

Abstract. We present an efficient construction of a *private disjointness testing* protocol that is secure against malicious provers and honest-but-curious (semi-honest) verifiers, without the use of random oracles. In a completely semi-honest setting, this construction implements a *private intersection cardinality* protocol. We formally define both private intersection cardinality and private disjointness testing protocols. We prove that our construction is secure under the *subgroup decision* and *subgroup computation* assumptions. A major advantage of our construction is that it does not require bilinear groups, random oracles, or non-interactive zero knowledge proofs. Applications of private intersection cardinality and disjointness testing protocols include privacy-preserving data mining and anonymous login systems.

Keywords: private disjointness testing, private intersection cardinality, subgroup decision assumption, private data mining, anonymous login

1 Introduction

Suppose two parties, Alice and Bob, each have a private database of values, respectively denoted A and B, where the set cardinalities $|A|$ and $|B|$ are publicly known. Alice wishes to learn whether their two sets are disjoint, that is, whether $A \cap B = \emptyset$, or how large the intersection is, that is, $|A \cap B|$. In doing so, Alice cannot reveal information about her set A to Bob, who in turn does not want to reveal information about his set B, other than the bit $A \cap B \stackrel{?}{=} \emptyset$ or, perhaps, the size of the intersection $|A \cap B|$. These are respectively the *private disjointness testing* (PDT) and *private intersection cardinality* (PIC) problems.

For example, Alice may be a law enforcement agent ensuring that no suspects under investigation purchased tickets on a flight operated by Bob. Alice cannot simply reveal her list of suspects to Bob without compromising her investigation. Nor can Bob disclose any passenger names without explicit subpoenas. Yet, both parties have an interest in alerting Alice whether any suspects are on Bob's flight.

As another example, suppose Bob wants to anonymously login to Alice's system. Bob needs to prove that one of his identities in a set B, which may be a singleton, is among the set of Alice's valid users, denoted A. Alice should be convinced that Bob is a legitimate user, without learning which specific user he is. Thus, both parties wish to determine whether $|A \cap B| \neq 0$.

These types of private set operations may be implemented by several existing techniques. Private set operations may be viewed as a general two-party secure computation problem, solvable by classic secure multiparty computation techniques [12, 22]. Zero-knowledge sets due to Micali, Rabin and Kilian [16], support private operations like disjointness testing, set union, and set intersection.

Unfortunately, these techniques have remained unused in practice due to their high computation, communication, and implementation costs. Oblivious polynomial evaluation protocols, such as those due to

* Susan's work was supported by an NDSEG Fellowship.
** Stephen's work was supported by NSF Grant CCR-0326277.

THE FNP PROTOCOL:

1. $\mathcal{V}$ chooses a random constant or irreducible polynomial $G(x)$.
2. $\mathcal{V}$ computes $f(x) = G(x) \cdot (\prod_{a_i \in A}(x - a_i)) = \sum \alpha_i x^i$.
3. If any $\alpha_i = 0$, restart the protocol.
4. $\mathcal{V}$ encrypts the coefficients of $f(x)$ with a homomorphic encryption scheme and sends the encryptions $c_i = E(\alpha_i)$ to $\mathcal{P}$.
5. Using the homomorphic properties of E, $\mathcal{P}$ obliviously evaluates $f(x)$ at some value b, obtaining $E(f(b))$.
6. $\mathcal{P}$ randomizes his evaluation as $c = E(Rf(b))$ and sends it to $\mathcal{V}$.
7. $\mathcal{V}$ decrypts c. If $D(c) = 0$, $\mathcal{V}$ knows $\mathcal{P}$'s value intersects with A.

Fig. 1. An overview of the Freedman, Nissim, and Pinkas (FNP) protocol

Naor and Pinkas [17], may also be applied to private set operations. However, using generalized oblivious polynomial evaluation for private set operations is inefficient in comparison to specialized protocols.

This paper builds on specialized private set operation protocols recently developed by Freedman, Nissim, and Pinkas (FNP) [11], and Kiayias and Mitrofanova (KM) [14], and offers a new construction that is more efficient in a malicious-prover setting. When both parties are honest-but-curious (semi-honest), the Hohenberger and Weis (HW) construction presented in this work is a *private intersection cardinality* protocol, where a verifier party (who is played by Alice in the above examples) learns $|A \cap B|$. The efficiency in this setting is equivalent to both FNP and KM, but is based on a different complexity assumption.

Note that in the context of "honest-verifier", we are using the term "honest" interchangeably with "semi-honest". This means the verifier is honest-but-curious about the values it receives and while abiding by the protocol, may examine any received values further to try to learn more about B. The notion of semi-honest or honest-but-curious was introduced in [12].

The HW construction improves on existing results in settings where the prover is malicious and the verifier is honest-but-curious. In this malicious-prover setting, the HW construction implements a *private disjointness testing* protocol. A malicious, polynomial-time bounded prover able to send arbitrary messages cannot convince a verifier that their sets intersect, unless they actually do. In the anonymous login example, Bob will not be able to login unless he possesses a legitimate identity string.

The HW honest-but-curious (semi-honest) private intersection cardinality protocol presented in this paper *as is* becomes a private disjointness testing protocol in the malicious-prover setting. By contrast, previous works require additional computations, such as adding zero-knowledge proofs [14] or homomorphic encryptions [11], to be made secure in a malicious-prover setting. Moreover, both FNP and KM require random oracles to be made secure in the presence of a malicious prover, whereas the HW construction does not.

1.1 The FNP Protocol Paradigm

The FNP protocol [11] is quite intuitive and simple, and is the design paradigm used in both the KM and HW protocols. An FNP invocation where Bob has a singleton set is informally outlined in Figure 1. To provide further technical details, suppose (G, E, D) is a semantically-secure homomorphic encryption scheme. Let $\mathcal{V}$ have set $A = \{a_1, \ldots, a_n\}$ and $\mathcal{P}$ have set $B = \{b_1, \ldots, b_m\}$.

As shown in Figure 1, the verifier (also known as Alice) first selects a random constant or irreducible polynomial $G(x)$ (i.e. $G(x)$ will have no roots). The verifier than computes $f(x) = G(x) \cdot (\prod_{a_i \in A}(x - a_i)) = \sum \alpha_i x^i$. Note that the roots of f are exactly the values in the set A. The verifier then encrypts the α coefficients of f under a public key pk that she chooses, and sends them to the prover. That is, $\mathcal{V}$ encrypts each coefficient as $c_i = E_{pk}(\alpha_i)$ with a homomorphic cryptosystem such as Paillier's [18, 19].

Recall that homomorphic cryptosystems like Paillier's allow a party given $E_{pk}(x)$ and $E_{pk}(y)$ to obliviously compute $E_{pk}(x) \cdot E_{pk}(y) = E_{pk}(x + y)$, or to compute $E_{pk}(x)^z = E_{pk}(x \cdot z)$, where z is some constant. Note that given the encryptions c_i, these homomorphic operations are sufficient to obliviously evaluate the polynomial f. For example, the encryptions $c_0 = E_{pk}(4)$ and $c_1 = E_{pk}(3)$ commit the polynomial $f(x) = 3x + 4$. A second party may evaluate this at a particular value $x = 2$, by computing

$$c_1^2 \cdot c_0 = E_{pk}(3 \cdot 2) \cdot E_{pk}(4) = E_{pk}(6 + 4) = E_{pk}(10) = E_{pk}(f(2))$$

Thus, given coefficients encrypted as c_i values, the prover (Bob) may obliviously evaluate $f(b_i)$ for each element $b_i \in B$. Note that if $b_i \in A$, then $f(b_i) = 0$. The prover will now randomize all his obliviously evaluated $f(b_i)$ values by homomorphically multiplying them by a random nonzero value. That is, he computes $E_{pk}(f(b_i))^r = E_{pk}(r \cdot f(b_i))$ where r is a random nonzero value. Thus, if $f(b_i) = 0$, then the encryption of $E_{pk}(r \cdot f(b_i)) = E_{pk}(0)$. Otherwise, $E_{pk}(r \cdot f(b_i))$ is some random value. This hides any information about elements in the prover's set that are *not* in the verifier's set.

The prover now sends each of these encrypted oblivious evaluations $E(r_i \cdot f(b_i))$ to the verifier. The verifier then decrypts and tests whether any of the resulting plaintexts are zero. If $b_i \in A$, then $f(b_i) = 0$, so if any decrypted values are zero, then the verifier believes there is an intersection with the prover's set. Note that the original FNP protocol reveals the elements in the intersection of the two sets, by having the prover return the ciphertext $E_{pk}(r \cdot f(b_i) + b_i)$ instead. Thus if $f(b_i) = 0$, the verifier will get the actual elements of the intersection – not just the cardinality.

We focus on applications where the prover explicitly does not want to reveal anything about his set, except the size or existence of the intersection. For instance, the anonymous login application cannot have the verifier learn the actual intersection values. This paper will only focus on the private intersection cardinality protocol version of FNP, although finding actual intersection values will be discussed further in Section 7.7.

In the KM protocol [14], the same techniques as FNP are applied, except that it uses a new primitive called *superposed encryption* based on Pedersen commitments [20]. Superposed encryption is closely related to a homomorphic ElGamal variant first used in voting schemes by Cramer, Gennaro, and Schoenmakers [9]. In the KM protocol the prover returns to the verifier a single ciphertext $E_{pk}(r \cdot |A \cap B|)$, where r is a random value. Thus, this is specifically a PDT protocol rather than a PIC protocol. The verifier accepts if the ciphertext decrypts to zero and rejects otherwise.

Both the FNP and KM constructions, based on Paillier's homomorphic encryption [18, 19] and Pedersen's commitment scheme [20], suffer from a crucial flaw: *malicious adversaries may simply encrypt or commit to zero values*. For instance, in the FNP case, someone can simply encrypt 0 with the public key and convince the verifier that an intersection exists when it does not. This is a critical failure which both FNP and KM immediately recognize and address. To cope with malicious provers, FNP proposes a fix that relies on the random oracle model (ROM), despite its inherent problems [2, 7].

Fixing KM against malicious adversaries requires both the use of random oracles as well as universally-composable (UC) commitments [6], which require the assumption of a common reference string. While relatively efficient, the best known UC commitment schemes are interactive and would increase communication complexity by a quadratic factor [5, 8, 10].

The weakness of FNP and KM in the face of malicious provers begs the question: Can we implement an efficient private disjointness testing protocol without the use of random oracles or costly sub-protocols? This paper answers this question in the affirmative.

1.2 Overview of the HW Construction

This section provides intuition for understanding the Hohenberger and Weis (HW) construction in the context of prior work. Essentially, the main difference is that in both the FNP and KM protocols, a prover convinces a verifier to accept by returning an encryption of zero. If the prover was honest, then if the verifier receives an encryption of a zero value, it implies some element in $\mathcal{P}$'s set is also in $\mathcal{V}$'s set. However, if the prover is malicious, then he can easily encrypt a zero value from scratch and send it to the verifier. To prevent this, both FNP and KM must add costly computations to check that the prover follows a specified protocol.

To cope with malicious provers, the HW construction essentially substitutes a cryptographic primitive dubbed "testable and homomorphic commitments" in the place of Paillier's homomorphic encryption. Instead of encryptions of zero, elements belonging to the intersection of the two sets will be encoded to have a specific order in a multiplicative group. In other words, a prover convinces a verifier that an intersection exists by returning elements of a specific order.

The necessary complexity-theoretic assumptions are that it is hard to for a prover to decide whether group elements belong to a particular subgroup of unknown order, and that it is hard to compute elements in the subgroup. Under this *subgroup computation assumption*, computing an element of this order is hard for a prover, unless he correctly follows the protocol (and there *is* a non-empty intersection). Thus, in the malicious-prover setting, the HW construction is sound by default, whereas FNP and KM must augment their protocols with costly computations in the random oracle model.

In the HW construction presented in Section 4, the verifier begins, as in FNP, by selecting a random polynomial $f(\cdot)$ whose roots correspond to set A. The verifier computes a *testable and homomorphic commitment* (THC) of each coefficient, which is essentially a BGN encryption [3] set in group $\mathbb{G}$, which has order $n = pq$ where p and q are large primes.

For each element $b_i \in B$, the prover uses THCs to compute a value that will be a random element in $\mathbb{G}$ if $b_i \notin A$ or will be a random element of order p if $b_i \in A$. The verifier, with knowledge of p and q, can test the order of each element returned by the prover. In this way, the verifier learns the cardinality of the intersection, just as in FNP.

The main benefit, however, is that a malicious prover cannot, under the subgroup computation problem, compute an element of order p from scratch. As will be proven in Appendix A.2, the HW construction remains sound in the malicious-prover setting without any augmentation. As in the FNP PDT protocol, the verifier can *potentially* learn the cardinality of the intersection, but is not *guaranteed* to do so when talking with a malicious prover. That is, if the prover happens to be honest, the verifier will learn the cardinality – but there is no way to know whether a prover is honest. Table 1 compares the behavior of FNP, KM, and the HW construction in different security settings.

1.3 Related Work

Kissner and Song [15] offer FNP-inspired schemes for solving several closely related privacy-preserving set operations like set disjointness, cardinality, and set union. They offer improved efficiency compared to FNP in the multiparty, honest-but-curious setting. Again, when translated to the malicious adversary model, their constructions require relatively costly zero-knowledge proof of knowledge sub-protocols. In all fairness, Kissner and Song address a richer set of problems than simple disjointness testing like set union, set intersection, and multiplicity testing. They also work in a multiparty model, so it is not surprising that their solutions require more computation.

Security Setting	FNP	KM	HW
Semi-Honest	Cardinality	Disjointness	**Cardinality**
Malicious Prover (Requirements)	Cardinality (ROM)	Disjointness (NIZK Proofs) (ROM)	**Disjointness** (**None**)
Malicious Verifier (Requirements)	Cardinality (Multiple Invocations)	Disjointness (UC-Commitments) (ROM)	See Section 7.1

Table 1. Three private set protocols compared in different security settings. ROM stands for "Random Oracle Model", NIZK for "Non-Interactive Zero Knowledge", and UC for "Universally Composable".

Constructions from both Pedersen's commitment scheme [20] and Paillier's homomorphic cryptosystem [18, 19] are both closely related to the "testable and homomorphic commitment" primitive presented in Section 4.2.

The Subgroup Decision Assumption (SDA) and the Subgroup Computation Assumption (SCA) described in Section 2.2 are both crucial to proving security of the construction presented in this paper. Yamamura and Saito apply the SDA to the private information retrieval problem [21]. The composite residuosity assumptions made by Paillier are also closely related.

A similar *bilinear* subgroup complexity assumption is made by Boneh, Goh, and Nissim for their 2DNF ciphertext evaluation scheme [3]. Groth, Ostrovsky, and Sahai also make the same complexity assumption to implement non-interactive zero knowledge proofs [13].

2 Preliminaries

2.1 Notation

Let $\mathbb{Z}$ be the integers. Let $\mathrm{negl}(\cdot)$ be a *negligible* function such that for all polynomials $p(\cdot)$ and all sufficiently large $k \in \mathbb{Z}$, we have $\mathrm{negl}(k) < 1/p(k)$. We will denote that two distributions C and D are perfectly indistinguishable using $C \approx D$ and computationally indistinguishable using $C \stackrel{c}{\approx} D$ notation. A $\mathcal{M}_{ppt}$ subscript will indicate that a interactive Turing machine $\mathcal{M}$ runs in probabilistic polynomial time. The value $ord(x)$ will be the order of an element x. The transcript $\mathsf{View}^{\mathcal{M}}[\mathcal{M}(x)\mathcal{N}(y)]$ will represent the view of algorithm $\mathcal{M}$ after interacting with algorithm $\mathcal{N}$ on inputs x and y, respectively. $\mathcal{M}$'s view includes its input, its randomness, and the public transcript of the protocol. We will denote a distribution of views over random inputs as $\{\mathsf{View}^{\mathcal{M}}[\mathcal{M}(x)\mathcal{N}(y)]\}$.

2.2 Complexity Assumptions

The complexity assumptions applied in the HW construction exist in various forms throughout the literature. The formalization here is closest to that of Yamamura and Saito [21]. Recently, Boneh, Goh, and Nissim introduced a stronger version of these assumptions for *bilinear* groups [3].

Definition 1 (Subgroup Decision Assumption (SDA) [3, 21]). *Let $S(1^k)$ be an algorithm that produces $(\mathbb{G}, p, q)$ where $\mathbb{G}$ is a group of composite order $n = pq$, and $p < q$ are k-bit primes. Then, we say that the*

subgroup decision *problem is hard in* $\mathbb{G}$ *if for all probabilistic polynomial time adversaries* $\mathcal{A}$*,*

$$\Pr\left[(\mathbb{G}, p, q) \leftarrow S(1^k);\ n = pq;\ x_0 \leftarrow \mathbb{G};\ x_1 \leftarrow x_0^q;\ b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}(\mathbb{G}, n, x_b) : b = b'\right] \leq \frac{1}{2} + \mathrm{negl}(k).$$

Basically, the SDA means that given the description of a group $\mathbb{G}$, in the form of a generator g, and its order $n = pq$, a probabilistic polynomial-time adversary cannot distinguish random elements of order p in $\mathbb{G}$ from random elements in $\mathbb{G}$. Clearly, if factoring is easy, then the SDA fails to hold. Similarly, someone able to compute discrete logarithms given $(\mathbb{G}, n, x)$ can decide this problem by computing $gcd(\log_g x, n)$, for some generator g. It is not clear how the SDA relates to the Decisional Diffie-Hellman (DDH) assumption.

Additionally, the security of the HW scheme requires the following computational assumption:

Definition 2 (Subgroup Computation Assumption (SCA)). *Let* $S(1^k)$ *be an algorithm that produces* $(\mathbb{G}, p, q)$ *where* $\mathbb{G}$ *is a group of composite order* $n = pq$*, and* $p < q$ *are* k*-bit primes. Then, we say that the* subgroup computation *problem is hard in* $\mathbb{G}$ *if for all probabilistic polynomial time adversaries* $\mathcal{A}$*,*

$$\Pr\left[(\mathbb{G}, p, q) \leftarrow S(1^k);\ n = pq;\ x \leftarrow \mathcal{A}(\mathbb{G}, n) : ord(x) = p\right] \leq \mathrm{negl}(k).$$

An example group where these assumptions may be applied is a subgroup $\mathbb{G}$ of order $n = pq$, consisting of the quadratic residues of $\mathbb{Z}_{p'}$, where $p' = 2pq + 1$ and p', p, q are all primes. Of course, the HW construction can also operate over the bilinear groups where Boneh et al. [3] assume the subgroup decision problem is hard. It is not clear that the SDA assumption implies SCA, or vice versa, although a relation between the two seems plausible. Further exploration of both assumptions could be valuable in other schemes as well.

3 Problem Definitions

This section formally defines private intersection cardinality (PIC) and private disjointness testing (PDT) protocols. Let 1^k be a security parameter in unary. Let Q be the domain of values for this protocol such that $|Q| \in \Theta(2^k)$. Let the universe U be the set of all $\mathrm{poly}(k)$-sized subsets of Q. For sets $A \in U$ and $B \in U$, define the *disjointness predicate* $D(A, B) = (A \cap B = \emptyset)$, that is, $D(A, B)$ will have value 1 if and only if A and B are disjoint.

Let a verifier $\mathcal{V}$ and a prover $\mathcal{P}$ be two probabilistic polynomial time interactive Turing machines. Each party takes an element of U (i.e. a $\mathrm{poly}(k)$-size subset of Q) as input. The interaction of $\mathcal{P}$ and $\mathcal{V}$ yields a result to $\mathcal{V}$ *only*.

3.1 Private Disjointness Testing Definition

Definition 3 (Honest-Verifier Private Disjointness Testing). *Two probabilistic polynomial time interactive Turing machines* $(\mathcal{P}, \mathcal{V})$ *define an* Honest-Verifier Private Disjointness Testing *protocol if the following conditions hold:*

1. **Completeness:** *For honest parties, the protocol works and the verifier learns the disjointness predicate; that is,*

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = D(A, B)\right] \geq (1 - \mathrm{negl}(k))$$

where the probability is taken over the randomness of $\mathcal{P}$ *and* $\mathcal{V}$*.*

2. **Soundness:** *For a random set $A \in U$, the probability that the prover will convince the verifier to accept is negligible; that is,*

$$\forall \mathcal{P}^*_{ppt},\ \Pr_{A \in U}\left[\mathcal{P}^*\mathcal{V}(A) \neq 0\right] \leq \mathrm{negl}(k)$$

where probability is taken over the choice of $A \in U$ and the randomness of $\mathcal{P}^*$ and $\mathcal{V}$.

3. **Malicious-Prover Zero Knowledge (MPZK):** *A malicious prover learns nothing about the verifier's set; that is,*

$$\exists \mathcal{S}_{ppt},\ \forall \mathcal{P}^*_{ppt},\ \forall A \in U,\ \{\mathit{View}^{\mathcal{P}^*}\left[\mathcal{P}^*\mathcal{V}(A)\right]\} \stackrel{c}{\approx} \{\mathit{View}^{\mathcal{P}^*}\left[\mathcal{P}^*\mathcal{S}(1^{|A|})\right]\}$$

4. **Honest-Verifier Perfect Zero Knowledge (HVPZK):** *An honest-but-curious verifier learns nothing about the prover's set beyond the size of the intersection; that is,*

$$\exists \mathcal{S}_{ppt},\ \forall A \in U,\ \forall B \in U,\ \{\mathit{View}^{\mathcal{V}}\left[\mathcal{P}(B)\mathcal{V}(A)\right]\} \approx \{\mathcal{S}(A, 1^{|B|}, 1^{|A \cap B|})\}$$

Note that an honest-but-curious verifier is allowed to *potentially* learn $|A \cap B|$, but he is not *guaranteed* to learn that value. One might define a stronger definition where rather than being provided $1^{|A \cap B|}$, the simulator would only be provided $D(A, B)$.

3.2 Private Intersection Cardinality Definition

Definition 4 (Honest-Verifier Private Intersection Cardinality). *An* Honest-Verifier Private Intersection Cardinality *protocol has the same setup as in Definition 3, except for the following differences:*

1. **Completeness:** *For honest parties, the protocol works and the verifier learns the cardinality predicate; that is,*

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = |A \cap B|\right] \geq (1 - \mathrm{negl}(k))$$

where probability is taken over the randomness of $\mathcal{P}$ and $\mathcal{V}$.

2. **Cardinality Soundness:** *A malicious prover can not convince an honest verifier that the cardinality is larger than it really is; that is,*

$$\forall \mathcal{P}^*_{ppt},\ \forall B \in U,\ \Pr_{A \in U}\left[\mathcal{P}^*(B)\mathcal{V}(A) > |A \cap B|\right] \leq \mathrm{negl}(k)$$

where probability is taken over the choice of $A \in U$ and the randomness of $\mathcal{P}$ and $\mathcal{V}$.

3.3 Informal Explanation of the Definitions

Completeness means that a correct execution between two honest parties will return the correct value to $\mathcal{V}$ with negligible chance for error. In a PDT protocol, the correct value is the disjointness predicate $D(A, B)$ and in a PIC protocol it is the intersection cardinality $|A \cap B|$.

PDT soundness implies that on a random input set $A \in U$, $\mathcal{V}$ has a negligible chance of obtaining a non-zero result when interacting with any malicious probabilistic polynomial-time prover $\mathcal{P}^*$. That is, unless $\mathcal{P}^*$ actually knows a value in $\mathcal{V}$'s set, or is extremely lucky, then $\mathcal{V}$ will not be fooled into thinking otherwise.

Neither the FNP nor KM protocols are sound by this definition. In those schemes, a verifier will believe that there is an intersection if it receives the value zero encrypted under a public-key. A malicious prover could trivially violate the soundness property by encrypting a zero value itself.

PIC soundness is similar to the PDT soundness definition, except that for any set B, and random set A, the protocol has a negligible chance of returning a value greater than $|A \cap B|$ to a verifier $\mathcal{V}$ interacting with $\mathcal{P}^*(B)$. The idea is that this prevents a malicious prover from doing trivial attacks like duplicating elements in its set B to inflate the cardinality returned to the verifier. Of course, a malicious prover can always run the protocol on some subset of B, which would with high probability under-report the cardinality. This is unavoidable and is why cardinality soundness is only concerned with over-reporting the cardinality. As it turns out, this property will be the reason why the HW construction in Section 4 is *not* an Honest-Verifier Private Intersection Cardinality protocol. Section 6 will discuss this further.

Since a verifier is allowed to potentially learn $|A \cap B|$ in both the PDT and PIC protocols, the zero knowledge definitions presented in this paper are the same. This relaxation appears in FNP as well, but not KM.

The Malicious-Prover Zero Knowledge (MPZK) property means that no probabilistic polynomial-time potentially malicious prover $\mathcal{P}^*$ can learn anything about a set A from an interaction with $\mathcal{V}$ that it could not simulate on its own. In other words, the verifier's set, for example a database of passwords, remains hidden from even malicious provers. Here the distributions are computationally indistinguishable. Any action that $\mathcal{V}$ takes as a result of a successful protocol invocation, such as allowing $\mathcal{P}^*$ to anonymously login, is considered outside the protocol definition.

Finally, the Honest-Verifier Perfect Zero Knowledge (HVPZK) property implies that a probabilistic polynomial-time semi-honest verifier $\mathcal{V}$ does not learn anything about B beyond the size of the set intersection. There is a subtle point here in the PDT protocol: the verifier is only *guaranteed* to learn the bit $D(A, B)$, but we allow an honest-but-curious verifier to *potentially* learn the size of the intersection. The flexibility suits the applications mentioned in the introduction. In fact, in the semi-honest setting, the distribution an adversary can simulate on its own is perfectly indistinguishable from a real transcript distribution.

The definitions in this paper do not explicitly consider auxiliary inputs in the zero-knowledge definitions in this paper. To do so, one need simply quantify over all polynomial-size advice strings and provide this string to both the party in question and the simulator.

4 HW Private Disjointness Testing

In this section, we present a construction that efficiently implements a PDT protocol. Appendix A proves that this construction securely meets the requirements of Definition 3. Overall, this construction is very similar to those of Freedman, Nissim, and Pinkas (FNP) [11] and Kiayias and Mitrofanova (KM) [14].

FNP and KM respectively rely on Paillier's homomorphic encryption system [18, 19] and a Pedersen commitment variant [20] as underlying primitives. This paper offers a new *testable and homomorphic commitment* (THC) primitive that will be used in a FNP-style oblivious polynomial evaluation scheme. The THC construction presented is reminiscent of both Paillier's and Pedersen's schemes. It is very similar to the encryption scheme for small messages due to Boneh, Goh, and Nissim (BGN) [3], but is used for the full range of messages.

The advantage of the HW construction is that it offers a stronger security guarantee than the basic FNP and KM protocols, with equivalent computation and communication costs. Although variants of both FNP and KM can be modified to offer stronger security, they require either the use of random oracles or significantly more computation.

4.1 Verifier System Setup

VERIFIER SYSTEM SETUP:

1. Run $S(1^k)$ to obtain $(\mathbb{G}, p, q)$.
2. Choose two random generators g and u from $\mathbb{G}$.
3. Compute $n = pq$ and $h = u^q$.
4. Publish $(\mathbb{G}, n)$ and keep (p, q, g, h) private.

Fig. 2. HW verifier system setup

As illustrated in Figure 2, the HW construction is initialized by a verifier that selects some group of order $n = pq$, where p and q are secret large primes. The verifier will also select two random generators g and u, and will compute $h = u^q$. Note that h is a random generator of the subgroup of order p.

The verifier only needs to publish $\mathbb{G}$ and n. The prover will not know p, q, h or even g. Learning h, p, or q would allow a malicious prover to spuriously convince the verifier that an intersection exists.

4.2 Testable and Homomorphic Commitments

The public order n and private values g and h may be used for a *testable and homomorphic commitment* (THC) scheme. This primitive will be the basis of the HW construction. Informally, a THC scheme supports the following operations:

- Commit: $\mathsf{Com}(m, r)$ a message m with randomness r,
- Addition: For all m, r, m', r', $\mathsf{Com}(m, r) \cdot \mathsf{Com}(m', r') = \mathsf{Com}(m + m', r + r')$,
- Constant Multiplication: For all m, r, c, $\mathsf{Com}(m, r)^c = \mathsf{Com}(cm, cr)$
- Equality Test: $\mathsf{Test}(\mathsf{Com}(m, r), x)$, returns 1 if $m = x$.

Testable and homomorphic commitments should be computationally hiding:

Definition 5 (Testable and Homomorphic Commitment Hiding Property).
Let n be an integer, and let a_0, a_1, r be values in $\mathbb{Z}_n^*$. Then, we say that a testable and homomorphic commitment Com set in a group $\mathbb{G}$ of order n is computationally hiding *over the distribution of r if*

$$\forall a_0, a_1 \in \mathbb{Z}_n^*, \{\mathbb{G}, n, a_0, a_1, \mathsf{Com}(a_0, r)\} \stackrel{c}{\approx} \{\mathbb{G}, n, a_0, a_1, \mathsf{Com}(a_1, r)\}$$

The encryption scheme for small messages due to BGN is very similar to the HW construction, except for two differences. First, we provide the adversary with even less information about the commitment; that is, the values g and h remain private. Secondly, BGN allow and support bilinear map operations, whereas we do not consider them. Similarly to their scheme, the HW testable and homomorphic commitment primitive operates as shown in Figure 3.

Lemma 1. *The testable and homomorphic commitment scheme described in Figure 3 is computationally hiding, i.e., it satisfies definition 5.*

This lemma follows, more or less, from the semantic security of the encryption scheme of Boneh, Goh, and Nissim. For completeness, however, Appendix A.3 will prove that this construction is computationally hiding.

TESTABLE AND HOMOMORPHIC COMMITMENTS OPERATIONS:

1. **Setup:** Let $S(1^k)$ be an algorithm that outputs $(\mathbb{G}, p, q)$ where $\mathbb{G}$ is a group of composite order $n = pq$, and $p < q$ are k-bit primes. Let g, u be random generators of $\mathbb{G}$ and let $h = u^q$. Publish n; keep all else private.
2. **Commit:** Given m and $r \in \mathbb{Z}_n^*$, compute: $\mathsf{Com}(m, r) = g^m h^r$
3. **Addition:** $\mathsf{Com}(m, r) \cdot \mathsf{Com}(m', r') = g^{m+m'} h^{r+r'} = \mathsf{Com}(m + m', r + r')$
4. **Constant Multiplication:** $\mathsf{Com}(m, r)^c = g^{cm} h^{cr} = \mathsf{Com}(cm, cr)$
5. **Equality Test:** If $\mathsf{Test}(\mathsf{Com}(m, r)) = (g^m h^r / g^x)^p = (g^p)^{m-x} = 1$, output 1; else, output 0.

Fig. 3. Testable and homomorphic commitment construction

4.3 Oblivious Polynomial Evaluation

Suppose a party knowing h has some polynomial $f(x) = \sum \alpha_i x^i \in \mathbb{Z}_q[x]$. This party can publish commitments to f's coefficients as $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i}$, where γ_i values are random. Let $s = \lceil \sqrt{n} \rceil$. Assuming p and q are not twin primes, we have that $p < s < q$. Let the group $\mathbb{Z}_s^*$ be the domain of set values. Due to the homomorphic properties of Com, anyone can obliviously evaluate a commitment to $f(z)$ for any $z \in \mathbb{Z}_s^*$.

The HW construction uses this ability by having a verifier $\mathcal{V}$ compute a polynomial f with A as its set of roots. $\mathcal{P}$ can then obliviously evaluate f and return the result to $\mathcal{V}$. Note, this is not a contribution due to HW. Similar constructions were proposed by Naor and Pinkus [17] and FNP [11]. It is also the basis of the KM scheme [14]. $\mathcal{V}$'s polynomial is constructed as shown in Figure 4.

OBLIVIOUS POLYNOMIAL EVALUATION:

1. $\mathcal{V}$ chooses a random constant or irreducible polynomial $G(x)$.
2. $\mathcal{V}$ computes $f(x) = G(x) \cdot (\prod_{a_i \in A}(x - a_i)) = \sum_{i=0}^{|A|} \alpha_i x^i \in \mathbb{Z}_q[x]$.
3. If any $\alpha_i = 0$, restart the protocol.
4. $\mathcal{V}$ chooses a random polynomial $r(x) = \sum_{i=0}^{|A|} \gamma_i x^i \in \mathbb{Z}_p[x]$.
5. $\mathcal{V}$ publishes commitments $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i}$, for $i = 0$ to $|A|$.

Fig. 4. HW oblivious polynomial evaluation

Given these commitments to the α_i coefficients, $\mathcal{P}$ may use the homomorphic operations to compute a commitment to $f(z)$ for an arbitrary point $z \in \mathbb{Z}_s^*$:

$\prod_i \mathsf{Com}(\alpha_i, \gamma_i)^{z^i} = g^{\sum_i \alpha_i z^i} h^{\sum_i \gamma_i z^i} = g^{f(z)} h^{r(z)} = \mathsf{Com}(f(z), r(z))$

Because $\mathcal{P}$ does not want to accidentally reveal information about values $z \notin A$ to $\mathcal{V}$, he can select a random $R \in \mathbb{Z}_n^*$ and compute the value $\mathsf{Com}(Rf(z), Rr(z)) = g^{Rf(z)} h^{Rr(z)} = \mathsf{Com}(f(z), r(z))^R$. If $f(z) \neq 0 \bmod q$, then $Rf(z)$ will be some random value in $\mathbb{Z}_n$, and $\mathsf{Com}(f(z), r(z))^R$ will be some random value in $\mathbb{G}$.

However, if $f(z) = 0 \bmod q$, then $g^{Rf(z)}$ will have order p (or 1). Since h has order p, this means that $\mathsf{Com}(f(z), r(z))^R$ will have order p, which can be tested by $\mathcal{V}$ by checking if the Test operation returns a 1 value. Thus, if $\mathcal{P}$ returns some value with order p, $\mathcal{V}$ concludes that $\mathcal{P}$ obliviously evaluated the polynomial at a root.

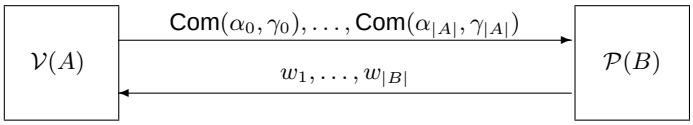

Fig. 5. An illustration of HW private disjointness testing

Recall that $\mathcal{P}$ does not know p, q, or even g or h. To erroneously convince $\mathcal{V}$ that he knows a root, a malicious $\mathcal{P}^*$ must produce some value of order p. Finding such a value is at least as hard as the Subgroup Computation Problem described in Definition 2.

5 HW Private Disjointness Testing

Given the oblivious polynomial evaluation protocol from the previous section, the HW construction to implement Private Disjointness Testing with a testable and homomorphic commitment primitive is quite simple. As mentioned, the overall protocol paradigm originally proposed by FNP [11]. Figure 5 illustrates the HW private disjointness testing protocol that is specified in Figure 6.

HW PRIVATE DISJOINTNESS TESTING:

1. $\mathcal{V}$ runs $S(1^k)$ to obtain $(\mathbb{G}, p, q)$, selects random generators g, u in $\mathbb{G}$, and computes $n = pq$ and $h = u^q$.
2. $\mathcal{V}$ publishes $(\mathbb{G}, n)$.
3. $\mathcal{V}$ and $\mathcal{P}$ announce $|A|$ and $|B|$ for respective input sets A and B, which are $\mathrm{poly}(k)$-sized subsets of $\mathbb{Z}_s^*$.
4. $\mathcal{V}$ publishes commitments to polynomial coefficients $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i} \in \mathbb{G}$ for $i = 0$ to $|A|$.
5. For each $b_j \in B$ selected in random order:
   (a) $\mathcal{P}$ obliviously evaluates $f(b_j)$ as $v_j = g^{f(b_j)} h^{r(b_j)}$.
   (b) $\mathcal{P}$ selects a random exponent $R_j \in \mathbb{Z}_n^*$.
   (c) $\mathcal{P}$ sends $\mathcal{V}$ the value $w_j = v_j^{R_j}$.
6. $\mathcal{V}$ halts if any $w_j = 1$.
7. $\mathcal{V}$ tests each w_j by computing w_j^p.
8. If any $w_j^p = 1$, then $\mathcal{V}$ concludes that $A \cap B \neq \emptyset$.
9. Otherwise, $\mathcal{V}$ concludes $A \cap B = \emptyset$.

Fig. 6. HW private disjointness testing

Theorem 1. *The HW construction is correct and secure, i.e., it satisfies Definition 3, under the Subgroup Decision and the Subgroup Computation assumptions.*

Remark: Note that when talking to an honest prover, a verifier will actually learn $|A \cap B|$ in this protocol by counting the number of elements returned with order p. We could change the protocol to obfuscate this value, but having the prover return a random number of copies of each element in his set. This would not be true zero-knowledge, but it would be good enough for many practical applications. After all, there is no guarantee that the number of elements with order p is the correct cardinality, because a malicious prover might evaluate the same value many times. This protocol can be modified to hide $|A \cap B|$ at a cost of

increased communication as discussed in Section 7.3. In many PDT applications this extra information is not a problem, but users should still be aware that cardinality is revealed when provers are honest.

5.1 Proof of Security

Theorem 1 is proven in four steps: completeness, soundness, malicious-prover zero knowledge, and honest-verifier zero knowledge. These proofs appear in Appendix A due to space considerations.

6 Semi-Honest Private Intersection Cardinality

The construction in Section 4 is *not* an Honest-Verifier Private Intersection Cardinality protocol. Unfortunately, there are trivial ways a malicious-prover can manipulate the actual cardinality value obtained by the verifier. The simplest attack would be to obliviously evaluate each element in B twice. The verifier will think the cardinality is $2 \cdot |A \cap B|$. By the HVPZK property, an honest verifier cannot detect this attack, otherwise it could distinguish different evaluations by the prover.

For this reason, the HW construction violates the Cardinality Soundness property from definition 4. However, we may consider a weaker PIC setting by assuming that both the prover and verifier are honest-but-curious (semi-honest). Recall that a honest-but-curious party will follow a protocol as specified, but may further examine any received values with the intention of learning more [12].

Definition 6 (Semi-Honest Private Intersection Cardinality). *An* Semi-Honest Intersection Cardinality *protocol has the same setup as in Definition 3, except for the following difference:*

Completeness: *For semi-honest parties, the protocol works and the verifier learns the cardinality predicate; that is,*

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = |A \cap B|\right] \geq (1 - \mathrm{negl}(k))$$

where probability is taken over the randomness of $\mathcal{P}$ and $\mathcal{V}$.

Corollary 1. *The HW construction from Section 4 implements a Semi-honest Private Intersection Cardinality Protocol, under the Subgroup Decision and the Subgroup Computation assumptions.*

Corollary 1 follows directly from the proof of Theorem 1.

7 Discussion

7.1 Malicious Verifiers

The HW construction is only secure against honest-but-curious verifiers. A malicious verifier $\mathcal{V}^*$ can choose arbitrary setup parameters $(\mathbb{G}, n)$, such as $\mathbb{G} = \mathbb{Z}_{p'}$ where $p' = 2n + 1$, and send $\mathcal{P}$ an arbitrary set of values $g^{c_i} \in \mathbb{G}$, where the c_i values define some polynomial $f(x) = \sum c_i x^i$. In response, a legitimate $\mathcal{P}$ will send values $w = g^{Rf(b)}$ for each $b \in B$, where R is chosen uniformly at random from $\mathbb{Z}_n^*$.

If $g^{f(b)}$ has order n, then w will be a random element of order n. However, a malicious $\mathcal{V}^*$ can design the polynomial $f(\cdot)$ to have different orders for different inputs. So, if $p' = 2pq + 1$, $\mathcal{V}^*$ might have two sets S, T such that $\forall s \in S, f(s) = 0 \bmod p$ and $\forall t \in T, f(t) = 0 \bmod q$. Thus, $\mathcal{V}^*$ would be able to distinguish how many elements of B were in either S or T. In fact, $\mathcal{V}^*$ could choose n to have many factors. This would allow her to test how many elements of B belonged to any of several different sets.

To make the HW construction secure against malicious verifiers, $\mathcal{V}$ could provide a zero knowledge proof that n was the product of two large primes p and q. $\mathcal{V}$ could then include a proof that each of her commitments was the product of at least one value with order p. Camenisch and Michels describe efficient zero knowledge proofs which can be applicable in this setting [4]. Of course, the costs of creating and verifying these proofs may be equivalent to the costs of the existing malicious verifier-secure protocols due to FNP and KM.

7.2 Computation and Communication Costs

The computation and communication costs of the HW construction are equivalent to the costs of FNP's malicious-prover secure scheme, except the HW construction offers security against malicious provers without random oracles. The costs of HW are as follows:

$\mathcal{V}$ **Computation Costs:** Computing α_i coefficients naively requires $O(|A|^2)$ modular additions and multiplications. Committing requires $O(|A|)$ modular exponentiations and multiplications. Testing whether responses have order p requires $O(|B|)$ modular exponentiations.

$\mathcal{P}$ **Computation Costs:** Using Horner's method, $\mathcal{P}$ can obliviously evaluate a d-degree polynomial with $O(d)$ modular exponentiations and multiplications. Normally, $\mathcal{P}$ will perform $O(|A||B|)$ operations; that is, one polynomial evaluation at a cost of $O(|A|)$ operations for each of the $|B|$ elements in $\mathcal{P}$'s set. However, as described in FNP, if the balanced hash-bucket scheme of Azar et al. [1] is employed $\mathcal{P}$ can perform only $O(|B| \ln \ln |A|)$ modular operations.

Communication Costs: The total exchange between $\mathcal{P}$ and $\mathcal{V}$ is $O(k(|A| + |B|))$ bits or alternatively $O(k(|A| \ln \ln |A| + |B|))$ if a hash-bucket optimization is used, where 1^k is the security parameter.

7.3 Hiding Set Sizes

In the HW construction, the size of the prover and verifier's sets is public information. In practice, however, the prover $\mathcal{P}$ with set B or the verifier $\mathcal{V}$ with set A might wish to mask the true size of their sets using well-known techniques. To do this, the verifier $\mathcal{V}$ can compute a random polynomial $f(\cdot)$ with roots in set A as normal, then multiply it by some irreducible polynomial of arbitrary degree d. Then, $\mathcal{P}$ (or anyone else) will only learn that $\mathcal{V}$'s set is of some size less or equal to $|A| + d$. Similarly, $\mathcal{P}$ can evaluate f on each value in B an arbitrary number of times. Each copy will be randomized by the regular protocol. This will maintain correctness of Private Disjointness Testing, but would obviously change the results of an honest-but-curious private intersection cardinality protocol, as described in Section 6.

7.4 Small Set Domains

The HW construction requires that sets A and B are small with respect to the domain of set values. Obviously, in the HW PDT protocol, if $|B| = \Theta(\sqrt{n})$, then a malicious adversary can factor n in time polynomial to the size of its input. This would allow an adversary to generate values of order p and violate the Soundness property.

7.5 Private Information Retrieval

Recalling Private Information Retrieval (PIR), one party will have a database of $m + 1$ bits $x_0, \ldots, x_m$, while a second party wishes to privately query a particular bit x_i without revealing i. Putting this in the

context of the HW construction, A would be the set of indices where x is 1 and $B = \{i\}$. Unfortunately, it may be the case that $|A|$ is large with respect to the domain $\mathbb{Z}_m^*$.

As a result, the requirement of small set domains mentioned in Section 7.4 precludes directly using the HW construction for PIR in general. Yamamura and Saito offer a simple PIR solution based on the SDA [21]. However, their PIR solution approach is very inefficient and requires $O(km)$ bits of communication to privately retrieve a single bit from a m-bit database, where k is a security parameter.

7.6 Multiparty Extensions

Another interesting variant to the 2-party PDT protocol is considering a multi-verifier, single-prover PDT scenario. For example, suppose that law enforcement agencies from different countries, in the role of verifiers, wish to be assured by an airline, in the role of the prover, that no one on *any* of their watch-lists is getting on the next flight. The law enforcement agencies neither trust each other nor the airline with their individual databases, yet may want to corroborate their watch lists (so as to possibly work together).

Suppose there are two verifiers. The HW construction may be extended as follows. First, each verifier computes his own values $n_i = p_i q_i$ and a group of known order $\prod_i n_i$ is published. Next, both verifiers publish commitments to their own polynomials using a random generator g from the group of order $n_1 n_2$ and, respectively, h_1 of order $(n_1 n_2)/p_1 = q_1 n_2$ and h_2 order $(n_1 n_2)/p_2 = n_1 q_2$. That is, values of the form $g^{\alpha_i} h_1^{r_i}$ and $g^{\beta_j} h_2^{r_j}$, where $f(x) = \sum \alpha_i x^i$ and $z(x) = \sum \beta_j x^j$. A third party can obliviously evaluate commitments to the sum of these polynomials. If the third party's set contains an element c_i such that $f(c_i) = z(c_i) = 0$, then this party can output elements $h_1^r h_2^{r'}$, which have order $q_1 q_2$.

This is interesting because no single party could compute elements of order $q_1 q_2$ by themselves; this only occurs when the airline makes an evaluation on an element contained in *both* of the law enforcement agencies' sets. Each agency, knowing q_1 and q_2 respectively, could collaborate to detect this fact and take further action. The benefit here is that the contents of the sets of the law enforcement agencies and the airline all remain private, up to knowledge of any three-way intersections. This digression is just to illustrate that unknown order subgroups might be applied in other interesting applications.

7.7 Finding Intersection Values with HW

As previously mentioned, basic FNP is actually a Private Intersection or Private Matching protocol. The verifier party learns which specific values are in the set intersection. Essentially, the prover will send homomorphic encryptions of the form $E_{pk}(r \cdot f(b) + b)$ for values $b \in B$. If $b \in A$, then $f(b) = 0$ and the verifier will receive an encryption of b. Otherwise, the verifier receives a random value.

Of course, this is still susceptible to malicious prover attacks. A malicious prover can encrypt any value he likes or can encrypt values like $E_{pk}(r_1 \cdot f(b_1) + r_2 \cdot f(b_2) + b_1)$, which can be interpreted as "If $(b_1 \in A)$ and $(b_2 \in A)$, then tell the verifier that $(b_1 \in A)$". FNP's fixes the problem by using the random oracle model to force a prover to use the encrypted coefficient values prepared by the verifier.

This begs the question of whether the HW testable and homomorphic commitment primitive could be used in a private intersection protocol. Initially, one may consider using the exact FNP construction and having the prover obliviously evaluate $g^{Rf(b)+b} h^r$. If $f(b) = 0$, raising this to the power q will result in the value $(g^q)^b$. The verifier can then check whether for any of its own values a, that $(g^q)^a = (g^q)^b$.

Unfortunately, like FNP, a malicious prover could also send conditional evaluations, like "if x is in A, then reveal that y is in B". This would violate the soundness of a private intersection protocol. Thus, a HW-style private intersection protocol offers no advantage over FNP. They have equivalent computation costs and the same level of security.

An open question is whether a HW testable and homomorphic commitment-based private intersection protocol may be constructed without the use of random oracles. It may be possible to leverage groups that have several different ordered subgroups as discussed in Section 7.6. The verifier might be able to commit A in a polynomial f that has roots in different subgroups. Then a malicious prover would not be able simply to mix-and-match oblivious evaluations of different b values. This idea is not fully developed and can be the subject of future work.

8 Conclusion and Open Questions

We presented an honest-verifier private disjointness testing protocol, which we refer to as HW. It is secure against malicious provers without requiring multiple invocations, bilinear groups, random oracles, non-interactive zero knowledge proofs, or universally-composable commitments. The related FNP and KM protocols require one or more of these assumptions to be made secure against malicious provers, while HW requires only the subgroup decision and subgroup computation assumptions.

There are several open questions and problems related to HW. First, disjointness testing is a fairly limited application. An open question is whether there are natural constructions of private set operations like union or intersection based on the subgroup assumptions. It is likely that several of the FNP-inspired privacy-preserving set operations due to Kissner and Song [15] may be adapted using this paper's testable and homomorphic commitment primitive.

The testable and homomorphic commitment based on the subgroup assumptions and described in Section 4.2 may be useful in other applications. Essentially, this commitment scheme allows one party to obliviously evaluate functions, and another party to test properties of the evaluation. In this paper there was a single property that was tested – whether a polynomial evaluated to zero.

However, testable and homomorphic commitment may be especially useful with groups with many subgroups of unknown order, as described in Section 7.6. This would allow a party to test several properties of an evaluation, or even several parties to test different properties independently. As discussed at the end of Section 7.7, groups with many subgroups of different order might be useful in developing a HW testable and homomorphic commitment-based private intersection protocol – where the verifier learns the actual values of the intersection.

More research must be focused on the hardness of both the subgroup decision and subgroup computation assumptions. First, does one assumption imply the other? Second, what is the relation, if any, between these assumptions and the Decisional or Computation Diffie-Hellman assumptions? Either subgroup assumption implies that factoring is hard, otherwise someone could just factor a group's order to obtain the order of its subgroups. Does assuming that factoring is hard imply either of the subgroup decision assumptions? These are all important questions that are relevant to both the HW construction and other works based on subgroup decision assumptions.

References

[1] AZAR, Y., BRODER, A. Z., KARLIN, A. R., AND UPFAL, E. Balanced allocations. *SIAM Journal on Computing 29*, 1 (1999), 180–200.

[2] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Computer and Communications Security – CCS* (1993), ACM Press, pp. 62–73.

[3] BONEH, D., GOH, E.-J., AND NISSIM, K. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography (TCC)* (2005), J. Kilian, Ed., vol. 3378 of *Lecture Notes in Computer Science*, Springer, pp. 325–341.

[4] CAMENISCH, J., AND MICHELS, M. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology – EUROCRYPT'99* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 107–122.

[5] CAMENISCH, J., AND SHOUP, V. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology – CRYPTO '03* (2003), D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 126–144.

[6] CANETTI, R., AND FISCHLIN, M. Universally composable commitments. In *Advances in Cryptology – CRYPTO '01* (2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, Springer, pp. 19–40.

[7] CANETTI, R., GOLDREICH, O., AND HALEVI, S. The random oracle methodology, revisited. *Journal of the ACM 51*, 4 (July 2004), 557–594.

[8] CANETTI, R., LINDELL, Y., OSTROVSKY, R., AND SAHAI, A. Universally composable two-party and multi-party secure computation. In *Symposium on Theory of Computation – STOC* (2002), J. H. Reif, Ed., ACM Press, pp. 495–503.

[9] CRAMER, R., GENNARO, R., AND SCHOENMAKERS, B. A secure and optimally efficient multi- authority election scheme. In *Advances in Cryptology – EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer, pp. 103–118.

[10] DAMGÅRD, I., AND NIELSEN, J. B. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology – CRYPTO '02* (2002), M. Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, Springer, pp. 581–596.

[11] FREEDMAN, M. J., NISSIM, K., AND PINKAS, B. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT '04* (May 2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 1–19.

[12] GOLDREICH, O., MICALI, S., AND WIGDERSON, A. How to play any mental game. In *Symposium on Theory of Computation – STOC* (January 1987), ACM Press, pp. 218–229.

[13] GROTH, J., OSTROVSKY, R., AND SAHAI, A. Perfect non-interactive zero knowledge for NP. In *Advances in Cryptology – EUROCRYPT '06* (To appear 2006), Springer.

[14] KIAYIAS, A., AND MITROFANOVA, A. Testing disjointness of private datasets. In *Financial Cryptography* (2005), A. S. Patrick and M. Yung, Eds., vol. 3570 of *Lecture Notes in Computer Science*, Springer, pp. 109–124.

[15] KISSNER, L., AND SONG, D. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO '05* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 241–257.

[16] MICALI, S., RABIN, M., AND KILIAN, J. Zero-knowledge sets. In *Foundations of Computer Science – FOCS* (2003), M. Sudan, Ed., IEEE Press, pp. 80–91.

[17] NAOR, M., AND PINKAS, B. Oblivious transfer and polynomial evaluation. In *Symposium on Theory of Computation – STOC* (May 1999), T. Leighton, Ed., ACM Press, pp. 245–254.

[18] PAILLIER, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99* (May 1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238.

[19] PAILLIER, P. Trapdooring discrete logarithms on elliptic curves over rings. In *Advances in Cryptology – ASIACRYPT '00* (2000), T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, Springer, pp. 573–584.

[20] PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91* (1991), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer, pp. 129–140.

[21] YAMAMURA, A., AND SAITO, T. Private information retrieval based on the private information retrieval based on subgroup membership problem. In *Australasian Conference on Information Security and Privacy* (2001), V. Varadharajan and Y. Mu, Eds., vol. 2119 of *Lecture Notes in Computer Science*, Springer, pp. 206–220.

[22] YAO, A. C. How to generate and exchange secrets. In *Foundations of Computer Science – FOCS* (1986), IEEE Press, pp. 162–167.

A Security Proofs

Theorem 1 is proven in four steps: completeness, soundness, malicious-prover zero knowledge, and honest-verifier zero knowledge.

A.1 Proof of Completeness

Recall the PDT completeness property:

$$\forall A \in U,\ \forall B \in U,\ \Pr\left[\mathcal{P}(B)\mathcal{V}(A) = D(A,B)\right] \geq (1 - \mathrm{negl}(k))$$

In the oblivious polynomial evaluation protocol discussed in Section 4.3, a verifier $\mathcal{V}$ sends $(|A|+1)$ different $\mathsf{Com}(\alpha_i, \gamma_i)$ values to a prover $\mathcal{P}$, where $f(x) = \sum \alpha_i x^i$ and $r(x) = \sum \gamma_i x^i$. $\mathcal{P}$ will respond with $|B|$ values $w_j = \mathsf{Com}(f(b_j), r(b_j))^{R_j}$ for random R_j. If the protocol is followed correctly, then with high probability $b_j \in A$ if and only if $ord(w_j) = p$.

Case 1: $b_j \in A \implies ord(w_j) = p$ w.h.p. over $r(x) \in \mathbb{Z}_p[x]$.

If $b_j \in A$ then $f(b_j) = kq$ for some k. Then we have that the value $\mathsf{Com}(f(b_j), r(b_j))^{R_j} = (g^{kq}h^{r(b_j)})^{R_j}$. Recalling that $n = pq$, $g^n = 1$, and that $h^p = 1$, we have that the value $w_j^p = (g^{kn}h^{r(b_j)p})^{R_j} = 1$.

Note that there is also a negligible chance that $R_j kq = R_j r(b_j) = 0 \bmod p$, i.e. $w_j = 1$. This will cause $\mathcal{V}$ to halt the protocol. Since $R_j \in \mathbb{Z}_n^*$, this would mean that $k = r(b_j) = 0 \bmod p$.

Recall that $r(x)$ is chosen uniformly at random from $\mathbb{Z}_p[x]$. Thus, the chance that a particular b_j is a root is at most $|A|/p$. The chance that none of the $|B|$ values are roots of $r(x)$ is:

$$\left(1 - \frac{|A|}{p}\right)^{|B|} = \left(\left(1 - \frac{1}{\frac{p}{|A|}}\right)^{\frac{p}{|A|}}\right)^{\frac{|A||B|}{p}} \approx e^{-\left(\frac{|A||B|}{p}\right)} > e^{-\left(\frac{\mathrm{poly}(k)}{2^k}\right)} > 1 - \mathrm{negl}(k)$$

Case 2: $b_j \notin A \implies ord(w_j) \neq p$.

If $b_j \notin A$, then $f(b_j) \neq 0 \bmod q$. So, $w_j^p = g^{R_j f(b_j) p} h^{R_j r(b_j) p} = g^{R_j f(b_j) p}$. Note that $R_j \in \mathbb{Z}_n^*$, so $R_j f(b_j) \neq 0 \bmod q$. Therefore, we have $R_j f(b_j) p \neq 0 \bmod n$ and can conclude that $w_j^p \neq 1$.

A.2 Proof of Soundness

Recall the PDT soundness property:

$$\forall \mathcal{P}^*_{ppt},\ \Pr_{A \in U}\left[\mathcal{P}^*\mathcal{V}(A) \neq 0\right] \leq \mathrm{negl}(k)$$

The verifier $\mathcal{V}$ will only accept when one of the $\mathrm{poly}(k)$ values sent to $\mathcal{V}$ by $\mathcal{P}^*$ has order p. Recall that (even a malicious) $\mathcal{P}^*$ only knows $(\mathbb{G}, n)$ and $\mathcal{V}$'s commitments. It does not know p, q, h or even g.

In the given soundness definition, $\mathcal{V}$ is operating with a *random* set $A \in U$. $\mathcal{P}^*$ has no *a priori* knowledge about A, other than $|A|$. By testable and homomorphic commitment hiding property, $\mathcal{P}^*$ can learn nothing about A from the commitments themselves.

It is worth elaborating on this particular definition of soundness. It assumes that a malicious prover knows nothing about the set he is trying to cause a spurious intersection with. Otherwise, if he has some partial knowledge, say that some element is in the set A with a $1/1000$ chance, a malicious prover could

cause a spurious intersection with with $1/1000$ probability. Our formulation captures the notion that malicious provers should not be able to trick verifiers that an intersection exists without the use of some previous knowledge of A.

Alternatively, one could formulate this as an experiment where an adversary $\mathcal{P}^*$ chooses a set B^* as input for an honest PDT prover $\mathcal{P}$. Any partial knowledge of A could be embedded in B^*. The probability that $\mathcal{V}$ believes there is an intersection interacting with $\mathcal{P}(B^*)$ would be non-negligibly different than when interacting directly with $\mathcal{P}^*$. However, the soundness formulation as given is clearer and captures the same properties.

With no information on A, $\mathcal{P}^*$ can try to evaluate $f(x)$ at $|B^*| = \text{poly}(k)$ random values and will fail to guess a member of A with probability approximately $e^{(-|A||B^*|/p)}$ which is greater than $1 - \text{negl}(k)$. Note by the Subgroup Decision Assumption, $\mathcal{P}^*$ won't actually be able to verify when he correctly guesses a value in A.

There is one caveat concerning the distribution of α coefficients. It could be the case that some coefficient, or linear combination of coefficients, has a non-negligible chance of being zero. Note that a zero coefficient corresponds to a commitment of the form $\mathsf{Com}(0, r_i) = g^0 h^{r_i}$. In other words, the commitment corresponding to a zero coefficient has order p. Thus, all $\mathcal{P}^*$ would have to do to break soundness is return this commitment to $\mathcal{V}$. The same applies if some linear combination of coefficients is zero with non-negligible probability.

To avoid this issue, α_i values are checked when $f(x)$ is created to ensure they are non-zero. Recall that to generate $f(x)$ the verifier will choose a random constant or irreducible polynomial $G(x)$ and multiply it by $\prod(x - a_i)$. If one $G(x)$ fails, another random irreducible polynomial can be chosen until all α_i values are non-zero. For each iteration, there is a high probability that no coefficients will be zero, so with high probability a constant number of iterations will be necessary.

Since the α_i coefficients are determined by some random irreducible polynomial $G(x)$, and in this case, a random set A, they are unpredictable to a prover $\mathcal{P}^*$. A malicious prover cannot send any trivial linear combinations of committed coefficients back to the verifier since the coefficients are determined entirely by a random A and $G(x)$.

Thus, $\mathcal{P}^*$ essentially must try to generate order p values directly from $(\mathbb{G}, n)$. We will use the Subgroup Computation Assumption (SCA), from Definition 2. The SCA asserts that it is difficult for a polynomial-time adversary, given the description of an efficiently sample-able group $\mathbb{G}$ of order $n = pq$, to find an element of order p. We will show that any adversary $\mathcal{A}$ that violates Soundness also violates the SCA.

Suppose we have some adversary $\mathcal{A}$ that violates Soundness. $\mathcal{A}$ would take $(\mathbb{G}, n)$ and $\mathcal{V}$'s commitments as input and would have a non-negligible probability of returning an element of order p. By the MPZK property, $\mathcal{A}$'s behavior cannot change significantly when we substitute $\mathcal{V}$'s commitments with random values.

Given such an adversary $\mathcal{A}$ and setup $(\mathbb{G}, n)$, we could feed it random values sampled from $\mathbb{G}$ and obtain an element of order p. Obviously, this violates the SCA. Thus by the facts that A is chosen uniformly at random, that f has a negligible probability of having zero coefficients, and by the SCA, the Soundness property from the PDT definition holds.

A.3 Proof of PDT Malicious-Prover Zero Knowledge

Recall the PDT Malicious-Prover Zero Knowledge (MPZK) property:

$$\exists \mathcal{S}_{ppt},\ \forall \mathcal{P}^*_{ppt},\ \forall A \in U,\ \{\mathsf{View}^{\mathcal{P}^*}\left[\mathcal{P}^* \mathcal{V}(A)\right]\} \overset{c}{\approx} \{\mathsf{View}^{\mathcal{P}^*}\left[\mathcal{P}^* \mathcal{S}(1^{|A|})\right]\}$$

The proof that a malicious-prover $\mathcal{P}^*$ is not able to glean any information about A from the values $\mathsf{Com}(\alpha_i, \gamma_i)$ follows from the hiding property of the testable and homomorphic commitment scheme from Lemma 1 (which in turn essentially follows from the semantic security of the BGN cryptosystem [3].) The proof will argue this step, then describe a zero-knowledge simulator.

First, the Subgroup Decision Assumption (SDA), from Definition 1, implies that Com is computationally hiding. Suppose the contrapositive, that Com is not hiding. Let $\mathcal{A}$ be a probabilistic polynomial time adversary that runs the indistinguishability under chosen plaintext attack (IND-CPA) experiment shown in Figure 7 for a given input $(\mathbb{G}, n)$ where $g, h \in \mathbb{G}$ and h has order p.

$\text{Exp}_{\mathcal{A}}^{IND-CPA}(\mathbb{G}, n, g, h)$:

1. $\mathcal{A}(\mathbb{G}, n) \rightarrow (a_0, a_1)$
2. $t \leftarrow \mathbb{Z}_n^*$
3. $b \leftarrow \{0, 1\}$
4. $\mathcal{A}(\mathbb{G}, n, g^{a_b} h^t) \rightarrow b'$

Fig. 7. Testable and homomorphic commitment IND-CPA experiment

Suppose, for the sake of contradiction, that $\Pr[b = b'] \geq 1/2 + 1/\text{poly}(k)$. If this is the case, we can construct an adversary $\mathcal{A}^*$ that violates the SDA assumption. To do so, given same input $(\mathbb{G}, n)$ and a challenge $x \in \mathbb{G}$, the adversary $\mathcal{A}^*$ must be able to distinguish whether $x = g^r$ or g^{rq} for $r \in \mathbb{Z}_n^*$. (We ignore the case where $x = 1$.)

Now, $\mathcal{A}^*$ selects a random generator $g \in \mathbb{G}$ and runs $\text{Exp}_{\mathcal{A}}^{IND-CPA}(\mathbb{G}, n, g, x)$ with the adversary $\mathcal{A}$. If $x = g^{rq}$ then it will have order p, and thus the $\mathcal{A}^*$ simulates $\text{Exp}_{\mathcal{A}}^{IND-CPA}$ perfectly for $\mathcal{A}$, so $\mathcal{A}$ will maintain a $1/2 + 1/\text{poly}(k)$ advantage.

If $x = g^r$, in step 4 of $\text{Exp}_{\mathcal{A}}^{IND-CPA}$, the adversary $\mathcal{A}$ will receive the value $g^{a_0} g^{rt}$ (w.l.o.g.). Because r and t are chosen uniformly at random from $\mathbb{Z}_n^*$, it's equally possible that $\mathcal{A}$ has received the value $g^{a_1} g^{r't'}$, where $r't' = rt + a_0 - a_1$. Thus, $\mathcal{A}$ has necessarily a $1/2$ chance of guessing b when $x = g^r$.

By running repeated $\text{Exp}_{\mathcal{A}}^{IND-CPA}(\mathbb{G}, n, g, x)$ experiments, $\mathcal{A}^*$ can observe $\mathcal{A}$'s performance at guessing b. If $\mathcal{A}$ displays a $1/2+1/\text{poly}(k)$ advantage, $\mathcal{A}^*$ will guess that $x = g^{rq}$. If $\mathcal{A}$ has no advantage, then $\mathcal{A}^*$ will guess that $x = g^r$. Thus, $\mathcal{A}^*$ can distinguish distributions of $(\mathbb{G}, n, g^{rq})$ from $(\mathbb{G}, n, g^r)$, violating the SDA.

Therefore, if the SDA holds, then the commitment scheme Com is computationally hiding. A malicious adversary cannot distinguish a commitment to a particular a_0 from a commitment to a random message. Based on this, we will construct a simulator $\mathcal{S}(1^{|A|})$ that will be indistinguishable from a verifier $\mathcal{V}(A)$ to any probabilistic polynomial time $\mathcal{P}^*$.

$\mathcal{S}$ is quite simple: it will send $|A|$ random values in $\mathbb{G}$ to $\mathcal{P}^*$. Because random values are individually indistinguishable from commitments to particular α_i coefficients, $\mathcal{P}^*$ will not be able to distinguish $\mathcal{S}(1^{|A|})$ from $\mathcal{V}(A)$. Thus, by the SDA, the Malicious-Prover Zero Knowledge property holds.

A.4 Proof of Honest-Verifier Perfect Zero Knowledge

Recall the PDT Honest-Verifier Perfect Zero Knowledge (HVPZK) property:

$$\exists \mathcal{S}_{ppt},\ \forall A \in U,\ \forall B \in U,\ \{\mathsf{View}^{\mathcal{V}}\big[\mathcal{P}(B)\mathcal{V}(A)\big]\} \approx \{\mathcal{S}(A, 1^{|B|}, 1^{|A \cap B|})\}$$

The HVPZK property implies that an adversary given $\mathsf{View}^{\mathcal{V}}[\mathcal{P}(B)\mathcal{V}(A)]$ cannot learn anything about B that it could not learn given $|A \cap B|$ and $|B|$. The quantification is over all choices of input A, which includes adversarial choices of A that might be based on some prior knowledge or some partial knowledge of a particular set B. By running the legitimate protocol, however, a semi-honest $\mathcal{V}$ should not learn anything beyond what is can conclude from the size of $|A \cap B|$.

Because the adversary is semi-honest, it cannot deviate from the protocol, manipulate its choice of $(\mathbb{G}, n)$, or manipulate its committed coefficients. Its power is equivalent to choosing a query set A and running the legitimate verifier $\mathcal{V}$. We will describe a simulator $\mathcal{S}$, that on inputs $(A, 1^{|B|}, 1^{|A \cap B|})$ produces a view that is perfectly indistinguishable from $\mathsf{View}^{\mathcal{V}}[\mathcal{P}(B)\mathcal{V}(A)]$.

HVPZK SIMULATOR:

1. $\mathcal{S}$ runs $S(1^k)$ to obtain $(\mathbb{G}, p, q)$, selects random generators g, u in $\mathbb{G}$, and computes $n = pq$ and $h = u^q$.
2. $\mathcal{S}$ publishes $(\mathbb{G}, n)$.
3. $\mathcal{S}$ announces $|A|$ and $|B|$.
4. $\mathcal{S}$ publishes committed polynomial coefficients $\mathsf{Com}(\alpha_i, \gamma_i) = g^{\alpha_i} h^{\gamma_i}$ in $\mathbb{G}$ for $i = 0$ to $|A|$, exactly as $\mathcal{V}$ would.
5. $\mathcal{S}$ ignores the values from step 3 and generates $|B|$ random elements of $\mathbb{G}$, raises a random selection of $|A \cap B|$ of these values to q (thereby making them of order p), leaves the other $|B| - |A \cap B|$ values as is, and outputs these $|B|$ elements as the response of $\mathcal{P}$.

Fig. 8. Honest-verifier perfect zero knowledge simulator

The HVPZK simulator works as shown in Figure 8 (recall the PDT protocol from Section 5). This simulator perfectly generates the distribution $\mathsf{View}^{\mathcal{P}}[\mathcal{P}(B)\mathcal{V}(A)]$. Here, $\mathcal{S}$ follows the same setup as $\mathcal{V}$ and $\mathcal{P}$ in steps 1, 2, and 3. In step 4, the response of the simulated prover is $|A \cap B|$ *random* values of order p in $\mathbb{G}$ and $|B| - |A \cap B|$ *random* values in $\mathbb{G}$. This is exactly the distribution than an honest prover would return. Thus, the two views are perfectly indistinguishable and the HVPZK property holds.