Efficient Set Operations in the Presence of Malicious Adversaries

Carmit Hazay[1,⋆] and Kobbi Nissim[2,⋆⋆]

[1] Dept. of Computer Science and Applied Mathematics,
Weizmann Institute and Interdisciplinary Center (IDC) Herzliya
carmit.hazay@weizmann.ac.il
[2] Dept. of Computer Science, Ben-Gurion University and Microsoft AI, Israel
kobbi@cs.bgu.ac.il

Abstract. We revisit the problem of constructing efficient secure two-party protocols for set-intersection and set-union, focusing on the model of malicious parties. Our main results are constant-round protocols that exhibit linear communication and a linear number of exponentiations with simulation based security. In the heart of these constructions is a technique based on a combination of a perfectly hiding commitment and an oblivious pseudorandom function evaluation protocol. Our protocols readily transform into protocols that are UC-secure.

Keywords: Secure two-party computation, Simulation based security, Set intersection, Set union, Oblivious pseudorandom function evaluation.

1 Introduction

Secure function evaluation (SFE) allows two distrusting parties to jointly compute a function of their respective inputs as if the computation is executed in an ideal setting where the parties send inputs to a trusted party that performs the computation and returns its result. Starting with the work of [37,20,6,4], it is by now well known that (in various settings, and considering semi-honest and malicious adversaries) any polynomial-time computation can be generically compiled into a secure function evaluation protocol with polynomial complexity. However, more often than not, the resulting protocols are inefficient for practical uses and hence attention was given to constructing efficient protocols for specific functions. This approach that proved quite successful for the semi-honest setting (see, e.g., [26,13,28,1,17,25,5,30,24,27]), while the malicious setting remained, at large, elusive (a notable exception is [1]).

We focus on the secure computation of basic set operations (intersection and union) where the parties P_1, P_2, holding input sets X, Y, respectively, wish to compute $X \cap Y$ or $X \cup Y$. These problems have been widely looked at by researchers in the last few years, and our main goal is to come up with protocols

⋆ Research supported by an Eshkol scholarship.
⋆⋆ Research partly supported by the Israel Science Foundation (grant No. 860/06).

P.Q. Nguyen and D. Pointcheval (Eds.): PKC 2010, LNCS 6056, pp. 312–331, 2010.

for set-intersection and set-union that are secure in the malicious setting and are of better complexity to those known.

We begin by briefly surveying the current constructions of two-party secure computation for set intersection and union that are most relevant to our work:

- Freedman, Nissim and Pinkas studied set intersection in [17]. They represent a set by a polynomial that zeros exactly on the set elements. Their construction for the semi-honest setting utilizes oblivious polynomial evaluation and a balanced allocation scheme and exhibits linear communication (counting field elements) and (almost) linear computation (counting modular exponentiations). See Section 3.

  They also present variants of the above protocol, for the cases where one of the parties is malicious and the other is semi-honest. For efficiency, generic zero-knowledge proofs of adherence to the protocol are avoided. The protocol for malicious P_1 (denoted *client* in [17]) and semi-honest P_2 (*server*) utilizes a cut-and-choose strategy and hence communication is inflated by a statistical security parameter. The protocol for malicious P_2 and semi-honest P_1 is in the random oracle model. A protocol that is secure in the fully malicious setup, that combines both techniques, is sketched in Section 3.1.
- Kissner and Song [25] used polynomials to represent multi-sets. Letting the roots of $Q_X(\cdot)$ and $Q_Y(\cdot)$ coincide with elements of the multi-sets X and Y, they observed that if $r(\cdot), s(\cdot)$ are polynomials chosen at random then the roots of $r(\cdot) \cdot Q_X(\cdot) + s(\cdot) \cdot Q_Y(\cdot)$ coincide with high probability with the multi-set $X \cap Y$. This beautiful observation yields a set-intersection protocol for the semi-honest case, where the parties use an additively homomorphic encryption scheme (the Paillier scheme is suggested in [25]) to perform the polynomial multiplication, introducing quadratic computation costs in the set sizes. For the security of the protocol, it is crucial that no party should be able to decrypt on her own. Hence, the secret key should be shared and joint decryption should be deployed. Assuming a trusted setup for the encryption scheme, the communication costs for the two-party case are as in the protocol for semi-honest parties of [17].

  For malicious parties [25] introduced generic zero-knowledge proofs for proving adherence to the prescribed protocol (e.g., zero-knowledge proofs of knowledge for the multiplication of the encrypted $Q_x(\cdot)$ with a randomly selected $r(\cdot)$). While this change seems to be of dire consequences to the protocol efficiency, the analysis in [25] ignores its effects. Furthermore, the costs of setting the shared key for the Paillier scheme are ignored in the analysis. To the best of our knowledge, there are currently no efficient techniques for generating the shared Paillier keys, which do not incorporate an external trusted dealer (the latter schemes include [14,15] referenced in [25]).

  In addition to that, Kissner and Song presented a protocol for the threshold set-union problem, where only the elements that appear in the combined inputs more than t times are learnt by the parties. Their protocol employs the same technique of polynomial multiplication and thus introduces quadratic computation costs as above.

– Hazay and Lindell [21] revisited secure set intersection, with the aim of achieving efficient protocols in presence of a more realistic adversarial behavior than in the benign semi-honest model, and under standard cryptographic assumptions. Two protocols were presented, one achieves security in the presence of malicious adversaries with one-sided simulatability, the other is secure in the presence of covert adversaries [2]. The main tool used in these protocols is a secure implementation of oblivious pseudorandom function evaluation.

  Having P_1, P_2 hold sets of sizes m_X, m_Y respectively, both protocols in [21] are constant round, and incur the communication of $O(m_X \cdot p(n) + m_Y)$ group elements and the computation of $O(m_X \cdot p(n) + m_Y)$ modular exponentiations, where set elements are taken from $\{0,1\}^{p(n)}$.

  We note that the protocols in [21] can be made secure in the malicious setup, e.g., by introducing a secure key selection step for the oblivious PRF and by adding zero-knowledge proofs of knowledge to show correctness at each step. Namely, for proving that the *same* PRF key is indeed used by party P_1 in each iteration and to enable extraction of its input (as a pseudorandom function is not necessarily invertible). While this would preserve the complexity of these protocols asymptotically (in m_X, m_Y), the introduction of such proofs would probably make them inefficient for practical use since there is no effieicnt known way to construct them.
– Recently, Jarecki, and Liu [23] presented a very efficient protocol for computing a pseudorandom function with a committed key (informally, this means that the same key is used in all invocations), and showed that it yields an efficient set-intersection protocol. The main restriction of this construction is that the input domain size of the pseudorandom function should be polynomial in the security parameter (curiously, the proof of security for the set-intersection protocol makes use of the ability to exhaustively search over the input domain, so removing the restriction on the input domain of the pseudorandom function does not immediately yield a set-intersection protocol for a super-polynomial domain).
– Finally, Dachman-Soled et al. [11] present a protocol for set-intersection in the presence of malicious adversaries *without* restricting the domain. Their construction uses polynomial evaluation but takes a different approach than ours by incorporating a secret sharing of the inputs to the polynomials. They avoid generic zero-knowledge by utilizing the fact that Shamir's secret sharing implies Reed Solomon code. Their protocol incurs communication of $O(mk^2 \log^2 n + kn)$ group elements and computation of $O(mnk \log n + mk^2 \log^2 n)$.

1.1 Our Contributions

Our main contributions are efficient set-intersection and set-union protocols that are secure in the setup of malicious parties. Our constructions are in the standard model, and are based on standard cryptographic assumptions (in particular, no

random oracle or a trusted setup). We begin by briefly describing the construction of Freedman et al. [17] for semi-honest parties that serves as our starting point.

Secure Set Intersection with Semi-Honest Parties. The main tool used in the construction of [17] is oblivious polynomial evaluation. The basic protocol works as follows:

1. Party P_1 chooses encryption/decryption keys $(pk, sk) \leftarrow G(1^n)$ for a homomorphic encryption scheme (G, E, D) and sends pk to P_2.
2. P_1 computes the coefficients of a polynomial $Q(\cdot)$ of degree m_X, with roots set to the m_X elements of X, and sends the encrypted coefficients to P_2.
3. For each element $y \in Y$ (in random order), party P_2 chooses a random value r (taken from an appropriate set depending on the encryption scheme), and uses the homomorphic properties of the encryption scheme to compute an encryption of $r \cdot Q(y) + y$. P_2 sends the encrypted values to P_1.
4. Upon receiving these encrypted values, P_1 extracts $X \cap Y$ by decrypting each value and then checking if the result is in X. Note that if $y \in X \cap Y$ then by the construction of the polynomial $Q(\cdot)$ we get that $r \cdot Q(y) + y = r \cdot 0 + y = y$. Otherwise, $r \cdot Q(y) + y$ is a random value that reveals no information about y and (with high probability) is not in X.

Note that the communication complexity of this simple scheme is linear in $m_X + m_Y$, as $m_X + 1$ encrypted values are sent from P_1 to P_2 (these are the encrypted coefficients of $Q(\cdot)$), and m_Y encrypted values are sent from P_2 to P_1 (i.e., $Q(y)$ for every $y \in Y$). However, the work performed by P_2 is high, as each of the m_Y oblivious polynomial evaluations includes performing $O(m_X)$ exponentiations, totaling in $O(m_X \cdot m_Y)$ exponentiations.

To save on computational work, Freedman et al. introduced a balanced allocation scheme into the protocol. Loosely speaking, they used the balanced allocation scheme of [3] with $B = \frac{m_X}{\log \log m_X}$ bins, each of size $M = O(m_X/B + \log \log B) = O(\log \log m_X)$. Party P_1 now uses the balanced allocation scheme to hash every $x \in X$ into one of the B bins resulting (with high probability) with each bin's load being at most M. Instead of a single polynomial of degree m_X party P_1 now constructs a degree-M polynomial for each of the B bins, i.e., polynomials $Q_1(\cdot), \ldots, Q_B(\cdot)$ such that the roots of $Q_i(\cdot)$ are the elements put in bin i. As some of the bins contain less than M elements, P_1 pads each polynomial with zero coefficients up to degree M. Upon receiving the encrypted polynomials, party P_2 obliviously evaluates the encryption of $r_0 \cdot Q_{h_0(y)}(y) + y$ and $r_1 \cdot Q_{h_1(y)}(y) + y$ for each of the two bins $h_0(y), h_1(y)$ in which y can be allocated, enabling P_1 to extract $X \cap Y$ as above.

Neglecting constant factors, the communication complexity is not affected as P_1 now sends $BM = O(m_X)$ encrypted values and P_2 replies with $2m_Y$ encrypted values. There is, however, a dramatic reduction in the work performed by P_2 as each of the oblivious polynomial evaluations amounts now to performing just $O(M)$ exponentiations, and P_2 performs $O(m_Y \cdot M) = O(m_Y \cdot \log \log m_X)$ exponentiations overall.

Our main goal is to come up with protocols that exhibit low asymptotic communication and computation costs in the presence of malicious behavior. Noting that asymptotic complexity does not reveal everything about a protocol's efficiency or practicality, we avoid using generic zero-knowledge proofs of adherence to the prescribed protocols, even when they involve relatively short statements, and costly set up commutations that make the efficient only for very large inputs. Our contributions are realized as follows,

Preventing the Players from Deviating from the Protocol: We inherit the oblivious polynomial evaluation and balanced allocation techniques used in [17]. On top of these we introduce an efficient zero-knowledge proof that P_1 uses to show that her encrypted polynomials were correctly produced (unlike in [17], our proof does not use a cut-and-choose strategy), and a technique preventing player P_2 from deviating meaningfully from the protocol. This technique combines a perfectly hiding commitment scheme with an oblivious pseudorandom function evaluation protocol.

Eliminating the Random Oracle: In some sense, our construction replaces the random oracle used in [17] in the case of a malicious sender with a PRF, but this 'replacement' is only in a very weak sense: In our construction P_2 holds the key for the pseudorandom function, and hence the function does not look random to P_2, nor does P_2 does not need to invoke the oblivious pseudorandom evaluation protocol to compute it. The consequence is that, unlike with the simulator for the protocol in the random oracle model that can easily monitor all invocations of the oracle, our simulator cannot extract P_2's input to the pseudorandom function.

We note that the protocols of [21] also use an oblivious pseudorandom function evaluation primitive, where the player analogous to P_2 knows the key for the function. Their usage of this primitive is, however, very unlike in our protocols. In the protocols of [21] the pseudorandom function is evaluated on the set of elements that P_2 holds, using the *same* PRF key for all evaluations. Whereas in our protocols it is evaluated on a random payload using (possibly) different keys. A payload s_y is a random element that is chosen independently for each element $y \in Y$ with the aim to fix the randomness used by P_2. Meaning that the randomness for the oblivious polynomial evaluation of y is determined by the PRF evaluation of s_y. Furthermore, the protocols in [21] are designed for the covert adversary model and for the one-sided simulatability model, and hence a technique enabling full simulation of P_2 is not needed, whereas our constructions allow simulation of both parties.

Choosing the Underlying Encryption Scheme: Our protocols make extensive use of a homomorphic encryption scheme, and would remain secure (with only small modifications) under a variety of choices. We chose to work with the El Gamal scheme (that is *multiplicatively* homomorphic) although it may seem that the more natural choice is the Paillier scheme [32], that is additively homomorphic (indeed, our initial constructions were based on the Paillier scheme).

Using the Paillier scheme, a subtle problem emerges (this was overlooked, e.g., in [17]). Recall that for the Paillier scheme $pk = N, sk = \phi(N)$. Now, if P_1 knows sk when she constructs her polynomials, then she may construct a polynomial $Q(\cdot)$ such that $Q(y) \notin \mathbb{Z}_N^*$ for some specific 'target' value y. This would allow her to learn about P_2's input beyond the intended protocol output. A possible solution is that P_1, P_2 would first engage in a protocol to jointly generate pk and shares of sk, whereas P_1 would learn sk only *after* committing to her polynomials. This, however, introduces high key setup costs, and the result is a protocol that exhibits low *asymptotic* costs, but, because of its high setup costs, its efficiency is gained only for very large inputs.

Efficiency: Our protocols for set intersection and set union $\pi_\cap, \pi_\cup$ are constant round, work in the standard model and do not require a trusted setup. The underlying encryption scheme is El Gamal where the keys are selected by party P_1. Both protocols do not employ any generic zero-knowledge proof.

Assuming the protocols of [16,21] (that require $p(n)$ oblivious transfers for realizing the oblivious pseudorandom function evaluation), we get that for sets $X, Y \subset \{0,1\}^{p(n)}$ of m_X, m_Y elements respectively, the costs of $\pi_\cap, \pi_\cup$ are of sending $O(m_X + m_Y \cdot p(n))$ group elements, and the computation of $O(m_X + m_Y \cdot (\log\log m_X + p(n)))$ modular exponentiations. Note that this is significantly better than $O(m_X \cdot m_Y)$.

A significant improvement can be achieved by using a more efficient pseudorandom function evaluation instead of using the function of [29] which requires a single oblivious transfer for every input bit. This is due to the fact that our protocol uses oblivious pseudorandom function evaluation as a black box. Furthermore, for set intersection, another significant improvement can be achieved if the size of the intersection $m_{X\cap Y}$ is allowed to be leaked (to P_2). The resulting protocol is of sending $O(m_X + m_{X\cap Y} \cdot p(n))$ and computing $O(m_X + m_Y \cdot \log\log m_X + m_{X\cap Y} \cdot p(n))$. When $m_{X\cap Y} \ll m_Y$ we get a protocol that is more efficient than that of [21]. This type of improvement does not apply for [21] as well since the parties apply the PRF directly on the input set Y and thus cannot deduce $m_{X\cap Y}$ before that.

UC Security: Our protocols readily transform into the UC framework as all our simulators are straight-line in an hybrid model with access to some specific zero-knowledge proofs. We show how to modify our set intersection protocol to one that is secure in the UC framework (in the common reference string model).

For lack of space, we focus only on the protocol for secure set intersection in the malicious model, and omit the more standard details of the construction, and its proof of security. The missing details and proofs can be found in the full version of this paper [22].

2 Preliminaries

Throughout the paper, we denote the security parameter by n, and, although not explicitly specified, input lengths are always assumed to be bounded by some

polynomial in n. A probabilistic machine is said to run in *polynomial-time* (PPT) if it runs in time that is polynomial in the security parameter n alone. A function $\mu(n)$ is *negligible* (in n) if for every polynomial $p(\cdot)$ there exists a value N such that $\mu(n) < \frac{1}{p(n)}$ for all $n > N$; i.e., $\mu(n) = n^{-\omega(1)}$.

2.1 Secure Two-Party Computation – Definitions

We use standard definitions of security for two party commputation in the malicious model, which we now briefly review. The reader is referred to [18, Chapter 7] for more details and motivating discussion.

We prove the security of our protocols in the setting of *malicious adversaries*, that may arbitrarily deviate from the specified protocol. Security is analyzed by comparing what an adversary can do in a *real* protocol execution to what it can do in an *ideal* scenario. In the ideal scenario, the computation involves an incorruptible *trusted third party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Informally, the protocol is secure if any adversary interacting in the real protocol (i.e., where no trusted third party exists) can do no more harm than what it could do in the ideal scenario. We consider the *static* setting where the adversary is only able to corrupt a party at the outset of the protocol. There are technical issues that arise, such as that it may be impossible to achieve fairness or guaranteed output delivery. E.g., it is possible for the an adversarial party to prevent an honest party from receiving outputs.

2.2 The El Gamal Encryption Scheme

The El Gamal encryption scheme operates on a cyclic group $\mathbb{G}$ of prime order q. We will work in the group $\mathbb{Z}_{q'}^*$ where $q' = 2q + 1$ is prime, and set $\mathbb{G}$ to be the subgroup of $\mathbb{Z}_{q'}$ of quadratic residues modulo q' (note that membership in $\mathbb{G}$ can be easily checked). Let g denote a random generator in $\mathbb{G}$, then the public and secret keys are $\langle \mathbb{G}, q, g, h \rangle$ and $\langle \mathbb{G}, q, g, x \rangle$ where $x \leftarrow_R \mathbb{Z}_q$ and $h = g^x$. A message $m \in \mathbb{G}$ is encrypted by choosing $y \leftarrow_R \mathbb{Z}_q$ and the ciphertext is $\langle g^y, h^y \cdot m \rangle$. A ciphertext $c = \langle \alpha, \beta \rangle$ is decrypted as $m = \beta / \alpha^x$. We use the property that given $y = \log_g \alpha$ one can reconstruct $m = \beta / h^y$ and hence a party encrypting m can prove knowledge of m by proving knowledge of y.

The semantic security of the El Gamal scheme follows from the hardness of decisional Diffie-Hellman (DDH) in $\mathbb{G}$. The El Gamal scheme is homomorphic relative to multiplication. I.e., if $\langle \alpha_1, \beta_1 \rangle$ encrypts m_1 and $\langle \alpha_2, \beta_2 \rangle$ encrypts m_2 then $\langle \alpha_1 \cdot \alpha_2, \beta_1 \cdot \beta_2 \rangle$ encrypts $m_1 m_2$. We additionally consider a modified version of El Gamal where the encryption is performed by choosing $y \leftarrow_R \mathbb{Z}_q$ and computing $\langle g^y, h^y \cdot g^m \rangle$. Decryption of a ciphertext $c = \langle a, b \rangle$ is performed by computing $g^m = b \cdot a^{-x}$. The fact that m cannot be efficiently recovered is not problematic for the way El Gamal is incorporated in our protocols. Moreover, this variant of El Gamal is additively homomorphic and can be used to perform oblivious linear computations (e.g., polynomial evaluation) in the exponent.

2.3 Perfectly Hiding Commitment

We use a perfectly-hiding commitment scheme $(\mathsf{com}, \mathsf{dec})$ with a zero-knowledge proof of knowledge π_{COM} for the relation

$$\mathcal{R}_{\text{COM}} = \left\{ \Big(c, (r, m) \Big) \mid c = \mathsf{com}(m; r) \right\},$$

where $\mathsf{com}(m; r)$ denotes the commitment to a message m using random coins r. We instantiate $\mathsf{com}(\cdot;\cdot)$ with Pedersen's commitment scheme [33], using the same underlying group $\mathbb{G}$ used for the El Gamal scheme. I.e., let $q' = 2q + 1$ where q', q are primes and let g, h be generators of the subgroup $\mathbb{G}$ of quadratic residues modulo q'. A commitment to m is then defined as $\mathsf{com}(m; r) = g^m h^r$ where $r \leftarrow_R \mathbb{Z}_{q-1}$. The scheme is perfectly hiding as for every m, r, m' there exists a single r' such that $g^m h^r = g^{m'} h^{r'}$. The scheme is binding assuming hardness of computing $\log_g h$. However, given $\log_g h$, it is possible to decommit any commitment c into any message $m \in \mathbb{Z}_q$. We instantiate π_{COM} with the proof of knowledge from [31] (this proof is not a zero-knowledge proof, yet can be modified using standard techniques [19]).

2.4 Zero-Knowledge Proofs

Our protocols employ zero-knowledge proofs of knowledge for the following relations (in the following, $\mathbb{G}$ is a group of prime order):

Type	Protocol	Relation/Language	Reference
ZKPK	π_{DL}	$\mathcal{R}_{\text{DL}} = \{((\mathbb{G}, g, h), x) \mid h = g^x\}$	[35]
ZKPK	π_{DDH}	$\mathcal{R}_{\text{DDH}} = \{((\mathbb{G}, g, g_1, g_2, g_3), x) \mid g_1 = g^x \wedge g_3 = g_2^x\}$	[7]
ZK	π_{NZ}	$\mathrm{L}_{\text{NZ}} = \left\{ (\mathbb{G}, g, h, \langle \alpha, \beta \rangle) \mid \begin{array}{l} \exists\ (m \neq 0, r) \text{ s.t.} \\ \alpha = g^r \wedge \beta = h^r g^m \end{array} \right\}$	Sec. 2.4

Zero-Knowledge Proof for $\mathbf{L_{NZ}}$. We use standard techniques for constructing a zero-knowledge proof for the language of encryptions $\langle \alpha, \beta \rangle$ of non-zero exponents of g:

$$\mathrm{L}_{\text{NZ}} = \{(\mathbb{G}, g, h, \langle \alpha, \beta \rangle) \mid \exists\ (m \neq 0, r) \text{ s.t. } \alpha = g^r \wedge\ \beta = h^r g^m\}.$$

The construction is based on a zero-knowledge protocol π_{MULT} for the language

$$\mathrm{L}_{\text{MULT}} = \left\{ (\mathbb{G}, g, h, c_1, c_2, c_3) \mid \begin{array}{l} \exists\ m, m' \in \mathbb{Z}_q \text{ s.t. } c_1, c_2, c_3 \text{ are} \\ \text{encryptions of } g^m, g^{m'}, g^{mm'} \text{ resp.} \end{array} \right\}.$$

π_{MULT} is a modification of a protocol by Damgård and M. Jurik [9] designed for the Paillier encryption scheme.

2.5 Balanced Allocation

We employ a scheme for randomly mapping elements into bins, as suggested in [17]. We use the balanced allocation scheme of [3] where elements are inserted

into B bins as follows. Let $h_0, h_1 : \{0,1\}^{p(n)} \to [B]$ be two randomly chosen hash functions mapping elements from $\{0,1\}^{p(n)}$ into bins $1, \ldots, B$. An element $x \in \{0,1\}^{p(n)}$ is inserted into the less occupied bin from $\{h_0(x), h_1(x)\}$, where ties are broken arbitrarily. If m elements are inserted, then except with negligible probability over the choice of the hash functions h_0, h_1, the maximum number of elements allocated to any single bin is at most $M = O(m/B + \log\log B)$. Setting $B = \frac{m}{\log\log m}$ we get that $M = O(\log\log m)$.[1] In the protocol we deviate insignificantly from the description above, and let P_1 choose seeds for two pseudorandom functions, that are used as the hash functions h_0, h_1.

2.6 Oblivious PRF Evaluation

We use a protocol π_{PRF} that obliviously evaluates a pseudorandom function in the presence of a malicious adversary. Let I_{PRF} be the indexing algorithm for a pseudorandom function ensemble, and let $k \leftarrow_R I_{\text{PRF}}(1^n)$ be a sampled key. The functionality F_{PRF} is defined as

$$(k, x) \mapsto (\lambda, F_{\text{PRF}}(k, x)). \tag{1}$$

The PRF may be instantiated with the Naor-Reingold pseudorandom function [29] with the protocol presented in [16] (and proven in [21]). The function is defined as

$$F_{\text{PRF}}((a_0, \ldots, a_n), x) = g^{a_0 \prod_{i=1}^{n} a_i^{x[i]}},$$

where $\mathbb{G}$ is a group of prime order q, g is a generator of $\mathbb{G}$, $a_i \in \mathbb{Z}_q$ and $x = (x[1], \ldots, x[n]) \in \{0,1\}^n$. The protocol involves executing an oblivious transfer for every bit of the input x. Combining this with the fact that n oblivious transfers runs require $11n + 29$ exponentiations using the protocol in [34] (the analysis in [34] includes the cost for generating a common reference string), one gets a constant-round protocol that securely computes $\mathcal{F}_{\text{PRF}}$ in the presence of malicious players using a constant number of exponentiations for every bit of the input x.

3 Secure Set Intersection

We now consider the functionality of *set intersection*, where each party's input consists of a *set*, and the size of the other party's input set. If the set sizes match, then the functionality outputs the intersection of these input sets to P_1. Otherwise P_1 is given $\perp$. More formally:

Definition 1. *Let X and Y be subsets of a predetermined domain (w.l.o.g., we assume $X, Y \subset \{0,1\}^{p(n)}$ for some polynomial $p()$ such that $2^{p(n)}$ is*

[1] A constant factor improvement is achieved using the *Always Go Left* scheme in [36] where $h_0 : \{0,1\}^{p(n)} \to [1, \ldots, \frac{B}{2}], h_1 : \{0,1\}^{p(n)} \to [\frac{b}{2} + 1, \ldots, B]$. An element x is inserted into the less occupied bin from $\{h_0(x), h_1(x)\}$; in case of a tie x is inserted into $h_0(x)$.

super-polynomial in n, and that the set elements can be represented as elements of some finite group), the functionality $\mathcal{F}_\cap$ is:

$$((X, m_Y), (Y, m_X)) \mapsto \begin{cases} (X \cap Y, \lambda) & \text{if } |X| = m_X,\ |Y| = m_Y \text{ and } X, Y \subseteq \{0,1\}^{p(n)} \\ (\perp, \lambda) & \text{otherwise} \end{cases}$$

In the rest of this section we present in detail our construction for a protocol realizing $\mathcal{F}_\cap$ in the presence of malicious adversaries. For completeness we include a description of the protocol by Freedman et al. for semi-honest parties:

Protocol 1. (set-intersection protocol secure in the presence of semi-honest parties):

- **Inputs:** *The input of P_1 is m_Y and a set $X \subseteq \{0,1\}^{p(n)}$ containing m_X items; the input of P_2 is m_X and a set $Y \subseteq \{0,1\}^{p(n)}$ containing m_Y items.*
- **Auxiliary inputs:** *A security parameter 1^n.*
- **The protocol:**
  1. **Key setup:** *P_1 chooses the secret and public keys (sk, pk) for the underlying homomorphic encryption scheme (e.g., Paillier or El Gamal). She sends pk to P_2.*
  2. **Setting the balanced allocation scheme:** *P_1 computes the parameters B, M for the scheme and chooses the seeds for two (pseudo-)random hash functions $h_0, h_1 : \{0,1\}^{p(n)} \to [B]$. She sends B, M, h_0, h_1 to P_2.*
  3. **Creating polynomials for the set X:** *For every $x \in X$, P_1 maps x into the less occupied bin from $\{h_0(x), h_1(x)\}$ (ties broken arbitrarily). Let $\mathcal{B}_i$ denote the set of elements mapped into bin i and let $Q_i(x) \stackrel{\text{def}}{=} \sum_{j=0}^{M} Q_{i,j} \cdot x^j$ denote a polynomial with the set of roots $\mathcal{B}_i$. P_1 encrypts the coefficients of the polynomials and sends the encrypted coefficients to P_2.*
  4. **Substituting in the polynomials:** *Let $y_1, \ldots, y_{m_Y}$ be a random ordering of the elements of set Y. P_2 does the following for all $\alpha \in \{1, \ldots, m_Y\}$:*
     (a) *He sets $\hat{h}_0 = h_0(y_\alpha)$, $\hat{h}_1 = h_1(y_\alpha)$.*
     (b) *He chooses two random elements in the underlying group of the homomorphic encryption scheme r_0, r_1. He then uses the homomorphic properties of the encryption scheme to compute an encryption of $r_0 \cdot Q_{\hat{h}_0}(y_\alpha) + y_\alpha$ and $r_1 \cdot Q_{\hat{h}_1}(y_\alpha) + y_\alpha$. Both encrypted values are sent to P_1.*
  5. **Computing the intersection:** *P_1 decrypts each received value. If the decrypted value is in X then P_1 records as part of her local output.*

Note that, since the parties are semi-honest, P_1 outputs $X \cap Y$ with probability negligibly close to 1: (i) For elements $y_\alpha \in X \cap Y$ we get that $Q_{h_0(y_\alpha)}(y_\alpha) = 0$ or $Q_{h_1(y_\alpha)}(y_\alpha) = 0$, hence one of the corresponding encrypted values is y_α itself, and P_1 would record it in its local output. (ii) For $y_\alpha \notin X \cap Y$ we get that $Q_{h_0(y_\alpha)}(y_\alpha) \neq 0$ and $Q_{h_1(y_\alpha)}(y_\alpha) \neq 0$ and hence corresponding encrypted values are two random values $r_0 + y$ and $r_1 + y$ that fall within X with only a negligible probability.

Efficiency. The protocol runs in a constant number of rounds. The communication costs are of sending the encrypted polynomials (BM values) and the encrypted $r_0 \cdot Q_{\hat{h}_0}(y_\alpha) + y_\alpha$ and $r_1 \cdot Q_{\hat{h}_1}(y_\alpha) + y_\alpha$ ($2m_Y$ values). Using the El Gamal or Paillier encryption schemes, the computation costs are of encrypting the polynomials ($O(BM)$ exponentiations) and of obliviously computing the encryptions of $r_0 \cdot Q_{\hat{h}_0}(y_\alpha) + y_\alpha$ and $r_1 \cdot Q_{\hat{h}_1}(y_\alpha) + y_\alpha$ ($O(Mm_Y)$ exponentiations). Overall, we get that the overall communication costs are of sending $O(m_X + m_Y)$ encryptions, and the computation costs are of performing $O(m_X + m_Y \log\log n)$ modular exponentiations.

3.1 Constructing a Protocol for Malicious Parties

We note a couple of issues that need to be addressed in transforming the above protocol for semi-honest parties to a protocol for malicious parties:

1. It is easy for P_1 to construct the B polynomials such that it would learn about elements that are not in the intersection $X \cap Y$. For instance, if $Q_i(\cdot)$ is identically zero then P_1 learns all elements $\{y \in Y : h_0(y) = i \text{ or } h_1(y) = i\}$. Similarly, if the sum of degrees of $Q_1, \ldots, Q_B$ exceeds m_X then P_1 may learn about more than m_X elements in P_2's input.

   To resolve these problems we introduce a zero-knowledge protocol for verifying that $Q_i \not\equiv 0$ for all $i \in \{1, \ldots, B\}$, and $\sum_{i \in \{1,\ldots,B\}} \mathsf{deg}(Q_i) = m_X$.
2. While party P_2 is supposed to send m_Y pairs of encryptions resulting from substituting a value y (known to P_2) in the (encrypted) polynomials $Q_{h_0(y)}$ and $Q_{h_1(y)}$ it may deviate from his prescribed computation. Thus, P_2's input to the protocol may be ill defined. A solution suggested in [17] solves this problem partially, as it deals with the case where each element y is substituted in a single polynomial. This solution avoids the standard usage of zero-knowledge proofs by P_2 that it indeed followed the protocol. Instead, it enables party P_1 to redo the entire computation supposedly carried out by P_2 on y and verify that its outcome is consistent with the messages received from P_2 (this is where the construction uses a random oracle).

   We remove the dependency on the random oracle and present a solution to the case where y is substituted in two polynomials.

3.2 Checking the Polynomials

Our set-intersection protocol utilizes a zero-knowledge proof of knowledge for the relation[2]

$$\mathcal{R}_{\text{POLY}} = \left\{ \left(\{q_{i,j}\}_{i,j}, m_X, pk \right), \left(\{Q_{i,j}, r_{i,j}\}_{i,j} \right) \middle| \begin{array}{c} \forall i,j \ q_{i,j} = E_{pk}(Q_{i,j}; r_{i,j}) \wedge \\ \sum_i \mathsf{deg}(Q_i(\cdot)) = m_X \wedge \\ \forall i, \ Q_i(\cdot) \not\equiv 0 \end{array} \right\}$$

where $i \in \{1, \ldots, B\}, j \in \{0, \ldots, M\}$.

[2] We will use the convention that the degree of a polynomial $Q_i(\cdot)$ can be chosen to be any integer j' such that $Q_{i,j} = 0$ for all $j \geq j'$, hence equality with m_X can always be achieved.

3.3 Secure Set-Intersection in the Presence of Malicious Adversaries

We now get to the main contribution of this work – a protocol that securely computes $\mathcal{F}_\cap$ in the presence of malicious adversaries, in the standard model; see Figure 1. The main ingredient is a subtle combination of an oblivious pseudo-random function evaluation protocol and a perfectly hiding commitment scheme. Somewhat counter intuitively, the oblivious PRF need not be committed (in a sense that the same key is being reused) – the proof of security shows that although party P_2 that controls the key may change it between invocations, this does not get him any advantage.

The oblivious PRF is used to save on using a (generic) zero-knowledge protocol for party P_2's adherence to the protocol. Recall that in the protocol of Freedman et al. P_1 learns for every $y \in Y$ two values: $r_0 \cdot Q_{h_0(y)}(y) + y$ and $r_1 \cdot Q_{h_1(y)}(y) + y$ where r_0, r_1 are randomly distributed. If $y \notin X$ then both values are random, and reveal no information about y. If, $y \in X$ then one of these values equals y. In our protocol, the 'payload' y of this computation is replaced by a secret s (meaning, P_1 learns two values: $r_0 \cdot Q_{h_0(y)}(y) + s$ and $r_1 \cdot Q_{h_1(y)}(y) + s$). The result of the polynomial evaluation step is, hence, that if $y \notin X$ then P_1 learns no information about s, and if $y \in X$ then P_1 learns s.

The crux of our construction is that the strings r_0, r_1 (as well as other) are not really random. These are pseudorandom strings that are directly derived from $F_{\text{PRF}}(k, s)$. What we get, is that if $y \notin X$ then P_1 learns nothing about s or y. If, on the other hand, $y \in X$ then P_1 learns s, and furthermore after P_1 invokes the oblivious PRF protocol, she can recover y and check that the computations P_2 performed based on the other 'random' strings were performed correctly.

A complication arises as P_2 (who selects the key k for the PRF) computes $F_{\text{PRF}}(k, s)$ by himself, and hence it is impossible for the simulator to extract s from this computation. We thus provide the simulator with an alternative means of extracting s (and also the corresponding y value) by having P_2 commit to both. To guarantee independence of inputs (i.e., that P_1 would not be able to choose his inputs depending on P_2's commitment or vise versa), this commitment is perfectly hiding and is performed *before* P_1 sends the encrypted polynomials representing her input set X.

We continue with a formal description of the protocol.

Protocol 2. ($\pi_\cap$ – secure set-intersection):

- **Inputs:** *The input of P_1 is m_Y and a set $X \subseteq \{0,1\}^{p(n)}$ containing m_X items; the input of P_2 is m_X and a set $Y \subseteq \{0,1\}^{p(n)}$ containing m_Y items (hence, both parties know m_X and m_Y).*
- **Auxiliary inputs:** *A security parameter 1^n, a prime q' such that $q' = 2q+1$ for a prime q. The group $\mathbb{G}$ is the subgroup of quadratic residues modulo q' and g is a generator of $\mathbb{G}$.*
- **Convention:** *Both parties check every received ciphertext for validity (i.e, that it is in $\mathbb{G}$), and abort otherwise.*

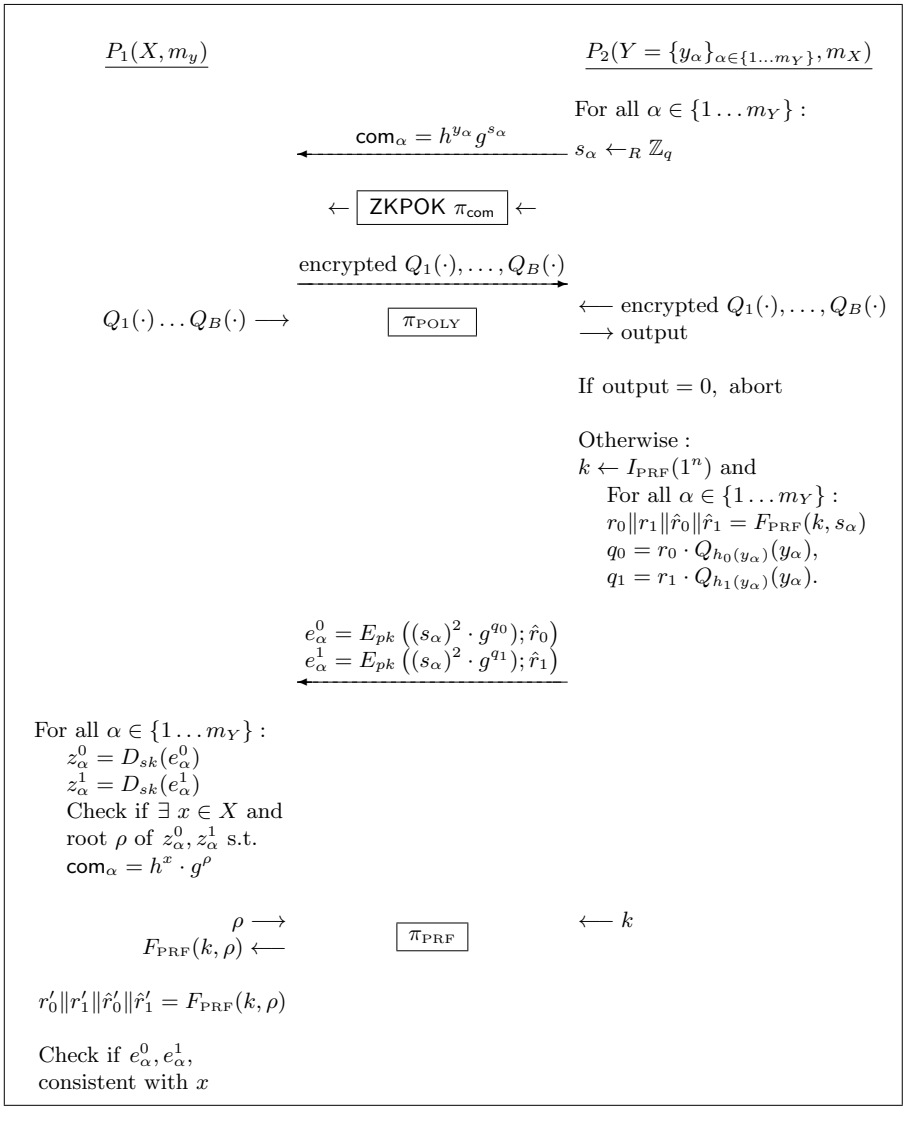

Fig. 1. A high-level diagram of $\pi_\cap$

– **The protocol:**

1. **Key setup for the encryption and commitment schemes:** P_1 *chooses* $t, t' \leftarrow_R \mathbb{Z}_q$*, sets* $h = g^t, h' = g^{t'}$ *and sends* h, h' *to* P_2*. The key for the Pedersen commitment scheme is* h*. The public and private keys for the El Gamal scheme are* $pk = h'$ *and* $sk = t'$*.* P_1 *proves knowledge of* $\log_g h$ *and* $\log_g h'$ *using the zero-knowledge proof of knowledge for* $\mathcal{R}_{\mathrm{DL}}$*.*

2. **Setting the balanced allocation scheme:** *P_1 computes the parameters B, M for the scheme and chooses the seeds for two (pseudo-)random hash functions $h_0, h_1 : \{0,1\}^{p(n)} \to [B]$. She sends B, M, h_0, h_1 to P_2 that checks that the parameters B, M were computed correctly, and aborts otherwise.*
3. P_2 **commits to his input set:** *Let $y_1, \ldots, y_{m_Y}$ be a random ordering of the elements of set Y. For all $\alpha \in \{1, \ldots, m_Y\}$, P_2 chooses $s_\alpha \leftarrow_R \mathbb{Z}_q$ and sends $\mathsf{com}_\alpha = \mathsf{com}(y_\alpha; s_\alpha) = h^{y_\alpha} g^{s_\alpha}$ to P_1. P_2 then proves the knowledge of y_α and s_α by invoking the zero-knowledge proof of knowledge for $\mathcal{R}_{\text{COM}}$.*
4. P_1 **Creates the polynomials representing her input set:** *For every $x \in X$, P_1 maps x into the less occupied bin from $\{h_0(x), h_1(x)\}$ (ties broken arbitrarily). Let $\mathcal{B}_i$ denote the set of elements mapped into bin i. P_1 constructs a polynomial $Q_i(x) \stackrel{\text{def}}{=} \sum_{j=0}^{M} Q_{i,j} \cdot x^j$ of degree at most M whose set of roots is $\mathcal{B}_i$.[3] P_1 encrypts the polynomials' coefficients, setting $q_{i,j} = E_{pk}(g^{Q_{i,j}}; r_{i,j})$, and sends the encrypted coefficients to P_2.*
5. **Checking the polynomials:** *P_1 and P_2 engage in a zero-knowledge execution π_{POLY} for which P_1 proves that the sets $\{q_{i,j}\}_{i \in \{1,\ldots,B\}, j \in \{0,\ldots,M\}}$ were computed correctly, using its witness $\{Q_{i,j}, r_{i,j}\}_{i \in \{1,\ldots,B\}, j \in \{0,\ldots,M\}}$. If the outcome is not 1 then P_2 aborts.*
6. **Evaluating the polynomials:** *P_2 chooses $k \leftarrow I_{\text{PRF}}(1^n)$. Then, P_2 performs the following for all $\alpha \in \{1, \ldots, m_Y\}$:*
   (a) *P_2 computes $F_{\text{PRF}}(k, s_\alpha)$ and parses the result to obtain pseudorandom strings r_0, r_1, $\hat{r}_0$, $\hat{r}_1$ of appropriate lengths for their usage below (i.e., $r_0 \| r_1 \| \hat{r}_0 \| \hat{r}_1 = F_{\text{PRF}}(k, s_\alpha)$).*
   (b) *He sets $\hat{h}_0 = h_0(y_\alpha)$ and $\hat{h}_1 = h_1(y_\alpha)$.*
   (c) *He uses the homomorphic properties of the encryption scheme to evaluate $e_\alpha^0 = E_{pk}(((s_\alpha)^2 \bmod q') \cdot g^{r_0 \cdot Q_{\hat{h}_0}(y_\alpha)}; \hat{r}_0)$ and $e_\alpha^1 = E_{pk}(((s_\alpha)^2 \bmod q') \cdot g^{r_1 \cdot Q_{\hat{h}_1}(y_\alpha)}; \hat{r}_1)$ (where $\hat{r}_0, \hat{r}_1$ denote here the randomness used in the re-encryptions). Then he sends e_α^0, e_α^1 to P_1.*
7. **Computing the intersection:** *For each $\alpha \in \{1, \ldots, m_Y\}$:*
   (a) *P_1 computes $z_\alpha^0 = D_{sk}(e_\alpha^0)$ and $z_\alpha^1 = D_{sk}(e_\alpha^1)$. For each of the (up to four) roots ρ of z_α^0, z_α^1 (roots are computed modulo $q' = 2q+1$ and the result is considered only if it falls within $\mathbb{Z}_q$), she checks if $\mathsf{com}_\alpha / g^\rho$ coincides with h^{x_α} for some $x_\alpha \in X$ (this can be done efficiently by creating a hash table for set $\{h^x : x \in X\}$), and if this is the case sets $\hat{s}_\alpha$ to the corresponding root and marks α.*
   (b) *P_1 and P_2 engage in an execution of the protocol for F_{PRF}. If α is marked, then P_1 enters $\hat{s}_\alpha$ as input, and otherwise she enters a zero. P_2 enters k as input. Let $r_0' \| r_1' \| \hat{r}_0' \| \hat{r}_1'$ denote P_1's output from this execution.*

[3] If $B_i = \emptyset$ then P_1 sets $Q_i(x) = 1$. Otherwise, if $|\mathcal{B}_i| < M$ then P_1 sets the $M+1-|\mathcal{B}_i|$ highest-degree coefficients of $Q_i(\cdot)$ to zero.

(c) *If α is marked, then P_1 checks that $e_\alpha^0 = E_{pk}((\hat{s}_\alpha)^2 \cdot g^{r'_0 \cdot Q_{h_0(x_\alpha)}(x_\alpha)}; \hat{r}'_0)$, and $e_\alpha^1 = E_{pk}((\hat{s}_\alpha)^2 \cdot g^{r'_1 \cdot Q_{h_1(x_\alpha)}(x_\alpha)}; \hat{r}'_1)$ result from applying the homomorphic operations on the encrypted polynomials and randomness r'_0, r'_1. If all checks succeed P_1 records x_α as part of her output.*

A word of explanation is needed for the computation done in Step 6. A natural choice for the payload is s_α itself. However, $s_\alpha \in \mathbb{Z}_q$ whereas the message space of the El Gamal encryption is $\mathbb{G}$. Noting that $\mathbb{Z}_q \subset \mathbb{Z}^*_{q'}$ (neglecting $0 \in \mathbb{Z}_q$), and that by squaring an element of $\mathbb{Z}_{q'}$ we get an element of $\mathbb{G}$, we get that treating s_α as an element of $\mathbb{Z}^*_{q'}$ and computing $(s_\alpha)^2 \bmod q'$ we get an element of $\mathbb{G}$. In this mapping, (up to) two elements of $\mathbb{Z}_q$ share an image in $\mathbb{G}$, and hence in Step 7a we need to recover and check both pre-images.

Before getting into the proof of security, we observe that if both parties are honest, then P_1 outputs $X \cap Y$ with probability negligibly close to one. In this case, if for an element $y_\alpha \in Y$ is holds that $y_\alpha \in X$ then one of $Q_{h_0(y_\alpha)}(y_\alpha)$, $Q_{h_1(y_\alpha)}(y_\alpha)$ is zero, and otherwise none of $Q_{h_0(y_\alpha)}(y_\alpha)$, $Q_{h_1(y_\alpha)}(y_\alpha)$ is zero. We get:

1. If $y_\alpha \in X \cap Y$ then one of e_α^0, e_α^1 encrypts $(s_\alpha)^2 \bmod q'$. Hence, there exists a root ρ of z_α^0, z_α^1 such that $\mathsf{com}_\alpha / g^\rho$ coincides with h^{x_α} for some $x_\alpha \in X$, resulting in P_1 marking α. Furthermore, as $r_0, r_1, \hat{r}_0, \hat{r}_1$ are derived from $F_{\text{PRF}}(k, s_\alpha)$, the check done by P_1 in Step 7 succeeds and P_1 records y_α in her output.
2. If $y_\alpha \notin X \cap Y$ then none of e_α^0, e_α^1 encrypts $(s_\alpha)^2 \bmod q'$ and (except for a negligible probability) $\mathsf{com}_\alpha / g^\rho$ coincides with h^{x_α} for no root ρ of z_α^0, z_α^1 and $x_\alpha \in X$. Hence, P_1 does not mark α, and y_α is not considered in Step 7, and not included in P_1's output.

Theorem 1. *Assume that π_{DL}, π_{PRF} and π_{POLY} are as described above, that (G, E, D) is the El Gamal encryption scheme, and that* com *is a perfectly-hiding commitment scheme. Then $\pi_\cap$ (Protocol 2) securely computes $\mathcal{F}_\cap$ in the presence of malicious adversaries.*

Efficiency. We first note that the protocol is constant round (as all its zero-knowledge proofs and subprotocols are constant round). The costs of using current implementations of F_{PRF} on inputs of length $p(n)$ is that of $p(n)$ oblivious transfer invocations [21], and hence of $O(p(n))$ modular exponentiations. We get that the overall communication costs are of sending $O(m_X + m_Y p(n))$ group elements, and the computation costs are of performing $O(m_X + m_Y(\log\log m_X + p(n)))$ modular exponentiations.

Optimizations. Note first that if the functionality is changed to allow P_2 learn the size of the intersection $m_{X\cap Y}$, then, in Step 7b, it is possible to avoid invoking π_{PRF} when α is not marked. This yields a protocol where $O(m_X + m_{X\cap Y} \cdot p(n))$ group elements are sent, and $O(m_X + m_Y \cdot \log\log m_X + m_{X\cap Y} \cdot p(n))$ exponentiation are computed. When $m_{X\cap Y} \ll m_Y$, this protocol is significantly more efficient than those suggested in [21] for weaker adversarial models. Furthermore, an improved scheme for oblivious pseudorandom function evaluation with overall complexity which is independent of the input length yields a better efficiency as well.

3.4 A Very Efficient Heuristic Construction

Note that we can now modify protocol $\pi_\cap$ to get a protocol in the random oracle model $\pi_\cap^{\text{RO}}$ by performing the following two changes: (1) the computation of $F_{\text{PRF}}(k, s_\alpha)$ performed by P_2 in Step 6a of $\pi_\cap$ is replaced with an invocation of the random oracle, i.e., P_2 computes $\mathcal{H}(s_\alpha)$; and (ii) the execution of the secure protocol for evaluating F_{PRF} by P_1 and P_2 in Step 7 of $\pi_\cap$ is replaced with an invocation of the random oracle by P_1, i.e., no communication is needed, and instead of providing s' to the protocol for F_{PRF}, P_1 computes $\mathcal{H}(s')$.

A typical proof of security in the random oracle model relies on the simulator's ability to record the inputs on which the random oracle is invoked, and the recorded information is used by the simulator for malicious P_2 while recovering his input. In other words, the proof of security relies on the property of the random oracle that the only way to learn any information about $\mathcal{H}(s)$ is to apply $\mathcal{H}$ on a *well defined input* s. Should $\pi_\cap^{\text{RO}}$ be implemented such that the invocations of the random oracle are replaced by a concrete computation of some function, it seems that this proof of security would collapse, even if very strong hardness assumptions are made with respect to this implementation.

Nevertheless, the situation in protocol $\pi_\cap$ is very different. Note, in particular, that the simulator for malicious P_2 cannot monitor P_2's input to F_{PRF} (nor is this notion of inputs to the function well defined). Instead, the simulator extracts s from the zero-knowledge proof of knowledge for the commitment on P_2's inputs in Step 3 of $\pi_\cap$. This is inherited by the modified protocol $\pi_\cap^{\text{RO}}$. Hence, should the random oracle calls in $\pi_\cap^{\text{RO}}$ be replaced with some primitive Gen, the proof of security may still hold with small modifications, given the hardness assumption on Gen (intuitively, some functions of the outcome of $\mathsf{Gen}(s)$ and s should look random). $\pi_\cap^{\text{RO}}$ can hence be viewed as an intermediate step between the protocol in [17] that utilizes a random oracle to cope with malicious parties, and the protocol suggested in the current paper. If the primitive Gen is realized efficiently (e.g., if its computation incurs a constant number of exponentiations), we get an extremely efficient protocol for $\mathcal{F}_\cap$, where the communication costs are of sending $O(m_X + m_Y)$ group elements, and the number of exponentiations is $O(m_X + m_Y \log\log m_X)$.

For the sake of completeness we include a formal description of protocol $\pi_\cap^{\mathsf{Gen}}$, that is identical to protocol $\pi_\cap$ except for the replacing every invocation of $F_{\text{PRF}}(k, \cdot)$ by a computation of $\mathsf{Gen}(\cdot)$. Note that unlike in an invocation of $F_{\text{PRF}}(k, \cdot)$, no communication is needed for computing $\mathsf{Gen}(\cdot)$.

Protocol 3. ($\pi_\cap^{\mathsf{Gen}}$ – secure set-intersection with a "generator"):

- **Inputs:** *The input of* P_1 *is* m_Y *and a set* $X \subseteq \{0,1\}^{p(n)}$ *containing* m_X *items; the input of* P_2 *is* m_X *and a set* $Y \subseteq \{0,1\}^{p(n)}$ *containing* m_Y *items (hence, both parties know* m_X *and* m_Y*).*
- **Auxiliary inputs:** *A security parameter* 1^n*, a prime* q' *such that* $q' = 2q+1$ *for a prime* q*. The group* $\mathbb{G}$ *is the subgroup of quadratic residues modulo* q' *and* g *is a generator of* $\mathbb{G}$*.*

– **Convention:** *Both parties check every received ciphertext for validity (i.e, that it is in $\mathbb{G}$), and abort otherwise.*

– **The protocol:**

1. **Key setup for the encryption and commitment schemes:** *P_1 chooses $t, t' \leftarrow_R \mathbb{Z}_q$, sets $h = g^t, h' = g^{t'}$ and sends h, h' to P_2. The key for the Pedersen commitment scheme is h. The public and private keys for the El Gamal scheme are $pk = h'$ and $sk = t'$. P_1 proves knowledge of $\log_g h$ and $\log_g h'$ using the zero-knowledge proof of knowledge for $\mathcal{R}_{\text{DL}}$.*
2. **Setting the balanced allocation scheme:** *P_1 computes the parameters B, M for the scheme and chooses the seeds for two (pseudo-)random hash functions $h_0, h_1 : \{0,1\}^{p(n)} \to [B]$. She sends B, M, h_0, h_1 to P_2 that checks that the parameters B, M were computed correctly, and aborts otherwise.*
3. **P_2 commits to his input set:** *Let $y_1, \ldots, y_{m_Y}$ be a random ordering of the elements of set Y. For all $\alpha \in \{1, \ldots, m_Y\}$, P_2 chooses $s_\alpha \leftarrow_R \mathbb{Z}_q$ and sends $\mathsf{com}_\alpha = \mathsf{com}(y_\alpha; s_\alpha) = h^{y_\alpha} g^{s_\alpha}$ to P_1. P_2 then proves the knowledge of y_α and s_α by invoking the zero-knowledge proof of knowledge for $\mathcal{R}_{\text{COM}}$.*
4. **P_1 Creates the polynomials representing her input set:** *For every $x \in X$, P_1 maps x into the less occupied bin from $\{h_0(x), h_1(x)\}$ (ties broken arbitrarily). Let $\mathcal{B}_i$ denote the set of elements mapped into bin i. P_1 constructs a polynomial $Q_i(x) \stackrel{\text{def}}{=} \sum_{j=0}^{M} Q_{i,j} \cdot x^j$ of degree at most M whose set of roots is $\mathcal{B}_i$.[4] P_1 encrypts the polynomials' coefficients, setting $q_{i,j} = E_{pk}(g^{Q_{i,j}}; r_{i,j})$, and sends the encrypted coefficients to P_2.*
5. **Checking the polynomials:** *P_1 and P_2 engage in a zero-knowledge execution π_{POLY} for which P_1 proves that the sets $\{q_{i,j}\}_{i \in \{1,\ldots,B\}, j \in \{0,\ldots,M\}}$ were computed correctly, using its witness $\{Q_{i,j}, r_{i,j}\}_{i \in \{1,\ldots,B\}, j \in \{0,\ldots,M\}}$. If the outcome is not 1 then P_2 aborts.*
6. **Evaluating the polynomials:** *For all $\alpha \in \{1, \ldots, m_Y\}$ P_2 performs the following :*
   (a) *He sets $\hat{h}_0 = h_0(y_\alpha)$ and $\hat{h}_1 = h_1(y_\alpha)$.*
   (b) *He parses $\mathsf{Gen}(s_\alpha)$ to obtain pseudorandom strings $r_1, r_2, \hat{r}_0, \hat{r}_1$ of appropriate lengths for their usage below (i.e., $r_1 \| r_2 \| \hat{r}_0 \| \hat{r}_1 = \mathsf{Gen}(s_\alpha)$).*
   He uses the homomorphic properties of the encryption scheme to evaluate $e^0_\alpha = E_{pk}(((s_\alpha)^2 \bmod q') \cdot g^{r_0 \cdot Q_{\hat{h}_0}(y_\alpha)}; \hat{r}_0)$ and $e^1_\alpha = E_{pk}(((s_\alpha)^2 \bmod q') \cdot g^{r_1 \cdot Q_{\hat{h}_1}(y_\alpha)}; \hat{r}_1)$ (where $\hat{r}_0, \hat{r}_1$ denote here the randomness used in the re-encryptions). Then he sends e^0_α, e^1_α to P_1.
7. **Computing the intersection:** *For each $\alpha \in \{1, \ldots, m_Y\}$:*
   (a) *P_1 computes $z^0_\alpha = D_{sk}(e^0_\alpha)$ and $z^1_\alpha = D_{sk}(e^1_\alpha)$. For each of the (up to four) roots ρ of z^0_α, z^1_α (roots are computed modulo $q' = 2q+1$ and the*

[4] If $B_i = \emptyset$ then P_1 sets $Q_i(x) = 1$. Otherwise, if $|\mathcal{B}_i| < M$ then P_1 sets the $M+1-|\mathcal{B}_i|$ highest-degree coefficients of $Q_i(\cdot)$ to zero.

result is considered only if it falls within $\mathbb{Z}_q$), she checks if $\mathsf{com}_\alpha / g^\rho$ coincides with h^{x_α} for some $x_\alpha \in X$ (this can be done efficiently by creating a hash table for set $\{h^x : x \in X\}$), and if this is the case sets $\hat{s}_\alpha$ to the corresponding root and marks α.

(b) If α is marked, then P_1 parses $\mathsf{Gen}(\hat{s}_\alpha)$ to obtain $r'_0, r'_1, \hat{r}'_0, \hat{r}'_1$.

(c) P_1 checks that $e^0_\alpha = E_{pk}((\hat{s}_\alpha)^2 \cdot g^{r'_0 \cdot Q_{h_0(x_\alpha)}(x_\alpha)}; \hat{r}'_0)$, and $e^1_\alpha = E_{pk}((\hat{s}_\alpha)^2 \cdot g^{r'_1 \cdot Q_{h_1(x_\alpha)}(x_\alpha)}; \hat{r}'_1)$ result from applying the homomorphic operations on the encrypted polynomials and randomness r'_0, r'_1. If all checks succeed P_1 records x_α as part of her output.

References

1. Aggarwal, G., Mishra, N., Pinkas, B.: Secure Computation of the kth-Ranked Element. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 40–55. Springer, Heidelberg (2004)
2. Aumann, Y., Lindell, Y.: Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 137–156. Springer, Heidelberg (2007)
3. Azar, Y., Broder, A.Z., Karlin, A.R., Upfal, E.: Balanced Allocations. SIAM Journal on Computing 29(1), 180–200 (1999)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non cryptographic fault tolerant distributed computations. In: 20th STOC, pp. 1–10 (1988)
5. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
6. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th STOC, pp. 11–19 (1988)
7. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
8. Damgård, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000)
9. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
10. Damgård, I., Nielsen, J.B.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 3–42. Springer, Heidelberg (2002)
11. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient Robust Private Set Intersection. In: Ghilardi, S. (ed.) ANCS 2009. LNCS, vol. 5479, pp. 125–142. Springer, Heidelberg (2009)
12. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1984)
13. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M.J., Wright, R.N.: Secure multiparty computation of approximations. ACM Transactions on Algorithms (TALG) 2(3), 435–472 (2006)

14. Fouque, P., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 573–584. Springer, Heidelberg (2000)
15. Fouque, P., Poupard, G., Stern, J.: Sharing decryption in the context of voting of lotteries. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 90–104. Springer, Heidelberg (2009)
16. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword Search and Oblivious Pseudorandom Functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005)
17. Freedman, M., Nissim, K., Pinkas, B.: Efficient Private Matching and Set-Intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
18. Goldreich, O.: Foundations of Cryptography: Volume 2 – Basic Applications. Cambridge University Press, Cambridge (2004)
19. Goldreich, O., Kahan, A.: How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology 9(3), 167–190 (1996)
20. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: 19th STOC, pp. 218–229 (1987)
21. Hazay, C., Lindell, Y.: Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
22. Hazay, C., Nissim, K.: Efficient Set Operations in the Presence of Malicious Adversaries. Cryptology ePrint Archive, Report 2009/594 (2009), http://eprint.iacr.org/
23. Jarecki, S., Liu, X.: Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
24. Kiltz, E., Mohassel, P., Weinreb, E., Franklin, M.K.: Secure Linear Algebra Using Linearly Recurrent Sequences. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 291–310. Springer, Heidelberg (2007)
25. Kissner, L., Song, D.X.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005); See technical report CMU-CS-05-113 for the full version
26. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. Journal of Cryptology 15(3), 177–206 (2002)
27. Mohassel, P., Weinreb, E.: Efficient Secure Linear Algebra in the Presence of Covert or Computationally Unbounded Adversaries. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 481–496. Springer, Heidelberg (2008)
28. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: 33th STOC, pp. 590–599 (2001)
29. Naor, M., Reingold, O.: Number-Theoretic Constructions of Ecient Pseudo-Random Functions. In: 38th FOCS, pp. 231–262 (1997)
30. Nissim, K., Weinreb, E.: Communication Efficient Secure Linear Algebra. In: 4th TCC, pp. 522–541 (2006)
31. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
32. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)

33. Pedersen, T.P.: Non-Interactive and Information-Theoretical Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
34. Peikert, C., Vaikuntanathan, V., Waters, B.: A Framework for Efficient and Composable Oblivious Transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
35. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
36. Vocking, B.: How Asymmetry Helps Load Balancing. Journal of the ACM 50(4), 568–589 (2003)
37. Yao, A.C.: Protocols for secure computations. In: 23rd FOCS, pp. 160–164 (1982)