

# Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model

Emiliano De Cristofaro<sup>1</sup>, Jihye Kim<sup>2</sup>, and Gene Tsudik<sup>1</sup>

<sup>1</sup> Computer Science Department, University of California, Irvine

<sup>2</sup> Department of Mathematical Sciences, Seoul National University

**Abstract.** Private Set Intersection (PSI) protocols allow one party (“client”) to compute an intersection of its input set with that of another party (“server”), such that the client learns nothing other than the set intersection and the server learns nothing beyond client input size. Prior work yielded a range of PSI protocols secure under different cryptographic assumptions. Protocols operating in the semi-honest model offer better (linear) complexity while those in the malicious model are often significantly more costly. In this paper, we construct PSI and Authorized PSI (APSI) protocols secure in the malicious model under standard cryptographic assumptions, with both *linear* communication and computational complexities. To the best of our knowledge, our APSI is the first solution to do so. Finally, we show that our linear PSI is appreciably more efficient than the state-of-the-art.

## 1 Introduction

Private set intersection (PSI) protocols allow two parties – a server and a client – to interact on their respective input sets, such that the client only learns the intersection of the two sets, while the server learns nothing (beyond the client input set size). PSI addresses several realistic privacy issues. Typical application examples include:

1. **Aviation Security:** The U.S. Department of Homeland Security (DHS) needs to check whether any passenger on each flight from/to the United States must be denied boarding or disembarkation, based on so-called *Terror Watch List*. Today, airlines surrender their entire passenger manifests to DHS, together with other sensitive information, such as credit card numbers. Besides privacy implications, this modus operandi poses liability issues with regard to (for the most part) innocent passengers’ data and concerns about potential data losses. Ideally, DHS would obtain information *only* pertaining to passengers on the list, while not disclosing any information to the airlines.
2. **Healthcare:** Insurance companies often need to obtain information about their insured patients from other parties, such as other insurance carriers or hospitals. The former cannot disclose the identity of inquired patients, whereas, the latter cannot provide any information on other patients.
3. **Law Enforcement:** Investigative agencies (e.g., the FBI) need to obtain information on suspects from other agencies, e.g., local police departments, the military, DMV, IRS, or employers. In many cases, it is dangerous (or simply forbidden) for the FBI to disclose subjects of investigation. For their part, other parties cannot

disclose their entire data-sets and need the FBI to access only desired information. Also, the FBI requests might need to be *pre-authorized* by some appropriate trusted authority (e.g., a federal judge, via a warrant). This way, the FBI can only obtain information related to legitimate requests.

## 1.1 Adversaries in PSI

Over the last years, PSI-related research has yielded several PSI constructs, with a wide range of adversarial models, security assumptions, and efficiency characteristics. One major distinguishing factor is the adversarial model which is typically either semi-honest or malicious. (Note that, in the rest of this paper, the term *adversary* refers to insiders, i.e., protocol participants. Outside adversaries are not considered, since their actions can be mitigated via standard network security techniques.)

Following Goldreich’s definition [Gol04], protocols secure in the presence of **semi-honest adversaries** (or honest-but-curious) assume that parties faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., set size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about the other party’s input. This model is formalized by requiring that each party does not learn more information that it would in an ideal implementation relying on a trusted third party (TTP).

In contrast, security in the presence of **malicious parties** allows *arbitrary* deviations from the protocol. In general, however, it does not prevent parties from refusing to participate in the protocol, modifying their private input sets, or prematurely aborting the protocol. Security in the malicious model is achieved if the adversary (interacting in the real protocol, without the TTP) can learn no more information than it could in the ideal scenario. In other words, a secure PSI emulates (in its real execution) the ideal execution that includes a trusted third party. This notion is formulated by requiring the existence of adversaries in the ideal execution model that can *simulate* adversarial behavior in the real execution model.

## 1.2 Authorized (Client) Input

Malicious parties cannot be prevented from modifying their input sets, even if a protocol is proven secure in the malicious model. Considering that the client learns the intersection while the server learns nothing, this appears a severe threat to server’s privacy. For instance, suppose that a malicious client faithfully follows the protocol, but populates its input set with its best guesses of the server set (especially, if the set is easy to exhaustively enumerate). This would maximize the amount of information it learns. In the extreme case, the client could even claim that its set contain all possible elements. Although the server could impose a limit on this size, the client could still vary its set over multiple protocol runs.

We claim that this issue cannot be effectively addressed without some mechanism to *authorize* client inputs. Consequently, a trusted certification authority (CA) is needed to certify input sets, as proposed in [DJKT09, CZ09]. This variant is called “Authorized Private Set Intersection” (APSI) in [DT10]. Note that the CA is an off-line entity; it is neither trusted, nor involved in, computing the intersection.

As discussed above, input authorization ensures that malicious clients cannot manipulate their inputs to harm server privacy. However, this does not help at all as far as manipulation of server inputs. One way towards security against malicious servers would be to introduce authorization for server input, along the same lines as client input authorization. Although this would likely yield protocols secure in the malicious model, we choose not to pursue this direction. The main reason is that, it is more natural for the client (who learns the intersection) to be authorized on its input, than for the server (who learns nothing). However, though it is outside the scope of this paper, we believe that enforcing both server and client input authorization is a subject worth investigating. Finally, we leave as an open question whether we can *reduce* security of PSI in the malicious model to authorization of both client and server inputs.

### 1.3 Technical Roadmap and Contributions

Over the last few years, several elegant (if not always efficient) PSI and APSI protocols have been proposed, that are secure in the malicious model, under standard assumptions [KS05, HL08, DSMRY09, CZ09, CKRS09, HN10]. Only [JL09] presents a linear-complexity PSI protocol secure in the malicious setting. Its proof requires that the domain of inputs to be restricted to polynomial in the security parameter and requires a Common Reference String model (CRS), where the reference string, including a safe RSA modulus, must be generated by a mutually trusted third party. Other results (such as [DT10]) construct linear-complexity PSI and APSI protocols secure in the semi-honest model, under assumptions of the *one-more-XXX* type [BNPS03], with much lower computational and communication complexity. (Note that we overview prior work in Section 2). As shown in [DT10], via both analysis and experiments, there is an appreciable efficiency gap between the two “families” of PSI/APSI protocols: those secure in the malicious and in the semi-honest models. In this paper, our main goal is to construct efficient PSI and APSI protocols secure under standard assumptions, with malicious participants (both server and client).

Our starting point are the linear-complexity protocols from [DT10] (specifically, Figure 2 and 3), which are secure only in the semi-honest model. First, we modify the APSI construct of [DT10] and obtain APSI protocol secure in the malicious model, under the standard RSA assumption (in ROM). Then, we modify its PSI counterpart: while the linear-complexity PSI protocol in [DT10] is secure under the One-More-Gap-DH assumption [BNPS03] against semi-honest parties, our modified variant is secure in the malicious model under the standard DDH assumption (again, in ROM). We present formal proofs for all proposed protocols.

Contributions of our work are:

1. To the best of our knowledge, our APSI protocol is the *first* result with *linear* communication and computational complexity, in the malicious model. (Previous work achieved quadratic computational complexity.)
2. Our PSI protocol also offers linear complexity. Although some prior work (i.e., [JL09]) also achieves the same asymptotic bound, we do not require the CRS model and our proof does not restrict input domain size. We also show that our protocol incurs *significantly reduced* constant factors.

3. We prove security of proposed protocols, in presence of malicious adversaries, under standard cryptographic (RSA and DDH) assumptions, in ROM.

**Organization.** Section 2 overviews previous work. Then, after some preliminaries in Section 3, we present our constructions in Sections 4 and 5. Next, Section 6 discusses the efficiency of our constructs and Section 7 concludes the paper.

## 2 Related Work

This section overviews prior work on PSI and APSI.

### 2.1 Prior Work on PSI

It is well known that PSI could be realized via general secure two-party computation [Yao82]. However, it is usually far more efficient to have dedicated protocols (see [FNP04, KS05]); which is the direction we pursue in this paper. From here on, we consider PSI as an interaction between a server  $S$  and a client  $C$ . The server set contains  $w$  items, while the client set  $-v$ .

Freedman, et al. [FNP04] introduce the concept of PSI and presented protocols based on *Oblivious Polynomial Evaluation* (OPE) [NP06]. The basic idea is to represent a set as a polynomial, with individual elements as its roots. The construction for the semi-honest setting incurs linear communication, and quadratic computational, complexity. Using Horner’s rule and balanced bucket allocation, the number of modular exponentiations can be reduced to  $O(w \log \log v)$  exponentiations for the server and  $O(w + v)$  exponentiations for the client. [FNP04] also gives constructions for a malicious client and semi-honest server. This protocol uses a cut-and-choose strategy, thus, the overhead is increased by a statistical security parameter. Also presented is a protocol secure in the presence of a malicious server and a semi-honest client in ROM.

Kissner and Song [KS05] propose OPE-based protocols for *mutual* PSI (as well as for additional set operations), and may involve more than two players. Protocols are secure in the standard model against semi-honest and also malicious adversaries. The former incurs quadratic ( $O(wv)$ ) computation (but linear communication) overhead. The latter uses (expensive) generic zero-knowledge proofs to prevent parties from deviating to the protocol. Later, Dachman-Soled, et al. [DSMRY09] present an improved PSI construction, based on [KS05]. Their construction incorporates a secret sharing of polynomial inputs. Since Shamir’s secret sharing [Sha84] implies Reed Solomon codes, they do not need generic zero-knowledge proofs. Complexity of their protocol amounts to  $O(wk^2 \log^2(v))$  in communication and  $O(wvk \log(v) + wk^2 \log^2(v))$  in computation, being  $k$  the security parameter.

Another family of protocols rely on so-called *Oblivious Pseudo-Random Functions* (OPRF-s). An OPRF is a two-party protocol (between a sender and a receiver) that securely computes a pseudorandom function  $f_k(\cdot)$  on key  $k$  contributed by the sender and input  $x$  contributed by the receiver, such that the former learns nothing from the interaction, and the latter learns only the value  $f_k(x)$ . OPRF-based PSI-s work as follows: Server  $S$  holds a secret random key  $k$ . Then, for each  $s_j \in \mathcal{S}$ ,  $S$  computes  $u_j = f_k(s_j)$ , and publishes (or sends the client) the set  $\mathcal{U} = \{u_1, \dots, u_w\}$ . Then,  $C$  and  $S$  engage in

an OPRF computation of  $f_k(c_i)$  for each  $c_i \in \mathcal{C}$  (of size  $v$ ), such that  $S$  learns nothing about  $\mathcal{C}$  (except the size) and  $C$  learns  $f_k(c_i)$ . Finally,  $C$  obtains  $c_i \in \mathcal{C} \cap \mathcal{S}$  if and only if  $f_k(c_i) \in \mathcal{U}$ . The idea of using OPRFs for PSI protocols is due to Hazay and Lindell [HL08], who propose one solution with security against malicious adversaries with one-sided simulatability, and one – against covert adversaries [AL07].

This protocol has been later improved by Jarecki and Liu [JL09], who proposed a protocol secure in the standard model in the presence of both malicious parties, based on the Decisional  $q$ -Diffie-Hellman Inversion assumption, in the Common Reference String (CRS) model, where a safe RSA modulus is generated by a trusted third party. Encryption operations are performed using an additively homomorphic encryption scheme, such as Camenisch and Shoup [CS03]. As pointed out in [JL09], this approach can be further optimized, based on the concurrent work in [BCC<sup>+</sup>09]. Assuming such improved construction, [JL09] incurs the following computational complexity: Let  $m$  be the number of bits needed to represent each set item; the server performs at least  $O(w)$  PRF evaluations, i.e., both  $m$ -bit and group exponentiations, plus  $O(v)$  group exponentiations, whereas, the client at least  $O(v)$   $m$ -bit exponentiations plus  $O(v)$  group exponentiations. We discuss in details the complexity of this solution later in the paper. Finally, note that the proof in [JL09] requires the ability to exhaustively search over the input domain, i.e., the input domain size of the PRF should be polynomial in the security parameter.

A recent result by Hazay and Nissim [HN10] presents an improved construction of OPE-based PSI based on [FNP04], but without ROM. Specifically, it introduces zero-knowledge proofs that allow client to demonstrate that encrypted polynomials are correctly produced. Also, it uses a technique based on a perfectly hiding commitment scheme with an OPRF evaluation protocol to prevent the server from deviating from the protocol. The PSI protocol in [HN10] incurs  $O(v + w(\log \log v + m))$  computational and  $O(v + w \cdot m)$  communication complexity, where  $m$  is the number of bits needed to represent a set item. Note that execution of the underlying OPRF in [HN10] requires  $m$  oblivious transfer invocations, and hence  $O(m)$  modular exponentiations, for each set item. However, such overhead can be avoided by instantiating the protocol in ROM. This protocol can be also optimized if the size of the intersection is allowed to be leaked to the server, in contrast to our strict privacy definitions (see Section 3.3). Nonetheless, the resulting protocol is of sending  $O(v + |\mathcal{S} \cap \mathcal{C}| \cdot m)$  and computing  $O(v + w \cdot \log \log v + |\mathcal{S} \cap \mathcal{C}| \cdot m)$ , which is still not linear. (Also recall that it is not clear how to enable convert the PSI construct of [HN10] into APSI.)

In another recent result, [DT10] (Fig.4) presents an adaptive PSI protocol based on blind-RSA signatures [Cha83], secure in the semi-honest model, under the One-More-RSA assumption [BNPS03], in ROM. Specifically, during an initialization phase, the server generates RSA keys  $(N, e, d)$  and commits to its set, by publishing the hash of the RSA signature of each item. During the interaction, the client obtains blind-RSA signatures of its items from the server. Thus, the server needs to compute  $O(w)$  RSA signatures during the initialization phase, and  $O(v)$  online. Whereas, the client (assuming  $e = 3$ ) only computes  $O(v)$  multiplications, thus making this construct particularly appealing for clients running on limited-resource devices.

[DT10] (Fig.3) includes another PSI secure in the presence of semi-honest adversaries, under the One-More-Gap-DH assumption, in ROM. Common inputs are primes  $p, q$  (with  $q|p-1$ , the order of a subgroup of  $\mathbb{Z}_p^*$ ) and a generator of the subgroup,  $g$ . First, the client computes the accumulator  $PCH = \prod_{i=1}^v (H(c_i))$  and sends  $X = PCH \cdot g^{R_c}$  for  $R_c$  random in  $\mathbb{Z}_q^*$ . Also, for  $i = 1, \dots, v$ , it computes  $PCH_i = PCH/H(c_i)$  and sends  $x_i = PCH_i \cdot g^{R_{c:i}}$  for  $R_{c:i}$  in  $\mathbb{Z}_q^*$ . The server picks a random  $R_s$  in  $\mathbb{Z}_q^*$ , sends  $Z = g^{R_s}$ , and, for each  $x_i$ , sends back  $x'_i = x_i^{R_s}$ . Then, for  $j = 1, \dots, w$ , it computes  $T_{s:j} = H'((X/H(s_j))^{R_s})$ . Finally, the client computes  $T_{c:i} = x'_i \cdot Z^{R_c} \cdot Z^{-R_{c:i}}$ , and learns that  $c_i \in \mathcal{C} \cap \mathcal{S}$  if  $T_{c:i} = T_{s:j}$ . Computational complexity of this protocol is  $O(w+v)$  and  $O(v)$  exponentiations (with short exponents) for the server and client, respectively.

## 2.2 Prior Work on APSI

Authorized Private Set Intersection (APSI) is defined in [DT10] to extend PSI to support *authorization* of client inputs. Each client input must be authorized (via signing) by some trusted authority, e.g., a CA. Recall the third example in Section 1: to obtain information on a suspect from her employer, the FBI needs to be duly authorized. APSI represents an authorization using a digital signature. Note that authorizations obtained from the CA are private to the client and cannot be disclosed to the server.

[DT10] shows that the PSI protocol in its Fig.3 (reviewed at the end of Section 2.1) can be instantiated in a RSA setting, where client input is a set of RSA signatures and the server obviously verifies them by modifying the protocol as follows. The client  $C$  needs to obtain from the CA signatures  $\sigma_i = H(c_i)^d$  (for input set  $\mathcal{C} = \{c_1, \dots, c_v\}$ ).  $C$  computes the accumulator  $PCH^* = \prod_{i=1}^v \sigma_i$  and sends  $X = PCH^* \cdot g^{R_c}$  for random  $R_c$ . Also, it computes  $PCH_i^* = PCH^*/\sigma_i$  and sends  $x_i = PCH_i^* \cdot g^{R_{c:i}}$  for random  $R_{c:i}$ . The server picks a random  $R_s$ , sends  $Z = g^{eR_s}$ , and, for each  $x_i$ , sends back  $x'_i = x_i^{eR_s}$ . Then, for  $j = 1, \dots, w$ , it computes  $T_{s:j} = H'((X^e/H(s_j))^{R_s})$ . Finally, the client computes  $T_{c:i} = x'_i \cdot Z^{R_c} \cdot Z^{-R_{c:i}}$ , and learns that  $c_i \in \mathcal{C} \cap \mathcal{S}$  if  $T_{c:i} = T_{s:j}$ . Asymptotic complexity of this solution is the same as that of the standard PSI presented above, i.e.,  $O(w+v)$  and  $O(v)$  exponentiations for the server and client, respectively. (Although short exponents are replaced with “RSA” exponents.) The resulting protocol is secure in the semi-honest model, under the standard RSA assumption, in ROM. Note that the use of “authorized” client inputs seems to increase server privacy: under the RSA assumption, the client does not learn any information about server inputs, unless it holds a valid RSA signature. In other words, there appears to be a strong correlation between server privacy and client’s difficulty of forging signatures.

A similar concept (adaptable to APSI) is Public-Key Encryption with Oblivious Keyword Search in [CKRS09]. It proposes an Identity-based cryptosystem (inspired by PEKS in [BDOP04]), where the client obtains authorized search trapdoors from a CA, and uses them to search over data encrypted by the server. The client learns only the information matching the authorized trapdoors, whereas, the server learns nothing. The protocol is secure in the presence of malicious adversaries in the standard model, under the Decision Bilinear Diffie-Hellman assumption [BF03]. It uses a modification of the Boyen-Waters IBE [BW06]. Even without taking into account zero-knowledge proofs,

the server would compute  $O(w)$  encryptions of [BW06] (each requiring 6 exponentiations and a representation of 6 group elements). The client would need to test each of the  $O(w)$  PEKS against its  $O(v)$  trapdoors, hence performing  $O(w \cdot v)$  decryptions.

Finally, [CZ09] introduces another similar notion – Private Intersection of Certified Sets. This construct allows a trusted third party to ensure that all protocol inputs are valid and bound to each protocol participant. The proposed protocol is *mutual* (i.e., both parties receive the intersection) and builds upon oblivious polynomial evaluation and achieves *quadratic* computation and communication overhead.

### 3 Preliminaries

In this section, we present our cryptographic assumptions and tools, as well as security model. We introduce our notation in Table 1.

**Table 1.** Notation

$a \leftarrow_r \mathcal{A}$	variable $a$ is chosen uniformly at random from set $\mathcal{A}$
$\kappa$	security parameter
$N = pq$	safe RSA modulus with at least $\kappa$ -bit security
$e, d$	public and private exponents of RSA
$Z_{N/2}$	1/2 of bit-size of $N$
$H()$	random oracle $H : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$
$H_1()$	random oracle $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$ ( $\mathbb{G}$ depends on the context)
$H_2()$	random oracle $H_2 : \mathbb{G} \times \mathbb{G} \times \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ ( $\mathbb{G}$ depends on the context)
$\mathcal{C}, \mathcal{S}$	client and server sets, respectively
$v, w$	sizes of $\mathcal{C}$ and $\mathcal{S}$ , respectively
$i \in [1, v]$	indices of elements of $\mathcal{C}$
$j \in [1, w]$	indices of elements of $\mathcal{S}$
$c_i, s_j$	$i$ -th and $j$ -th elements of $\mathcal{C}$ and $\mathcal{S}$ , respectively
$hc_i, hs_j$	$H_1(c_i)$ and $H_1(s_j)$ , respectively
$\sigma_i$	$H_1(c_i)^d$ , RSA-signature on client item

#### 3.1 Cryptographic Assumptions

**Definition 1.** Let  $\mathbb{G}$  be a cyclic group and let  $g$  be its generator. Assume that the bit-length of the group size is  $l$ . The DDH problem is hard in  $\mathbb{G}$  if for every efficient algorithm  $A$  the probability:

$$\left| \Pr[x, y \leftarrow_r \{0, 1\}^l : A(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \leftarrow_r \{0, 1\}^l : A(g, g^x, g^y, g^z) = 1] \right|$$

is a negligible function of  $\kappa$ .

**Definition 2.** Let  $\text{RSA-Gen}(1^\kappa)$  be an algorithm that outputs so-called “safe RSA instances”, i.e. pairs  $(n, e)$  where  $n = pq$ ,  $e$  is a small prime such that  $\text{gcd}(e, \phi(n)) = 1$ , and  $p, q$  are random  $\kappa$ -bit primes subject to the constraint that  $p = 2p' + 1, q = 2q' + 1$  for prime  $p', q', p' \neq q'$ . The RSA problem is hard if, for every efficient algorithm  $A$ , the probability:

$$\Pr[(n, e) \leftarrow \text{RSA-Gen}(1^\kappa), z \leftarrow \mathbb{Z}_n^* : A(n, e, z) = y \text{ s.t. } y^e = z \pmod{n}]$$

is a negligible function of  $\kappa$ .

### 3.2 Tools

In this section, we consider signature of knowledge of a discrete logarithm and equality of two discrete logarithms in a cyclic group  $\mathbb{G} = \langle g \rangle$ . In particular, we consider  $\mathbb{G}$  where either the order of  $\mathbb{G}$  is known or the order of  $\mathbb{G}$  is unknown but its bit-length  $l$  is publicly known. Fujisaki and Okamoto [FO97] show that (under the strong RSA assumption) standard proofs of knowledge that work in a group of known order are also proofs of knowledge in this setting. We define discrete logarithm of  $y \in \mathbb{G}$  with respect to base  $g$  as any integer  $x \in \mathbb{Z}$  such that  $y = g^x$  in  $\mathbb{G}$ . We assume a security parameter  $\epsilon > 1$ .

**Definition 3.** (*ZK of DL over a known order group*) Let  $y, g \in \mathbb{G}$  of order  $q$ . A pair  $(c, s) \in \{0, 1\}^\kappa \times \mathbb{Z}_q$  verifying  $c = H(y||g||g^s y^c||m)$  is a signature of knowledge of the discrete logarithm of  $y = g^x$  w.r.t. base  $g$ , on message  $m \in \{0, 1\}^*$ .

**Definition 4.** (*ZK of DL over an unknown order group*) Let  $y, g \in \mathbb{G}$  where the group order is unknown, but its bit-length is known as  $l$  bits. A pair  $(c, s) \in \{0, 1\}^\kappa \times \pm\{0, 1\}^{\epsilon(l+\kappa)+1}$  verifying  $c = H(y||g||g^s y^c||m)$  is a signature of knowledge of the discrete logarithm of  $y = g^x$  w.r.t. base  $g$ , on message  $m \in \{0, 1\}^*$ .

The player in possession of the secret  $x = \log_g y$  can compute the signature by choosing a random  $t \in \mathbb{Z}_q$  (or  $\pm\{0, 1\}^{\epsilon(l+\kappa)}$ ) and then computing  $c$  and  $s$  as:  $c = H(y||g||g^t||m)$  and  $s = t - cx$  in  $\mathbb{Z}_q$  (or in  $\mathbb{Z}$ ).

**Definition 5.** (*ZK of EDL over a known order group*) Let  $y_1, y_2, g, h \in \mathbb{G}$  of order  $q$ . A pair  $(c, s) \in \{0, 1\}^\kappa \times \mathbb{Z}_q$  verifying  $c = H(y_1||y_2||g||h||g^s y_1^c||h^s y_2^c||m)$  is a signature of knowledge of the discrete logarithm of both  $y_1 = g^x$  w.r.t. base  $g$  and  $y_2 = h^x$  w.r.t. base  $h$ , on message  $m \in \{0, 1\}^*$ .

**Definition 6.** (*ZK of EDL over an unknown order group*) Let  $y_1, y_2, g, h \in \mathbb{G}$  where the group order is unknown, but its bit-length is known as  $l$  bits. A pair  $(c, s) \in \{0, 1\}^\kappa \times \pm\{0, 1\}^{\epsilon(l+\kappa)+1}$  verifying  $c = H(y_1||y_2||g||h||g^s y_1^c||h^s y_2^c||m)$  is a signature of knowledge of the discrete logarithm of both  $y_1 = g^x$  w.r.t. base  $g$  and  $y_2 = h^x$  w.r.t. base  $h$ , on message  $m \in \{0, 1\}^*$ .

The player in possession of the secret  $x = \log_g y_1 = \log_h y_2$  can compute the signature by choosing a random  $t \in \mathbb{Z}_q$  (or  $\pm\{0, 1\}^{\epsilon(l+\kappa)}$ ) and then computing  $c$  and  $s$  as:  $c = H(y_1||y_2||g||h||g^t||h^t||m)$  and  $s = t - cx$  in  $\mathbb{Z}_q$  (or in  $\mathbb{Z}$ ).

### 3.3 Security Model

We assume a malicious adversary that behaves arbitrarily. Informally, a protocol is secure in this model if no adversary interacting in the real protocol (where no TTP exists) can learn any more from a real execution than from an execution that takes place in the ideal world. In other words, for any adversary that successfully attacks a real protocol, there exists a simulator that successfully attacks the same protocol in the ideal world.

We now define ideal functionalities of PSI and APSI. In particular, in contrast to PSI, APSI employs an (off-line) CA with algorithms (**KGen**, **Sign**, **Ver**). The CA generates a key-pair  $(sk, pk) \leftarrow \text{KGen}$ , publishes its public key  $pk$ , and, on client input  $c_i$ , it issues a signature  $\sigma_i = \text{Sign}(sk, c_i)$  s.t.  $\text{Ver}(pk, \sigma_i, c_i) = 1$ .



**Definition 7.** The ideal functionality  $\mathcal{F}_{\text{APSI}}$  of an APSI protocol betw. server  $S$  on input  $\mathcal{S} = \{s_1, \dots, s_w\}$  and client  $C$  on input  $\mathcal{C} = \{(c_1, \sigma_1), \dots, (c_v, \sigma_v)\}$  is defined as:

$$\mathcal{F}_{\text{APSI}} : (\mathcal{S}, \mathcal{C}) \rightarrow (\perp, \mathcal{S} \cap \{c_i \mid c_i \in \mathcal{C} \wedge \mathbf{Ver}(pk, \sigma_i, c_i) = 1\})$$

where  $w, v$  are the public input to  $\mathcal{F}_{\text{APSI}}$ .

**Definition 8.** The ideal functionality  $\mathcal{F}_{\text{PSI}}$  of a PSI between server  $S$  on input  $\mathcal{S} = \{s_1, \dots, s_w\}$  and client  $C$  on input  $\mathcal{C} = \{c_1, \dots, c_v\}$  is defined as follows:

$$\mathcal{F}_{\text{PSI}} : (\mathcal{S}, \mathcal{C}) \rightarrow (\perp, \mathcal{S} \cap \mathcal{C})$$

where  $w, v$  are the public input to  $\mathcal{F}_{\text{PSI}}$ .

## 4 APSI Protocol

We now present our protocol for secure computation of authorized set intersection. We start from the APSI protocol of [DT10] (reviewed in Section 2.2), secure in the semi-honest model. We describe a modified version that securely implements the  $\mathcal{F}_{\text{APSI}}$  functionality in the *malicious* model, in ROM, under the RSA and DDH assumptions.

The CA (trusted third party that authorizes client input) is realized with the following algorithms:

- **KGen:** On input of security parameter  $\kappa$ , this algorithm generates safe RSA modulus  $N = pq$  where  $p = 2p' + 1$ ,  $q = 2q' + 1$  and picks a random element  $g, g'$  s.t.  $\langle -1 \rangle \times \langle g \rangle \equiv \langle -1 \rangle \times \langle g' \rangle \equiv \mathbb{Z}_N^*$ . RSA exponents  $(e, d)$  are chosen in the standard way:  $e$  is a small prime and  $d = e^{-1} \bmod \phi(N)$ . The algorithm also fixes hash function  $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$  and  $H_2 : \mathbb{Z}_N^* \times \mathbb{Z}_N^* \times \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ . The secret key is  $(p, q, d)$  and the public parameters are:  $N, e, g, g', H_1(), H_2()$ .
- **Sign:** On input of  $c_i$ , this algorithm issues an authorization  $\sigma_i = H_1(c_i)^d \bmod N$ .
- **Ver:** On input of  $(\sigma_i, c_i)$ , this algorithm verifies whether  $\sigma_i^e = H_1(c_i) \bmod N$ .

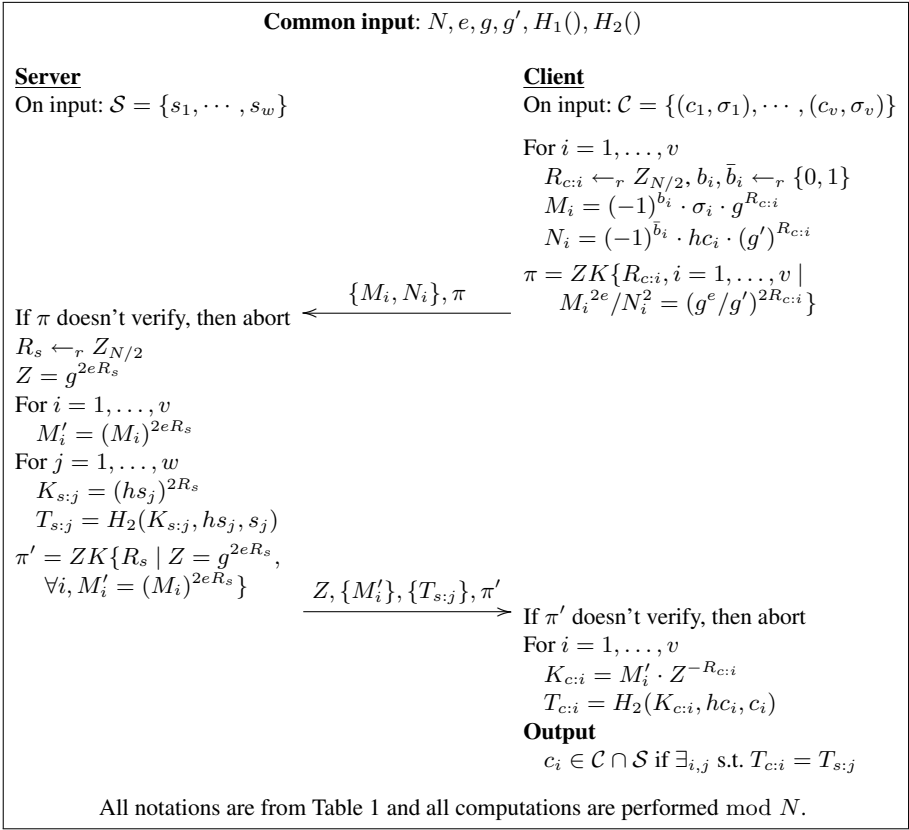
The resulting protocol is presented in **Figure 1**.

**Theorem 1.** If RSA and DDH problems are hard, and  $\pi, \pi'$  are zero-knowledge proofs, then the protocol in Figure 1 is a secure computation of  $\mathcal{F}_{\text{APSI}}$  in ROM.

**Proof.** [Construction of an ideal world  $\text{SIM}_s$  from a malicious real-world server  $S^*$ ]

The simulator  $\text{SIM}_s$  is built as follows:

- **Setup:**  $\text{SIM}_s$  executes **KGen** and publishes public parameters  $N, e, g, g'$ .
- **Hash queries to  $H_1$  and  $H_2$ :**  $\text{SIM}_s$  constructs two tables  $T_1 = (q, h_q)$  and  $T_2 = ((k, h'_q, q'), t)$  to answer, respectively, the  $H_1$  and  $H_2$  queries. Specifically:
  - On query  $q$  to  $H_1$ ,  $\text{SIM}_s$  checks if  $\exists (q, h_q) \in T_1$ : If so, it returns  $h_q$ , otherwise it responds  $h_q \leftarrow_r \mathbb{Z}_N^*$ , and stores  $(q, h_q)$  to  $T_1$ .
  - On query  $(k, h'_q, q')$  to  $H_2$ ,  $\text{SIM}_s$  checks if  $\exists ((k, h'_q, q'), t) \in T_2$ : If so, it returns  $t$ , otherwise it responds  $t \leftarrow_r \{0, 1\}^\kappa$  to  $H_2$ , and stores  $((k, h'_q, q'), t)$  to  $T_2$ .



**Fig. 1.** Our APSI Protocol with linear complexity secure against malicious adversaries

– **Simulation of the real-world client  $C$  and the ideal-world server  $\bar{S}$ :**

1.  $\text{SIM}_s$  picks  $M'_i \leftarrow_r \mathbb{Z}_N^*$ ,  $N'_i \leftarrow_r \mathbb{Z}_N^*$  and computes  $M_i = (M'_i)^2$ ,  $N_i = (N'_i)^2$  for each  $i = 1, \dots, v$ .
2.  $\text{SIM}_s$  sends  $\{M_i, N_i\}_{i=1, \dots, v}$  and simulates the proof  $\pi$ .
3. After getting  $(Z, \{M'_i\}_{i=1, \dots, v}, \{T_{s:j}\}_{j=1, \dots, w})$ , and interacting with  $S^*$  as verifier in the proof  $\pi'$ , if the proof  $\pi'$  verifies,  $\text{SIM}_s$  runs the extractor algorithm for  $R_s$ . Otherwise, it aborts.
  - (a) For each  $T_{s:j}$ ,  $\text{SIM}_s$  checks if  $\exists(q, h_q) \in T_1$  and  $\exists((k, h'_q, q'), t) \in T_2$ , s.t.  $q = q'$ ,  $h_q = h'_q$ ,  $k = (h_q)^{2R_s}$  and  $t = T_{s:j}$ . If so, add  $q$  to  $\mathcal{S}$ ; otherwise, add a dummy item into  $\mathcal{S}$ .
  - (b) Then  $\text{SIM}_s$  plays the role of the ideal-world server, which uses  $\mathcal{S}$  to respond to ideal client  $\bar{C}$ 's queries.

Since the distribution of  $\{M_i, N_i\}_{i=1, \dots, v}$  sent by  $\text{SIM}_s$  is identical to the distribution produced by the real client  $C$  and the  $\pi$  proof system is zero-knowledge,  $S^*$ 's views when interacting with the real client  $C$  and with the simulator  $\text{SIM}_s$  are indistinguishable.

[Output of (honest) real client  $C$  interacting with  $S^*$ ]

Now we consider the output of the honest real client  $C$  interacting with  $S^*$ . By soundness of proof  $\pi'$ , message  $Z$  and  $M'_i$  sent by  $S^*$  is  $Z = g^{eR_s}$  and  $M'_i = (M_i)^{eR_s}$  for  $i = 1, \dots, v$ . Then,  $C$ 's final output is a set containing all  $c_i$ 's, such that  $H_2(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i) \in \{T_{s:j}\}$ . In other words, for each  $c_i$ ,  $C$  outputs  $c_i$  if  $\exists j$  s.t.  $H_2(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i) = T_{s:j}$ . Since  $H_2$  is a random oracle, there are two possibilities:

1.  $S^*$  computes  $T_{s:j}$  from  $H_2((hs_j)^{2R_s}, hs_j, s_j)$  for  $s_j = c_i$ . Since  $\text{SIM}_s$  described above extracts  $s_j = c_i$  and adds  $s_j$  in  $\mathcal{S}$ , the ideal world  $\overline{C}$  also output  $c_i$  on its input  $c_i$ .
2.  $S^*$  did not query  $H_2$  on  $(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i)$  but  $H_2(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i)$  happens to be equal to  $T_{s:j}$ . This event occurs with negligible probability bounded by  $v \cdot w \cdot 2^{-\kappa}$ .

Therefore, with probability  $1 - v \cdot w \cdot 2^{-\kappa}$ , the real-world client  $C$  interacting with  $S^*$  and the ideal-world client  $\overline{C}$  interacting with  $\text{SIM}_s$  yield identical outputs.

[Construction of an ideal world  $\text{SIM}_e$  from a malicious real-world client  $C^*$ ]

The simulator  $\text{SIM}_e$  is formed as follows:

- **Setup and hash queries to  $H_1$  and  $H_2$ :** Same as Setup and  $H_1$  and  $H_2$  responses described above in construction of  $\text{SIM}_s$ .
- **Authorization queries:** On input  $m$ ,  $\text{SIM}_e$  responds with  $(m, \sigma)$  where  $\sigma = (H_1(m))^d$  and stores  $(m, \sigma)$  to table  $T_3$ .
- **Simulation of real-world server  $S$  and ideal-world client  $\overline{C}$ :**

1. After getting  $\{M_i, N_i\}_{i=1, \dots, v}$ , and interacting with  $C^*$  as verifier in the proof  $\pi$ ,  $\text{SIM}_e$  checks if proof  $\pi$  verifies. If not, it aborts. Otherwise, it runs the extractor algorithm for  $\{R_{c:i}\}$  and computes  $\pm(hc_i, \sigma_i)$  s.t.  $hc_i = \sigma^e$ .
2. For each  $\pm(hc_i, \sigma_i)$ :
  - If  $\nexists (q, h_q) \in T_1$  s.t.  $h_q = \pm hc_i$  then add a dummy item  $(\delta, \sigma_\delta)$  to  $\mathcal{C}$  where  $\delta$  and  $\sigma_\delta$  are randomly selected from the respective domain.
  - If  $\exists (q, h_q) \in T_1$  s.t.  $h_q = \pm hc_i$ , but  $\nexists (m, \sigma) \in T_3$  s.t.  $\sigma = \pm \sigma_i$  then output fail<sub>1</sub> and abort.
  - If  $\exists (q, h_q) \in T_1$  s.t.  $h_q = \pm hc_i$  and  $\exists (m, \sigma) \in T_3$  s.t.  $\sigma = \pm \sigma_i$ , then add  $(q, \pm \sigma)$  to the set  $\mathcal{C}$ .
3.  $\text{SIM}_e$  plays the role of the client in the ideal-world. On input  $\mathcal{C} = \{(c_1, \sigma_1), \dots, (c_v, \sigma_v)\}$ ,  $\text{SIM}_e$  interacts with the ideal-world server  $\overline{S}$  through the TTP.
4. On getting intersection  $L = \{c'_1, \dots, c'_{|L|}\}$ , with  $|L| \leq v$  from the ideal-world interaction,  $\text{SIM}_e$  forms  $\mathcal{S} = \Pi(c'_1, \dots, c'_{|L|}, \delta'_1, \dots, \delta'_{w-|L|+1})$ , where  $\delta'$ 's are dummy items and  $\Pi$  is a permutation function.
5.  $\text{SIM}_e$  picks  $R_s \leftarrow_r \mathbb{Z}_{N/2}$ , and computes  $Z = g^{2eR_s}$  and  $M'_i = (M_i)^{2eR_s}$  for  $i = 1, \dots, v$ .
6. For each  $s_j \in \mathcal{S}$ :
  - If  $s_j \in L$ , compute  $T_{s:j} = H_2((hs_j)^{2R_s}, hs_j, s_j)$ .
  - If  $s_j \notin L$ , compute  $T_{s:j} \leftarrow_r \{0, 1\}^\kappa$ .
7.  $\text{SIM}_e$  returns  $Z, \{M'_i\}_{i=1, \dots, v}, \{T_{s:j}\}_{j=1, \dots, w}$  to  $C^*$  and simulates the proof  $\pi'$ .

**Claim 1.** If event  $\text{fail}_1$  occurs with non-negligible probability, then  $C^*$  can be used to break the RSA assumption.

We describe the reduction algorithm using a modified simulator algorithm called  $\mathcal{Ch}_1$  that takes an RSA challenge  $(N', e', z)$  as an input and tries to output  $z^{(e')^{-1}}$ .  $\mathcal{Ch}_1$  follows the  $\text{SIM}_c$  as described above, except:

- **Setup:** On input  $(N', e', z)$ ,  $\mathcal{Ch}_1$  sets  $N = N'$ ,  $e = e'$  and picks generator  $g$ ,  $g' \leftarrow_r \mathbb{Z}_N^*$ . (Note that random  $g$  in  $\mathbb{Z}_N^*$  matches that chosen by a real key generation with probability about  $1/2$ .)
- **Authorization queries:** On input  $m$ ,  $\mathcal{Ch}_1$  responds with  $(m, \sigma)$  with  $\sigma \leftarrow_r \mathbb{Z}_N^*$ , assign  $H_1(m) = \sigma^e$ , and records  $(m, \sigma)$  to  $T_3$ .
- **Hash queries to  $H_1$ :** On query  $H_1$  on  $q$ , if  $\nexists(q, h_q) \in T_1$  then  $\mathcal{Ch}_1$  responds  $h_q = z(r_q)^e$  where  $r_q \leftarrow_r \mathbb{Z}_N$ , and stores  $(q, r_q, h_q)$  to  $T_1$ . (Since  $r_q$  is uniformly distributed in  $\mathbb{Z}_N$ , the distribution of  $h_q$  is also uniformly distributed in  $\mathbb{Z}_N$ .)

Assume that  $\text{fail}_1$  occurs on  $(hc_i, \sigma_i)$ . Then,  $\mathcal{Ch}_1$  extracts entry  $(q, r_q, h_q) \in T_1$  s.t.  $h_q = hc_i$  and outputs  $\sigma_i/r_q$ , which breaks the RSA assumption.

Now unless the  $\text{fail}_1$  event occurs, the views interacting with the  $\text{SIM}_c$  and with the real protocol are different only in the computation of  $T_{s;j}$  for  $s_j \in \mathcal{S}$  but  $s_j \notin L$ . Let  $\text{fail}_2$  be the event that  $C^*$  queries  $H_2$  on  $((hs_j)^{2R_s}, hs_j, s_j)$  for  $s_j \in \mathcal{S}$  and  $s_j \notin L$ .

**Claim 2.** If event  $\text{fail}_2$  occurs with non-negligible probability, then  $C^*$  can be used to break the DDH assumption.

We describe reduction algorithm  $\mathcal{Ch}_2$  that takes a DDH challenge  $(N', f, \alpha = f^a \pmod{N'}, \beta = f^b \pmod{N'}, \gamma)$  as input and outputs the DDH answer using  $C^*$ .  $\mathcal{Ch}_2$  follows the  $\text{SIM}_c$  algorithm as we describe above, except that:

- **Setup:** On input  $(N', f, \alpha, \beta, \gamma)$ ,  $\mathcal{Ch}_2$  sets  $N = N'$ ,  $g = f$  and picks generator  $g' \leftarrow_r \mathbb{Z}_N^*$  and odd  $e \leftarrow_r \mathbb{Z}_N$ .
- **Authorization queries:** Same as in  $\mathcal{Ch}_1$  simulation.
- **Hash queries to  $H_1$ :** On query  $q$  to  $H_1$ , if  $\nexists(q, h_q) \in T_1$  then  $\mathcal{Ch}_2$  responds with  $h_q = \beta g^{r_q}$  where  $r_q \leftarrow_r \mathbb{Z}_{N/2}$ , and records  $(q, r_q, h_q)$  to  $T_1$ . (Since  $r_q$  is random  $\mathbb{Z}_{N/2}$ , the distribution of  $h_q$  is computationally indistinguishable from the uniform distribution of  $\mathbb{Z}_N^*$ .)
- **In computation for  $Z, \{M_i\}, \{T_{s;j}\}$ :**
  - $\mathcal{Ch}_2$  sets  $Z = A^{2e}$  and computes  $M'_i = \gamma^2(\alpha)^{2r_q + 2eR_c:i}$  for  $i = 1, \dots, v$  (instead of picking  $R_s$  and computing  $Z = g^{2eR_s}$  and  $M'_i = (M_i)^{2eR_s}$ ).
  - For each  $s_j \in \mathcal{S}$ , if  $s_j \in L$ ,  $\mathcal{Ch}_2$  computes  $T_{s;j} = H_2(\gamma^2(\alpha)^{2r_q}, hs_j, s_j)$ .

Given  $\alpha = g^a (= g^{R_s})$  and  $\beta = g^b$ , we replace  $g^{ab}$  by  $\gamma$  in the above simulation of  $M_i$  and  $T_{s;j}$ . Thus,  $C^*$ 's views when interacting with the real server  $S$  and with the simulator  $\mathcal{Ch}_2$  are indistinguishable under that DDH assumption. Assume that  $\text{fail}_2$  occurs, i.e.,  $C^*$  makes a query to  $H_2$  on  $((hs_j)^{2R_s}, hs_j, s_j)$  for  $s_j \in \mathcal{S}$  but  $s_j \notin L$ .  $\mathcal{Ch}_2$  checks if  $\exists(q, r_q, h_q) \in T_1$  and  $\exists((k, h'_q, q'), t) \in T_2$  s.t.  $q = q'$ ,  $h_q = h'_q$ ,  $k = \gamma^2(\alpha)^{2r_q}$  for each  $q \in \mathcal{S}$  but  $q \notin L$ . If so,  $\mathcal{Ch}_2$  outputs True. Otherwise,  $\mathcal{Ch}_2$  outputs False. Thus, the DDH assumption is broken.

Therefore, since  $\text{fail}_1$  and  $\text{fail}_2$  events occur with negligible probability,  $C^*$ 's view in the protocol with the real-world server  $S$  and in the interaction with  $\text{SIM}_c$  is negligible.

[The output of honest real server  $S$  interacting with  $C^*$ ]

Finally, the real-world  $S$  interacting with  $C^*$  in the real protocol outputs  $\perp$  and the ideal-world  $\bar{S}$  interacting with  $\text{SIM}_c$  gets  $\perp$ . This ends proof of Theorem 1.

Our APSI protocol differs from the one in [DT10] in the following:

- We modify inputs to the protocol and add efficient zero-knowledge proofs to prevent client and server from deviating from the protocol and to enable extraction of inputs.
- We multiply client inputs by  $-1$  to a random bit to: (1) ensure that they are uniformly distributed in  $QR_N$ , and (2) simplify reduction to the RSA problem.
- We do not use “accumulated” values, such as  $PCH_i^*$ , as they are not needed either for protocol security or for input extraction during simulation.

## 5 PSI Protocol

This section presents our protocol for secure computation of set intersection. It is a modified version of the PSI protocol of [DT10] (reviewed in Section 2.1), secure in the semi-honest model under the One-More-Gap-DH assumption (in ROM). We amend it to obtain a protocol that securely implements  $\mathcal{F}_{\text{PSI}}$  in the *malicious* model under the DDH assumptions (in ROM). We assume that  $\text{KGen}$  generates  $p, q, g, g', g''$  where  $p$  and  $q$  are primes, such that  $q|p-1$  and  $g, g', g''$  are generators of  $\mathbb{Z}_q^*$ .

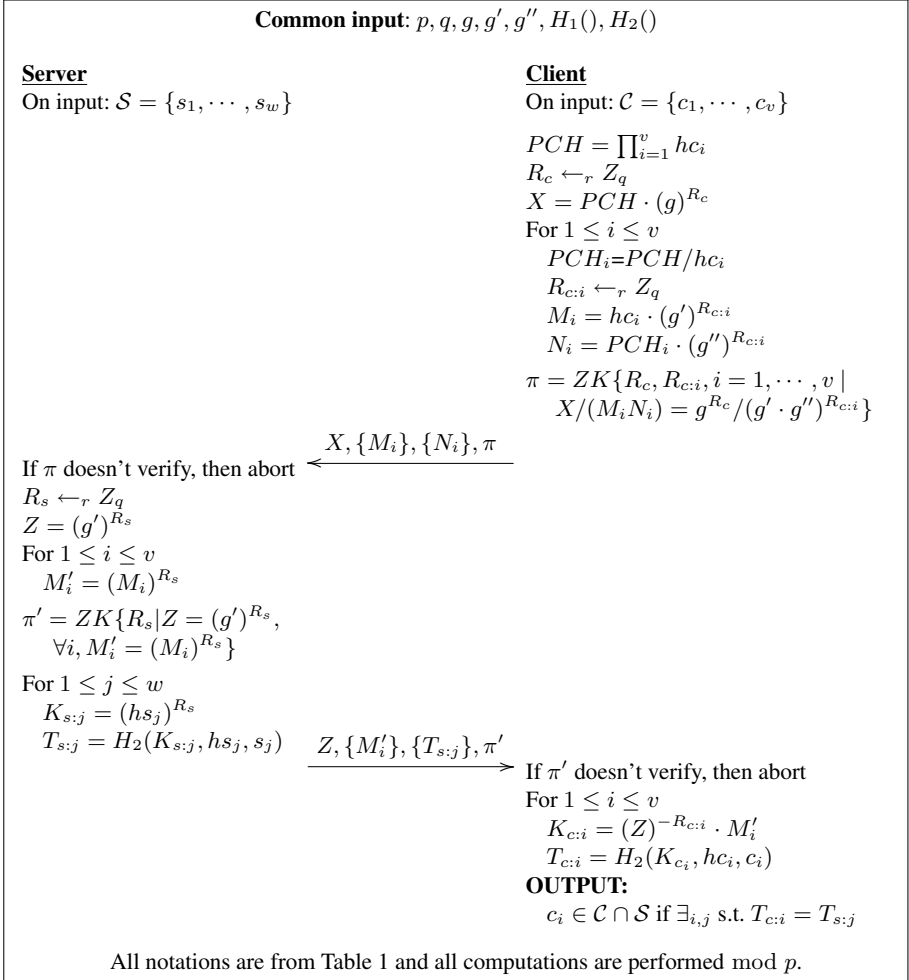
The resulting protocol is presented in **Figure 2**.

**Theorem 2.** *If the DDH problem is hard and  $\pi, \pi'$  are zero-knowledge proofs, the protocol in Figure 2 is a secure computation of  $\mathcal{F}_{\text{PSI}}$  in ROM.*

**Proof.** [Construction of an ideal world  $\text{SIM}_s$  from malicious real-world server  $S^*$ ]

Simulator  $\text{SIM}_s$  is built as follows:

- **Setup:**  $\text{SIM}_s$  executes  $\text{KGen}$  and publishes public parameters  $p, q, g, g', g''$ .
- **Queries  $H_1$  and  $H_2$ :**  $\text{SIM}_s$  creates two tables  $T_1 = (q, h_q)$  and  $T_2 = ((k, h'_q, q'), t)$  to answer, respectively,  $H_1$  and  $H_2$  queries. Specifically,
  - On query  $q$  to  $H_1$ ,  $\text{SIM}_s$  checks if  $\exists (q, h_q) \in T_1$ : If so, it returns  $h_q$ , otherwise it responds  $h_q \leftarrow_r \mathbb{Z}_p^*$ , and stores  $(q, h_q)$  to  $T_1$ .
  - On query  $(k, h'_q, q')$  to  $H_2$ ,  $\text{SIM}_s$  checks if  $\exists ((k, h'_q, q'), t) \in T_2$ : If so, it returns  $t$ , otherwise it responds  $t \leftarrow_r \{0, 1\}^\kappa$  to  $H_2$ , and stores  $((k, h'_q, q'), t)$  to  $T_2$ .
- **Simulation of real-world client  $C$  and ideal-world server  $\bar{S}$ :**
  1.  $\text{SIM}_s$  picks  $X \leftarrow_r \mathbb{Z}_p^*$  and  $\{M_i, N_i \mid M_i \leftarrow_r \mathbb{Z}_p^*, N_i \leftarrow_r \mathbb{Z}_p^*\}$  (for  $i = 1, \dots, v$ ).
  2.  $\text{SIM}_s$  sends  $X, \{M_i, N_i\}_{i=1, \dots, v}$  and simulates proof  $\pi$ .
  3. After getting  $(Z, \{M'_i\}_{i=1, \dots, v}, \{T_{s:j}\}_{j=1, \dots, w})$ , and interacting with  $S^*$  as verifier in proof  $\pi'$ , if  $\pi'$  verifies,  $\text{SIM}_s$  runs the extractor algorithm for  $R_s$ . Otherwise, it aborts.
    - (a) For each  $T_{s:j}$ ,  $\text{SIM}_s$  checks if  $\exists (q, h_q) \in T_1$  and  $\exists ((k, h'_q, q'), t) \in T_2$ , s.t.  $q = q', h_q = h'_q, k = (h_q)^{R_s}$  and  $t = T_{s:j}$ . If so, add  $q$  to  $\mathcal{S}$ ; otherwise, add a dummy item into  $\mathcal{S}$ .
    - (b) Then  $\text{SIM}_s$  plays the role of the ideal-world server, which uses  $\mathcal{S}$  to respond to ideal client  $\bar{C}$ 's queries.



**Fig. 2.** Our PSI Protocol with linear complexity secure against malicious adversaries

Since the distribution of  $X, \{M_i, N_i\}_{i=1, \dots, v}$  sent by  $\text{SIM}_s$  is identical to the distribution produced by the real client  $C$  and the  $\pi$  proof system is zero-knowledge,  $S^*$ 's views when interacting with real client  $C$  and with simulator  $\text{SIM}_s$  are indistinguishable.

[Output of the honest real client  $C$  interacting with  $S^*$ ]

Now we consider output of honest real client  $C$  interacting with  $S^*$ . By soundness of  $\pi'$ , message  $Z$  and  $M'_i$  sent by  $S^*$  is  $Z = (g')^{R_s}$  and  $M'_i = (M_i)^{R_s}$  for  $i = 1, \dots, v$ . Then  $C$ 's final output is a set containing all  $c_i$ 's such that  $H_2(M'_i Z^{-R_{c:i}}, hc_i, c_i) \in \{T_{s:j}\}$ . In other words, for each  $c_i$ ,  $C$  outputs  $c_i$  if  $\exists j$  s.t.  $H_2(M'_i Z^{-R_{c:i}}, hc_i, c_i) = T_{s:j}$ . Since  $H_2$  is a random oracle, there are two possibilities:

1.  $S^*$  computes  $T_{s:j}$  from  $H_2((hs_j)^{2R_s}, hs_j, s_j)$  for  $s_j = c_i$ . Since  $\text{SIM}_s$  described above extracts  $s_j = c_i$  and adds  $s_j$  in  $\mathcal{S}$ , ideal world  $\overline{C}$  also output  $c_i$  on its input  $c_i$ .
2.  $S^*$  did not query  $H_2$  on  $(M'_i Z^{-R_{c:i}}, hc_i, c_i)$  but  $H_2(M'_i Z^{-R_{c:i}}, hc_i, c_i)$  happens to be equal to  $T_{s:j}$ . This event occurs with negligible probability bounded by  $v \cdot w \cdot 2^{-\kappa}$ .

Therefore, with probability  $1 - v \cdot w \cdot 2^{-\kappa}$ , real-world client  $C$  interacting with  $S^*$  and ideal-world client  $\overline{C}$  interacting with  $\text{SIM}_s$  produce identical output.

[Construction of ideal world  $\text{SIM}_c$  from malicious real-world client  $C^*$ ]

Simulator  $\text{SIM}_c$  is formed as follows:

- **Setup and hash queries to  $H_1$  and  $H_2$ :** Same as Setup and  $H_1$  and  $H_2$  responses described above in construction of  $\text{SIM}_s$ .
- **Simulation of real-world server  $S$  and ideal-world client  $\overline{C}$ :**
  1. After getting  $(X, \{M_i\}, \{N_i\})$ , and interacting with  $C^*$  as verifier in proof  $\pi$ ,  $\text{SIM}_c$  checks if  $\pi$  verifies. If not, it aborts. Otherwise, it runs the extractor algorithm for  $R_c, \{R_{c:i}\}$  and computes  $hc_1, \dots, hc_v$ .
  2. For each  $hc_i$ , if  $\exists(q, h_q) \in T_1$  s.t.  $h_q = hc_i$  then add  $q$  to the set  $\mathcal{C}$ . Otherwise, add a dummy item to  $\mathcal{C}$ .
  3.  $\text{SIM}_c$  plays the role of the client in the ideal-world. On input  $\mathcal{C} = \{c_1, \dots, c_v\}$ ,  $\text{SIM}_c$  interacts with the ideal-world server  $\overline{S}$  through the TTP.
  4. On getting intersection  $L = \{c'_1, \dots, c'_{|L|}\}$ , with  $|L| \leq v$  from the ideal-world interaction,  $\text{SIM}_c$  forms  $\mathcal{S} = \Pi(c'_1, \dots, c'_{|L|}, \delta'_1, \dots, \delta'_{w-|L|+1})$ , where  $\delta'_i$ 's are dummy items and  $\Pi$  is a permutation function.
  5.  $\text{SIM}_c$  picks  $R_s \leftarrow_r \mathbb{Z}_q$ , and computes  $Z = g^{R_s}$  and  $M'_i = (M_i)^{R_s}$  for  $i=1, \dots, v$ .
  6. For each  $s_j \in \mathcal{S}$ :
    - If  $s_j \in L$ , compute  $T_{s:j} = H_2((hs_j)^{R_s}, hs_j, s_j)$ .
    - If  $s_j \notin L$ , compute  $T_{s:j} \leftarrow_r \{0, 1\}^\kappa$ .
  7.  $\text{SIM}_c$  returns  $Z, \{M'_i\}, \{T_{s:j}\}$  to  $C^*$  and simulates proof  $\pi'$ .

Let fail be the event that  $C^*$  queries  $H_2$  on  $((hs_j)^{R_s}, hs_j, s_j)$  for  $s_j \in \mathcal{S}$  and  $s_j \notin L$ . Similar to the argument in the proof of Theorem 1, if fail event does not occur, since the  $\pi'$  is zero-knowledge, we argue that  $C^*$ 's views in the real game with real-world server  $S$  and in the interaction with simulator  $\text{SIM}_c$  constructed above are indistinguishable.

**Claim.** If event fail occurs with non-negligible probability, then  $C^*$  can be used to break the DDH assumption.

We describe the reduction algorithm called  $\mathcal{Ch}$  that takes a DDH problem  $(p', q', f, \alpha = f^a \pmod{p'}, \beta = f^b \pmod{p'}, \gamma)$  as an input and tries to output the answer using  $C^*$ .  $\mathcal{Ch}$  follows the  $\text{SIM}_c$  algorithm as we describe above, except that:

- **Setup:** On input  $(p', q', f, \alpha, \beta, \gamma)$ ,  $\mathcal{Ch}_2$  sets  $p = p', q = q', g' = f$  and picks generator  $g, g'' \leftarrow_r \mathbb{Z}_q^*$ .
- **Hash queries to  $H_1$ :** On query  $q$  to  $H_1$ , if  $\nexists(q, h_q) \in T_1$  then  $\mathcal{Ch}_2$  responds with  $h_q = \beta(g')^{r_q}$  where  $r_q \leftarrow_r \mathbb{Z}_q$ , and records  $(q, r_q, h_q)$  to  $T_1$ .

– **In computation for  $Z, \{M'_i\}, \{T_{s:j}\}$ :**

- $Ch_2$  sets  $Z = A$  and computes  $M'_i = C(A)^{r_q + R_{c:i}}$ .
- For each  $s_j \in \mathcal{S}$ , if  $s_j \in L$ ,  $Ch_2$  computes  $T_{s:j} = H_2(C(A)^{r_q}, h_{s_j}, s_j)$ .

Using an argument similar to that in the proof of Theorem 1,  $C^*$ 's views, when interacting with real server  $S$  and with simulator  $Ch_2$ , are indistinguishable under the DDH assumption. Assume that fail occurs, i.e.,  $C^*$  makes a query to  $H_2$  on  $((hs_j)^{R_s}, h_{s_j}, s_j)$  for  $s_j \in \mathcal{S}$  but  $s_j \notin L$ .  $Ch$  checks if  $\exists(q, r_q, h_q) \in T_1$  and  $\exists((k, h'_q, q'), t) \in T_2$  s.t.  $q = q'$ ,  $h_q = h'_q$ ,  $k = C(A)^{r_q}$  for each  $q \in \mathcal{S}$  and  $q \notin L$ . If so,  $Ch$  outputs True. Otherwise,  $Ch_2$  outputs False. Thus,  $Ch$  solves the DDH problem.

Since fail occurs with negligible probability,  $C^*$ 's view in the protocol with the real-world server  $S$  and in interaction with  $SIM_c$  is negligible.

[Output of honest real server  $S$  interacting with  $C^*$ ]

Finally, real-world  $S$  interacting with  $C^*$  in the real protocol outputs  $\perp$  and ideal-world  $\bar{S}$  interacting with  $SIM_c$  gets  $\perp$ .

## 6 Protocols Efficiency

In this section, we analyze the efficiency of our protocols and compare them to prior results. We summarize different features and estimated asymptotic complexities of prior work on Authorized Private Set Intersection and Private Set Intersection, respectively, as well as those of our protocols, in Table 2 and 3. Recall that we use  $w$  and  $v$  to denote the number of elements in the server and client input sets, respectively. Also, we specify whether they can support the extension for *data transfer* – a PSI variant introduced in [DT10] and discussed in details in the extended version of the paper [DKT10].

Note that our APSI protocol (in Figure 1) is, to the best of our knowledge, the only such construct, secure in the malicious model, with *linear* communication and computational complexity.

**Comparing our PSI [Fig. 2] to [JL09].** Our PSI protocol achieves the same (linear) asymptotic overhead as in prior work [JL09], although, in ROM. However, the underlying cryptographic operations of [JL09], hidden in the big  $O()$  notation, are much more expensive than those in Figure 2, as we discuss below.

First, recall that, on average, each  $q$ -bit multi-exponentiation mod  $p$  involves  $(1.5 \cdot |q|)$  multiplications of  $p$ -bit numbers. Whereas, each  $q$ -bit fixed-based exponentiation mod  $p$  incurs only  $(0.5 \cdot |q|)$  multiplications. From now on, we denote with  $m$  a modular multiplication of  $p$ -bit numbers, and we assume  $|p| = 1024$ .

Observe that the PSI protocol in Figure 2, in the malicious model, incurs the total cost of  $(240w + 960v)m$ . Due to space limitation, we refer to our extended version [DKT10] for all the details of our estimation.

In order to count the number of operations of [JL09], we use the optimized OPRF construction due to [BCC<sup>+</sup>09] and we use standard non-interactive ZK in ROM. We select set items to be drawn from a 40-bit domain.<sup>1</sup> The total cost of [JL09], in the malicious model, amounts to  $(80w + 81320v)m$ . (Again, refer to [DKT10] for the details).

<sup>1</sup> Recall that, in the proof of [JL09], there seems to be a trade-off between the input domain size and security loss, as the simulator needs to exhaustively search over the input domain.



**Table 2.** Comparison of Authorized Private Set Intersection protocols

Protocol	Standard Model	Malicious Model	Assumption	Communication Complexity	Server	Client	Data Transfer
[CKRS09]	✓	✓	BDH	$O(w)$	$O(w)$ enc's of [BW06]	$O(w \cdot v)$ dec's of [BW06]	✓
[CZ09]	✓	✓	Strong RSA	$O(w + v)$	$O(w \cdot v)$ exps	$O(w \cdot v)$ exps	✗
Fig.2 of [DT10]	✗	✗	RSA	$O(w + v)$	$O(w + v)$ exps	$O(v)$ exps	✓
Our APSI	✗	✓	RSA	$O(w + v)$	$O(w + v)$ exps	$O(v)$ exps	✓

**Table 3.** Comparison of Private Set Intersection protocols

Protocol	Standard Model	Malicious Model	Assumption	Communication Complexity	Server	Client	Data Transfer
[FNP04]	✓	✗	Homom. Encr.	$O(w + v)$	$O(w \log \log v)$ exps	$O(w + v)$ exps	✓
[KS05]	✓	✓	Homom. Encr.	$O(w + v)$	$O(w \cdot v)$ exps	$O(w \cdot v)$ exps	✗
[JL09]	✓	✓	Decisional q-DH, CRS	$O(w + v)$	$O(w + v)$ exps	$O(v)$ exps	✓
[HN10]	✓	✓	DDH	$O(w + v)$	$O(w \log \log v)$ exps	$O(w + v)$ exps	✗
Fig.3 of [DT10]	✗	✗	One-More Gap-DH	$O(w + v)$	$O(w + v)$ exps	$O(v)$ exps	✓
Fig.4 of [DT10]	✗	✗	One-More RSA	$O(w + v)$	$O(w + v)$ exps	$O(v)$ mults	✓
Our PSI	✗	✓	DDH	$O(w + v)$	$O(w + v)$ exps	$O(v)$ exps	✓

Selecting, for instance,  $w = v$ , protocol in Figure 2 would require **as low as 1.5%** of the total modular multiplications incurred by [JL09] (even with the optimized OPRF construction and using non-interactive ZK in ROM). Only when  $w/v \gg 500$ , [JL09] incurs lower cost. Furthermore, recall that, although secure in the standard model, the PSI construct in [JL09], when compared to ours, has three major drawbacks: (1) The size of set items should be polynomial in the security parameter, whereas, in our protocol, items can be taken from  $\{0, 1\}^*$ , (2) It requires Decisional q-DH assumption and Common Reference String (CRS) model, where a safe RSA modulus must be generated by a mutually trusted party, and (3) It is not clear how to convert it into APSI.

We conclude that, though in ROM (as opposed to [JL09]), our PSI protocol significantly improves performance of prior PSI results, secure in the malicious model, while avoiding several restrictions.

## 7 Conclusion

In this paper, we presented PSI and APSI protocols secure in the malicious model under standard cryptographic assumptions, with linear communication and computational complexities. Proposed protocols offer better efficiency than prior work. In particular, our APSI protocol is the first technique to achieve linear computational complexity. Our efficiency claims are supported by detailed performance comparison.

**Acknowledgements.** This research was supported by the US Intelligence Advanced Research Projects Activity (IARPA) under grant number FA8750-09-2-0071 as well as by the Basic Science Research Program through the National Research Foundation of Korea, funded by the Ministry of Education, Science and Technology (2009-0070095) and by 'Developing Future Internet Network Model - Mathematical Approach' of the National Institute for Mathematical Sciences. Jihye Kim is the corresponding author for this paper. We also would like to thank Xiaomin Liu and Jae Hong Seo for their helpful comments.

## References

- [AL07] 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)
- [BCC<sup>+</sup>09] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
- [BDOP04] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
- [BF03] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. *SIAM Journal of Computing* 32(3), 586–615 (2003)
- [BNPS03] Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology* 16(3), 185–215 (2003)
- [BW06] Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
- [Cha83] Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO (1983)
- [CKRS09] Camenisch, J., Kohlweiss, M., Rial, A., Sheedy, C.: Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 196–214. Springer, Heidelberg (2009)
- [CS03] Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
- [CZ09] Camenisch, J., Zaverucha, G.: Private intersection of certified sets. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 108–127. Springer, Heidelberg (2009)
- [DJKT09] De Cristofaro, E., Jarecki, S., Kim, J., Tsudik, G.: Privacy-Preserving Policy-Based Information Transfer. In: Goldberg, I., Atallah, M.J. (eds.) Privacy Enhancing Technologies. LNCS, vol. 5672, pp. 164–183. Springer, Heidelberg (2009)
- [DKT10] De Cristofaro, E., Kim, J., Tsudik, G.: Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model, *Cryptology ePrint Archive* (2010), <http://eprint.iacr.org/2010/469>
- [DSMRY09] Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient Robust Private Set Intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009)

- [DT10] De Cristofaro, E., Tsudik, G.: Practical Private Set Intersection Protocols with Linear Complexity. In: Financial Cryptography and Data Security (2010)
- [FNP04] Freedman, M.J., 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)
- [FO97] Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
- [Gol04] Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge Univ. Press, Cambridge (2004)
- [HL08] 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)
- [HN10] Hazay, C., Nissim, K.: Efficient Set Operations in the Presence of Malicious Adversaries. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010)
- [JL09] 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)
- [KS05] Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
- [NP06] Naor, M., Pinkas, B.: Oblivious polynomial evaluation. *SIAM Journal on Computing* 35(5), 1254–1284 (2006)
- [Sha84] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
- [Yao82] Yao, A.C.: Protocols for secure computations. In: FOCS, pp. 160–164 (1982)