Compact and Malicious Private Set Intersection for Small Sets*

Mike Rosulek[1] and Ni Trieu[2]

[1]Oregon State University, rosulekm@eecs.oregonstate.edu
[2]Arizona State University, nitrieu@asu.edu

October 2, 2021

Abstract

We describe a protocol for two-party private set intersection (PSI) based on Diffie-Hellman key agreement. The protocol is proven secure against malicious parties, in the ideal permutation + random oracle model.

For small sets (500 items or fewer), our protocol requires the least time and communication of any known PSI protocol, even ones that are only *semi-honest* secure and ones that are not based on Diffie-Hellman. It is one of the few significant improvements to the 20-year old classical Diffie-Hellman PSI protocol of Huberman, Franklin, and Hogg (ACM Elec. Commerce 1999).

Our protocol is actually a generic transformation that constructs PSI from a class of key agreement protocols. This transformation is inspired by a technique of Cho, Dachman-Soled, and Jarecki (CT-RSA 2016), which we streamline and optimize in several important ways to achieve our superior efficiency.

1 Introduction

In a private set intersection (PSI) protocol, Alice provides an input set X of items, Bob provides an input set Y, then one or both of them learn $X \cap Y$, without learning anything about their opponent's items not in the intersection. Many of the most compelling real-world applications of secure multiparty computation are direct applications of PSI, or close variants of PSI such as private contact discovery [MPGP09, DRRT18].

PSI state of the art. Recently, PSI protocols have been the focus of significant concrete performance improvements (see [HEK12, DCW13, PSZ14, PSSZ15, KKRT16, RR17a, RR17b, CDCS18, PRTY19, CM20, PRTY20]). There are several protocol paradigms for PSI, but in this work we focus on the two most practical approaches: **Diffie-Hellman** and **OT-extension**. Other protocol paradigms (FHE, RSA, generic MPC) are many orders of magnitude slower.

Diffie-Hellman protocols. The first and arguably simplest PSI protocol is due to Huberman, Franklin, and Hogg [HFH99], but with roots as far back as Meadows [Mea86]. It is a semi-honest protocol that requires exponentiations in a Diffie-Hellman group proportional to the number of items in the sets. Because this protocol follows so elegantly from Diffie-Hellman key agreement, there is a rather limited design space of variants for semi-honest security (one variant is implicit in [JL10]). The DH-PSI protocol has been strengthened for malicious security in several works. The most efficient to date is due to De Cristofaro, Kim, and Tsudik [DKT10]. Another efficient, malicious variant is due to Jarecki & Liu [JL10], although it achieves a functionality that slightly relaxes the input independence security guarantee.

*Author version of a paper published at ACM CCS 2021, https://doi.org/10.1145/3460120.3484778

OT-extension protocols. The other category of PSI protocols is based on OT extension. With OT extension [Bea96, IKNP03], parties can generate many instances of oblivious transfer with only a small constant number of public-key operations. By basing PSI on many OTs, the number of public-key operations (exponentiations) in the resulting PSI protocol scales only with the security parameter, and not with the size of the input sets. PSI protocols in this category include [PSZ14, PSSZ15, KKRT16, RR17a, RR17b, PRTY19, PRTY20, CM20].

As a general rule, OT-based protocols are (significantly) faster but require more communication than Diffie-Hellman-based protocols. However, recent work of Pinkas et al. [PRTY19] presented an OT-based protocol with slightly less communication (and running time) than Diffie-Hellman-based PSI.

Why Care About Diffie-Hellman PSI? Since DH-based PSI is much slower (with exponentiations linear in the number of items) than OT-based PSI, what is the value in studying it? We suggest several reasons:

- In some scenarios, **communication cost** is overwhelmingly more important than computation cost. For a concrete example, Ion *et al.* [IKN+17, IKN+19] report on their real-world deployment of a PSI-like functionality within Google. They chose to deploy Diffie-Hellman PSI, and justified their choice as follows:

  "Somewhat surprisingly, for the offline 'batch computing' scenarios we consider, ***communication costs are far more important than computation.*** *This is especially the case for a secure protocol involving multiple businesses, where servers cannot be co-located (Wide area network solutions). Networks are inherently shared, and it is much less expensive to add CPUs to a shared network than to expand network capacity."* [from [IKN+19], bold formatting not in the original]

  Our improved DH-PSI protocol has the lowest communication among DH-based and OT-based protocols.[1]

- Consider the regime of **PSI on small sets.** For example, the PrivateDrop [HHS+21] system enhances Apple's AirDrop feature by performing a PSI of one user's entire address book (a few thousand items) with another user's own personal identifiers (*e.g.*, phone numbers and email addresses; perhaps 10 items), in order to determine whether one user appears in the other's address book. In another example, two parties may wish to use PSI of their available calendar times to schedule a meeting (∼360 half-hour slots during business hours in a single month). DH-based PSI protocols are the cheapest for these input sizes (equal-size sets of a few hundred items, or sets of highly unbalanced size where the larger set is a few thousand items); our improvements to DH-PSI give even further improvements.

  OT-based PSI protocols use OT extension, whose "base OTs" each require public-key operations (exponentiations). Concretely, using the most efficient 1-out-of-2 OT protocol to date [MR19], 128 base OTs cost $3 \times 128 = 384$ group elements of communication and $5 \times 128 = 640$ exponentiations. This is already more expensive than our improved DH-PSI protocol on sets of size 200, meaning that our protocol is **necessarily cheaper than any OT-extension-based protocol** for sets of this size. In fact the breakeven point, where OT-based protocols overtake ours, is between 500 and 1000 items on a fast network (10Gbps) and beyond 1000 items for a slow network (50Mbps).

- For OT-based PSI protocols, the performance gap between semi-honest and malicious is quite narrow due to recent improvements in malicious PSI by [PRTY20]. The case for DH-based PSI is much different, where the most efficient malicious PSI is 5× slower and requires 2.5× more communication. Our new approach essentially **closes the performance gap** between semi-honest and malicious, for DH-based PSI.

- Finally, the semi-honest DH-PSI protocol of [HFH99] is a truly classic protocol that has not been improved upon in over 20 years. Our new semi-honest protocol variant is the first to improve the communication cost of DH-PSI, and the improvement is not minor (over 40%). Even our malicious variant is more efficient than the classic semi-honest protocol. The only comparable improvement that we know of is due to Jarecki & Liu [JL10] who show how to improve only the *computational cost*, by about 5-15% in our experience.

[1] Some protocols based on FHE or RSA [DT10, ADT11] have even lower communication, but are several orders of magnitude higher in computation cost.

1.1 Related Work

Since its introduction, several techniques have been proposed to improve PSI's performance. In this section, we give an overview on existing efficient PSI protocols with more focus on the solutions that have linear-communication complexity due to public-key techniques. From here on, we assume that each set has n items, where each item has σ-bit length. We let λ and κ denote the statistical and computational security parameters, respectively.

The earliest PSI protocols were presented in the 1980s-1990s [Mea86, HFH99] and proven secure against semi-honest adversaries, in the random oracle model. These protocols remain the basis for comparison among Diffie-Hellman-based protocols.

Freedman et al. [FNP04] introduced PSI protocols secure against semi-honest and malicious adversaries in the standard model. Their protocol was based on oblivious polynomial evaluation (OPE) which is implemented using additively homomorphic encryption (AHE), such as Paillier encryption scheme. Relying on the OPE technique, Kissner and Song [KS05] proposed protocols for different set operations, such as set-intersection and set-union with quadratic computation and communication complexity in the size of dataset. Dachman-Soled et al. [DMRY09] present an improved construction of PSI protocol [KS05], which achieves communication of $O(n\kappa^2 \log^2(n) + \kappa n)$ group elements and $O(n^2\kappa \log(n) + n\kappa^2 \log^2(n))$ exponentiations in the presence of malicious adversaries. They avoid generic zero-knowledge due to the fact that Shamir's secret sharing implies a Reed-Solomon code. Later, Hazay and Nissim [HN10] extend OPE-based PSI protocol, and combine the efficiency of perfectly hiding commitment scheme with an OPRF evaluation protocol. The PSI protocol in [HN10] incurs communication of $O(n(1 + \log \sigma))$ group elements, and computation of $O(n(1 + \log\log(n) + \log(\sigma))$ modular exponentiations. Later, other variants of the problem were also investigated such as size-hiding set intersection [BFT16, CDCS18], PSI cardinality [DGT12, DD15], Private Intersection-Sum [IKN+19]. Here we highlight public-key based PSI protocols with linear-complexity.

Semi-honest PSI protocols. The current state-of-the-art for semi-honest PSI (independent of whether the protocols are based on DH or not) are the protocols of [KKRT16, PRTY19, CM20], with the best protocol depending on the relative cost of computation vs communication. Our protocol involves encoding values into polynomials, and this technique appears in some form in several PSI protocols. One such protocol is due to Cho, Dachman-Soled, and Jarecki [CDJ16]. Our protocol builds heavily on theirs, and we discuss it in more detail later. Another protocol of Pinkas *et al.* [PRTY19] is based on OT extension but also encodes certain values in a polynomial. Until our work, this protocol has had the lowest communication, excluding protocols based on expensive FHE or RSA accumulators.

For RSA-based PSI approaches, to the best of our knowledge, the work of Cristofaro and Tsudik [DT10], and its improvement [ADT11] proposed PSI protocols with lowest communication in this semi-honest setting. These protocol are based on RSA accumulators. The latter protocol achieves communication that is only marginally more than the insecure protocol for intersection (in which parties simply send hashes of their inputs). However, their computational requirements (at least $n \log(n)$ RSA exponentiations) make the protocol prohibitively expensive in practice due to the cost of RSA operations. We give further comparisons to the RSA approach later in Section 5.2.

Malicious PSI protocols. Jarecki and Liu [JL09] proposed the first linear-complexity PSI protocol based on OPRF in the presence of malicious adversaries. They constructed an OPRF protocol for the Dodis-Yampolskiy PRF $f_k(x) = g^{1/(k+x)}$, which requires $O(1)$ modular exponentiations and has constant-round communication. However, the secure computation protocol for their OPRF functionality is in the Common Reference String (CRS) model, where the CRS includes a safe RSA composite that either must be pre-generated by a trusted party or implies high overhead when produced in the secure two-party computation model. Another limitation of this protocol is that its security proof runs an exhaustive search over the input domain. This implies that the domain of the inputs should be polynomial in the security parameter.

De Cristofaro et al. [DKT10] presented a PSI protocol secure in the malicious setting, which achieves the same asymptotic bound as the previous work [JL09] without restricting the input domain size, and does not require the CRS model. Their PSI protocol incurs computation of $11n + 3$ modular exponentiations in a cyclic group.

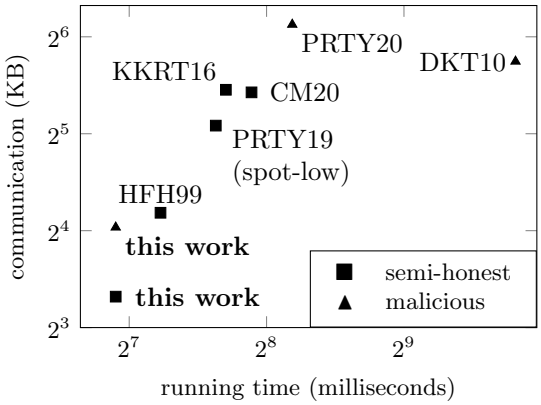

Figure 1: Time vs communication for PSI protocols on $n = 256$ items; LAN setting. Both axes are in log-scale.

Jarecki and Liu [JL10] is a concurrent work with [DKT10]. Their protocol [JL10] requires only $5n$ modular exponentiations for computing the adaptive set intersection in the presence of malicious adversaries, but under a One-More Gap Diffie-Hellman (OMGDH) assumption, which assumes that the One-More Diffie-Hellman problem is hard even when the DDH problem is easy.

Currently, the fastest malicious 2-party PSI protocols are due to Pinkas et al. [PRTY20], and more recently Rindal & Schoppmann [RS21]. They are not based on Diffie-Hellman, but on efficient OT extension or vector OLE [BCG+19]. The protocol of [RS21] is efficient when the set size is sufficiently large (e.g. $n > 2^{20}$), but it has significant fixed costs that make it inefficient for smaller sets.

In Table 1, we show the theoretical communication complexity of our protocol compared with the semi-honest and malicious protocols.

Protocol	**Communication**	$n = n_1 = n_2$ 2^8	2^9	2^{10}	2^{12}	2^{16}	2^{20}	Hardness Assumption
Semi Honest								
DH-PSI	$(\phi + \lambda + \log(n_1 n_2))n_1 + \phi n_2$	$568n$	$570n$	$572n$	$576n$	$584n$	$592n$	CDH
KKRT [KKRT16]	$(3+s)(\lambda + \log(n_1 n_2))n_1 + 1.2\ell n_2 + \lvert\texttt{baseOT}\rvert$	$1349n$	$1388n$	$1418n$	$1094n$	$1032n$	$1018n$	
SpOT-low [PRTY19]	$1.02(\lambda + \log_2(n_2) + 2)n_1 + \ell n_2 + \lvert\texttt{baseOT}\rvert$	$483n$	$493n$	$495n$	$499n$	$515n$	$532n$	
SpOT-fast [PRTY19]	$2(\lambda + \log(n_1 n_2))n_1 + \ell(1 + 1/\lambda)n_2 + \lvert\texttt{baseOT}\rvert$	$547n$	$559n$	$563n$	$571n$	$595n$	$619n$	CDH
PaXoS [PRTY20]	$(\lambda + \log_2(n_1 n_2))n_1 + \ell(2.4n_2 + \lambda + \chi) + \lvert\texttt{baseOT}\rvert$	$1074n$	$1095n$	$1097n$	$1101n$	$1128n$	$1155n$	
CM [CM20]	$(\lambda + \log(n_1 n_2))n_1 + 4.8\kappa n_2 + \lvert\texttt{baseOT}\rvert$	$670n$	$672n$	$674n$	$678n$	$686n$	$694n$	
VOLE-PSI (PaXoS)[RS21]	$(\lambda + \log(n_1 n_2))n_1 + 2^{17}\kappa n_2^{0.05} + 2.4\kappa n_2 + \lvert\texttt{baseOT}\rvert$	$86838n$	$45128n$	$23538n$	$6580n$	$825n$	$419n$	LPN+CDH
VOLE-PSI (interpolation)[RS21]	$(\lambda + \log(n_1 n_2))n_1 + 2^{17}\kappa n_2^{0.05} + \kappa n_2 + \lvert\texttt{baseOT}\rvert$	$86659n$	$44948n$	$23358n$	$6400n$	$646n$	**$240n$**	LPN+CDH
Ours (var. 1)	$(\lambda + \log(n_1 n_2))n_1 + \phi n_2 + \phi$	**$312n$**	**$314n$**	**$316n$**	**$320n$**	**$328n$**	$336n$	CDH
Ours (var. 2 – no ideal perm.)	$(\lambda + \log(n_1 n_2))n_1 + \phi n_2 + 2\phi$	**$312n$**	**$314n$**	**$316n$**	**$320n$**	**$328n$**	$336n$	ODH
Malicious								
DKT [DKT10]	$2\kappa n_1 + 6\phi n_2 + 2\phi$	$1792n$	$1792n$	$1792n$	$1792n$	$1792n$	$1792n$	CDH
JL [JL10]	$2\kappa n_1 + 3\phi n_2$	$1024n$	$1024n$	$1024n$	$1024n$	$1024n$	$1024n$	OMGDH
Hazay [Haz18]	$\phi(n_1 + n_2)\log(n_1 + n_2)$	$4608n$	$5120n$	$5632n$	$6656n$	$8704n$	$10752n$	CDH
PaXoS [PRTY20]	$2\kappa n_1 + \ell(2.4n_2 + 2\lambda + \chi) + \lambda(2.4n_2 + 2\ell) + \lvert\texttt{baseOT}\rvert$	$1370n$	$1389n$	$1389n$	$1389n$	$1408n$	$1427n$	CDH
VOLE-PSI (PaXoS)[RS21]	$2\kappa n_1 + 2^{17}\kappa n_2^{0.05} + 2.4\kappa n_2 + \lvert\texttt{baseOT}\rvert$	$87038n$	$45326n$	$23734n$	$6772n$	$1009n$	$595n$	LPN+CDH
VOLE-PSI (interpolation)[RS21]	$2\kappa n_1 + 2^{17}\kappa n_2^{0.05} + \kappa n_2 + \lvert\texttt{baseOT}\rvert$	$86859n$	$45146n$	$23554n$	$6592n$	$830n$	**$416n$**	LPN+CDH
Ours	$2\kappa n_1 + \phi n_2 + \phi$	**$512n$**	**$512n$**	**$512n$**	**$512n$**	**$512n$**	$512n$	ODH

Table 1: Theoretical communication costs of PSI protocols (in bits), calculated using computational security $\kappa = 128$ and statistical security $\lambda = 40$. The cost of base OTs are independent of input size and equal to 5κ, which are ignored in the columns $n = n_1 = n_2$. n_1 and n_2 are the input sizes of the sender and receiver respectively. ϕ is the size of elliptic curve group elements (256 is used here). ℓ is width of OT extension matrix (depends on n_1 and protocol). χ is the upper bound on the number of cycles in a cuckoo graph of PaXoS. The hardness assumptions that we list do not include random oracle or ideal permutation.

1.2 Summary of Our Results

We show how to transform any KA protocol (with pseudorandom messages and a natural non-malleability property) into a PSI protocol.

CDJ starting point. Our starting point is an approach of Cho, Dachman-Soled, and Jarecki (CDJ). Suppose Alice holds items $x_1, \ldots, x_n$ and Bob has items $y_1, \ldots, y_n$. Each party will run n instances of a (malicious) secure *string equality test* protocol, one for each of their inputs. Consider Alice's equality-test-protocol instance corresponding to item x_i. How will she send the protocol messages to Bob so that (1) if Bob also has x_i, then he will associate it with this instance (of the equality-test protocol) and not some other one, (2) if Bob doesn't have x_i, he won't know whether Alice was running an instance associated with x_i?

The main insight of CDJ — inspired by a technique originally due to Manulis, Pinkas, and Poettering [MPP10] — is to **embed protocol messages in a polynomial.** For each message of the equality-test protocol, Alice will interpolate a polynomial P such that $P(x_i)$ equals the next message for the ith equality test instance. When Bob receives the polynomial, he can evaluate it at each of his y_i inputs, respond to each one, and encode them into a polynomial of his own. Importantly, if the equality-test protocol messages are sufficiently random, then the polynomial P hides the x_i values of Alice.

Our improvements. We improve this CDJ paradigm in several dimensions. (1) Instead of embedding messages from a malicious-secure string-equality protocol into a polynomial, we can embed messages from a **plain key agreement (KA) protocol.** (2) We show that one party can avoid embedding n KA messages into a polynomial, and instead send only one KA message. This reduces the total communication significantly. (3) We simplify the protocol to use an ideal permutation in place of an ideal cipher.

In more detail, the CDJ mechanism has the parties run n instances of string equality tests. Each equality test will return either TRUE or FALSE, indicating which items are in the intersection. We observe that full-fledged equality tests are overkill for CDJ. Instead, let the parties run n instances of plain KA, embedded into polynomials according to their PSI inputs. Each of these KA instances terminates with an output key. If Alice and Bob hold a common item, then they will have a key in common. If Alice has an item that Bob doesn't (or vice-versa), we show that Alice computes a key that looks random to Bob. Hence, for PSI it suffices for the parties to simply compare their set of KA outputs in the clear.

Not only are key agreement protocols conceptually simpler and more concretely efficient than string equality test protocols — they are also **inputless.** As a result, KA protocols have the property that *their first protocol message can be reused* for many instances. This is not necessarily true for a string equality test protocol, where the party's input string would typically be "baked into" the first protocol message. In terms of the PSI protocol, this means that our protocol does not require a large polynomial of degree n (for n items) for the first message. Instead, Alice can send just a single KA protocol message, to which Bob computes n KA responses.

For a two-message KA protocol (like Diffie-Hellman), the fact that the second message is pseudorandom ensures that the polynomial hides the input set. By adding random oracle calls in a few selected places, we provide a "hook" for the simulator to extract malicious parties' inputs, yielding a malicious-secure PSI protocol.

Finally, the CDJ mechanism uses an ideal cipher for technical reasons (giving the simulator the ability to "program" outputs of the polynomial). We show that a simpler ideal permutation suffices.

Performance of the Diffie-Hellman Instantiation. When our new PSI paradigm is instantiated with Diffie-Hellman KA, we obtain the most efficient DH-based PSI protocol to date. For malicious security we require the **oracle Diffie-Hellman (ODH)** assumption [ABR01] to hold in the cyclic group. For semi-honest security we give two variants of the protocol — one that requires only the standard CDH or DDH assumption, and another that only requires ODH but *completely avoids the ideal permutation.*

Our protocol is both **faster and uses less communication than any other protocol**, when the set sizes are small (less than 1000 items) — even considering semi-honest protocols and protocols based on OT extension, which are faster on large sets. For $n = 256$ items, our malicious protocol is 18-30% faster (depending on the network speed) and uses 10% less communication than the next best (semi-honest) protocol. Our semi-honest variants use 45% less communication than the next best. See Figure 1 for a complete comparison.

To the best of our knowledge, ours is the **first significant improvement in communication cost to the 20-year old classic DH-PSI protocol**, due to [HFH99]. *We reduce the communication cost while simultaneously promoting it from semi-honest to malicious security.* The classic semi-honest DH-PSI protocol of [HFH99] requires total communication $2n$ group elements plus n hashes; the total computation is

PARAMETERS: Size of parties' sets: n for honest parties and n' for corrupt parties.

FUNCTIONALITY:

- Wait for input $Y \subseteq \{0,1\}^*$ from receiver. Abort if $|Y| > n$ and the receiver is honest, or if $|Y| > n'$ and the receiver is corrupt.
- Wait for input $X \subseteq \{0,1\}^*$ from sender and abort if $|X| > n$.
- Give output $X \cap Y$ to the receiver.

Figure 2: PSI ideal functionality.

$4n$ variable-base exponentiations. In our protocol, the total communication is only $n+1$ group elements plus n hashes; the total computation is $3n$ fixed-base exponentiations, $2n$ variable-base exponentiations, and 2 polynomial interpolation/multi-evaluations of a degree-n polynomial. The leading *malicious* DH-based PSI protocol is due to De Cristofaro, Kim, and Tsudik [DKT10]; its total communication is $6n$ group elements plus n hashes; the total computation is $2n$ fixed-based exponentiations and $4n$ variable-base exponentiations. Our malicious protocol is over $30\times$ faster and uses 80% less communication.

1.3 Updates Since Initial Publication

October 2021: We have revised the security of our semi-honest protocol variant. The original security proof had a bug which we have repaired. We also include a new variant of the semi-honest protocol that does not require an ideal permutation, but adds one extra field element of communication and requires the non-malleability property from the underlying KA scheme (the same property needed for our malicious protocol).

2 Preliminaries

2.1 Security Model

Secure two-party computation allows mutually distrusting parties to jointly perform a computation on their private inputs without revealing any additional information except for the result itself. There are two adversarial models, which are usually considered. In the semi-honest model, the adversary is assumed to follow the protocol, but may try to learn information from the protocol transcript. In the malicious model, the adversary follows an arbitrary polynomial-time strategy, and feasibility holds in the presence of both types of attacks.

2.2 PSI functionality

In Figure 2, we formally describe the PSI functionality, which allows 2 parties to compute the intersection of their datasets without revealing any additional information.

Note that the functionality allows a corrupt receiver to have more input items (n') than is "advertised" (n). This property reflects the fact that our protocol can't *tightly* enforce the number of items held by the receiver. This is a common feature of PSI protocols, shared in particular by all the fastest malicious-secure PSI protocols [RR17a, RR17b, PRTY20]. We discuss specific relationship between n' and n achieved by our protocol in Section 4.1.

2.3 Polynomial Operations

A common implementation of polynomial interpolation and multi-point evaluation is based on Lagrange algorithm, which costs $O(n^2)$ field operations. This implementation typically uses for low-degree polynomials. However, when n is very large (e.g. $n = 2^{20}$) this algorithm is completely impractical. In this work, we use the faster algorithms [MB72] which achieves computational complexity of $O(n \log^2 n)$ arithmetic

operations. At the high level idea, the algorithms for both problems follow the divide-and-conquer approach. Particularly, the problem is reduced to two half-size problems after each iteration. Each combination of individual solutions from two half-size problems to the full-size solution costs $O(n \log n)$. Therefore, the total complexity of polynomial interpolation and multi-point evaluation is $O(n \log^2 n)$.

Given $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{F}$ and $Y = \{y_1, \ldots, y_n\} \subseteq \mathbb{F}$, we use $P = \mathsf{interpol}_{\mathbb{F}}(\{(x_1, y_1), \ldots, (x_n, y_n)\})$ to refer to polynomial interpolation which finds the unique $(n-1)$-degree polynomial P that satisfies $P(x_i) = y_i$ for all $i \in [n]$.

2.4 Ideal Permutation

In the ideal permutation model, all parties have oracle access to a random permutation Π on $\{0,1\}^n$ and its inverse Π^{-1}. We write $\Pi^{\pm}$ to refer to the pair of these oracles. In the proof of security, the simulator answers the interface of $\Pi^{\pm}$, meaning that it can observe all queries and program the responses. The ideal permutation model is similar to, but weaker than, the *ideal cipher model*. An ideal cipher is a family of ideal permutations, one for each key.

The ideal permutation assumption has recently become popular in practical MPC implementations, because it allows one to base cryptographic operations on a *fixed-key block cipher* — *i.e.*, to use hardware-accelerated AES instructions without computing the AES key schedule. Ideal permutations have been used to realize efficient hashing functions for garbled circuits and OT extension [BHKR13, GKWY20]. Our work requires an ideal permutation that supports key-agreement messages as inputs, therefore our implementation uses Rijndael-256 rather than AES (whose block size is only 128). We note that other options are available to instantiate an ideal permutation. For example, symmetric-key constructions that use the sponge methodology [BDPV08] all use an efficient underlying ideal permutation.

3 Key Agreement Preliminaries

We construct PSI from 2-round key-agreement protocols. A **2-round key agreement protocol** KA has several parameters:

- $\mathsf{KA}.\mathcal{R}$ is the space of random coins for the two parties.
- $\mathsf{KA}.\mathcal{M}$ is the space of possible messages for Party 2.
- $\mathsf{KA}.\mathcal{K}$ is the space of possible output keys.

A key agreement protocol consists of algorithms: $\mathsf{KA.msg}_1$, $\mathsf{KA.msg}_2$, $\mathsf{KA.key}_1$, $\mathsf{KA.key}_2$, which correspond to an interactive key agreement protocol as shown in Figure 3.

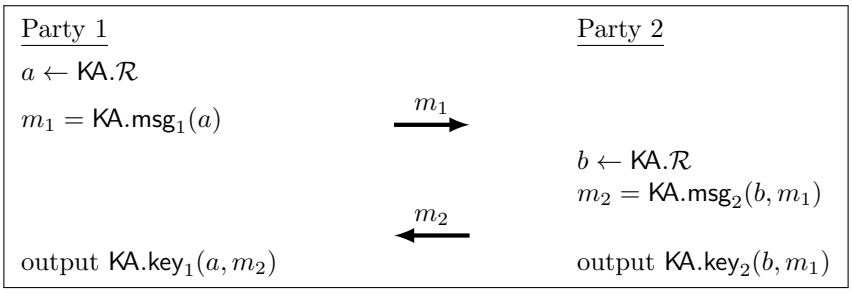

Figure 3: Generic 2-round key agreement protocol

In some 2-round key agreement protocols, the second message m_2 does not depend on the first message m_1, and we can write $m_2 = \mathsf{KA.msg}_2(b)$ instead of $m_2 = \mathsf{KA.msg}_2(b, m_1)$. In these cases, m_1 and m_2 can be sent simultaneously (or in either order), and we say that the key agreement protocol is **one-round**.

3.1 Security Properties

Different instantiations of our PSI protocol will require the following security properties of a key agreement protocol. Note that Definition 4 and strongly uniform KA (SU-KA) [FMV19] are similar, but our definition is specialized to 1-round KA.

Definition 1. *A KA scheme is* ***correct*** *if, when executed honestly as shown in Figure 3, the two parties give identical output. In other words, for all* $a, b \in \mathsf{KA}.\mathcal{R}$*:*

$$\mathsf{KA}.\mathsf{key}_1(a, \mathsf{KA}.\mathsf{msg}_2(b, \mathsf{KA}.\mathsf{msg}_1(a))) = \mathsf{KA}.\mathsf{key}_2(b, \mathsf{KA}.\mathsf{msg}_1(a))$$

Definition 2. *A KA scheme is* ***secure against an eavesdropper*** *if the following distributions are indistinguishable:*

$a, b \leftarrow \mathsf{KA}.\mathcal{R}$ $m_1 = \mathsf{KA}.\mathsf{msg}_1(a)$ $m_2 = \mathsf{KA}.\mathsf{msg}_2(b, m_1)$ $k = \mathsf{KA}.\mathsf{key}_2(b, m_1)$ return (m_1, m_2, k)	$a, b \leftarrow \mathsf{KA}.\mathcal{R}$ $m_1 = \mathsf{KA}.\mathsf{msg}_1(a)$ $m_2 = \mathsf{KA}.\mathsf{msg}_2(b, m_1)$ $k \leftarrow \mathsf{KA}.\mathcal{K}$ return (m_1, m_2, k)

Definition 3. *A KA scheme is* ***non-malleable*** *if it is secure (in the sense of Definition 2) against an eavesdropper that has oracle access to* $\mathsf{KA}.\mathsf{key}_1(a, \cdot)$*, provided the eavesdropper never queries the oracle on* m_2*. Formally, the following distributions are indistinguishable, for every PPT* $\mathcal{A}$ *that never queries its oracle on input* m_2*:*

$a, b \leftarrow \mathsf{KA}.\mathcal{R}$ $m_1 = \mathsf{KA}.\mathsf{msg}_1(a)$ $m_2 = \mathsf{KA}.\mathsf{msg}_2(b, m_1)$ $k = \mathsf{KA}.\mathsf{key}_2(b, m_1)$ return $\mathcal{A}^{\mathsf{KA}.\mathsf{key}_1(a,\cdot)}(m_1, m_2, k)$	$a, b \leftarrow \mathsf{KA}.\mathcal{R}$ $m_1 = \mathsf{KA}.\mathsf{msg}_1(a)$ $m_2 = \mathsf{KA}.\mathsf{msg}_2(b, m_1)$ $k \leftarrow \mathsf{KA}.\mathcal{K}$ return $\mathcal{A}^{\mathsf{KA}.\mathsf{key}_1(a,\cdot)}(m_1, m_2, k)$

Definition 4. *A KA scheme has* ***pseudorandom second messages*** *if* m_2 *is indistinguishable from random, even to someone who chooses* m_1 *adversarially. Formally, the following distributions are indistinguishable for all PPT* $\mathcal{A}$*:*

$(view, \widetilde{m}_1) \leftarrow \mathcal{A}$ $b \leftarrow \mathsf{KA}.\mathcal{R}$ $m_2 = \mathsf{KA}.\mathsf{msg}_2(b, \widetilde{m}_1)$ return $(view, m_2)$	$(view, \widetilde{m}_1) \leftarrow \mathcal{A}$ $m_2 \leftarrow \mathsf{KA}.\mathcal{M}$ return $(view, m_2)$

3.2 Diffie-Hellman Instantiation

The classic Diffie-Hellman key agreement protocol is a one-round KA protocol (meaning that the two messages can be sent simultaneously). It is parameterized by a cyclic group $\mathbb{G} = \langle g \rangle$ of order q, and defined as:

- $\mathsf{KA}.\mathcal{R} = \mathbb{Z}_q$ (space of private randomness)
- $\mathsf{KA}.\mathcal{M} = \mathbb{G}$ (space of second party's protocol messages)
- $\mathsf{KA}.\mathsf{msg}_1(a) = g^a$
- $\mathsf{KA}.\mathsf{msg}_2(b) = g^b$

In this work we consider the "hashed" variant of DH which is secure under the computational Diffie-Hellman (CDH) assumption in the random oracle model. Let $H : \mathbb{G} \to \{0,1\}^\ell$ be a random oracle, then:

- $\mathsf{KA}.\mathcal{K} = \{0,1\}^\ell$ (space of output keys)
- $\mathsf{KA}.\mathsf{key}_1(a, g^b) = H((g^b)^a)$
- $\mathsf{KA}.\mathsf{key}_2(b, g^a) = H((g^a)^b)$

Elligator DHKA

Modern applications of DHKA use elliptic curves for the underlying cyclic group, due to their compact size (*e.g.*, group elements with representations roughly 2κ bits, for κ bits of security). However, encodings of elliptic curve elements are rather conspicuous, and can easily be distinguished from uniformly distributed *strings*. Our PSI protocols require the KA protocol messages (specifically, m_2) to be pseudorandom as strings.

In [BHKL13], Bernstein *et al.* explicitly consider the question of encoding elliptic curve elements so that the resulting Diffie-Hellman protocol has pseudorandom messages (viewed as strings). Formally, they define an encoding mechanism called **elligator** with the following properties:

- There are efficient encoding/decoding functions $\mathsf{dec}, \mathsf{enc}$ which are inverses, where $\mathrm{im}(\mathsf{enc}) \subseteq \{0,1\}^t$ is a set of strings and $\mathrm{im}(\mathsf{dec}) \subseteq \mathcal{E}$ is a subset of elliptic curve points.
- The size of $\mathrm{im}(\mathsf{enc})$ is very close to 2^t, so that the uniform distribution over encodings is indistinguishable from the uniform distribution over $\{0,1\}^t$
- The size of $\mathrm{im}(\mathsf{dec})$ is a constant fraction (typically close to $1/2$) of the size of the elliptic curve.
- It is possible to efficiently test for membership in $\mathrm{im}(\mathsf{enc})$ (and hence also in $\mathrm{im}(\mathsf{dec})$).

After defining such an elligator encoding method for Edwards curves, Bernstein *et al.* propose to modify Diffie-Hellman key agreement as follows:

- $\mathsf{KA}.\mathcal{R} = \{r \in \mathbb{Z}_q \mid g^r \in \mathrm{im}(\mathsf{dec})\}$.
- $\mathsf{KA}.\mathcal{M} = \{0,1\}^t$
- $\mathsf{KA}.\mathsf{msg}_1(a) = \mathsf{enc}(g^a)$
- $\mathsf{KA}.\mathsf{msg}_2(b) = \mathsf{enc}(g^b)$
- $\mathsf{KA}.\mathsf{key}_1(a, s_b) = H(\mathsf{dec}(s_b)^a)$
- $\mathsf{KA}.\mathsf{key}_2(b, s_a) = H(\mathsf{dec}(s_a)^b)$

In other words, the parties condition their randomness to always sample a point in the "elligator subset" $\mathrm{im}(\mathsf{dec})$ of the curve. In practice, each party would repeatedly sample an exponent $r \leftarrow \mathbb{Z}_q$ and retry until finding one in the elligator subset. Since $|\mathrm{im}(\mathsf{dec})|/|\mathcal{E}|$ is constant, only a constant number of trials is needed before successfully hitting $\mathrm{im}(\mathsf{dec})$. Furthermore, the concrete security of DHKA is degraded by only a small constant factor.

Due to the desired properties of the elligator encoding, the protocol messages are uniform in $\mathrm{im}(\mathsf{enc})$ and hence pseudorandom in $\{0,1\}^t$.

Security properties

The security of hashed DHKA against an eavesdropper (Definition 2) is standard and follows from the CDH assumption.

The "pseudorandom second messages" property (Definition 4) of the elligator-DHKA protocol follows from the properties of elligator discussed above. Note that in DHKA, m_2 doesn't depend on m_1, so the adversary's ability to choose m_1 in Definition 4 is irrelevant.

Finally, the "non-malleable" property (Definition 3) of hashed DHKA is equivalent to the **oracle DH (ODH)** assumption proposed by Abdalla, Bellare, and Rogaway [ABR01]. Roughly speaking, the ODH assumption is that $g^a, g^b, H(g^{ab})$ is indistinguishable from random in the presence of an oracle for $X \mapsto H(X^a)$, as long as the distinguisher doesn't query that oracle on g^b. Here H is the hash function / random oracle used in hashed DHKA. In [ABR01] it is shown that the ODH assumption holds in the generic group model when H is a random oracle.

4 Malicious PSI from Key Agreement

In this section we present our main result, a malicious 2-party PSI protocol. Our protocol requires the following building blocks:

- A 2-round KA protocol KA. Recall that $\mathsf{KA}.\mathcal{M}$ is the space of possible protocol messages. We require $\mathsf{KA}.\mathcal{M} = \mathbb{F}$ for some finite field $\mathbb{F}$, and that the KA protocol has pseudorandom messages in this field. We also require the KA protocol to be non-malleable in the sense of Definition 3.
- Parties have oracle access to an ideal permutation Π, Π^{-1} defined over the same field $\mathbb{F}$. We write $\Pi^{\pm}$ to refer to the two functions Π, Π^{-1} collectively. Parties also have access to random oracles H_1, H_2.

As a concrete example, we can choose hashed DHKA with elligator encodings (see Section 3.2), whose protocol messages are pseudorandom in $\{0,1\}^\ell$, and then set $\mathbb{F}$ be to the field $GF(2^\ell)$. Under the ODH assumption, hashed DHKA is also non-malleable. We give more details about instantiating our protocol with Diffie-Hellman in Section 5.

Protocol Overview. Following the overview given in Section 1, the sender sends the first KA message. Intuitively, the receiver prepares a polynomial P such that $P(y_i)$ is a KA response that it chooses, for each y_i in its set. If KA responses are pseudorandom then the polynomial P hides the identities of the y_i-values.

However, for technical reasons we make the receiver prepare a polynomial such that $P(H_1(y_i)) = \Pi^{-1}(m_i)$ where H_1 is a random oracle, Π is an ideal permutation, and m_i is the KA response. The presence of random oracle H_1 helps the simulator extract from a corrupt receiver (from observing its H_1-queries). The presence of the ideal permutation helps the simulator (against both corrupt parties), by programming Π to output KA messages chosen by the simulator.

Finally, the sender can interpret $\Pi(P(H_1(x_i)))$ as a KA response, for each x_i in its set, and compute the corresponding KA output k_i. For each x_i, the sender sends $H_2(x_i, k_i)$ to the receiver. The presence of this random oracle again helps the simulator extract from a corrupt sender.

The protocol is described formally in Figure 4. $\mathsf{interpol}_{\mathbb{F}}$ denotes polynomial interpolation over field $\mathbb{F}$, as discussed in Section 2.3.

Lemma 5. *The protocol of Figure 4 is UC-secure against a malicious sender, if KA has pseudorandom messages (Definition 4), $\Pi^{\pm}$ is an ideal permutation, and H_2 is a random oracle.*

Before giving the proof, we first sketch the main idea of the simulator. When the simulator sees the set K provided by the adversary, it needs to extract a set of items that "explains" the effect of K on the honest party. The elements of K are supposed to have the form $H_2(x_i, k_i)$, where k_i is the "correct" KA output for item x_i. The simulator observes all queries to H_2, so it can see which H_2-outputs are placed into K — but how can the simulator check that some k_i is the "correct" KA output corresponding to item x_i? To do this, we let the simulator program Π so that every output of Π is a KA message for which it knows the randomness. Now for any x, the simulator can compute the corresponding KA output, using the KA randomness it associates with $\Pi(P(H_1(x)))$.

Proof. We first describe the behavior of the simulator.

- The simulator honestly plays the role of random oracle H_2. For every query $H_2(x,k)$ made by the adversary, the simulator records the input-output tuple $(x, k, H_2(x,k))$ in a set $\mathcal{O}_2$.
- For every query of the form $\Pi(f)$ made after the message m is sent, the simulator chooses a random $b_f \leftarrow \mathsf{KA}.\mathcal{R}$ and simulates $\mathsf{KA}.\mathsf{msg}_2(b_f, m)$ as the output of $\Pi(f)$.
- In step 4, the simulator sends a uniform polynomial P.
- Upon receiving K in step 6, the simulator defines the set

$$\tilde{X} = \{x \mid \exists k' : \left(x, \mathsf{KA}.\mathsf{key}_2(b_{P(H_1(x))}, m), k'\right) \in \mathcal{O}_2 \text{ and } k' \in K\}$$

and sends $\tilde{X}$ to the ideal PSI functionality (which causes the honest receiver to obtain output $\tilde{X} \cap Y$).

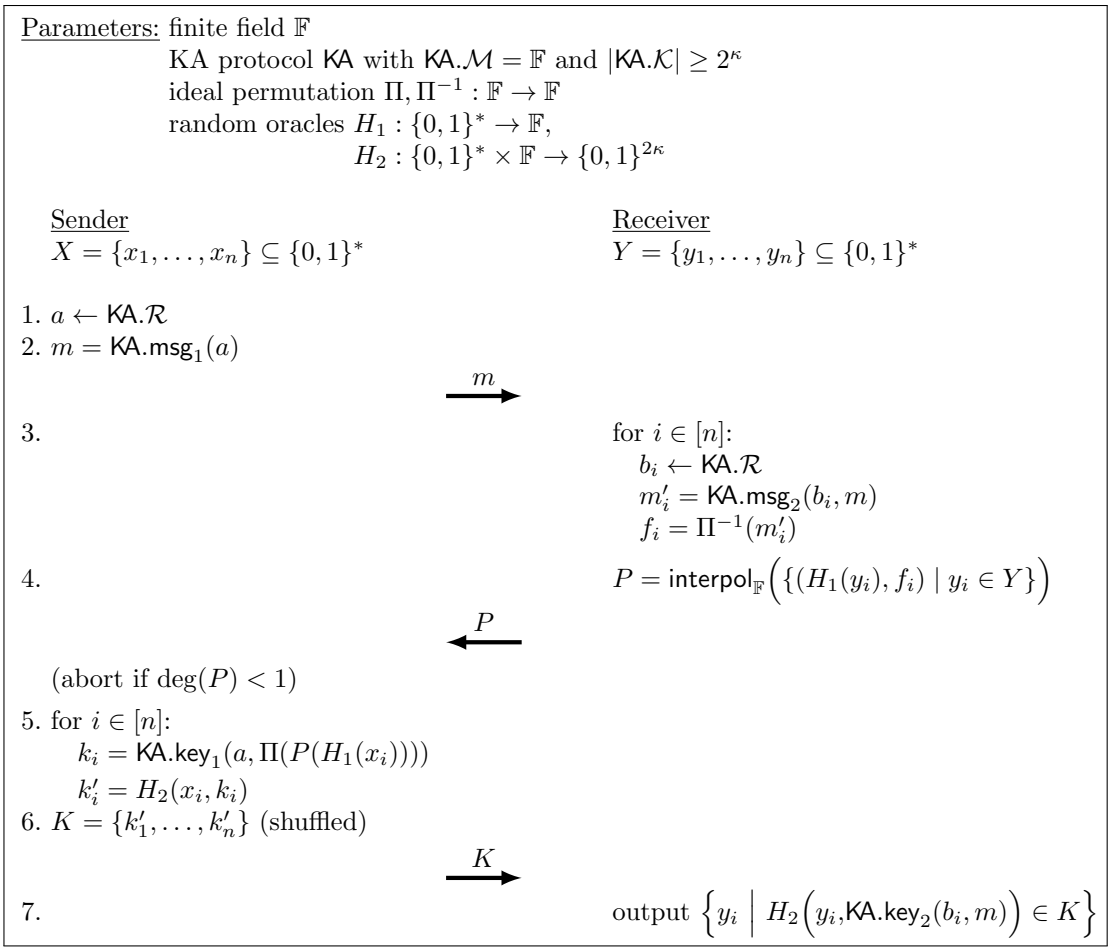

Figure 4: Our malicious PSI protocol.

We prove that this simulation is indistinguishable from the real interation via the following sequence of hybrids.

Hybrid 0: The real interaction, with the receiver running honestly with input Y and giving its output to the environment according to the protocol specification.

Hybrid 1: Same as the previous hybrid, except for how $\Pi^{\pm}$ is simulated. A query to Π (resp. Π^{-1}) is *fresh* if it was never made before, and its value is not determined by previous queries to Π^{-1} (resp. Π) and the fact that Π/Π^{-1} are inverses. In this hybrid, all fresh queries (by either the adversary or honest party) to Π and Π^{-1} are answered with a uniformly random response. The interaction aborts if this leads to Π or Π^{-1} repeating an output. This change is indistinguishable from the standard permutation switching lemma.

Hybrid 2: Same as the previous hybrid, except for how P is generated. In step 3, P is generated by interpolating through points of the form $\Pi^{-1}(\mathsf{KA}.\mathsf{msg}_2(b_i, m))$. In this hybrid we abort if these queries to Π^{-1} are not fresh — *i.e.*, if $\mathsf{KA}.\mathsf{msg}_2(b_i, m)$ previously occurred as either an adversary's query to Π^{-1} or as an output of an adversary's query to Π.

If the $\mathsf{KA}.\mathsf{msg}_2(b_i, m)$ terms were independently and uniformly random, then this abort would happen with probability bounded by $nq/|\mathbb{F}|$, when the adversary makes q oracle queries. By the pseudorandom property of the KA scheme, each $\mathsf{KA}.\mathsf{msg}_2(b_i, m)$ is indistinguishable from random, so the abort probability is negligibly close to $nq/|\mathbb{F}|$. Either way, the probability is negligible, so the hybrids are indistinguishable.

Now conditioned on not aborting, we have that each $\Pi^{-1}(\mathsf{KA}.\mathsf{msg}_2(\cdots))$ is a fresh and *uniform* value. Hence, P is distributed as a uniform polynomial, independent of the y_i values. Then this interaction is

identical to one in which we *first* choose a uniform polynomial P and then later program $\Pi(P(H_1(y_i))) = \mathsf{KA.msg}_2(b_i, m)$ for each $y_i \in Y$ (aborting if Π is already programmed on this point).

Hybrid 3: Same as the previous hybrid, except for how Π is simulated. For every fresh query $\Pi(f)$ made after the adversary sends m, sample $b_f \leftarrow \mathsf{KA}.\mathcal{R}$ and respond with $\mathsf{KA.msg}_2(b_f, m)$ (instead of responding with a uniform result). This change is indistinguishable by the pseudorandomness property of KA.

Note that we have already been simulating $\Pi(P(H_1(y_i)))$ in this way for $y_i \in Y$, but with different variable names (randomness b_i rather than b_f for $f = P(H_1(y_i))$). If we rename randomness b_i (for $y_i \in Y$) to $b_{P(H_1(y_i))}$ then we program Π in the same way for all inputs, with no special case for the elements of Y. In doing so, the honest party's output is computed via:

$$\{y_i \in Y \mid H_2\Big(y_i, \mathsf{KA.key}_2(b_{P(H_1(y_i))}, m)\Big) \in K\}$$

Hybrid 4: The honest receiver queries H_2 to determine its final output (in the expression above). In this hybrid we abort if one of those H_2 queries is *fresh* (meaning that the adversary did not make that query) and yet the result is in K. The probability of a fresh query's output being an element of K is $|K|/|\mathbb{F}| = n/|\mathbb{F}|$, which is negligible. Therefore this change is indistinguishable.

Suppose the final hybrid maintains the list $\mathcal{O}_2$ as described earlier — *i.e.*, $(x, k, k') \in \mathcal{O}_2$ means that the adversary queried $H_2(x, k)$ and got a result k'. Since the receiver only "recognizes" values that the adversary has already queried to H_2, this final hybrid is identical to one in which the receiver's output is computed as:

$$\{y_i \in Y \mid \exists k' : \Big(y_i, \mathsf{KA.key}_2(b_{P(H_1(y_i))}, m), k'\Big) \in \mathcal{O}_2 \text{ and } k' \in K\}$$

But this is logically equivalent to:

$$Y \cap \underbrace{\{x \mid \exists k' : \Big(x, \mathsf{KA.key}_2(b_{P(H_1(x))}, m), k'\Big) \in \mathcal{O}_2 \text{ and } k' \in K\}}_{\tilde{X}}$$

Here $\tilde{X}$ is the set that the simulator can define. Hence this hybrid is identical to the ideal interaction involving the simulator defined earlier. □

Lemma 6. *The protocol of Figure 4 is UC-secure against a malicious receiver, if* KA *is non-malleable (Definition 3),* $|\mathsf{KA}.\mathcal{K}| \geq 2^\kappa$*,* H_1, H_2 *are random oracles, and* $\Pi^\pm$ *is an ideal permutation.*

Before giving the proof, we first sketch the main idea of the simulator. The simulator's job, when the adversary gives the polynomial P, is to extract a set $\tilde{Y}$ that it can send to the ideal functionality. Then, after learning $X \cap \tilde{Y}$, it simulates the message K appropriately. Intuitively, we want to make a distinction between KA instances where the receiver *participates* versus KA instances where the receiver *acts as an eavesdropper.* The former instances will correspond to the items of $\tilde{Y}$ and the latter instances will contribute to KA outputs (and elements of K) that look random.

The honest sender will interpret $\Pi(P(H_1(x)))$ as a KA message, for every $x \in X$. The receiver only "controls" this value if: (1) it made a query to $H_1(x)$; (2) it made a *backwards query* to Π^{-1} that resulted in the value $P(H_1(x))$. If on the other hand the adversary chose $P(H_1(x))$ first and only then made a *forward query* at $\Pi(P(H_1(x)))$, then intuitively it will have no control over the resulting value.

The simulator observes all queries to $\Pi^\pm$ and to H_1, and can therefore use these criteria to identify which KA instances will give outputs that the receiver can recognize. All other KA outputs can safely be replaced with random.

We draw the reader's attention to two subtleties in the proof: Suppose the adversary queries Π to obtain some KA message m^*. Since (intuitively) the adversary has no control over m^*, we would like to argue that the corresponding $\mathsf{KA.key}(m^*)$ (slightly abusing notation here) looks random. But suppose the adversary programs P so that $\Pi(P(H_1(y))) = m^*$ and $\Pi(P(H_1(y'))) = m^* + 1$. If the sender has both inputs y and y', then she will compute and send $\mathsf{KA.key}(m^*)$ and $\mathsf{KA.key}(m^* + 1)$. Does the former KA output look random even in the presence of the latter? It does if the KA protocol is **non-malleable** in the sense of Definition 3.

Another subtlety is that the receiver may choose its polynomial P to have "collisions" in the sense that $P(H_1(y)) = P(H_1(y'))$. This is not a problem or an attack *per se*, but it means that the hybrids in the

proof must be structured carefully. The goal of the proof is to justify that the sender's messages of the form $H_2(x_i, \mathsf{KA.key}(\Pi(P(H_1(x_i)))))$ can be replaced with random values, for all x_i not in the intersection. But the sequence of hybrids does not replace these real values with random one at a time. Instead, we replace $\Pi(P(H_1(\cdot)))$ *outputs*, one at a time, with KA messages chosen by the simulator. Then we can argue that $\mathsf{KA.key}(\Pi(P(H_1(x_i))))$ is indistinguishable from random for **possibly many** values of x_i that give the same $P(H_1(x_i))$.

Proof. We first formally describe the behavior of the simulator:

- The simulator honestly plays the role of random oracle H_1 and ideal permutation $\Pi^{\pm}$. For every query $H_1(y)$ made by the adversary, record y in a set $\mathcal{O}_1$. For every query $\Pi^{-1}(m) = f$, where **there was no previous query** of the form $\Pi(f) = m$, record f in a set $\mathcal{O}_\Pi$.
- The simulator runs steps 1–2 honestly.
- Upon receiving P in step 4, the simulator defines the set

$$\tilde{Y} = \{y \mid y \in \mathcal{O}_1 \text{ and } P(H_1(y)) \in \mathcal{O}_\Pi\}$$

  and sends $\tilde{Y}$ to the ideal PSI functionality.
- Upon receiving output $Z = X \cap \tilde{Y}$ from the functionality, the simulator computes $k_z = \mathsf{KA.key}_1(a, \Pi(P(H_1(z))))$ for each $z \in Z$. Define $K = \{H_2(z, k_z) \mid z \in Z\}$ and then keep adding uniformly random values to K until $|K| = |X|$. The simulator finally sends this K to the adversary.

We prove that this simulation is indistinguishable from the real interation via the following sequence of hybrids.

Hybrid 0: The real interaction, with the sender running honestly on input X. In particular, the protocol message K is generated as follows:

$$K = \left\{ H_2\Big(x, \mathsf{KA.key}_1\big(a, \Pi(P(H_1(x)))\big)\Big) \mid x \in X \right\}$$

The lists $\mathcal{O}_1$ and $\mathcal{O}_\Pi$ are also maintained, as defined above.

Hybrid 1: Same as the previous hybrid, except the interaction aborts in step 5 if there is an $x \in X$ where $x \notin \mathcal{O}_1$ and yet $P(H_1(x)) \in \mathcal{O}_\Pi$. In other words, the adversary nevery queried $H_1(x)$ and yet $P(H_1(x))$ is a value that it previously received as output from Π^{-1}.

It suffices to show that the probability of such an abort is negligible. For any $f \in \mathcal{O}_\Pi$, the polynomial equation $P(\cdot) = f$ has at most n solutions, since P is a polynomial of degree n, and not the zero polynomial (that would mean P is a constant polynomial and the sender would have already aborted in step 4). Since $H_1(x)$ is a fresh query never made before (until the simulated sender makes it), it is uniformly distributed in $\mathbb{F}$ and therefore has at most $n/|\mathbb{F}|$ probability of satisfying $P(H_1(x)) = f$. Suppose the adversary makes a total of q queries to its oracles. By a union bound over all n choices of $x \in X$ and q choices of $f \in \mathcal{O}_\Pi$, the total probability of this event is $n^2 q/|\mathbb{F}|$, which is negligible.

Hybrid $(2, i)$, for $i \in [q]$: Same as the previous hybrid, except for the following changes. For the first i queries of the form $\Pi(f) = m$, where there was no previous query to $\Pi^{-1}(m)$, add f to the set $\mathcal{S}_i$. Note that $\mathcal{S}_i$ and $\mathcal{O}_\Pi$ are necessarily disjoint (based on whether Π or Π^{-1} was queried first). Intuitively, $\mathcal{S}_i$ are the first i Π-outputs (interpreted in the protocol as KA protocol messages) that the adversary has no control over. Then compute K instead as:

$$K = \left\{ H_2\Big(x, \mathsf{KA.key}_1\big(a, \Pi(P(H_1(x)))\big)\Big) \mid x \in X \text{ and } P(H_1(x)) \notin \mathcal{S}_i \right\}$$

and thereafter add uniformly random elements to K until $|K| = n$. Note that there may be many values of x giving the same $P(H_1(x))$ output, so there may be many values of x treated differently between Hybrids $(2, i)$ and $(2, i+1)$.

It should be clear that Hybrid $(2, 0)$ is identical to Hybrid 2, since $\mathcal{S}_0 = \emptyset$ and the new condition is always true. In Lemma 7 we prove that Hybrids $(2, i)$ and $(2, i+1)$ are indistinguishable.

Hybrid 3: We rewrite Hybrid $(2, q)$ for clarity. In this hybrid, every $\Pi(f) = m$ that is known in the interaction is represented in either $\mathcal{S}_q$ (for those known by an initial Π-query) or $\mathcal{O}_\Pi$ (for those known by an initial Π^{-1} query). In other words, these two sets form a partition of all known $\Pi(f) = m$ points.

Let us consider how the set K is computed in this hybrid. The condition $P(H_1(x)) \notin \mathcal{S}_q$ is equivalent to $P(H_1(x)) \in \mathcal{O}_\Pi$, meaning that we can write:

$$K = \left\{ H_2\Big(x, \mathsf{KA.key}_1\big(a, \Pi(P(H_1(x)))\big)\Big) \mid x \in X \text{ and } P(H_1(x)) \in \mathcal{O}_\Pi \right\}$$

(padded with random values).

Recall that the interaction aborts if there is any $x \notin \mathcal{O}_1$ but $P(H_1(x)) \in \mathcal{O}_\Pi$. In other words, conditioned on even reaching this point in the interaction, $P(H_1(x)) \in \mathcal{O}_\Pi$ implies $x \in \mathcal{O}_1$. Hence we can further rewrite the definition of K as:

$$K = \left\{ H_2\Big(x, \mathsf{KA.key}_1\big(a, \Pi(P(H_1(x)))\big)\Big) \mid \begin{array}{c} x \in X \cap \mathcal{O}_1 \\ \text{and } P(H_1(x)) \in \mathcal{O}_\Pi \end{array} \right\}$$

Now, suppose we define $\tilde{Y} = \{y \mid y \in \mathcal{O}_1 \text{ and } P(H_1(y)) \in \mathcal{O}_\Pi\}$. Then K can be rewritten in the equivalent form:

$$K = \left\{ H_2\Big(x, \mathsf{KA.key}_1\big(a, \Pi(P(H_1(x)))\big)\Big) \mid x \in X \cap \tilde{Y} \right\}$$

In this form, it is now clear that the hybrid corresponds to the behavior of the ideal interaction. That is, the simulator computes $\tilde{Y}$, and then computes K based only on the contents of $Z = X \cap \tilde{Y}$, its output from the functionality. □

Lemma 7. *Hybrids $(2, i-1)$ and $(2, i)$ are indistinguishable, if the KA protocol is non-malleable (Definition 3) and $|\mathsf{KA}.\mathcal{K}| \geq 2^\kappa$.*

Proof. The hybrids differ only in the following way: Hybrid $(2, i)$ replaces $\mathsf{KA.key}_1(a, \Pi(f^*))$ with random, in the event that f^* was the ith query to Π (with no corresponding prior Π^{-1} query).

Recall that in the game that defines non-malleability of a KA, the distinguisher receives $(m_1 = \mathsf{KA.msg}_1(a), m_2, k)$ and also gets access to an oracle for $\mathbb{K}(\cdot) = \mathsf{KA.key}_1(a, \cdot)$, which it cannot query on m_2. Below is a reduction algorithm that is a distinguisher for the non-malleability game:

$\underline{\mathcal{R}^{\mathbb{K}}(m_1, m_2, k)}$:

- Run Hybrid $(2, i-1)$ against the adversary, using m_1 as the PSI protocol message m.
- Maintain set $\mathcal{S}_{i-1}$ as described. On the ith query to Π (*i.e.*, the value that would have been added to $\mathcal{S}_i$), let f^* denote the input and simulate $m_2 = \Pi(f^*)$ as the response.
- For every expression of the form $\mathsf{KA.key}_1(a, \Pi(P(H_1(x))))$ used in the definition of K:
  - If $P(H_1(x)) = f^*$ then replace the entire expression with k (input to this reduction algorithm).
  - Otherwise, replace the entire expression with the result of $\mathbb{K}(\Pi(P(H_1(x))))$, where $\mathbb{K}$ is the reduction algorithm's oracle. Since Π is a permutation, we have $\Pi(P(H_1(x))) \neq \Pi(f^*) = m_2$; in other words, the oracle $\mathbb{K}$ is never invoked on m_2.

Intuitively, this reduction algorithm runs the hybrid interaction without knowing the KA randomness a. Instead, a is used implicitly via m_1, k, and the oracle $\mathbb{K}$.

When the input k is the correct key $k = \mathsf{KA.key}_1(a, m_2)$, then the simulation exactly matches Hybrid $(2, i-1)$, since the reduction correctly uses k in place of the expression $\mathsf{KA.key}_1(a, \Pi(f^*)) = \mathsf{KA.key}_1(a, m_2)$.

Now consider the case that k is a random key. Then whenever $P(H_1(x)) = f^*$, the value $H_2(x, k)$ is added to K. Since H_2 is a random oracle, and since k is uniform (and $|k| \geq \kappa$), outputs $H_2(x, k)$ are indistinguishable from random, even for multiple values of x (*e.g.*, in the case where the adversary constructs P so that $P(H_1(x)) = f^*$ for several values of x). In summary, when k is uniform, the simulation is indistinguishable from Hybrid $(2, i)$ in which a random value is added to the set K in these cases. The non-malleabiity of KA means that these two cases are indistinguishable. □

Optimizations. When KA is a one-round key agreement protocol (*i.e.*, message 2 doesn't depend on message 1, as in the Diffie-Hellman instantiation), then the two messages m and K from the sender can be combined. This leads to a **2-round PSI protocol** where the first flow is P from the receiver and the second flow is m, K from the sender.

Note that the direction of the last message (H_2 outputs from sender to receiver) is important. It is not possible to save a round of communication by letting the receiver send H_2 outputs to the sender. These H_2 outputs are computed using the result of a KA between a common a (chosen by the sender) and various b_i (chosen by the receiver). Knowing a, the sender can compute the "correct" H_2 for *any* x, so the receiver would expose a dictionary attack by sending their set of H_2 outputs.

If security is required against only **semi-honest** adversaries, then the protocol can be streamlined slightly. We describe two semi-honest variants in Appendix A, which have slightly improved performance and slightly more favorable hardness assumptions.

Two other possible optimizations are presented in Appendix B.

Costs. The sender must compute one KA message and n KA keys/outputs. The receiver computes n KA responses and n KA keys/outputs. Both parties make n queries to each of H_1, H_2, and $\Pi^{\pm}$. Finally, the receiver must interpolate a polynomial of degree n, and the sender must evaluate such a polynomial on n points. These are both possible with $O(n \log^2 n)$ field operations, as described in Section 2.3.

The total communication cost of the protocol consists of: (1) 1 KA message from the sender, (2) n field elements (each equivalent in size to a KA response) from the receiver to describe P, (3) n outputs of H_2, each 2κ bits.

4.1 Size of Adversary's Set

Recall that we consider an ideal functionality in which a corrupt party can provide an input set that is "larger than advertised." If a corrupt party (specifically, the receiver) provides an input that is as large as the universe of possible items, then PSI provides no security whatsoever. Hence, it is important to bound the size of the set that the simulator extracts.

Corrupt Sender. The sender gives a set K during the protocol, which is supposed to contain H_2-outputs. The simulator extracts by finding x such that $H_2(x, k) \in K$, for an appropriate value k. Since the output of H_2 is 2κ bits, the probability of the adversary encountering a collision in H_2 is negligible. Hence for each item in K, there is at most one preimage known to the adversary/simulator and hence at most one item that will be included in the extracted set $\tilde{X}$.

In other words, the simulator extracts an input set for a corrupt sender of size at most $|K| = n$. The protocol strictly enforces the size of a corrupt sender's input set.

Corrupt Receiver. The simulator for a corrupt receiver extracts their input set as

$$\tilde{Y} = \{y \mid y \in \mathcal{O}_1 \text{ and } P(H_1(y)) \in \mathcal{O}_\Pi\}$$

Abstractly speaking, the adversary sees q outputs of H_1, and it sees q outputs of Π. In the simulation, outputs of both H_1 and Π are uniform. The adversary then generates a polynomial P of degree less than n (and greater than 0) and the simulator checks whether $P(\alpha) = \beta$ for all outputs α from H_1 and all outputs β from Π. The number of such pairs is the size of the set that is extracted. The question is therefore **how many random points can the adversary fit on a degree $< n$ polynomial?**

CDJ shows that if the size of the field is $2^{n\omega(\log \kappa)}$ then with overwhelming probability no polynomial can fit more points than its degree suggests. However, such a large field leads to *quadratic* total communication (n coefficients in a field of more than 2^n elements). We instead prefer to stick to a field of minimum size (large enough only to encode a KA message) and obtain bounds on the number of items.

Definition 8. *Let $\mathbb{F}$ be a field and define the $\mathsf{PolyOverfit}_{\mathbb{F}}^{n,n'}(q)$ game against an adversary $\mathcal{A}$ to be as follows:*

sample $\alpha_1, \ldots, \alpha_q, \beta_1, \ldots, \beta_q \leftarrow \mathbb{F}$
$P \leftarrow \mathcal{A}(\alpha_1, \ldots, \alpha_q, \beta_1, \ldots, \beta_q)$
if $0 < \deg(P) < n$ and $\left|\{\alpha_i \mid P(\alpha_i) \in \{\beta_1, \ldots, \beta_q\}\}\right| \geq n'$:
    declare $\mathcal{A}$ the winner

In other words, the adversary tries to generate a polynomial that hits some β_j on at least n' distinct α_i's.

We say that $\mathsf{PolyOverfit}_{\mathbb{F}}^{n,n'}$ is hard if for all polynomial q and all PPT $\mathcal{A}$, the adversary wins with negligible probability.

Proposition 9. *If $\mathsf{PolyOverfit}_{\mathbb{F}}^{n,n'}$ is hard, then the simulator for a corrupt receiver in our PSI protocol outputs a set of size bounded by n', except with negligible probability.*

In Appendix C we show the following using a standard compression argument. If such an "overfitting" polynomial existed, it could be used to generate a compressed representation of the α_i's and β_i's, which is impossible if they are uniform.

Lemma 10. *The probability of winning $\mathsf{PolyOverfit}_{\mathbb{F}}^{n,n'}(q)$ is at most $(q^2 n)^{n'}/|\mathbb{F}|^{n'-n}$.*

Some concrete examples of this bound for $|\mathbb{F}| = 2^{256}$ are given below:

q	n	n'	bound
2^{107}	2^{10}	$8n+4$	2^{-128}
2^{115}	2^{10}	$16n+8$	2^{-128}
2^{102}	2^{20}	$8n+4$	2^{-128}
2^{110}	2^{20}	$16n+8$	2^{-128}

For example, when running the protocol for $n = 2^{10}$ items, the adversary will not be able to have an effective input of size $8n + 4$, with high probability.

We emphasize that the above bound is unconditional, meaning that for the parameters above, such an "overfitting" polynomial simply does not exist except with negligible probability. It seems reasonable to conjecture that even when such polynomials exist, finding them is hard for PPT adversaries. If such a claim were proven, it would imply a tighter enforcement of set sizes in our protocol.

We also emphasize that all malicious PSI protocols based on OT extension have similar "slack" in the size of corrupt parties' sets. In [RR17a] a bound of $n' = 6n$ is stated; in [RR17b] a bound of $n' = 4n$ is stated; and in [PRTY20] a range of bounds $n' = 2.4cn$ for $c \in \{2, 3, 4, 5\}$ is given for different parameters.

5 Experimental Results

5.1 Implementation

In order to evaluate the performance of our PSI protocol, we built and evaluated an implementation. Our complete implementation is available on GitHub: `https://github.com/osu-crypto/MiniPSI`. Below, we discuss how the various components were instantiated.

Key Agreement. We instantiate DHKA using elliptic curve groups, and hash g^{ab} with SHA2. As mentioned previously, this variant of DHKA is non-malleable (Definition 3) under the ODH assumption.

An elliptic curve consists of the solutions (x, y) in a field $\mathbb{F}_q$ to the Weierstrass equation $y^2 = x^3 + Ax^2 + B$ or Montgomery equation $y^2 = x^3 + Ax^2 + x$. Depending on the curve parameters, EC shows different shapes on the plane. In this work, we choose the Curve25519 Montgomery curve, since it is recommended for elligator [imp]. This Curve25519 is defined over $GF(q = 2^{255} - 19)$ and its curve parameter A has the value 486662.

We implement the elligator encoding based on [BHKL13]. The encoding takes a curve point and outputs a pseudorandom string of 256 bits. The point (x, y) has an inverse map if it satisfies two conditions: the x

n	Protocol	Sec.	Comm. (KB)	Running time (milliseconds) 10 Gbps Offline	10 Gbps Online	10 Gbps Total	50 Mbps Offline	50 Mbps Online	50 Mbps Total
2^7	Classic DH [HFH99]	SH	9.09	—	81.1	81.1	—	241.1	241.1
	KKRT [KKRT16]		22.22	180.1	21.2	201.3	339.0	499.1	838.1
	Ours		**4.99**	29.2	29.3	**58.5**	29.2	172	**201.2**
	SpOT-low [PRTY19]	1M	26.70	139.5	24.9	164.4	570.9	185.6	756.5
	CM [CM20]		32.00	203.9	32.3	236.2	554.0	349.2	903.2
	DKT [DKT10]	2M	31.48	—	492.0	492.0	—	1918.8	1918.8
	PaXoS [PRTY20]		40.96	250.2	34.8	285	665.2	536.1	1201.3
	Ours		**8.19**	32	30.1	**62.1**	32.2	189.6	**221.8**
2^8	Classic DH [HFH99]	SH	18.18	—	149.8	149.8	—	321.7	321.7
	KKRT [KKRT16]		43.8	181.2	27.1	208.3	341.2	507.1	848.3
	Ours		**9.98**	57.4	59.1	**116.5**	58.1	212.5	**270.6**
	SpOT-low [PRTY19]	1M	33.90	138.8	59.2	198.0	565.7	216.8	782.5
	CM [CM20]		43.00	205.1	32.0	237.1	623.3	361.3	984.6
	DKT [DKT10]	2M	62.74	—	898.0	898.0	—	3081.8	3081.8
	PaXoS [PRTY20]		69.83	255.3	35.9	291.2	668.1	552.04	1220.14
	Ours		**16.38**	58.4	61.1	**119.5**	62.1	225.5	**287.6**
2^9	Classic DH [HFH99]	SH	36.86	—	248.2	248.2	—	430.3	430.3
	KKRT [KKRT16]		94.64	183.2	40.8	224.0	342.0	656.9	998.9
	Ours		**20.48**	96.3	110.0	**206.3**	106.2	268.9	**375.1**
	SpOT-low [PRTY19]	1M	48.40	139.0	116.8	255.8	571.0	266.7	837.7
	CM [CM20]		64.00	207.1	28.3	235.4	633.5	355.1	988.6
	DKT [DKT10]	2M	125.27	—	1720.0	1720.0	—	5966.2	5966.2
	PaXoS [PRTY20]		127.56	256.3	54.1	310.4	671.1	554.04	1225.14
	Ours		**32.77**	98.3	112.9	**211.2**	115.1	275.1	**390.2**
2^{10}	Classic DH [HFH99]	SH	73.73	—	375.2	375.2	—	574.2	574.2
	KKRT [KKRT16]		188.64	185.4	42.6	228.0	345.1	554.1	899.2
	Ours		**40.96**	149.1	252.4	401.5	155	379.7	**534.7**
	SpOT-low [PRTY19]	1M	77.20	140.0	239.6	379.6	570.5	358.2	928.7
	CM [CM20]		105.00	207.4	36.5	**243.9**	633.6	359.5	993.1
	DKT [DKT10]	2M	250.32	—	3028.0	3028.0	—	10111.2	10111.2
	PaXoS [PRTY20]		243.01	258.4	94.1	352.5	671.2	560.1	1231.3
	Ours		**65.54**	155.6	268.9	424.5	164	393.9	**557.9**

Table 2: Communication cost in KB and running time in milliseconds of PSI protocols on the set size n. "SH", "1M", and "2M" refer to semi-honest, 1-sided malicious and 2-sided malicious protocol, respectively. Cells with "—" denote setting not supported or program out of memory.

value is not equal to the curve parameter A; and $-2x(x+a)$ must be a square. Therefore, we keep sampling points until these conditions are hold. According [BHKL13, imp] and confirmed by our experiment, the success probability is $\frac{1}{2}$. The elligator encoding of such a valid point is defined by $r = \sqrt{(\frac{-1}{2})(\frac{x}{x+A})^b}$, where $b = 1$ if $v \leq \frac{q-1}{2}$, otherwise, $b = -1$.

The decoding function takes a string r and produces the x coordinate of a point on Curve25519. The value x can be computed as $x = ed - (1-e)\frac{A}{2}$, where $d = \frac{-A}{1+2r^2}$ and $e = (d^3 + Ad^2 + d)^{\frac{q-1}{2}}$.

We implemented elligator on top of the Curve25519 implementation from `libsodium`. From our experimental evaluation, `libsodium` is about 10× faster than `miracl` library.

The length of elligator encodings is slightly less than 256 bits. In order to promote these encodings to be uniform in $\{0,1\}^{256}$, we can append a few extra uniform bits which are ignored during decoding. These additional bits can be considered as part of the randomness in the KA protocol, and they cause the protocol messages to be pseudorandom in $\mathbb{F} = \{0,1\}^{256}$.

n_2	n_1	Protocol	Sec.	Comm. (MB)	Running time (seconds) 10 Gbps Online	10 Gbps Total	50 Mbps Online	50 Mbps Total	1 Mbps Online	1 Mbps Total
2^{12}	2^{12}	Classic DH [HFH99]	SH	0.29	0.86	0.86	1.13	1.13	2.26	2.26
		KKRT [KKRT16]		0.56	0.03	**0.2**	0.57	**0.91**	5.09	5.47
		Ours		**0.16**	0.59	1.07	0.91	1.39	1.71	**2.17**
		SpOT-low [PRTY19]	1M	0.25	0.72	0.88	1.04	1.61	2.79	3.36
		CM [CM20]		0.36	0.08	0.28	0.51	1.15	3.11	3.74
		DKT [DKT10]	2M	0.83	12.12	12.12	36.35	36.35	97.06	97.06
		PaXoS [PRTY20]		0.94	0.14	0.4	0.97	1.64	5.26	5.93
		Ours		**0.16**	0.62	1.08	0.95	**1.41**	1.75	**2.22**
	2^{8}	Classic DH [HFH99]	SH	0.17	0.48	0.48	0.83	0.83	1.24	1.24
		KKRT [KKRT16]		0.28	0.02	**0.2**	0.67	1.01	4.64	5.01
		Ours		**0.26**	0.41	0.47	0.59	**0.65**	0.59	**0.69**
		SpOT-low [PRTY19]	1M	0.24	0.65	0.79	0.33	0.89	0.38	0.95
		CM [CM20]		0.32	0.20	0.27	0.63	1.13	3.11	3.74
		Ours	2M	**0.14**	0.43	0.49	0.62	**0.66**	0.63	**0.67**
2^{16}	2^{16}	Classic DH [HFH99]	SH	4.78	10.38	11.58	17.6	17.6	38.53	38.53
		KKRT [KKRT16]		6.73	0.21	**0.44**	2.53	**2.92**	74.15	74.57
		Ours		**2.69**	8.96	16.25	10.9	16.64	25.03	**31.16**
		SpOT-low [PRTY19]	1M	3.9	12.61	12.81	15.76	16.33	40.15	40.71
		CM [CM20]		5.34	0.54	0.75	1.72	2.35	45.11	45.75
		DKT [DKT10]	2M	13.33	216.83	216.83	845.63	845.63	1929.76	1929.76
		PaXoS [PRTY20]		14.79	0.25	0.52	4.27	4.95	48.34	49.02
		Ours		**4.19**	8.89	15.95	11.1	18.64	27.73	**36.16**
	2^{12}	Classic DH [HFH99]	SH	2.82	6.45	6.45	14.01	14.01	22.37	22.37
		KKRT [KKRT16]		4.55	0.11	**0.32**	1.59	1.92	39.7	40.02
		Ours		**0.72**	4.71	5.17	5.91	6.37	7.10	7.46
		SpOT-low [PRTY19]	1M	3.42	1.55	1.75	1.34	**1.91**	6.10	**6.67**
		CM [CM20]		4.82	0.50	0.70	1.37	2.01	40.38	41.01
		Ours	2M	**2.23**	4.71	5.97	6.01	7.47	7.90	8.96
2^{20}	2^{20}	Classic DH [HFH99]	SH	77.59	189.87	189.87	290.82	290.82	717.08	717.08
		KKRT [KKRT16]		133.00	3.51	**4.18**	27.42	27.5	1153.23	1154
		Ours		**44.04**	144.64	245.06	150.93	251.69	452.7	**554.26**
		SpOT-low [PRTY19]	1M	63.18	270.69	270.88	310.83	311.4	687.77	688.34
		CM [CM20]		86.16	7.94	8.15	16.56	**17.17**	726.81	727.46
		DKT [DKT10]	2M	213.00	5121	5121	—	—	—	—
		PaXoS [PRTY20]		236.47	5.01	5.29	46.13	46.81	798.26	798.94
		Ours		**67.11**	148.94	251.06	161.93	267.69	489.7	**597.26**
	2^{16}	Classic DH [HFH99]	SH	46.14	104.57	104.57	170.82	170.82	371.77	371.77
		KKRT [KKRT16]		74.20	1.86	**2.32**	17.5	18.25	609.49	610.25
		Ours		**12.58**	92.5	98.24	104.44	108.92	109.41	**117.81**
		SpOT-low [PRTY19]	1M	55.53	218.65	218.85	15.82	16.39	128.43	129.10
		CM [CM20]		76.77	7.50	7.70	15.66	**16.26**	721.81	722.45
		Ours	2M	**35.65**	94.1	99.91	105.44	113.02	120.49	**121.81**

Table 3: Communication cost in MB and running time in seconds of PSI protocols; the sender and receiver set size is n_1 and n_2, respectively. "SH", "1M", and "2M" refer to semi-honest, 1-sided malicious and 2-sided malicious protocol, respectively. Cells with "—" denote setting not supported or program out of memory. [DKT10] and [PRTY20] implementations do not support sets of different sizes.

Other Primitives. We instantiate the necessary random oracles using SHA2. Since the elliptic curves have 256-bit encodings, we need an ideal permutation $\Pi^{\pm}$ defined over $\{0,1\}^{256}$. In our implementation we use Rijndal-256 with a fixed key as the ideal permutation.

Polynomial Operations. Our protocol requires the receiver to generate a polynomial of degree n, and the sender to evaluate it on n points. It is known that these problems could be solved by Lagrange interpolation and Horner evaluation which requires $O(n^2)$ field operations. However, when n is very large (e.g. $n = 2^{20}$) this becomes impractical. Moenck and Borodin [MB72] describe algorithms for these problems in $O(n \log^2(n))$ field operations, which make them a better fit for our protocol.

Security Parameters. All evaluations were performed with a PSI item length of 128 bits, computational security parameter $\kappa = 128$ bits, and a statistical security parameter $\lambda = 40$ bits.

5.2 Experiments and Evaluation

Experimental Setup. We implement our protocol in C++, and run our protocol on a single Intel Xeon with 2.30GHz and 256GB RAM. The parties communicate over a simulated 10Gbps network with 0.2ms round-trip time for LAN setting. We also run all protocols in WAN setting with 80ms round-trip time and two different network bandwidths 50 Mbps, and 1 Mbps.

Protocol Evaluation. In the following, we benchmark the state of the art semi-honest and malicious PSI protocols [HFH99, DKT10, KKRT16, PRTY19, CM20, PRTY20]. We now briefly discuss several protocols not included in our comparison: The Jarecki-Liu protocol [JL10] is a malicious-secure, DH-based protocol. However, it achieves a weaker ideal functionality where the adversary can choose items adaptively. The recent PSI protocol of Rindal & Schoppmann [RS21] is based on silent vector-OLE, and is extremely efficient for large sets. However, its implementation is not yet publicly available and its high fixed costs make it inefficient for small sets (as illustrated in Table 1). The work of Chen *et al.* [CHLR18] is the state-of-the-art (one-sided) malicious FHE-based PSI. Its first step is essentially classic DH-PSI, before even doing any FHE operations. Since our entire protocol is more efficient than DH-PSI, we expect ours would be much faster than theirs for small-to-medium-size sets.

We also do not include RSA-based PSI protocols [DT10, ADT11], by which we mean protocols that require at least one RSA exponentiation per item. RSA elements are $16\times$ ($= 4096/256$) larger than elliptic curve (ECC) elements. A simple benchmark on our experimental hardware (`openssl speed rsa4096 ecdhx25519`) shows that RSA-4096 exponentiation is $100\times$ slower than ECC exponentiation (even RSA-2048 was $20\times$ slower). Therefore, RSA-based protocols will always be $\sim 100\times$ slower than ours. If they send one RSA value per item, they will have $16\times$ more communication than ours.

We report detailed comparisons in Table 2 and Table 3 for small set size $\{2^7, 2^8, 2^9, 2^{10}\}$ and large set size $n \in \{2^{12}, 2^{16}, 2^{20}\}$. As expected, our protocol shows a significant performance improvement when the set is small.

We note that our poly-DH PSI protocol is very amenable to precomputation (by precomputing exponentiation). When reporting performance of these protocols, we split total running time into two phases:

- **Offline:** operations like generating random pairs (r_i, g^{r_i}), which can be done without any interaction and before the inputs are known.
- **Online:** everything else, starting when the parties have determined their inputs.

Bandwidth Comparison. Our polynomial-based protocol requires the lowest communication among all PSI protocols. The communication of our polynomial-based protocol is approximately $2\times$ smaller than that of classic DH PSI. Compared to malicious DH-based PSI protocol [DKT10] (DKT), our protocol shows about $3-4\times$ improvement.

Consider a semi-honest PSI with unequal set size, the communication cost is $(n_1|\mathbb{G}| + n_2\ell)$ bits for the polynomial-based PSI protocol, and about $(n_1 + n_2)|\mathbb{G}| + n_2\ell)$ bits for classic DH-based PSI. Concretely, for $n_1 = 2^{16}$ and $n_2 = 2^{20}$, the polynomial-based protocol takes 12.58 MB of communication while classic DH PSI needs 46.14 MB, a $3.67\times$ improvement.

We also compare bandwidth to the state-of-the-art OT-based semi-honest PSI protocols [KKRT16, PRTY19, CM20] and malicious PSI protocol [PRTY20]. Note that [KKRT16] (KKRT), [PRTY20](PaXoS) are the fastest PSI protocol to date and [CM20] (CM) has the fastest in networks with moderate bandwidth

(e.g., 30-100 Mbps) while [PRTY19] (SpOT-low) has the least communication among practical semi-honest protocols. The communication cost of our protocol is about $3-4.6\times$, $1.4-1.7\times$, and $3.7-7.8\times$ less than that of [KKRT16], [PRTY19], and [PRTY20], respectively.

Runtime Comparison. For small set (e.g $n = 2^9$), our polynomial-based protocol is faster than all DH-based and OT-based schemes in both LAN and WAN settings. Starting from $n = 2^{10}$, our protocol is slower than the OT-based protocols in LAN setting. However, bench-marking all protocols in the WAN setting with 1 Mbps network bandwidth and 80 ms round-trip latency, our protocol shows an $1-3.17\times$ faster than others due to the fact that the communication cost is smallest.

The polynomial-based protocol shows its benefit in the unbalanced setting where the sender's set size is larger than the receiver's set size ($n_2 > n_1$). It means that the sender only needs to send the receiver a short fingerprint ℓ per each item in his set while in DH-based protocol the sender additional requires to send a group element per each item. Since the implementation of PaXoS and DKT does not support to compute a PSI for asymmetric set, we omit to report their performance costs. Table 3 shows that in most of the cases the running time of our polynomial-based protocol is faster than other semi-honest protocols. Consequently, our protocol is faster than other malicious protocols. For $n_1 = 2^{16}$ and $n_2 = 2^{20}$ in WAN setting with 1Mpbs bandwidth, the baseline DH protocol runs in 574.26 seconds, while the polynomial-based protocol requires 117.81 seconds, a factor of $4.9\times$ and $3.1\times$ improvement, respectively.

A summary of the state of the art (including this work) is presented in Figure 1 where the running time is measured in the LAN setting. Our PSI prototocol's performance is mostly unaffected by changing the network bandwidth and latency, due to its extremely low communication complexity.

Conclusions. For small sets ($n \leq 512$) our protocol is the best in terms of both communication and computation. As we previously discussed in Section 1, on sets of this size our protocol is less expensive than the base OTs required for OT extension and PSI protocols that are based on OTs.

Acknowledgements

The first author is partially supported by a Facebook research award. The second author is partially supported by NSF awards #2031799, #2101052, and #2115075. We are grateful to the CCS 2021 anonymous reviewers whose feedback was instrumental in improving several aspects of this paper.

References

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, Heidelberg, April 2001.

[ADT11] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) size matters: Size-hiding private set intersection. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 156–173. Springer, Heidelberg, March 2011.

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

[BFT16] Tatiana Bradley, Sky Faber, and Gene Tsudik. Bounded size-hiding private set intersection. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 449–467. Springer, Heidelberg, August / September 2016.

[BHKL13] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 967–980. ACM Press, November 2013.

[BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy*, pages 478–492. IEEE Computer Society Press, May 2013.

[CDCS18] Andrea Cerulli, Emiliano De Cristofaro, and Claudio Soriente. Nothing refreshes like a repsi: Reactive private set intersection. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security*, pages 280–300, Cham, 2018. Springer International Publishing.

[CDJ16] Chongwon Cho, Dana Dachman-Soled, and Stanislaw Jarecki. Efficient concurrent covert computation of string equality and set intersection. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 164–179. Springer, Heidelberg, February / March 2016.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Heidelberg, August 2020.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 789–800. ACM Press, November 2013.

[DD15] Sumit Kumar Debnath and Ratna Dutta. Secure and efficient private set intersection cardinality using bloom filter. In Javier Lopez and Chris J. Mitchell, editors, *ISC 2015*, volume 9290 of *LNCS*, pages 209–226. Springer, Heidelberg, September 2015.

[DGT12] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *CANS 12*, volume 7712 of *LNCS*, pages 218–231. Springer, Heidelberg, December 2012.

[DKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, Heidelberg, December 2010.

[DMRY09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 125–142. Springer, Heidelberg, June 2009.

[DRRT18] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. Pir-psi: Scaling private contact discovery. *Proceedings on Privacy Enhancing Technologies*, 2018(4):159 – 178, 2018.

[DT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, Heidelberg, January 2010.

[FMV19] Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 111–130. Springer, Heidelberg, December 2019.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy*, pages 825–841. IEEE Computer Society Press, May 2020.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 395–425, Virtual Event, August 2021. Springer, Heidelberg.

[Haz18] Carmit Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. *J. Cryptol.*, 31(2):537–586, April 2018.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

[HFH99] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *ACM CONFERENCE ON ELECTRONIC COMMERCE*. ACM, 1999.

[HHS+21] Alexander Heinrich, Matthias Hollick, Thomas Schneider, Milan Stute, and Christian Weinert. Privatedrop: Practical privacy-preserving authentication for apple airdrop. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[HN10] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 312–331. Springer, Heidelberg, May 2010.

[IKN+17] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017. https://eprint.iacr.org/2017/738.

[IKN+19] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. On deploying secure computing commercially: Private intersection-sum protocols and their business applications. Cryptology ePrint Archive, Report 2019/723, 2019. https://eprint.iacr.org/2019/723.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[imp] https://www.imperialviolet.org/2013/12/25/elligator.html.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, Heidelberg, March 2009.

[JL10] Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 418–435. Springer, Heidelberg, September 2010.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005.

[MB72] R. Moenck and Allan Borodin. Fast modular transforms via division. In *Switching and Automata Theory*, pages 90–96, 1972.

[Mea86] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134, April 1986.

[MPGP09] Ghita Mezzour, Adrian Perrig, Virgil D. Gligor, and Panos Papadimitratos. Privacy-preserving relationship path discovery in social networks. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 189–208. Springer, Heidelberg, December 2009.

[MPP10] Mark Manulis, Benny Pinkas, and Bertram Poettering. Privacy-preserving group discovery with linear complexity. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 420–437. Springer, Heidelberg, June 2010.

[MR19] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 309–326. ACM Press, November 2019.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 515–530. USENIX Association, 2015.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 797–812. USENIX Association, 2014.

[RR17a] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 235–259. Springer, Heidelberg, April / May 2017.

[RR17b] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017.

[RS21] Peter Rindal and Phillipp Schoppmann. Vole-psi: Fast oprf and circuit-psi from vector-ole. Cryptology ePrint Archive, Report 2021/266, 2021. https://eprint.iacr.org/2021/266.

A Semi-Honest Variants

In this section we show two semi-honest variants of our protocol.

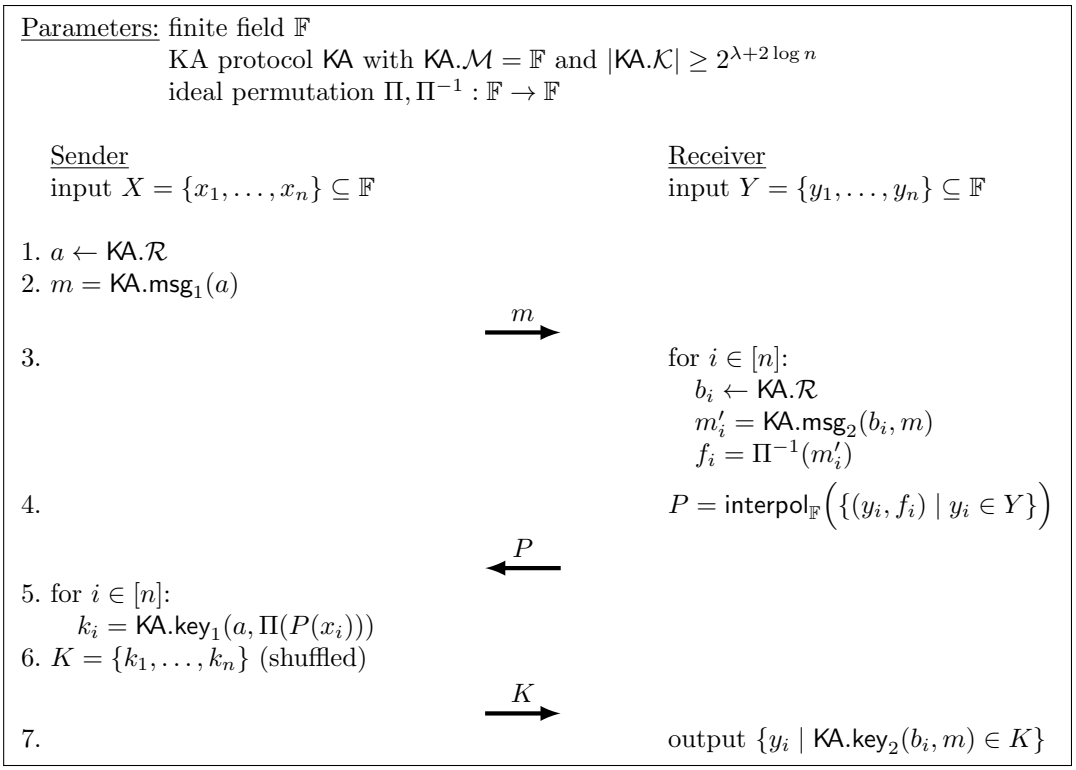

Figure 5: Semi-honest variant #1 of our protocol.

A.1 Variant 1

The first protocol variant closely matches the malicious variant. The only changes are:

- The polynomial can be interpolated on values $P(y_i)$ instead of $P(H_1(y_i))$; H_1 was used only to help extract a malicious party's input, which is not a concern in the semi-honest model.
- Instead of sending values of the form $H_2(x_i, k_i)$, the sender can simply send the k_i values. Again, H_2 was used only to extract. Furthermore, the k_i values can have length of only $\lambda + 2\log(n)$ in order to ensure correctness with probability $1 - 2^{-\lambda}$.

The details of this protocol are given in Figure 5. The correctness of the protocol boils down to the following observations:

- Suppose $x_i = y_j$ for some i, j (*i.e.*, x_i is an item of the sender that is in the intersection). Then by construction we have:

$$\begin{aligned}\mathsf{KA}.\mathsf{key}_1(a, \Pi(P(x_i))) &= \mathsf{KA}.\mathsf{key}_1(a, \Pi(f_j)) \\ &= \mathsf{KA}.\mathsf{key}_1(a, m'_j) \\ &= \mathsf{KA}.\mathsf{key}_1(a, \mathsf{KA}.\mathsf{msg}_2(b_j, m))\end{aligned}$$

  By the correctness of the KA protocol, this is equal to $\mathsf{KA}.\mathsf{key}_2(b_j, m)$, and the receiver will indeed include y_j $(= x_i)$ in the output.

- Suppose $x_i \notin Y$. In this case, our security proof will argue that the corresponding k_i value (computed by the sender) is pseudorandom. The receiver only produces incorrect output if this k_i happens to match one of the $\mathsf{KA}.\mathsf{key}_2(b_j, m)$ values computed by the receiver. For this particular k_i, this event happens with probability (negligibly close to) $n/|\mathsf{KA}.\mathcal{K}|$. With a union bound over at most n such k_i values, the overall probability of incorrect output is at most (negligibly close to) $n^2/|\mathsf{KA}.\mathcal{K}|$.

To limit the correctness error to a concrete value $2^{-\lambda}$ (for example, $\lambda = 40$ in our implementation), it suffices to use a KA protocol with $|\mathsf{KA}.\mathcal{K}| \geq 2^{\lambda+2\log n}$.

Lemma 11. *The protocol of Figure 5 is secure against a semi-honest sender, if* KA *is a pseudorandom-message KA (Definition 4) and* $\Pi^{\pm}$ *is an ideal permutation.*

Proof. Since the only protocol message from the receiver is P, it suffices to show how to simulate P. In fact, we simply show that P is indistinguishable from a polynomial of appropriate degree with coefficients chosen uniformly in $\mathbb{F}$.

First, consider replacing "$b_i \leftarrow \mathsf{KA}.\mathcal{R}; m'_i = \mathsf{KA}.\mathsf{msg}_2(b_i, m)$" in step 3 with "$m'_i \leftarrow \mathsf{KA}.\mathcal{M}\ (= \mathbb{F})$". This change is indistinguishable to the sender by the pseudorandom-message property of KA. Then, since Π is a permutation, we see that f_i becomes uniformly distributed on $\mathbb{F}$. Finally, interpolating a polynomial on a set of points $\{(y_i, f_i)\}$, where each f_i is uniform in $\mathbb{F}$, results in a uniformly chosen polynomial, independent of the y_i values. □

Lemma 12. *The protocol of Figure 5 is secure against a semi-honest receiver, if* KA *is a secure KA (Definition 2) and* $\Pi^{\pm}$ *is an ideal permutation.*

Proof. First, we discuss the intuition of the proof. For each $x_i \in X$, the sender interprets $\Pi(P(x_i))$ as a KA protocol message. When $x_i \in X \setminus Y$, the receiver never actively chooses the value at $P(x_i)$, and so presumably does not know the secret randomness of the KA message $\Pi(P(x_i))$. From the receiver's perspective, this is just like watching a KA instance between two external parties, so the resulting key k_i should look random. We can formalize this by having the simulator *program* $\Pi(P(x_i))$ to be a KA protocol message whose underlying randomness is explicitly unknown to the receiver.

Hybrid 0: The real interaction, where both parties run the protocol honestly on inputs X and Y, and the ideal permutation $\Pi^{\pm}$ is simulated honestly.

Hybrid 1: Same as the real interaction, except we abort if there exists $x^* \in X \setminus Y$ and $y^* \in Y$ such that $P(x^*) = P(y^*)$. Note that P is distributed indistinguishably from a uniform polynomial, independent of any X and Y values. Given a fixed $x^* \neq y^*$, the probability that $P(x^*) = P(y^*)$ for a random polynomial is $1/|\mathbb{F}|$. By a union bound over all choices of x^* and y^*, the probability of an abort in this hybrid is at most $n^2/|\mathbb{F}|$, which is negligible. Hence, the hybrids are indistinguishable.

Hybrid 2: Same as Hybrid 1, except in how the ideal permutation is simulated. Without loss of generality, the adversary does not make any $\Pi^{\pm}$ queries until after the protocol transcript is generated. This hybrid generates the polynomial P as in the real protocol. Then, for each $x_i \in X \setminus Y$, we know that there has been no query to $\Pi(P(x_i))$ so far — otherwise we would already have aborted, since this means $P(x_i)$ collides with some $P(y_j)$. The hybrid chooses $r_i \leftarrow \mathsf{KA}.\mathsf{key}_2$ and programs $\Pi(P(x_i))$ to be equal to $\mathsf{KA}.\mathsf{msg}_2(r_i, m)$.

The hybrids are indistinguishable because honestly generated $\mathsf{KA}.\mathsf{msg}_2$ values are indistinguishable from random. Note that in this hybrid, every k_i that the sender computes is in response to a $\mathsf{KA}.\mathsf{msg}_2$ message *with randomness known to the hybrid.* For $x \in X \cap Y$, the sender responds to a KA message explicitly programmed into P. For $x \in X \setminus Y$, the sender responds to a value chosen above while programming Π.

Hybrid 3: Same as Hybrid 2, except how the k_i values are computed:

for $i \in [n]$:
  if $x_i \in Y$, *i.e.*, $x_i = y_j$ for some $y_j \in Y$:
    $k_i = \mathsf{KA}.\mathsf{key}_2(b_j, m)$
  else:
    $k_i = \mathsf{KA}.\mathsf{key}_2(r_i, m)$

We have simply replaced all $\mathsf{KA}.\mathsf{key}_1(a, \cdot)$ computations of the sender with the corresponding $\mathsf{KA}.\mathsf{key}_2$ computations. The hybrids are identically distributed by the correctness of KA.

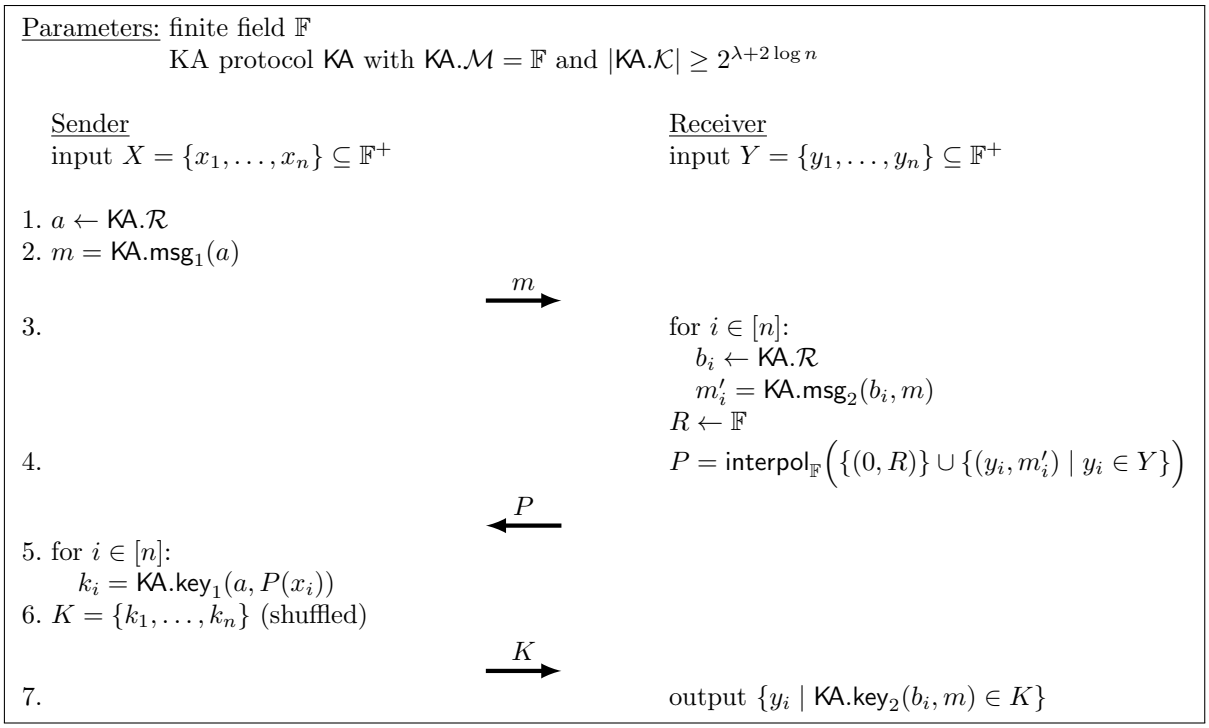

Figure 6: Semi-honest variant #2 of our protocol.

Hybrid $(4, h)$*, for* $h \in [n+1]$*:* Same as Hybrid 3, except that K is computed as follows:

for $i \in [n]$:
  if $x_i \in Y$, *i.e.*, $x_i = y_j$ for some $y_j \in Y$:
    $k_i = \mathsf{KA.key}_2(b_j, m)$
  else if $i < h$:
    $k_i \leftarrow \mathsf{KA}.\mathcal{K}$
  else:
    $k_i = \mathsf{KA.key}_2(r_i, m)$

Clearly if $h = 0$ then the new if-clause is not reachable and the hybrid is identical to Hybrid 3. The difference between Hybrids $(4, h-1)$ and $(4, h)$ is the treatment of the value k_h when $x_h \notin Y$. It is straightforward to reduce this change of hybrids to the standard security of KA (Definition 2) as follows: Given (m_1, m_2, k^*), use $m = m_1$ as the sender's first message, program $\Pi(P(x_h)) = m_2$, and set $k_h = k^*$. If k^* is a real KA key then we get Hybrid $(4, h-1)$, and if it is a random key then we get Hybrid $(4, h)$. Hence, the hybrids are indistinguishable.

Simulator: We rewrite the computation of the K-values in Hybrid $(4, n+1)$ in an equivalent way. Note that in this hybrid, the the final else-branch is not reachable. Hence the set K is generated as follows:

$$K = \{\mathsf{KA.key}_2(b_j, m) \mid y_j \in X \cap Y\} \cup \{k'_1, \ldots, k'_{|X \setminus Y|}\}$$

where each k'_i is chosen uniformly. This interaction is distributed identically to Hybrid $(4, n+1)$, and clearly it can be carried out with only knowledge of $X \cap Y$, Y and $|X \setminus Y|$ (which can be inferred from $|X|$). Hence, this hybrid defines our final simulator. □

A.2 Variant #2

We also describe a semi-honest variant that does not require an ideal permutation. However, the protocol requires the receiver to send 1 additional field element, and it requires the KA protocol to satisfy the stronger non-malleability property (Definition 3 — the same property used for the malicious protocol variant).

Recall how the ideal permutation was used in the previous semi-honest proof. While the receiver interpolated the polynomial P so that $\Pi(P(y_i))$ equals chosen KA messages, the simulator in the hybrid proof used Π to program other outputs $\Pi(P(x))$ for $x \notin Y$. These other outputs need to be "programmed" in order to reduce security to that of KA.

Now consider an alternative approach that we take in this section. There is no longer any ideal permutation, so the receiver interpolates P such that $P(y_i)$ is a chosen KA message. The receiver also chooses a random $R \leftarrow \mathbb{F}$ and ensures that $P(0) = R$ (we assume that 0 is not a valid item for the PSI inputs). Now the degree of the polynomial is one larger than otherwise, and the value R becomes part of the receiver's protocol view. The simulator can "program" the output of $P(x^*)$ for some $x^* \notin Y$ by interpolating over the y_i values and x^*, instead of the y_i values and zero. Then it can simply solve for the correct $R = P(0)$ to include in the receiver's simulated view.

With this trick, the hybrid simulators can change one KA output at a time. However, while it replaces one KA output with random, the simulator still needs to simulate the other KA outputs — *i.e.*, $\mathsf{KA.key}_1(a, P(x_i))$ for $x_i \notin Y$. Unlike in the previous protocol variant, here we cannot guarantee that $P(x_i)$ is an honestly-generated KA message. There is no "other way" (*i.e.*, using $\mathsf{KA.key}_2$) to compute this KA output. Hence, we must give the simulator oracle access to $\mathsf{KA.key}_1(a, \cdot)$, just like in the security proof of the malicious variant.

The details of the protocol are given in Figure 6. Security against a semi-honest sender follows from the same reasoning as the previous semi-honest variant. We focus on security against semi-honest receiver.

Lemma 13. *The protocol of Figure 6 is secure against a semi-honest receiver, if KA is non-malleable (Definition 3) and has pseudorandom second messages (Definition 4).*

Proof. We prove security in a sequence of hybrids. Hybrid 0 is the real protocol interaction.

Hybrid 1: Same as the real interaction, except we abort if there exists $x^* \in X \setminus Y$ and $y^* \in Y$ such that $P(x^*) = P(y^*)$. This hybrid is indistinguishable from Hybrid 0, following the same logic as in the previous proof.

Hybrid $(2, h)$ for $h \in [n]$: Same as Hybrid 1, except in the way the k_i values are computed:

$$
\begin{array}{l}
\text{for } i \in [n]\text{:} \\
\quad \text{if } x_i \in X \cap Y \text{ or } i > h\text{:} \\
\quad\quad k_i = \mathsf{KA.key}_1(a, P(x_i)) \\
\quad \text{else:} \\
\quad\quad k_i \leftarrow \mathsf{KA}.\mathcal{K}
\end{array}
$$

Since the if-branch is always taken when $h = 0$, Hybrid $(2, 0)$ is identical to Hybrid 1. Below, we prove that Hybrids $(2, h-1)$ and $(2, h)$ are indistinguishable for all h.

Simulation: Hybrid $(2, n)$ is our final simulation. Note that Hybrid $(2, n)$ is equivalent to one where the k_i values are computed as:

$$
\begin{array}{l}
\text{for } i \in [n]\text{:} \\
\quad \text{if } x_i \in X \cap Y\text{:} \\
\quad\quad k_i = \mathsf{KA.key}_1(a, P(x_i)) \\
\quad \text{else:} \\
\quad\quad k_i \leftarrow \mathsf{KA}.\mathcal{K}
\end{array}
$$

Hence, these k_i values can be generated knowing only $X \cap Y$ and $|X \setminus Y|$. □

Lemma 14. *Hybrids $(2, h-1)$ and $(2, h)$, defined above, are indistinguishable when KA is non-malleable and has pseudorandom second messages.*

Proof. Fix h and note that the hybrids differ only in how k_h is generated, in the case that $x_h \notin X \cap Y$. The lemma is trivial in the case that $x_h \in X \cap Y$, so hereafter we assume $x_h \notin X \cap Y$. To prove the lemma, we

define the following reduction algorithm:

$\mathcal{A}^{\mathbb{K}}(m_1, m_2, k^*; X, Y, x_h)$:
  for $i \in [n]$:
    $b_i \leftarrow \mathsf{KA}.\mathcal{R}$
    $m'_i = \mathsf{KA.msg}_1(b_i, m_1)$
  $P = \mathsf{interpol}_{\mathbb{F}}(\{(x_h, m_2)\} \cup \{(y_i, m'_i) \mid y_i \in Y\})$
  $R = P(0)$
  abort if $\exists x^* \in X \setminus Y, y^* \in Y$ with $P(x^*) = P(y^*)$
  for $i \in [n]$:
    if $x_i \in X \cap Y$ or $i > h - 1$:
      $k_i = \mathbb{K}(P(x_i))$
    else if $i = h$ and $x_h \notin X \cap Y$
      $k_i = k_h = k^*$
    else:
      $k_i \leftarrow \mathsf{KA}.\mathcal{K}$
  return $(m_1, K = \{k_1, \ldots, k_n\}; \{b_1, \ldots, b_n\}, R)$

Intuitively, this algorithm receives a purported KA transcript (m_1, m_2, k^*) as input, and generates a simulated view by letting m_1 be the sender's PSI message, programming P so that $P(x_h) = m_2$ and letting k^* be the value k_h that the sender sends corresponding to item x_h. It also has access to an oracle $\mathbb{K}$ which it uses to generate other k_i values.

Claim: When this reduction algorithm is run as distinguisher in the "real" experiment from Definition 3, it outputs a view which is indistinguishable from Hybrid $(2, h-1)$. Indeed, m_1 is generated as $\mathsf{KA.msg}_1(a)$, k_h is generated as $\mathsf{KA.key}_1(a, P(x_h)) = \mathsf{KA.key}_1(a, m_2)$, and the other k_i values are generated correctly using the $\mathbb{K}(\cdot) = \mathsf{KA.key}_1(a, \cdot)$ oracle. All of these values are as in Hybrid $(2, h-1)$. The only difference is how P is generated — instead of interpolating P over $\{(0, R)\} \cup \{(y_i, m'_i)\}$ values, here we interpolate over $\{(x_h, m_2)\} \cup \{(y_i, m'_i)\}$ values. But since m_2 is pseudorandom, the two ways of computing P are indistinguishable. Note also that $\mathbb{K}$ is called only on values of the form $\mathbb{K}(P(x_i))$ for $x_i \in Y$. But no $y \in Y$ has $P(y) = P(x_h) = m_2$, since we would already have aborted in that case. Hence the $\mathbb{K}$ oracle is never queried at m_2, as required in Definition 3.

Claim: When this reduction algorithm is run as distinguisher in the "random" experiment from Definition 3, it outputs a view which is indistinguishable from Hybrid $(2, h)$. The logic here is almost identical, except now $k_h = k^*$ is random, as in Hybrid $(2, h)$ □

B Optimizations

For all of these optimizations, we leave it as an exercise for the reader to verify that the security proofs hold when using the optimizations.

Elligator. Our protocol requires a KA protocol whose *second* message is pseudorandom, since only the second KA message is encoded into a polynomial. Elligator-DHKA requires parties to re-sample randomness until they "hit" the elligator subset of the elliptic curve. Only the receiver needs to do this in our PSI protocol; the sender does not need to use elligator encodings for their KA message.

Alternatives to Polynomials. Our PSI protocol requires the receiver to interpolate a polynomial over n points, and the sender to evaluate that polynomial on n points, where n is the size of their sets (*e.g.*, $n = 1\text{M}$). Each of these procedures cost $O(n \log^2 n)$ field operations.

One way to reduce the cost of this step is to encode the same information in a different way. The purpose of P is to convey mappings of the form $y_i \mapsto f_i$ in a way that hides the y_i values. Concurrent to this work, Garimella *et al.* [GPR+21] introduced *oblivious key-value stores (OKVS)*, which are an abstraction that provides the properties that our protocol requires. They present an efficient OKVS alternative to polynomials that has linear encoding time, but at a small (∼35%) increase in communication size. This data

structure can be used in our protocol to replace polynomials, however: (1) Polynomial interpolation over small (degree < 1000) polynomials is a very small contribution to the protocol's overall cost (even using a simpler quadratic algorithm) compared to the cost of elliptic curve exponentiations. (2) Even a 35% increase in size significantly undermines our protocol's contribution of minimal communication cost.

C Polynomial Overfitting

Recall the polynomial overfitting game $\mathsf{PolyOverfit}_{\mathbb{F}}^{n,n'}(q)$:

sample $\alpha_1, \ldots, \alpha_q \leftarrow \mathbb{F}$
sample $\beta, \ldots, \beta_q \leftarrow \mathbb{F}$
give $\{\alpha_1, \ldots, \alpha_q\}$ and $\{\beta_1, \ldots, \beta_q\}$ to $\mathcal{A}$
$\mathcal{A}$ outputs a polynomial P
if $0 < \deg(P) < n$ and $P(\alpha_i) \in \{\beta_1, \ldots, \beta_q\}$ for at least n' distinct α_i:
  $\mathcal{A}$ wins the game
else $\mathcal{A}$ loses the game

We prove an unconditional bound for winning this game, based on a compression argument.

Proposition 15. *Let $E : A \to B$ and $D : B \to A$ be functions. Then $\Pr_{a \leftarrow A}[D(E(a)) = a] \leq |B|/|A|$.*

Proof. If $a \notin \mathsf{range}(D)$, then we can never have $D(E(a)) = a$. Furthermore, $|\mathsf{range}(D)| \leq |B|$. □

Lemma 16. *The probability of any (computationally unbounded) adversary winning $\mathsf{PolyOverfit}_{\mathbb{F}}^{n,n'}(q)$ is at most*

$$(q^2 n)^{n'} / |\mathbb{F}|^{n'-n}$$

Proof. Let $\mathcal{A}$ be an adversary that wins the game with probability ϵ. Using $\mathcal{A}$ we can compress a list $(\alpha_1, \ldots, \alpha_q, \beta_1, \ldots, \beta_q)$ by giving the following information (in this order):

- the output polynomial $P \leftarrow \mathcal{A}(\alpha_1, \ldots, \alpha_1, \beta_1, \ldots, \beta_q)$
- a bipartite graph G with left and right vertex sets $[q]$, left-degree 1, and an edge from left vertex i to right vertex j if $P(\alpha_i) = \beta_j$
- for every connected component in G (in some canonical order):
  - If the component is a singleton right vertex i, give β_i
  - If the component contains a left vertex, let i be the lowest numbered left vertex in the component, and give α_i. Then for every left vertex i except the lowest numbered one, give an index v such that α_i is the vth root of $P(\cdot) - P(\alpha_i)$ in lexicographic order.

Recovering the α_i and β_i inputs from this information amounts to labeling each vertex in G with the appropriate α_i or β_j, which can be done in a straight-forward way.

Note that the graph G has $2q$ vertices, and if it has e edges then it has $c = 2q - e$ connected components. The number of possible "compressed encodings" is at most the product of the following terms:

- $|\mathbb{F}|^n$, for the number of polynomials P
- $(q^2)^e$, for (an upper bound on) the number of bipartite graphs with $q + q$ vertices and e edges.
- $|\mathbb{F}|^c$, for listing one α_i or β_j per component
- n^e, for the other information in each connected component — there are at most e left vertices in nontrivial connected components, and each index v names one of the n roots of a deg $< n$ polynomial.

Hence, the number of such encodings is bounded by:

$$|\mathbb{F}|^n \cdot q^{2e} \cdot |\mathbb{F}|^{2q-e} \cdot n^e = |\mathbb{F}|^{2q+n} \cdot \left(\frac{q^2 n}{|\mathbb{F}|}\right)^e$$

Assume that the quantity in parentheses is less than 1, since if it is not then the probability bound in the statement of the lemma exceeds 1 and is therefore trivial. When $\mathcal{A}$ wins the game, then $e \geq n'$ and the number of encodings is bounded by:

$$|\mathbb{F}|^{2q+n} \cdot \left(\frac{q^2 n}{|\mathbb{F}|}\right)^{n'}$$

Yet the number of inputs to this compression algorithm is $|\mathbb{F}|^{2q}$. Hence the compression cannot succeed with probability better than the ratio of inputs to outputs:

$$\frac{|\mathbb{F}|^{2q+n-n'} \cdot q^{2n'} \cdot n^{n'}}{|\mathbb{F}|^{2q}} = \frac{(q^2 n)^{n'}}{|\mathbb{F}|^{n'-n}}$$

□