Testing Disjointness of Private Datasets

Aggelos Kiayias[1,⋆] and Antonina Mitrofanova[2]

[1] Computer Science and Engineering,
University of Connecticut Storrs, CT, USA
aggelos@cse.uconn.edu

[2] Computer Science, Rutgers University,
New Brunswick, NJ, USA
amitrofa@cs.rutgers.edu

Abstract. Two parties, say Alice and Bob, possess two sets of elements that belong to a universe of possible values and wish to test whether these sets are disjoint or not. In this paper we consider the above problem in the setting where Alice and Bob wish to disclose no information to each other about their sets beyond the single bit: "whether the intersection is empty or not." This problem has many applications in commercial settings where two mutually distrustful parties wish to decide with minimum possible disclosure whether there is any overlap between their private datasets. We present three protocols that solve the above problem that meet different efficiency and security objectives and data representation scenarios. Our protocols are based on Homomorphic encryption and in our security analysis, we consider the semi-honest setting as well as the malicious setting. Our most efficient construction for a large universe in terms of overall communication complexity uses a new encryption primitive that we introduce called "superposed encryption." We formalize this notion and provide a construction that may be of independent interest. For dealing with the malicious adversarial setting we take advantage of recent efficient constructions of Universally-Composable commitments based on verifiable encryption as well as zero-knowledge proofs of language membership.

1 Introduction

Suppose that Alice and Bob, wish to test whether their stock-portfolios share any common stocks. Nevertheless they are mutually distrustful and they are reluctant to reveal the contents of their portfolios to each other or to a third party. More generally, Alice and Bob possess two datasets drawn from a universe of publicly known values and wish to find out whether the intersection of their sets is empty or not with the minimum possible disclosure of information: only a single bit should become known, "whether the two datasets are disjoint or not." We call this a *private disjointness test.*

⋆ Research partly supported by NSF CAREER Award CNS-0447808.

A private disjointness test is a useful primitive in various online collaboration procedures. Indeed, depending on the disjointness of their two datasets, Alice and Bob may then proceed to different courses of action, e.g., in the case of a positive answer, Alice and Bob may seek further authorization and proceed to actually compute the intersection of their two datasets; on the other hand, in case of a negative outcome Alice and Bob may terminate their negotiation. The minimum disclosure property of a single bit output that we require is optimal from the point of view of decision-making based on private collections of data.

Problem Formulation. Let us formulate the problem that we put forth in more concrete terms: the two participants of a private disjointness test, Alice and Bob (and henceforth called A and B) possess two subsets S_A and S_B of the universe Ω; they wish to extract the bit "$S_A \bigcap S_B \overset{?}{=} \emptyset$" without disclosing any information about S_A, S_B to each other (except the sizes of S_A and S_B that are publicly known). We will make the abstraction that $\Omega = \{1, \ldots, N\}$ as for a publicly known universe set Ω it is sufficient for the two parties to use the indices of the actual elements that are contained in their datasets instead of the elements themselves (using a predetermined ordering). Each dataset may be represented in two possible ways: either using its characteristic bitstring (and thus the two players in this case possess inputs of length N bits) or using a list of indices listed explicitly i.e., the length of each player's input $N_X \log N$ where $N_X = \#S_X$ (the number of elements of the X player's dataset) where $X \in \{A, B\}$.

A protocol solution for a private disjointness test will be called a Private Intersection Predicate Evaluation (PIPE) protocol. Given a correct PIPE protocol, we will be interested in various objectives such as (i) the level of security of each player against the other, (ii) minimization of the total communication and time complexity based on the input sizes of the two players, (iii) minimization of the number of rounds of interaction between the two players. Note that we will consider only PIPE protocols where one of the two players (selected to be A) has output. This is a standard assumption and consistent with a client-server type of interaction between the two parties. Naturally players may execute a PIPE protocol changing roles if output is desired in both sides.

Our Results. We present three PIPE protocols that satisfy different aspects of the objectives above; PIPE protocol #1 is suitable for settings where each player represents its input as a characteristic bitstring (this setting is suitable when the universe is not substantially larger than the size of the datasets); our protocol is very efficient and achieves communication complexity linear in N in only a single round of interaction between the two players (we note that this communication is optimal for the general case of the disjointness problem, cf. the overview of related work in the following paragraph). The setting where datasets are substantially smaller compared to the universe size and one wishes to construct protocols that are sublinear in N is more challenging. Our PIPE protocol #2 applies to this setting and operates efficiently with $N_A \times N_B$ total communication using N_B rounds of interaction. Finally, our PIPE protocol #3, operating in the same setting as the second protocol, requires more communication, proportional to $\binom{N_A+N_B}{N_B}$ but reduces the number of rounds to a single round of interaction between the two players.

It should be stressed that protocols such as PIPE #2 and PIPE #3, reveal the size of the datasets of the players while PIPE #1 does not. This seems unavoidable if one wants to obtain sublinear communication. Nevertheless, it is straightforward that players need only reveal an *upper bound* on the number of elements of their dataset (this can be done by adjoining to the universe a sufficient amount of "dummy" elements different for each player that can be used for dataset padding). We defer further details for the full version.

We first describe our protocols in the so called semi-honest setting (parties are assumed to follow the protocol specifications) and then we consider their extension to the malicious setting (players may deviate arbitrarily from protocol specifications). We provide a concrete formulation of the malicious adversarial setting and efficient transformations of our PIPE protocols (of asymptotically the same complexity) to this setting, taking advantage of Universally-Composable commitments based on a recent verifiable encryption scheme. From a technical viewpoint we remark that, in our attempt to provide a communication efficient protocol for the setting where each party encodes his dataset as a list of values, we came up with a notion of public-key encryption (which we formalize and realize as part of the description of PIPE protocol #2) that is called **superposed encryption** and may have further applications in concrete secure function evaluation protocols.

Related Work. A private disjointness test falls into a general category of secure protocols that allow two parties to compare information they possess without leaking it. In turn, such protocols can be formulated and solved in terms of two-party Secure Function Evaluation [25] (see also [18]).

The special case of a private disjointness test where each party has a single element in the dataset has been studied extensively and is called a "private equality test." Protocols for private equality tests were considered in [13, 22, 20]. The problem of securely computing the *intersection* of two private datasets was considered in [22] and more recently in [15]. A related problem to a private disjointness test is scalar multiplication. A concurrent to the present work investigation of the notion of scalar products in the context of private data mining procedures appeared in [17].

The disjointness test problem itself was considered from the communication complexity perspective (without taking into account privacy) and a linear lower bound was shown [19, 23] even in the probabilistic setting (that allows an amount of error in the test). This suggests that there is no chance for a sublinear communication solution in the size of the universe in the worst case.

PIPE Protocols Based on Existing Techniques. As mentioned above, a private disjointness test can be formulated in the setting of secure function evaluation over a circuit. We note that protocol constructions based on secure circuit evaluation in the sense of [25] are not particularly efficient; nevertheless, they may be well within reach of current computational capabilities depending on the circuit as some recent results suggest [21]. Regarding building a private disjointness test over a circuit, we consider the two scenarios of input encoding: (i) input to each player is by the characteristic bitstring of the subset, (ii) input is by the subset as a list of elements. Using the formulation of secure two-party evaluation of Goldreich [18] and a security parameter ℓ, we have the following: regarding (i), we can design a circuit for testing disjointness that uses $2N$ AND gates and $2N-1$ XOR gates something that will result in a protocol with communication complexity $16N\ell + 2N$ bits (using $2N\ell$ oblivious transfers). Regarding (ii), (assuming subset sizes of N_A and N_B for the two players respectively) we can design a circuit that contains $N_A N_B(3\log_2(N) - 1))$ XOR-gates and $N_A N_B(\log_2(N) + 1)$ AND-gates that will produce a protocol of total communication $8N_A N_B \ell(\log_2(N) + 1) + N_A \log_2(N) + N_B \log_2(N)$ bits (using $N_A N_B(\log_2(N)+1)$ oblivious transfers). The number of rounds required by the two protocols is $O(N)$ in the first case and $O(N_A N_B + \log N)$ in the second case. Evidently our PIPE constructions compare very favorably to generic secure function evaluation techniques – e.g. for case (i) our PIPE protocol #1 has total communication of $2(N+1)\ell$ bits and a single round of interaction, where for the case (ii) our PIPE protocol #2 has total communication of $3(N_B(N_A + 2)\ell$ bits with N_B rounds of interaction; finally, our PIPE protocol #3 (applying to case (ii) as well) has a single round of interaction (at the cost of substantially larger communication though).

Beyond secure function evaluation techniques, the most related to the present work previous protocol constructions are those of [15]. This latter paper deals with the problem of computation of the actual intersection set of two private datasets; moreover the protocol construction of [15] (as noted in that paper) can also be easily modified to a protocol that reveals the size of the intersection only (not its elements); a private disjointness test nevertheless has a much more stringent security requirement (only one bit of information must be revealed — and perhaps an upper bound on the size of *both* datasets); for this reason it appears to be much more challenging to achieve. Another related problem to the intersection computation that is mentioned by [15] is "private threshold matching" that requires the computation of whether the intersection is larger than a specified threshold. Naturally a private disjointness test is a special case of this problem; nevertheless, no efficient protocol construction of a private disjointness test that is entirely independent from generic secure function evaluation techniques is known for this problem (cf. [15]).

Regarding our notion of superposed encryption we remark that it can be paralleled w.r.t. functionality to a $(2, 2)$ threshold homomorphic encryption with the main difference being in that in a superposed scheme key-generation is executed locally without communication and independently assuming fixed public-parameters. Concurrently and independently to the present work applications of $(2, 2)$-threshold-homomorphic encryption in two-party secure computations were considered in [24].

Organization. In section 2 we present the cryptographic primitives that are used in our constructions. In section 3 we present our three private intersection evaluation protocols as well as the notion of superposed encryption. Finally, in section 4 we consider the malicious adversary setting.

Due to lack of space we have omitted the proofs of our theorems from this extended abstract. They will appear in the full version of this work.

2 Preliminaries

Homomorphic Encryption. An encryption scheme is a triple $\langle K, E, D\rangle$ of algorithms defined as follows: the key generation algorithm K on input 1^ℓ (where ℓ is the key length) outputs a public key pk and a secret key sk. The encryption function E_{pk} uses the public key pk for its operation $E_{pk} : R \times P \to C$. In this case, P is the plaintext space, C is the ciphertext space and R is the randomness space (all parameterized by ℓ). At the same time, the decryption function $D_{sk} : C \to P$ uses the secret key sk so that for any plaintext $m \in P$, if $E_{pk}(r, p) = c$, then $D_{sk}(c) = p$ for any $r \in R$. Homomorphic encryption adds to the above the following requirements: there exist binary operations $+$, $\oplus$, $\odot$ defined over the spaces P, R, C so that $\langle P, +\rangle$, $\langle R, \oplus\rangle$ are the groups written additively and $\langle C, \odot\rangle$ – multiplicatively. We say that an encryption scheme is Homomorphic if for all $r_1, r_2 \in R$ and all $x_1, x_2 \in P$ it holds that

$$E_{pk}(r_1, x_1) \odot E_{pk}(r_2, x_2) = E_{pk}(r_1 \oplus r_2, x_1 + x_2)$$

Informally, this means that if we want to "add" plaintexts that are encrypted, we may "multiply" their corresponding ciphertexts. As a result, we can "add" any two plaintexts (by multiplying corresponding ciphertexts); also we can multiply an encrypted plaintext by an integer constant, by raising its corresponding ciphertext to the power that is equal to the integer constant — which is essentially multiplying a ciphertext by itself a number of times; note that this can be done efficiently by using standard repeated squaring.

ElGamal Homomorphic Encryption. We will employ a standard variant of ElGamal encryption [12]. This variant of ElGamal has been employed numerous times in the past (e.g., in the context of e-voting [8]). This public-key encryption scheme is a triple $\langle K, E, D\rangle$ defined as follows:

- Key-generation K. Given a security parameter ℓ, the probabilistic algorithm $K(1^\ell)$ outputs a public-key $pk := \langle p, g, h, f\rangle$ and the corresponding secret-key x so that the following are satisfied: (i) p is a ℓ-bit prime number so that $(p-1)/2 = q$ is also a prime number. (ii) g is an element of order q in $\mathbf{Z}_p^*$. (iii) $h, f \in \langle g\rangle$ are randomly selected. (iv) $x = \log_g h$.
- Encryption E. Given public-key $pk = \langle p, g, h, f\rangle$ and a plaintext $m \in \mathbf{Z}_q$, E samples $r \leftarrow_R \mathbf{Z}_q$ and returns $\langle g^r, h^r f^m\rangle$.
- Decryption D. Given secret-key x and a ciphertext $\langle G, H\rangle$ the decryption algorithm returns $G^{-x}H(\bmod p)$. Note that this will only return f^m, never-

theless this would be sufficient for our setting as, given a ciphertext $\langle g^r, h^r f^m \rangle$ we will only be interested in testing whether $m \stackrel{?}{=} 0$ (something that can easily be done by testing $G^{-x}H \stackrel{?}{\equiv}_p 1$).

Observe that the above encryption scheme is homomorphic: indeed, the randomness space R, the plaintext space P and the ciphertext space C satisfy the following: (i) $R = P = \mathbf{Z}_q$ and $(R, \oplus)$, $(P, +)$ are additive groups by setting the operations $\oplus, +$ to be addition modulo q. (ii) $C \subseteq \mathbf{Z}_p^* \times \mathbf{Z}_p^*$ and it holds that $(C, \odot)$ is a multiplicative group when $\odot$ is defined as pointwise multiplication modulo p. (iii) it holds that for any $r_1, r_2 \in R$, x_1, x_2, and $pk = \langle p, g, h, f \rangle$,

$$E_{pk}(r_1, x_1) \odot E_{pk}(r_2, x_2) = \langle g^{r_1}, h^{r_1} f^{x_1} \rangle \odot \langle g^{r_2}, h^{r_2} f^{x_2} \rangle = \langle g^{r_1+r_2}, h^{r_1+r_2} f^{x_1+x_2} \rangle$$

Interactive Protocols. A two-party interactive protocol $\mathcal{P}$ is specified by a pair of probabilistic Interactive Turing machines $\langle \mathcal{A}, \mathcal{B} \rangle$. Each TM has input tape, private work tapes, output tape, as well as both have access to a communication tape; one of them is designated to make the first move. An execution of a protocol $\mathcal{P}$ is denoted by $\mathsf{exec}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$ where a is the private input for $\mathcal{A}$, b is the private input for $\mathcal{B}$. We will denote as $\mathsf{out}_{\mathcal{A}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$ and $\mathsf{out}_{\mathcal{B}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$ the private outputs of the two ITM's. We write $\mathsf{out}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$ for the concatenation of the outputs of the two parties.

Here we will consider protocols where only player $\mathcal{A}$ has output. We say that a protocol $\mathcal{P}$ computes a certain functionality f if it holds that for all a, b $\mathsf{out}_{\mathcal{A}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle = f(a, b)$ (note that $\mathsf{out}_{\mathcal{B}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$ is not relevant in this case and it may be designated to just $\top$, a dummy "accept" symbol).

For a given protocol $\mathcal{P} = \langle \mathcal{A}, \mathcal{B} \rangle$ we define as $\mathsf{view}_{\mathcal{A}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$ the random variable tuple $\langle a, \rho, m_1, \ldots, m_k \rangle$ where ρ is the internal coin tosses of $\mathcal{A}$ and $m_1, \ldots, m_k$ are the messages received from $\mathcal{B}$. In a similar fashion we define $\mathsf{view}_{\mathsf{B}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$.

Definition 1. *A protocol $\mathcal{P}$ computing a functionality f is said to be* private w.r.t. semi-honest behavior *if the following hold true for all a, b: there exists a simulator $\mathcal{S}$ (respectively $\mathcal{S}'$) so that $\mathcal{S}(a, f(a, b))$ (respectively $\mathcal{S}'(b)$) is computationally indistinguishable from* $\mathsf{view}_{\mathsf{A}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$. *(respectively* $\mathsf{view}_{\mathsf{B}}\langle \mathcal{A}(a), \mathcal{B}(b) \rangle$*).*

Privacy w.r.t. malicious behavior is a bit more complicated as in this case we cannot assume that either party follows the protocol $\mathcal{P}$ as it is specified (i.e., either party may abort or transmit elements that do not follow the specifications).

The way that security is dealt in this case is by comparing the player's views with respect to an "ideal" protocol implementation. In particular, an ideal two-party protocol for the functionality f operates with the help of a trusted-third party $\mathcal{T}$ as follows: player $\mathcal{A}$ transmits to $\mathcal{T}$ the value a; player $\mathcal{B}$ transmits to $\mathcal{T}$ the value b. $\mathcal{T}$ computes the value $f(a, b)$ and transmits it to player $\mathcal{A}$ while it sends the value $\top$ to player $\mathcal{B}$. If either player fails to transmit its input to $\mathcal{T}$ then $\mathcal{T}$ returns $\perp$ to both parties. Normally $\mathcal{A}, \mathcal{B}$ output whatever output is given by $\mathcal{T}$ in their private output tape.

Privacy w.r.t. malicious behavior then is defined (informally) in the following fashion: for any implementation of player $\mathcal{B}$ denoted by $\mathcal{B}^*$ in the real world there exists a $\mathcal{B}^*_{ideal}$ that operates in the ideal world so that the random variables $\mathsf{out}\langle\mathcal{A}(a), \mathcal{B}^*(b)\rangle$ and $\mathsf{out}\langle\mathcal{A}_{ideal}(a), \mathcal{B}^*_{ideal}(b)\rangle$, are computationally indistinguishable. Likewise, for any implementation of player $\mathcal{A}$ denoted by $\mathcal{A}^*$ in the real world there exists a $\mathcal{A}^*_{ideal}$ that operates in the ideal world so that the random variables $\mathsf{out}\langle\mathcal{A}^*(a), \mathcal{B}(b)\rangle$ and $\mathsf{out}\langle\mathcal{A}^*_{ideal}(a), \mathcal{B}_{ideal}(b)\rangle$, are computationally indistinguishable. A formal definition of the above will be given in the full version — it is omitted here due to lack of space; we refer to [18] for more details w.r.t. secure two-party function evaluation.

Universally Composable Commitments. A commitment is a scheme that allows to a party called the committer holding a value x to submit a value $C(x)$ to a party called the receiver so that the two following properties are satisfied (i) hiding: the value $C(x)$ does not reveal any information about x, (ii) binding: the committer can "open" $C(x)$ to reveal that it is a commitment to x in a unique way (this is called the decommitment phase). Universally-Composable (UC) commitments, introduced by Canetti and Fischlin [3] is a very useful tool for proving security in the malicious setting for secure function evaluation protocols. Informally, a UC-commitment simulates an ideal commitment scheme where the committer submits x to a trusted third party $\mathcal{T}$ in the commitment phase, and in the decommitment phase, $\mathcal{T}$ simply transfers x to the receiver. More efficient UC-commitments were presented by Damgard and Nielsen [10].

Verifiable encryption of discrete-logarithms was suggested by Camenisch and Shoup in [2] and is a useful tool for constructing UC-commitments. In the construction suggested in [2] (using the common reference string model, cf. [9]) a UC-commitment scheme can be constructed as a pair $\langle\psi, C\rangle$ where $C = \gamma_1^x \gamma_2^r$ and ψ is a verifiable encryption of the pair x, r. In a real execution the relative discrete-logarithms of γ_1, γ_2 and the secret-key of the verifiable encryption are unknown; this allows one to prove that the commitment scheme is computationally binding and hiding. In the simulation, on the other hand, the simulator controls the common reference string so that the secret-key of the verifiable encryption as well as the relative discrete-logarithm of γ_1 base γ_2 are known; this allows the simulator to extract the committed value as well as equivocate the committed value (open the commitment in an arbitrary way).

Zero-knowledge Proofs of Language Membership. Proofs of language membership were introduced in [16]. A proof of knowledge of language membership is a protocol between parties, the prover and the verifier, that allows a prover to show knowledge of a witness w so that $(w, x) \in R$ holds, where R is a polynomial-time relation and x is a publicly known value; the existence of such witness suggests that x belongs to the NP-language L defined as $x \in L \leftrightarrow \exists w : (w, x) \in R$. Such a protocol is called a proof of knowledge if it satisfies the property that the verifier cannot extract any knowledge about w except for the fact that such w exists and it is known to the prover (this requires the existence of a knowledge extractor for the protocol, see [1]).

In our context, we will consider *efficient* and *non-interactive* zero-knowledge proofs of language membership that deal with languages of committed and encrypted values. Regarding non-interactiveness, we will resort to the Fiat-Shamir heuristics [14] that employ a random oracle for providing the challenge for the prover. Security will be argued in the random oracle model. Regarding efficiency, we stress that we will not resort to generic zero-knowledge arguments for NP-languages. Instead, we will rely on three-move zero-knowledge proofs of knowledge, cf. [7], that are specifically designed for the encryption and commitment schemes at hand. Below we describe the proofs that we will employ:

Proofs of knowledge of a discrete-log representation. $\mathsf{PK}(x_1, x_2 : y = g_1^{x_1} g_2^{x_2})$. This is a standard protocol used extensively in various constructions; it originates from [4].

Proof of knowledge of Equality of Discrete-logarithms. $\mathsf{PK}(x_1, x_2 : y_1 = g_1^{x_1} \wedge y_2 = g_2^{x_2})$, also a standard protocol used extensively; it originates from [5].
Proof of knowledge of a Product. $\mathsf{PK}(x_1, x_2 : y = g^{x_1 \cdot x_2})$; this is slightly more tricky than the above, as it requires at least one separate discrete-log based commitment to one of the two values; see e.g., [6].

We will also consider OR/AND compositions of the above protocols [11, 7].

3 Private Intersection Predicate Evaluation

In this section we give a formal definition of Private Intersection Predicate Evaluation (PIPE) protocols and we present three protocol constructions for different settings and objectives that allow the computation of the intersection predicate of the two datasets possessed by the two players.

Definition 2. *A two-party Private Intersection Predicate Evaluation (*PIPE*) protocol is a pair $\langle \mathcal{A}, \mathcal{B} \rangle$ of ITM's that have the following properties:*

- *(Correctness) For any $S_A, S_B \subseteq \{1, \ldots, N\}$, let $f(S_A, S_B) \in \{0, 1\}$ so that $f(S_A, S_B) = 1$ if and only if $S_A \cap S_B \neq \emptyset$. $\mathcal{A}, \mathcal{B}$ is a correct* PIPE *protocol if it is a two-party protocol that computes the functionality f.*
- *(Security) It will be argued in the semi-honest and malicious behavior setting according to the definitions of section 2.*

The time and communication complexity of PIPE *protocols will be measured over the parameters $N, N_A = \#S_A, N_B = \#S_B$ (note that $S_A = a_1, .., b_{N_A}$ and $S_B = b_1, .., b_{N_B}$) as well as a security parameter ℓ; we will also measure the number of rounds that a protocol requires (a round = a full interaction between A and B).*

Note that we will somewhat relax the definition of security above for protocols PIPE #2 and PIPE #3 in order to achieve sublinear communication in the size of the universe. In particular we will allow to the ideal functionality to release upper bounds on the list sizes in addition to the single bit output (in fact , without loss of generality, we will assume that the ideal functionality releases the sizes of the datasets).

3.1 Two-Party PIPE Protocol #1

Consider two players A and B that have sets of values $S_A = \{a_1, \ldots, a_{N_A}\}$ and $S_B = \{b_1, \ldots, b_{N_B}\}$ where $S_A, S_B \subseteq [N]$ and $[N] = \{1, \ldots, N\}$. The two datasets are stored by the players in the form of their characteristic bitstring of length N: S_X is represented by a bitstring $Bit_X = \langle bit_1^X, \ldots, bit_N^X \rangle$ so that $bit_j^X = 1$ iff $j \in S_X$, where $X \in \{A, B\}$.

Step 1. Player A executes the key generation algorithm for a public-key encryption $\langle K, E, D \rangle$ to obtain pk, sk. After this, player A sends to the player B pk and the encryption of Bit_A, as follows: $\langle c_1, c_2, \ldots, c_N \rangle$, where $c_j = E_{pk}(bit_j^A)$.

Step 2. Upon receiving $\langle c_1, \ldots, c_N \rangle$ and pk from player A, player B computes a ciphertext c as follows: Let $random \neq 0$ be some random number drawn uniformly from R (the randomness space of the encryption), then:

$$c = (c_1^{bit_1^B} \odot \cdots \odot c_N^{bit_N^B})^{random} \odot E_{pk}(0)$$

Note that $E_{pk}(0)$ is used to refresh the randomness of the ciphertext c. The above expression is equivalent to :

$$c = E_{pk}((bit_1^A \cdot bit_1^B + \cdots + bit_N^A \cdot bit_N^B) \times random + 0)$$

Observe that if $bit_j^A = 1$ and $bit_j^B = 1$, then $bit_j^A \cdot bit_j^B$ will produce 1. In all other cases, the result of multiplication will be 0. It follows that we will obtain $1's$ in all cases of $j \in S_A \bigcap S_B$. The sum of all multiplications together can result in 0 only if $S_A \bigcap S_B = \emptyset$. The sum is multiplied by some random number so that it becomes impossible to determine how many elements belong to the intersection. Player B sends back the value of c to the player A .

Step 3. Player A, using his secret-key, tests whether the decryption of c is equal to 0 or not; if the decryption is 0, A concludes that $S_A \bigcap S_B = \emptyset$; otherwise, if the decryption is non-zero, player A concludes that there must be at least one element in the joint intersection.

Theorem 1. *The above protocol is a correct PIPE protocol that is secure in the semi-honest model under the* CPA *security of* $\langle K, E, D \rangle$.

Regarding efficiency, if ℓ is the security parameter, and the key-generation, encryption and decryption require $k(\ell), e(\ell), d(\ell)$ time respectively, it holds that (i) the total communication complexity is $2(N+1)\ell$ bits (ii) the time-complexity of player A is $k(\ell) + Ne(\ell) + d(\ell)$, and (iii) the time-complexity of player B is $N\ell + N_B\ell^2 + \ell^3 + e(\ell)$.

3.2 Two-Party PIPE Protocol # 2

In the protocol of this section, we will employ a type of a public-key encryption scheme that we call (two-player) superposed encryption. In this kind of encryption scheme, there are two parties each with a pair of public and secret-keys defined over the same public-parameters (e.g., the same prime modulus).

The functionality of the encryption scheme extends regular public-key encryption in the following way: given a plaintext m' and a ciphertext c that encrypts a plaintext m under one player's public-key, one can superpose c with m' to obtain a "superposed ciphertext" c' that can be decrypted by either player to a random ciphertext that hides the plaintext $m \cdot m'$ and can be decrypted by the other player. Formally, superposed encryption is a sequence of procedures $\langle K, K', E, E^{\mathsf{ext}}, D, D^{\mathsf{sup}} \rangle$ defined as follows:

– The key generation algorithm K is comprised by an initial key generation step that produces the public parameter $param$, as well as K' that produces the public-key and secret-key for each user (given the parameter $param$).

  Now given $param \leftarrow K(\ell)$ and $(pk_A, sk_A), (pk_B, sk_B) \leftarrow K'(param)$:
– The two encryption functions are specified as follows: $E_{pk_X} : P \to C$ and $E^{\mathsf{sup},X}_{pk_A,pk_B} : P \times C \to C^{\mathsf{sup}}$ for each player $X \in \{A, B\}$.
– The encryption function E_{pk_X} is homomorphic for the plaintext $(P, +)$, randomness $(R, \oplus)$ and ciphertext group $(C, \odot)$. Moreover, $(P, +, \cdot)$ is a ring.
– The superposed encryptions $E^{\mathsf{sup},X}_{pk_A,pk_B}(m, E_{pk_{\overline{X}}}(m'))$ and $E^{\mathsf{sup},X}_{pk_A,pk_B}(m', E_{pk_{\overline{X}}}(m))$ are indistinguishable for any fixed m, m', where X is a player, $X \in \{A, B\}$, and $\overline{X}$ is the other player, $\overline{X} \in \{A, B\} - \{X\}$.
– The decryption functions satisfy the following conditions:
  • $D_{sk_X}(E_{pk_X}(m)) = m$ if $X \in \{A, B\}$, for all $m \in P$.
  • For any fixed $c \in E_{pk_X}(m')$, it holds that if c' is distributed according to $D^{\mathsf{sup}}_{sk_X}(E^{\mathsf{sup},\overline{X}}_{pk_A,pk_B}(m, c))$, then c' is uniformly distributed over $E_{pk_{\overline{X}}}(m \cdot m')$.
  where $X \in \{A, B\}$ and $\overline{X}$ is the single element of $\{A, B\} - \{X\}$.

Next we define the appropriate notion of security for superposed encryption. Observe the differences from regular semantic security of public-key encryption: the adversary is allowed to *select a ciphertext and a public-key* over which the challenge plaintext will be superposed.

The superposed encryption CPA Game $G^{\mathcal{B}}_{\mathsf{cpa}}$ (denoted by $G^{\mathcal{B}}_{\mathsf{cpa}}(1^\ell)$):

```
1. $param \leftarrow K(1^\ell)$;
2. $(pk_A, sk_A) \leftarrow K'(param)$;
3. $\langle aux, pk_B, c, m_0, m_1 \rangle \leftarrow \mathcal{B}(\mathsf{choose}, 1^\ell, param, pk_A)$
4. Choose $b \leftarrow_R \{0, 1\}$;
5. Set $c^* \leftarrow E^{\mathsf{sup},A}_{pk_A,pk_B}(m_b, c)$;
6. $b^* \leftarrow \mathcal{B}(\mathsf{guess}, aux, c^*)$;
7. if $b = b^*$ return $\top$ else return $\bot$;
```

Note that the above game assumes that player B is the "attacker." The identical game where player A is the attacker, will be denoted by $G^{\mathcal{A}}_{\mathsf{cpa}}(1^\ell)$.

Definition 3. *A superposed encryption scheme satisfies* CPA *security provided that for any* PPT *attackers* $\mathcal{A}, \mathcal{B}$ *it holds that* $2\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{cpa}}(1^\ell) = \top] - 1$ *and* $2\mathbf{Prob}[G^{\mathcal{B}}_{\mathsf{cpa}}(1^\ell) = \top] - 1$ *are negligible functions in the security parameter* ℓ.

Below we show that superposed encryption CPA security implies regular CPA security of the underlying public-key encryption scheme.

Theorem 2. *Let $\langle K, K', E, E^{\mathsf{sup}}, D, D^{\mathsf{sup}} \rangle$ be a superposed encryption scheme that satisfies CPA security. Then, the underlying public-key encryption $\langle K'', E, D \rangle$ (where K'' is the composition of K and K') is a CPA pk encryption scheme.*

As a side note, the opposite does not hold necessarily and CPA security of the underlying public-key encryption does not appear to imply CPA security for the superposed scheme.

Construction. The scheme that we will use is based on ElGamal encryption and it is as follows:

– K, given 1^ℓ, samples a prime number p such that $p = 2q+1$ with q also prime. Then it selects a $g' \in \mathbf{Z}_p^*$ and sets $g \leftarrow (g')^2 (\mathrm{mod} p)$ and selects $h \leftarrow_R \langle g \rangle$. The public parameter is $param := \langle p, q, g, h \rangle$. The user key generation operates as follows: given $\langle p, q, g \rangle$ it samples $x \leftarrow_R \mathbf{Z}_q$ and sets $y_X := g^x (\bmod p)$, with $pk_X = y_X$ and $sk_X = x$.
– The encryption function E_{pk_X} selects $r \leftarrow_R \mathbf{Z}_q$ and returns $\langle g^r, y_X^r h^m \rangle$ (for $X \in \{A, B\}$). The ciphertext encryption function $E^{\mathsf{sup},A}_{pk_A,pk_B}$ takes a ciphertext pair $\langle G, H \rangle$ and the plaintext m, it samples $r, r' \leftarrow_R \mathbf{Z}_q$ and returns the triple $\langle g^r, G^m g^{r'}, y_X^r y_{\overline{X}}^{r'} H^m \rangle$. Likewise the ciphertext encryption $E^{\mathsf{sup},B}_{pk_A,pk_B}$ takes a ciphertext pair $\langle G, H \rangle$ and the plaintext m, it samples $r, r' \leftarrow_R \mathbf{Z}_q$ and returns the triple $\langle G^m g^r, g^{r'}, y_X^r y_{\overline{X}}^{r'} H^m \rangle$.
The decryption function $D^{\mathsf{sup}}_{\mathsf{sk_A}}(c)$ (respectively $D^{\mathsf{sup}}_{\mathsf{sk_B}}(c)$) for $c = \langle G_A, G_B, Y \rangle$ it returns the ciphertext $\langle G_B, Y G_A^{-sk_A} \rangle$ (respectively $\langle G_A, Y G_B^{-sk_B} \rangle$).

To see that the above scheme is a superposed encryption observe:

$$D^{\mathsf{sup}}_{sk_A}(E^{\mathsf{sup},B}_{pk_A,pk_B}(m, E_{\mathsf{pk_A}}(m'))) = D^{\mathsf{sup}}_{sk_A}(\langle g^r, g^{r'}, y_A^r y_B^{r'} h^{m \cdot m'} \rangle) = \langle g^{r'}, y_B^{r'} h^{m \cdot m'} \rangle$$

Theorem 3. *The superposed encryption scheme presented above satisfies CPA security under the DDH assumption.*

The PIPE protocol. Suppose that $\langle K, E, E^{\mathsf{sup}}, D, D^{\mathsf{sup}} \rangle$ is a superposed encryption, and the two players possess the lists S_A, S_B respectively that are subsets of $[N]$ with $N_A = \#S_A$ and $N_B = \#S_B$.

Step 0. The two players A, B receive as public joint input $param \leftarrow K(1^\ell)$ and each one executes separately the key-generation algorithm to obtain $(pk_A, sk_A) \leftarrow K_A(param)$, $(pk_B, sk_B) \leftarrow K_B(param)$.

Player A selects $\alpha_0, \ldots, \alpha_{N_A} \in \mathbf{Z}_q$ such that the polynomial $f(x) := \alpha_0 + \alpha_1 x + \ldots \alpha_{N_A} x^{N_A}$ has the property that $f(a) = 0$ if and only if $a \in S_A$.

Also player A computes $c^* = E^{\mathsf{sup},B}_{pk_A,pk_B}(1, E_{pk_A}(1))$.

The following steps are repeated for $j = 1, \ldots, N_B$:

Step $\boldsymbol{j}$.1. Player A computes $c = D^{\mathsf{sup}}_{sk_A}(c^*)$. Player A transmits to player B the sequence of superposed ciphertexts,

$$\langle c^*_0, \ldots, c^*_{N_A} \rangle = \langle E^{\mathsf{sup},A}_{pk_A,pk_B}(\alpha_0, c), \ldots, E^{\mathsf{sup},A}_{pk_A,pk_B}(\alpha_{N_A}, c) \rangle$$

Step $\boldsymbol{j}$.2. Player B decrypts the superposed ciphertexts as follows:

$$\langle c_0, \ldots, c_{N_A} \rangle = \langle D^{\mathsf{sup}}_{sk_B}(c^*_0), \ldots, D^{\mathsf{sup}}_{sk_B}(c^*_{N_A}) \rangle$$

Observe that, if $j = 1$,

$$\langle c_0, \ldots, c_{N_A} \rangle \in \langle E_{pk_A}(\alpha_0), \ldots, E_{pk_A}(\alpha_{N_A}) \rangle$$

or for $j > 1$,

$$\langle c_0, \ldots, c_{N_A} \rangle \in \langle E_{pk_A}(f(b_1) \ldots f(b_{j-1}) \cdot \alpha_0), \ldots, E_{pk_A}(f(b_1) \ldots f(b_{j-1}) \cdot \alpha_{N_A}) \rangle$$

Following this, player B computes

$$c' = E_{pk_A}(0) \cdot c_0 \cdot (c_1)^{b_j} \ldots (c_{N_A})^{b_j^{N_A}} \in E_{pk_A}(f(b_1) \ldots f(b_j))$$

Observe that, c' is uniformly distributed over $E_{pk_A}(f(b_1) \ldots f(b_j))$. Then player B computes an encryption of the ciphertext c' using the superposed encryption function

$$c^* = E^{\mathsf{sup},B}_{pk_A,pk_B}(1, c')$$

and transmits c^* to player A; the step $j + 1$ is executed now.

In the final round when $j = N_B$ the following modifications are made:

Final Round. Player B in step $\mathbf{N_B.2}$ does not compute c^* using the superposed encryption $E^{\mathsf{sup},B}_{pk_A,pk_B}$; instead, he computes a regular ciphertext c^{Fin} as follows:

$$c^{\mathsf{Fin}} = (c')^{random} \cdot E_{pk_A}(0)$$

where $random \leftarrow_R \mathbf{Z}_q - \{0\}$. Observe that $c^{\mathsf{Fin}} \in E_{pk_A}(0)$ if $f(b_1) \ldots f(b_{N_B}) = 0$; otherwise, $c^{\mathsf{Fin}} \in E_{pk_A}(s)$ where s is a random non-zero element of $\mathbf{Z}_q$.

When player A receives c^{Fin}, he computes $s = D_{sk_A}(c^{\mathsf{Fin}})$ and concludes that the intersection is empty if $s \neq 0$, or that there exists an intersection in case of $s = 0$.

Theorem 4. *The protocol described above is a correct* PIPE *protocol that is secure in the semi-honest model under the* CPA *security of the superposed pk-encryption.*

Regarding efficiency, we need N_B-rounds and for security parameter ℓ and if the key-generation, encryption and decryption require $k(\ell), e(\ell), d(\ell)$ time respectively, it holds that the total communication is $3N_B(N_A+2)\ell$ bits; the time-complexity of player A is $\Theta(k(\ell) + N_B N_A e(\ell) + N_B d(\ell))$; the time-complexity of player B is $\Theta(k(\ell) + N_B e(\ell) + N_B N_A d(\ell) + N_A N_B \ell^2 + N_A N_B \ell^2 \log N + \ell^3)$.

3.3 Two-Party PIPE Protocol # 3

Checking whether intersection exists for players that possess relatively small lists of essentially constant size (compared to the universe $[N]$) can be phrased in the context of multivariate polynomial evaluation to allow a protocol with optimal round complexity. Suppose now that players A and B have small lists of values $S_A = \{a_1, \ldots, a_{N_A}\}$ and $S_B = \{b_1, \ldots, b_{N_B}\}$ where $N_A \ll N$ and $N_B \ll N$. First, player A selects $\langle \alpha_0, \alpha_1, \ldots, \alpha_{N_A} \rangle$ for the polynomial

$$Pol(z) = \sum_{u=0}^{N_A} \alpha_u z^u = \alpha_0 + \alpha_1 z + \alpha_2 z^2 + \cdots + \alpha_{N_A} z^{N_A}$$

so that $Pol(a) = 0$ iff $a \in S_A$.

As in the case of PIPE protocol #2, player A wants to evaluate $F(b_1, b_2, \ldots, b_{N_B}) = \prod_{j=1}^{N_B} Pol(b_j)$ in order to find if $S_A \bigcap S_B$ is non-empty. If at least one $Pol(b_y)$ is equal to 0 (i.e., b_y is in the $S_A \bigcap S_B$), then obviously $F(b_1, b_2, \ldots, b_t)$ will evaluate in 0.

A direct application of the techniques of [15] in this setting will result in a protocol of communication complexity $\Theta((N_A+1)^{N_B})$, as the total number of coefficients of $F(b_1, b_2, \ldots, b_{N_B})$ is $(N_A+1)^{N_B}$. In the remaining of the section, using the observation that many of coefficients are repeated, we will reduce the communication complexity to $\Theta(\binom{N_A+N_B}{N_B})$. Observe that the number of distinct coefficients in the computation of F equals the number of possible multisets of size N_B from the alphabet of size N_A+1. Therefore, the number of distinct coefficients is $\binom{N_A+1+N_B-1}{N_B} = \binom{N_A+N_B}{N_B}$.

Let us denote the array of such coefficients as $Cf = \langle cf[1], \ldots, cf[\binom{N_A+N_B}{N_B}] \rangle$.

Let $\mathcal{I}$ contain all monotonically decreasing tuples $\langle i_1, \ldots, i_{N_B} \rangle$ of $\{0, \ldots, N_A\}^{N_B}$ so that $i_\ell \geq i_{\ell'}$ for $\ell > \ell'$. For $j = 1, \ldots, \binom{N_A+N_B}{N_B}$ we define $cf[j]$ to be $cf[j] = \alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_{N_B}}$ where $\langle i_1, \ldots, i_{N_B} \rangle$ is the j-th tuple of $\mathcal{I}$.

To describe the protocol, we need to specify an order in which the tuples $\langle i_1, \ldots, i_{N_B} \rangle$ are generated: we will use an inverse lexicographic order in $\mathcal{I}$, i.e., $\langle N_A, \ldots, N_A \rangle$ is the first element and the subsequent elements are in the order are defined by the function $next[\langle i_1, ..., i_{N_B} \rangle] = \langle i_1, ..., i_{t-1}, i_t - 1, i_t - 1, \ldots, i_t - 1 \rangle$ where $t = min\{1, ..., N_B\}$ with the property $i_{t+1} = \cdots = i_{N_B} = 0$. Note, that if $i_{N_B} \neq 0$ then $t = N_B$ (the last element).

The protocol description is as follows:

Step 1. Player A executes the key generation algorithm for a public-key encryption $\langle K, E, D \rangle$ to obtain pk, sk.

Then, player A sends to the player B the encryption of Cf, as follows: $\langle \xi[1], \xi[2], \ldots, \xi[\binom{N_A+N_B}{N_B}] \rangle$, where $\xi[j] = E_{pk}(cf[j])$:

$$\boldsymbol{c} = \langle E_{pk}(cf[1]), E_{pk}(cf[2]), \ldots, E_{pk}(cf[\binom{N_A+N_B}{N_B}]) \rangle$$

Step 2. Let $oc_{\boldsymbol{i}}[j]$ equal the number of times the element j occurs in the monotonically decreasing tuple $\boldsymbol{i} = \langle i_1, \ldots, i_{N_B} \rangle$. Observe that for such a tuple, the value $T_{\boldsymbol{i}} = \frac{N_B!}{oc_{\boldsymbol{i}}[0]!oc_{\boldsymbol{i}}[1]!\ldots oc_{\boldsymbol{i}}[N_A]!}$ corresponds to the number of times the coefficient $cf[\boldsymbol{i}]$ is repeated in F. Note, $oc_{\boldsymbol{i}}[0] + \cdots + oc_{\boldsymbol{i}}[N_A] = N_B$.

Note that a tuple $\boldsymbol{i} = \langle i_1, i_2, \ldots, i_{N_B}\rangle$ can be permuted $T_{\boldsymbol{i}}$ times in total; let $\langle i_1^{(j)}, i_2^{(j)}, \ldots, i_{N_B}^{(j)}\rangle$ denote the j-th permutation of this tuple. If $v = 1, \ldots, \binom{N_A+N_B}{N_B}$, let $\boldsymbol{i}[v]$ denote the v-th monotonically decreasing tuple of $\{0, \ldots, N_A\}^{N_B}$ that will be also denoted by $\langle i_1[v], \ldots i_{N_B}[v]\rangle$. Player B upon receiving $\boldsymbol{c} = \langle \xi[1], \ldots, \xi[\binom{N_A+N_B}{N_B}]\rangle$, will perform the following: let *random* $\neq 0$ be some random number drawn uniformly from P (the plaintext space); then, player B computes:

$$F^{en} = (\bigodot_{v=1}^{\binom{N_A+N_B}{N_B}} \xi[v]^{\sum_{j=1}^{T_{\boldsymbol{i}[v]}} \prod_{\ell=1}^{N_B} b_\ell^{i_\ell^{(j)}[v]}})^{random} \odot E_{pk}(0)$$

The above expression is equivalent to :

$$F^{en} = E_{pk}((\sum_{v=1}^{\binom{N_A+N_B}{N_B}} cf[v] \sum_{j=1}^{T_{\boldsymbol{i}[v]}} \prod_{\ell=1}^{N_B} b_\ell^{i_\ell^{(j)}[v]})random) = E_{pk}(F(b_1, \ldots, b_{N_B})random)$$

The evaluation of F^{en} will result in 0 only if $S_A \bigcap S_B \neq \emptyset$. Player B sends back the value of F^{en} to the player A .

Theorem 5. *The protocol for players with small sets is a correct PIPE protocol that is secure in the semi-honest model under the* CPA *security of* $\langle K, E, D\rangle$.

This protocol's total communication complexity is $\binom{N_A+N_B}{N_B} + 1$ ciphertexts. The time-complexity on the side of player A is $\binom{N_A+N_B}{N_B}$ encryptions and one decryption; the complexity for player B is $\binom{N_A+N_B}{N_B}$ ciphertext multiplications, one encryption, and $\binom{N_A+N_B}{N_B}$ exponentiations; note that the time of computing the exponents $\sum_{j=1}^{T_{\boldsymbol{i}[v]}} \prod_{\ell=1}^{N_B} b_\ell^{i_\ell^{(j)}[v]}$ for $v = 1, \ldots, \binom{N_A+N_B}{N_B}$ is proportional to $(N_A + 1)^{N_B}$ (equal to the total number of terms that should be computed for the multivariate polynomial); nevertheless, the values of these terms can be precomputed by player B since they only involve the variables from the player's B list. Thus, the online complexity will be proportional to $\binom{N_A+N_B}{N_B}$ steps.

4 Dealing with Malicious Parties

In this section we outline how the PIPE protocols presented in the previous sections can be modified so that they can be proven secure in the setting where either party is allowed to operate in a malicious way (as opposed to semi-honest). The basic tools that will be necessary for the transformation are universally composable commitments and zero-knowledge proofs of language membership (see section 2). Our general approach for all three protocols will be as follows:

players will provide UC-commitments for their private inputs and will use non-interactive zero-knowledge proofs of language membership to ensure that their computations are consistent with the UC-commitments. Security will be subsequently argued in the random oracle model.

PIPE Protocol #1 for Malicious Parties. Let $\mathsf{Com}_{\mathsf{crs}}(r, x)$ be of the form $\langle \gamma_1^r \gamma_2^x (\bmod p), \psi \rangle$ where the first component is a Pedersen-like discrete-log based commitment scheme that is made universable composable using verifiable encryption of [2], i.e., ψ is a verifiable encryption of r, x; note that γ_1, γ_2 belong to the common reference string crs. We will write $\mathsf{Com}_{\mathsf{crs}}(x)$ to denote the random variable defined by $\mathsf{Com}_{\mathsf{crs}}(r, x)$ over all possible random coin tosses r.

Step 1. Player A executes the key generation algorithm for a public-key encryption $\langle K, E, D \rangle$ to obtain pk, sk. Player A sends to player B the ciphertexts $\langle c_1, c_2, \ldots, c_N \rangle$, where $c_j := E_{pk}(bit_j^A)$ as well as the commitments $\delta_j := \mathsf{Com}_{\mathsf{crs}}(bit_j^A)$ for $j = 1, \ldots, N$. Player A accompanies each pair (c_j, δ_j) with a non-interactive proof of language membership $\mathsf{PK}(r, r', x : c_j = E_{pk}(r, x) \wedge \delta_j = \mathsf{Com}_{\mathsf{crs}}(r', x) \wedge x \in \{0, 1\})$.

Step 2. Player B computes the ciphertext c as in the semi-honest case, $c = (c_1^{bit_1^B} \odot \cdots \odot c_N^{bit_N^B})^{random} \odot E_{pk}(r, 0)$. where $r \leftarrow_R R$. Player B also computes the commitments $\delta'_j = \mathsf{Com}_{\mathsf{crs}}(bit_j^B)$ for $j = 1, \ldots, N$ and $\delta'_{N+1} = \mathsf{Com}_{\mathsf{crs}}(random)$ as well as $\delta'_{N+2} = \mathsf{Com}_{\mathsf{crs}}(r)$; player B transmits to player A the values $c, \delta'_1, \ldots, \delta'_N$ as well as a non-interactive proof of knowledge $\mathsf{PK}(r'_1, \ldots, r'_{N+2}, x_1, \ldots, x_N, r', random : c = (c_1^{x_1} \odot \cdots \odot c_N^{x_N})^{random} \odot E_{pk}(r, 0) \wedge_{j=1}^{N} [\delta_j = \mathsf{Com}_{\mathsf{crs}}(r'_j, x_j) \wedge x_j \in \{0, 1\}] \wedge \delta'_{N+1} = \mathsf{Com}_{\mathsf{crs}}(r'_{N+1}, random) \wedge \delta'_{N+2} = \mathsf{Com}_{\mathsf{crs}}(r'_{N+2}, r))$. Note that such proof of knowledge can be constructed efficiently, cf. section 2.

Step 3. This is the same as in the semi-honest case: player A tests whether the decryption of c is equal to 0 or not.

We note that either player aborts the protocol in case some of the non-interactive proofs of knowledge do not verify.

Theorem 6. *The above protocol is a correct PIPE protocol that is secure in the malicious setting in the random oracle model.*

PIPE protocol #2 and #3 for malicious parties. Due to lack of space we omit from this extended abstract the transformation of PIPE protocols # 2 and # 3 in the malicious adversary setting. We remark that the transformation is based on the same principles as above, i.e., the employment of UC-commitments and the appropriate non-interactive proofs of language membership that show that players' moves are consistent with their commitments.

References

1. Mihir Bellare and Oded Goldreich, *On Defining Proofs of Knowledge*, CRYPTO 1992: 390-420.
2. Jan Camenisch, Victor Shoup, *Practical Verifiable Encryption and Decryption of Discrete Logarithms*, CRYPTO 2003: 126-144

3. Ran Canetti and Marc Fischlin, *Universally Composable Commitments*, CRYPTO 2001: 19-40
4. David Chaum, Jan-Hendrik Evertse and Jeroen van de Graaf, *An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations*, EUROCRYPT 1987: 127-141
5. D. Chaum and T. Pedersen *Wallet databases with observers*, In Advances in Cryptology – Crypto '92, pages 89-105, 1992.
6. Ronald Cramer and Ivan Damgard *Zero-Knowledge Proofs for Finite Field Arithmetic; or: Can Zero-Knowledge be for Free?*, CRYPTO 1998: 424-441
7. Ronald Cramer, Ivan Damgard, Berry Schoenmakers, *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, CRYPTO 1994: 174-187
8. Ronald Cramer, Rosario Gennaro and Berry Schoenmakers, *A Secure and Optimally Efficient Multi-Authority Election Scheme*, EUROCRYPT 1997, pp. 103-118.
9. Ivan Damgard, *Efficient Concurrent Zero-Knowledge in the Auxiliary String Model* EUROCRYPT 2000: 418-430
10. Ivan Damgard, Jesper Buus Nielsen, *Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor*, CRYPTO 2002. pp. 581-596.
11. Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano and Moti Yung, *On Monotone Formula Closure of SZK*, FOCS 1994: 454-465.
12. T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, IT-31(4):469–472, July 1985.
13. Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. Communications of the ACM, 39(5):77–85, 1996.
14. Amos Fiat and Adi Shamir *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, CRYPTO 1986: 186-194.
15. Michael Freedman, Kobbi Nissim and Benny Pinkas, *Efficient private matching and set intersection*, EUROCRYPT 2004.
16. Shafi Goldwasser, Silvio Micali and Charles Rackoff, *The Knowledge Complexity of Interactive Proof Systems* SIAM J. Comput. 18(1): 186-208 (1989)
17. Bart Goethals, Sven Laur, Helger Lipmaa and Taneli Mielikäinen, *On Secure Scalar Product Computation for Privacy-Preserving Data Mining.* , The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004), December 2–3, 2004.
18. Oded Goldreich, *Secure Multi-Party Computation*, unpublished manuscript, 2002. `http://www.wisdom.weizmann.ac.il/ oded/pp.html`.
19. B. Kalyanasundaram and G. Schnitger. *The probabilistic communication complexity of set intersection*, SIAM Journal on Discrete Math, 5(5):545–557, 1992.
20. Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In Advances in Cryptology, ASIACRYPT 2003, pp. 416–433.
21. D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. *The Fairplay project*, http://www.cs.huji.ac.il/labs/danss/FairPlay.
22. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In Proc. 31st Annual ACM Symposium on Theory of Computing, pages 245–254, Atlanta, Georgia, May 1999.
23. Alexander A. Razborov, *On the Distributional Complexity of Disjointness*, Theor. Comput. Sci. 106(2): 385-390 (1992)
24. Berry Schoenmakers and Pim Tuyls, *Practical Two-Party Computation Based on the Conditional Gate*, ASIACRYPT 2004, pp. 119-136.
25. A. C. Yao, *How to generate and exchange secrets*, In Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pages 162-167, 1986.