Private Set Intersection from Pseudorandom Correlation Generators

Dung Bui[1] and Geoffroy Couteau[2]

[1] IRIF, Université de Paris
bui@irif.fr
[2] CNRS, IRIF, Université de Paris
couteau@irif.fr

Abstract. Pseudorandom correlation generators (PCG) allow two parties to generate long correlated pseudorandom strings with minimal communication. Since secure computation applications typically benefit from such protocols, we explore the use of PCG to improve private set intersection (PSI) protocols. We obtain two main results.
In our first result, we construct a new highly optimized semi-honest PSI. Our protocol builds upon the protocol of (Kolesnikov et al., CCS 2016), and significantly improves it using multiple optimizations, including a new oblivious pseudorandom function (built from a PCG for the subfield-VOLE correlation), and a new technique to handle a generalized variant of Cuckoo hashing tailored to our setting. For sets with elements of size ℓ bits with $\ell \leq 70$, our protocol outperforms all known PSI protocols, by as much as 42% when $\ell = 32$ and with $n = 2^{20}$ items (compared to the best known protocol of (Rindal and Schoppmann, Eurocrypt 2021), enhanced with recent improvements). For these parameters, the communication of our protocol is extremely small: only $129n$ bits of total communication.
In our second result, we use a PCG for a new correlation, called the subfield ring-OLE correlation. We construct a new protocol with attracting features: competitive communication with the state of the art, fully malicious security in the *standard model* (no random oracle or tailored assumptions on hash functions). To our knowledge, our protocol outperforms by a large margin all previous protocols in the standard model, and is competitive even with ROM-based protocols. Furthermore, our protocol leads to a batch non-interactive PSI, where (after a one-time short interaction) a client can broadcast a single compact encoding of its dataset, and compute its intersection with the datasets of multiple servers after receiving a single message from each server.

1 Introduction

Private Set Intersection (PSI) is a cryptographic primitive that allows parties to jointly compute the set of all common elements between their datasets, without leaking any value outside of the intersection. It is a special case of secure multi-party computation (MPC). PSI enjoys a wide array of real-life applications; it is perhaps the most actively researched concrete functionality in secure computation, and has been the target of a tremendous number of works, see [PSZ14, PSSZ15, KKRT16, RR17, KRTW19, PSWW18, PRTY19, PRTY20, CM20, RS21, GPR+21, RT21] and references therein for a sample. As a consequence of this intense research effort, modern PSI protocols now achieve impressive efficiency features, communicating only a few hundred bits per database items, and processing millions of items in seconds.

Paradigms for PSI. Many approaches to PSI have been designed over the year. While some early proposals relied on generic secure computation methods [HEK12], the most efficient solutions to date rely on specialized techniques. While this fails to capture all proposals, the most prominent modern PSI protocols can be categorized in four groups.

Key exchange-based. Early PSI protocols relied on techniques based on the Diffie-Hellman key-exchange [Mea86, HFH99], and were extended to the malicious setting in [DKT10, JL10]. While these protocols have relatively low communication, the need to compute multiple exponentiations for each item gives them relatively poor performances. Nevertheless, recent improvements to these old protocols [RT21] show that this approach can be somewhat competitive for very small sets, when communication is a very scarce resource.

Cuckoo hashing-based. Cuckoo hashing [PR04] is an efficient hashing technique where items are hashed into a linear number of bins while guaranteeing that no bin will contain more than one item. For most of the past decades, the top-performing PSI protocols [PSZ14,PSSZ15,KKRT16,HV17,RR17,CLR17, OOS17,PSWW18,PSZ18] relied on Cuckoo hashing techniques, usually combined with fast oblivious transfer extension [IKNP03]. While the more recent OKVS-based protocols now outperform Cuckoo-hashing-based (CH-based) protocols, the protocol of [KKRT16] remains among the most competitive PSI protocols.

OKVS-based. Starting with the work of [PRTY19], a new approach to PSI has been design, which relies on a carefully chosen encoding of the datasets with oblivious data structures. The first protocols required expensive polynomial interpolation [PRTY19, CM20], but a more efficient data structure called PaXoS was introduced in [PRTY20], which significantly changed the state of affairs, and led to the most efficient PSI protocol known to date [RS21]. The required data structures have been recently abstracted out in [GPR$^+$21] under the name *oblivious key-value stores* (OKVS), and the work introduced significant improvements in the efficiency and security guarantees of OKVS constructions. As a consequence, a combination of [RS21] with the latest OKVS of [GPR$^+$21] leads to a very low-communication PSI protocol (for mid to large databases), still with a very good concrete efficiency.

Polynomial manipulation. Eventually, starting with the work of [FNP04], several protocols have been designed that represent the datasets as polynomials over finite fields, and compute set operations through operations on these polynomials. Notable works include [KS05,DMRY09,HN10,Haz15, FHNP16, GS19, GN19]. While these protocols are usually much slower than those based on Cuckoo hashing or OKVS, they offer some advantages, such as realizing stronger functionalities (like threshold private set intersection [GS19]) and providing security in the standard model (in contrast, the most efficient OKVS- or CH-based protocols require the random oracle model, or some ad hoc family of correlation-robustness assumptions).

Improving PSI with pseudorandom correlation generators. Pseudorandom correlation generators (PCG) have been introduced in the works of [BCG$^+$17, BCGI18, BCG$^+$19b] and have been the subject of a long and fruitful line of work [BCG$^+$17, BCGI18, BCG$^+$19b, BCG$^+$19a, SGRR19, BCG$^+$20b, BCG$^+$20a, YWL$^+$20, CRR21, WYKW21]. At a high level, a PCG allows two party to securely stretch long pseudorandom correlated strings from short, correlated seeds. Securely sharing correlated random strings is a crucial component in most modern secure computation protocols, which operate in the preprocessing model; PCG allows to realize this functionality with almost no communication. Among their many applications, PCGs allow to construct *silent oblivious transfer extension* protocols [BCG$^+$19a], which can realize (pseudorandom) OT extension with minimal (logarithmic) communication.

Since the top-performing PSI protocols rely on efficient OT extension, using PCG-based techniques to improve their efficiency is a natural idea. And indeed, this was done recently for OKVS-based PSI in [RS21], leading to the most efficient PSI protocol known to date.

To give a single datapoint, computing the intersection between two databases of size $n = 2^{20}$ with the protocol of [RS21] communicates as little as $426n$ bits in total. In addition, some of the tools used in [RS21] have been significantly improved since: replacing their OKVS (which is the PaXoS OKVS of [PRTY20]) by the more recent 3H-GCT OKVS of [GPR$^+$21], and replacing their PCG (which is the one from [WYKW21]) by the recent PCG of [CRR21], the cost goes down to an impressive $247n$ bits of total communication. In comparison, even the *insecure* approach of exchanging the hashes of all items in the databases already requires $160n$ bits of communication. OKVS-based PSI protocols are now firmly established as the leading paradigm in the field, and the use of PCGs to reduce their communication overhead even more seems to further widen the gap with the other paradigms.

1.1 Our Contributions

We thoroughly investigate how pseudorandom correlation generators can improve the design of PSI protocols, in several of the previous paradigms. We obtain two main contributions.

Beating OKVS PSI with CH PSI, or "KKRT strikes back". We uncover a crucial distinction between CH-based and OKVS-based PSI protocols, which had never been pointed out to our knowledge. At a high level, Cuckoo hashing-based protocols possess a feature that makes them considerably more "PCG-friendly" than OKVS-based PSI. While the work of [RS21] demonstrated that PCGs can significantly reduce the communication of OKVS-based PSI protocol,[3] we show that their effect on Cuckoo hashing-based protocols is much deeper. Indeed, we show how a careful use of PCGs in KKRT [KKRT16], the leading Cuckoo hashing-based protocol, does not only significantly reduce its communication overhead: it also makes the communication complexity *entirely independent* of the computational security parameter (a feature that no previous protocol ever enjoyed to our knowledge). This in turns opens the door to several optimizations, including well-known optimizations (the use of permutation-based hashing to further reduce communication, which was previously incompatible with KKRT) and new techniques (including a significant generalization of KKRT which enables the use of a new variant of Cuckoo hashing to reduce the communication even further). Along the way, we notice an error in the security analysis of KKRT. Fixing the error requires introducing a new notion of correlation-robustness, which we do in the context of our construction. Our new notion allows in particular to repair the analysis of the KKRT protocol.

As a result, our first contribution shakes the dominant position of OKVS-based PSI protocols. To give a datapoint, for $n = 2^{20}$, the KKRT protocol communicates $1008n$ bits. We reduce this communication to $283n$ bits in the general case (a 72% reduction). For the common case of databases with small entries, we achieve even more impressive reductions: for example, with 64-bit entries, our protocol communicates $252n$ bits, and with 32-bit entries, it communicates only $189n$ bits (a 82% reduction compared to KKRT). This means that for databases with entries smaller than 64 bits, the Cuckoo hashing approach becomes more efficient than OKVS-based PSI, even when using all of the latest optimizations on the latter. None of these communication improvements come at the cost of computation: the computation overhead is actually reduced by more than 33% compared to KKRT.

In many concrete scenario, one might be fine with a lower statistical security parameter: a statistical failure probability of 2^{-30}, for example, might be deemed perfectly acceptable. Because our new construction depends *solely* on this parameter (and not on the computational security parameter), reducing λ from 40 to 30 makes our protocol stand out even more compared to the state of the art. For example, for inputs of length $\ell = 32$ and $\lambda = 30$, the protocol of [RS21] (enhanced with recent advances in VOLE and OKVS) has communication $237n$ at $n = 2^{20}$, while our protocol communicates $169n$ bits. Furthermore, a technique of [TLP$^+$17] can be used to reduce the size of the last message (intuitively, the trick is to sort the values in increasing order, and then send the differences between consecutive values; these differences are $\approx \log n$ bits shorter than the values themselves on average). This trick applies to both the protocol of [RS21] and ours, but its effect on our protocol is twice larger (because we send twice more values in the last round). The price to pay is quite mild (sorting a list of $2n$ values). Concretely, with $\lambda = 30$ and using this technique in both [RS21] and our protocol, our approach achieves lower communications already for $\ell \leq 77$ bits. We provide further datapoints and comparisons to the state of the art on Table 1.

Efficient PSI in the standard model. In our second contribution, we design a new "polynomial-based" PSI protocol, following the high level structure of previous works [KS05, GS19, GN19]. To this end, we introduce the notion of PCG for the *subfield ring-OLE* correlation, and show how a simple variant of the recent PCG for ring-OLE of [BCG$^+$20b] leads to efficient instantiations of this primitive. Then, we describe a new PSI protocol built on top of this PCG, which enjoys a number of very interesting features.

Security features. Our PSI protocol is in the standard model: unlike our first protocol, it does not require the random oracle model, or any tailor-made correlation-robustness assumptions. We rely solely on the (relatively well-established) ring-LPN assumption over polynomial rings with irreducible polynomials. To our knowledge, our protocol is the first standard model protocol which offers competitive performances compared to protocols using the random oracle heuristic or tailored assumptions. Furthermore, our PSI protocol enjoys full malicious security (for both parties) *almost for free.* This stems

[3] More precisely, [RS21] focused on using PCGs to improve the protocol of [PRTY20], which uses PaXoS; the work of [GPR$^+$21] observed that the technique actually works with any OKVS.

Table 1. Comparison of the communication cost of several PSI protocols in the semi-honest model, for various choices of the database size n (we assume that both parties have a database of the same size) and statistical security parameter λ. ℓ denote the bit-length of the inputs in the database; we set the computational security parameter κ to 128.

	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{20}$	$n = 2^{24}$
Statistical security parameter $\lambda = 40$				
[KKRT16]	$972n$	$984n$	$1008n$	$1032n$
[PRTY19] low[*]	$505n$	$509n$	$513n$	$517n$
[PRTY19] fast[*]	$588n$	$603n$	$619n$	$635n$
[CM20]	$682n$	$678n$	$694n$	$702n$
[PRTY20]	$1258n$	$1208n$	$1268n$	$1302n$
[RS21]	$2038n$	$914n$	$426n$	$398n$
[RS21] enhanced[**]	$271n$	$249n$	$247n$	$254n$
Ours ($\ell = 64$)	$275n$	$253n$	$252n$	$259n$
Ours ($\ell = 48$)	$244n$	$222n$	$221n$	$229n$
Ours ($\ell = 32$)	$212n$	$190n$	$189n$	$196n$
Statistical security parameter $\lambda = 30$				
[RS21] enhanced[**]	$261n$	$239n$	$237n$	$244n$
Ours ($\ell = 64$)	$255n$	$233n$	$232n$	$239n$
Ours ($\ell = 48$)	$224n$	$202n$	$201n$	$209n$
Ours ($\ell = 32$)	$192n$	$170n$	$169n$	$176n$
Statistical security parameter $\lambda = 30$, using the technique of [TLP$^+$17]				
[RS21] enhanced[**]	$247n$	$223n$	$217n$	$220n$
Ours ($\ell = 64$)	$227n$	$201n$	$191n$	$191n$
Ours ($\ell = 48$)	$196n$	$170n$	$161n$	$161n$
Ours ($\ell = 32$)	$166n$	$138n$	$129n$	$128n$

[*] PRTY19 has two variants, SpOT-low (lowest communication, higher computation) and SpOT-fast (higher communication, better computation). Both use expensive polynomial interpolation and require significantly more computation compared to all other protocols in this table.

[**] Using the 3H-GCT OKVS of [GPR$^+$21] instead of PaXoS, and the VOLE of [CRR21] instead of the one from [WYKW21].

from the use of PCGs, which allows to confine the "price" of achieving malicious security to the distributed seed generation only, which has logarithmic communication and computation (in the set size n).

Efficiency features. Our PSI protocol enjoys a very low communication, considerably lower than all previous PSI protocols in the standard model which we are aware of (excluding iO- or FHE-based protocol, which can have very low communication but very poor concrete efficiency). In fact, communication-wise, our PSI protocol is even competitive with the best ROM-based PSI protocols. Concretely, for sets of size n with ℓ-bit entries, our protocol communicates $(2\ell + 3\lambda + 6\log n) \cdot n + o(n)$ bits. To give a single datapoint, for $\ell = 32$ and $n = 2^{20}$, we estimate the total communication to be $334n$ bits. This is only mildly larger than the best maliciously secure protocol [RS21], which communicates $257n$ bits in the same setting, with comparable computation (it also uses polynomial interpolation), but without standard model security.

Note that our second protocol shares with our first protocol the (previously never achieved) feature of having a communication independent of κ. Our protocol requires more computation compared to the best ROM-based protocols, due to its use of polynomial interpolation. However, it still allows for very fast PSI computation (we estimate a few seconds to compute the intersection between databases of size 2^{20}, on one core of a standard laptop). Concretely, the protocol requires only

– a single degree-n polynomial interpolation, one FFT over a polynomial ring with degree-$2n$ polynomials, and 3 multiplications of degree-n polynomials for the receiver, and

– a single degree-n polynomial interpolation, one FFT as above, 2 multiplications of degree-n polynomials, and a single n-multipoint polynomial evaluation for the sender.

Furthermore, both polynomial interpolations only have to be performed over a field $\mathbb{F}$, of size $|\mathbb{F}| \approx 2^{\ell}$ where ℓ is the bit size of the set items (e.g. 32 or 64 bits), and the multipoint evaluation is over a field of size $\lambda + 2\log n$ bits. This stand in stark contrasts with previous state of the art protocols [PRTY19] that relied on polynomial interpolation (*on top* of using the ROM), where the interpolations and multipoint evaluations had to be performed over a very large field $\mathbb{F}$ of size $|\mathbb{F}| \approx 2^{400}$. By using a cyclotomic ring, the FFTs and polynomial multiplications are much faster than the interpolations.

Batch non-interactive PSI. On top of these security and efficiency features, the structure of our protocol allows to obtain a powerful interaction pattern: it leads to a batch non-interactive PSI, where after a short interaction with each server, a client C with set X can broadcast a *single* encoding of their database, and receive afterwards at anytime a single message from each server S_i with set X_i, from which they can decode $X \cap X_i$.

In essence, we build upon the fact that the PCG for subfield ring-OLE correlations is *programmable*, which means that we can enforce that a target party will receive the same pseudorandom string accross executions with many different parties. Concretely, we achieve the following form of *batch non-interactive PSI* between a client C with database X and multiple servers S_i with datasets X_i (all of size n):

1. In a preprocessing phase, C interacts with each of the servers, using $O(\log n)$ communication *and* computation in each interaction, in a small constant number of rounds.
2. Then, C performs a single $\tilde{O}(n)$ cost local computation, and broadcasts a single $2\ell n$-size *encoding* E_X of X.
3. Each server S_i can, at any time, send a single message $M_i = m(X_i, E_X)$, of length $3(\lambda + 2\log n)n$, using $\tilde{O}(n)$ computation.
4. Eventually, given X and M_i, the client C can run a $\tilde{O}(n)$ cost decoding procedure and recover $X \cap X_i$, without further interaction.

When the number of servers becomes large, our batch PSI protocol can lead to very strong savings for the client compared to executing a PSI protocol individually with each server, and provide a solution competitive even with the best ROM-based PSI protocols. Furthermore, in this setting, the amortized communication (per PSI instance) is reduced to $(2\ell/N_S + 3\lambda + 6\log n) \cdot n + o(n)$, where N_S denotes the number of servers.

2 Technical Overview

We focus in this technical overview on our first contribution; the second contribution is conceptually simpler (using mostly standard linear algebra lemmas) and the exposition in Section 5.1 should be easy to follow.

2.1 The KKRT protocol

Let us provide a high level overview of KKRT. At a high level, it combines Cuckoo hashing with a *batch related-key oblivious pseudorandom function* (BaRK-OPRF). A BaRK-OPRF is a protocol between a receiver, Alice, and a sender, Bob, where Alice has inputs $(x_1, \cdots, x_n)$. Bob receives one global key Δ and n local keys $(K_1, \cdots, K_n)$, and Alice learns $(F_{\Delta,K_1}(x_1), \cdots, F_{\Delta,K_n}(x_n))$. In KKRT, the BaRK-OPRF has the following structure: $F_{\Delta,K_i}(x_i) = H(i, K_i \oplus (\Delta \wedge \mathsf{Enc}(x_i)))$, where

– H is a hash function, modeled as a random oracle (or satisfying a tailor-made *Hamming correlation robustness* assumption),
– Enc is the encoding procedure of a suitable good error-correcting code,
– $\Delta \wedge \mathsf{Enc}(x_i)$ denotes the bitwise AND of Δ and $\mathsf{Enc}(x_i)$.

The intuition underlying the security is that for any value $x \neq x_i$, $\mathsf{Enc}(x)$ and $\mathsf{Enc}(x_i)$ differ on many positions (because the code has high minimum distance), hence $K_i \oplus \varDelta \wedge \mathsf{Enc}(x)$ contains a lot of entropy, even given $K_i \oplus \varDelta \wedge \mathsf{Enc}(x_i)$. Under a suitable assumption on the hash function, $H(K_i \oplus \varDelta \wedge \mathsf{Enc}(x))$ therefore looks random even given $F_{\Delta,K_i}(x_i)$, for any $x \neq x_i$. Note that to turn this intuition into a proof, the authors of KKRT introduced a formal notion of Hamming correlation robustness. However, their definition has a flaw which makes it impossible to instantiate (there is a simple attack on any instantiation, even with a random oracle). When analyzing the security of our variant, we will provide a (more involved) corrected notion of correlation robustness, and formally prove that it is satisfied by a random oracle.

The above construction can be instantiated quite efficiency using the IKNP oblivious transfer extension [IKNP03]. It does not, however, directly imply a PSI protocol: since the keys K_i are distinct for each input, it can only help comparing items in the same position. To obtain a PSI protocol, KKRT uses hashing to map the datasets to bins, and reduce the intersection computation to a bin-by-bin comparison. With a naive hashing strategy, this would lead to a maximum of $\log n / \log\log n$ items per bin (for datasets of size n), which would still induce a significant overhead (since all pairs of items of Alice and Bob in the same bin must be compared).

To achieve better efficiency, the authors use *Cuckoo hashing with a stash*. In Cuckoo hashing, the items are mapped into $c \cdot n$ bins (where $c > 1$ is usually a small constant) using d hash functions $(h_1, \cdots, h_d)$, such that each item x is mapped to the bin number $h_i(x)$ for some $i \leq d$, and every bin contains at most a *single* item. The insertion follows a simple greedy procedure: x is inserted at the $h_1(x)$ location, and if an item y was already present, it is evicted and re-inserted at its next "authorized" location, $h_2(y)$, possibly evicting an item in turn. The process continues until insertion succeeds, or some threshold number of items have been evicted. If an item fails to be inserted, it is added to a special stack, called the *stash*.

Using Cuckoo hashing, Alice maps her dataset X to a length-cn vector $(x_1, \cdots, x_{cn})$ (possibly adding dummy items in bins that remained empty), and runs the BaRK-OPRF protocol with Bob, obtaining $(F_{\Delta,K_i}(x_i))_{i\leq cn}$. Then, Bob maps his own dataset Y to the $c{\cdot}n$ bins, this time using *all* hash functions $h_1, \cdots, h_d$ (that is, every $y \in Y$ is mapped to the bins $(h_1(y), \cdots, h_d(y))$), and computes $F_{\Delta,K_i}(y)$ for all items y in the i-th bin, for $i = 1$ to $c \cdot n$. In the end, Bob randomly shuffles and sends to Alice all these PRF evaluations. Alice computes the intersection with her own PRF evaluations, and learns $X \cap Y$.

KKRT also showed how to handle the items for which insertion failed (which are stored in the stash). However, the work of [PSZ18] heuristically analyzed the failure probability of generalized Cuckoo hashing with $k \geq 2$ hash functions, using large scale simulations and extrapolations. Based on their extrapolations, they determined that using as little as 3 hash functions and $N = 1.3 \cdot n$ bins suffices to guarantee a statistical failure probability below 2^{-40} (i.e., a stash size $s = 0$ with probability at least $1 - 2^{-40}$) for large enough set sizes (around $n = 2^{20}$).

Cost of KKRT. Using the heuristic parameters of [PSZ18] to get rid of the stash, the communication of KKRT consists in executing a BaRK-OPRF on $c{\cdot}n$ inputs, and sending $dn = 3n$ hashes. The BaRK-OPRF requires around $6\kappa n$ bits of communication, where κ is a computational security parameter (typically, $\kappa = 128$). The factor 6 overhead mainly comes from the fact that $\varDelta \wedge \mathsf{Enc}(x)$ must retain $\approx \kappa$ bits of entropy, hence a large-ish value of $\varDelta$ is required. Sending the $3n$ hashes costs $3 \cdot (\lambda + 2\log n) \cdot n$ bits of communication, where λ is a statistical security parameter (typically, $\lambda = 40$ is the most common choice). Indeed, the hash outputs can be truncated to $\lambda + 2\log n$ bits while maintaining the probability of collisions between the hashes of any pair $(x, y) \in X \times Y$ of distinct elements below $2^{-\lambda}$. For $n = 2^{20}$, this leads to the claimed $1008n$ bits of overall communication.

2.2 Pseudorandom Correlation Generators

In our new protocol, we will reuse the KKRT template, but replace the BaRK-OPRF with a subfield-VOLE-based construction. Doing so enables several new optimizations of the scheme, which we describe afterwards. We start by recalling the notion of pseudorandom correlation generator, and that of subfield vector OLE. A pseudorandom correlation generator is a pair of algorithm $(\mathsf{Gen}, \mathsf{Expand})$, where Gen distributes a pair of small seeds (s_0, s_1), and $\mathsf{Expand}(i, s_i)$ stretches a seed into a long pseudorandom string. Informally, a PCG for a target *correlation* satisfies two properties:

- **Correctness.** Given $(s_0, s_1) \leftarrow_r \mathsf{Gen}(1^\kappa)$, the pair $(x_0, x_1) = ((\mathsf{Expand}(0, s_0), \mathsf{Expand}(1, s_1))$ is indistinguishable from a random sample from the target correlation.
- **Security.** Given s_0, the string $x_1 = \mathsf{Expand}(1, s_1)$ is indistinguishable from a uniformly random string sampled *conditioned on satisfying the right correlation with* $\mathsf{Expand}(0, s_0)$. The converse property holds as well.

In addition, a PCG should satisfy some *shortness* conditions: the seeds (s_0, s_1) should be significantly smaller than a sample from the target correlation.

A subfield vector oblivious linear evaluation generator (sVOLE generator) is a PCG for the following two-party correlation: Alice gets a pair of random vectors $(\mathbf{u}, \mathbf{v})$, and Bob gets a random scalar Δ and the vector $\mathbf{w} = \Delta \cdot \mathbf{u} + \mathbf{v}$, where $\mathbf{v}, \mathbf{w} \in \mathbb{F}^n$ for some field $\mathbb{F}$, $\Delta \in \mathbb{F}$, and $\mathbf{u} \in (\mathbb{F}')^n$, where $\mathbb{F}'$ is a subfield of $\mathbb{F}$. Efficient PCGs for the sVOLE correlation have been designed in [BCG+19b,BCG+19a,CRR21], with seed sizes logarithmic in n, and the latest protocol of [CRR21] achieve extremely impressive efficiency features (around 300ms to generate an sVOLE correlation of length 10^7 on one core of a standard laptop; using this sVOLE to achieve OT extension results in a protocol using 37% less computation and $\sim 1300\times$ less communication than the standard IKNP protocol).

In the following, we will sometimes slightly abuse the notion of PCG, and use formulations such as "Alice and Bob use a PCG protocol". What this means is the following: Alice and Bob use some dedicated two-party computation protocol to distributively and securely generate seeds $(s_0, s_1) \leftarrow_r \mathsf{Gen}(1^\kappa)$, such that Alice gets s_0 and Bob gets s_1; then, Alice and Bob *locally* expand these seeds into a pseudorandom instance of the target correlation. Efficient protocols for distributing PCG seeds have been introduced in [BCG+19a], and typically have communication linear in the seed length – that is, logarithmic in n.

2.3 A New sVOLE-Based BaRK-OPRF

Subfield-VOLE leads to a simple and natural construction of BaRK-OPRF. Let ℓ be the bitlength of Alice's inputs, and let $\mathbf{x} = (x_1, \cdots, x_n)$ be the inputs of Alice, viewed as elements of $\mathbb{F}_{2^\ell}$. We assume for simplicity that ℓ divides κ, the computational security parameter. Alice and Bob use an sVOLE protocol (e.g. [CRR21]) over the field $\mathbb{F}_{2^\kappa}$, with subfield $\mathbb{F}_{2^\ell}$; let $(\mathbf{u}, \mathbf{v})$ be the output of Alice, and $(\Delta, \mathbf{w})$ be the output of Bob. Recall that $\mathbf{w} = \Delta \cdot \mathbf{u} + \mathbf{v}$. Alice sends $\mathbf{z} = \mathbf{x} - \mathbf{u}$ to Bob, who defines the BaRK-OPRF keys to be Δ and $(K_1, \cdots, K_n) = \Delta \cdot \mathbf{z} + \mathbf{w}$. The BaRK-OPRF is defined as follows: $F_{\Delta, K_i}(y) = H(i, K_i - \Delta \cdot y)$ (all operations are over $\mathbb{F}_{2^\kappa}$). Eventually, Alice outputs $(H(i, v_i))_{i \le n}$. Observe that

$$\begin{aligned} H(i, v_i) &= H(i, w_i - \Delta u_i) = H(i, K_i - \Delta(z_i + u_i)) \\ &= H(i, K_i - \Delta \cdot x_i) = F_{\Delta, K_i}(x_i). \end{aligned}$$

Compared to the KKRT BaRK-OPRF, this construction has two significant advantages:

- Because it uses sVOLE, the bitwise AND is now replaced by a field multiplication. In particular, this means that we do not need anymore to use error-correcting codes, and that $y \cdot \Delta$ retains the entire entropy of Δ. In other words, it suffices for Δ to be κ-bit long to achieve κ bits of security for the construction (in contrast, KKRT had to use around 5κ bits). We prove this formally by reducing security to the (standard) correlation robustness of H, and by analyzing the exact security guarantees when modeling H as a random oracle.
- Perhaps most importantly, the use of *subfield* VOLE allows us to completely decorrelate the size of $\mathbf{u}$ from that of Δ, something which can fundamentally not be achieved with the INKP OT extension. Concretely, this means that $\mathbf{u}$ only needs to mask the input vector $\mathbf{x}$ of Alice. If $\mathbf{x} \in \mathbb{F}_{2^\ell}^n$, then so do $\mathbf{u}$ and $\mathbf{z}$: the communication now depends solely on the input size.

In total, our BaRK-OPRF communicates $\ell \cdot n$ bits, plus the cost of distributing the seeds for the sVOLE generator. Using the protocol of [BCG+19a] to distribute the seeds[4] adds a $(2 \log n + 9) \cdot t\kappa$ overhead, where t is a computational security parameter for the underlying LPN assumption, which is slightly smaller than κ (for example, according to Table 1 of [BCG+19a], $t = 118$ suffices to get

[4] This protocol uses a length-t reverse VOLE protocol as a blackbox, which we instantiate with the construction of [ADI+17].

128 bits of security for the underlying LPN assumption, when $n = 2^{20}$). This cost is logarithmic in n, hence its effect on the overall communication vanishes for large enough n. Concretely, for $n = 2^{20}$, this amounts to a total communication of $(\ell + 0.7) \cdot n$ bits (where the seed distribution contributes only $0.7n$).

2.4 Combining the New OPRF with Permutation-Based Hashing

Plugging our new BaRK-OPRF into KKRT, and using the same parameters for Cuckoo hashing, leads to a protocol with total communication $(1.3 \cdot \ell + 3 \cdot (\lambda + 2\log n))n + o(n)$ bits (where the $o(n)$ terms capture the costs of distributing the PCG seeds). Concretely, for $n = 2^{20}$ and $\ell = 32$ (resp. 64), this already brings the cost down, from $1008n$ bits to $282n$ bits (resp. $324n$ bits). However, this can be further improved using the well-established notion of *permutation-based* hashing [PSSZ15]. Concretely, storing a full ℓ-bit item x in the i-th bin induces some redundancy, since being in the i-th bins already implies that $h_j(x) = i$ for some $j \in \{1, 2, 3\}$. Building upon this observation, the work of [PSSZ15] devised a more efficient hashing strategy, where an item x is written as $x_L || x_R$, where x_L is $\log(1.3n)$-bit long. The item x is inserted by mapping x_R to the bin $x_L \oplus f(x_R)$, where f is a k-wise independent hash function, for some large enough k. This guarantees that no collision occurs, because if two items x, x' end up mapping the same value to the same bin, this means that $x_R = x'_R$ and $x_L \oplus f(x_R) = x'_L \oplus f'(x'_R)$, hence $x = x'$. When multiple hash functions are used, as in Cuckoo hashing, the index of the hash function must be appended to x_R.

While permutation-based hashing is a well-known optimization, it does not provide any communication savings for KKRT (nor for any of the OKVS-based PSI protocols). In our protocol, however, it further reduces the communication to $(1.3 \cdot (\ell - \log(1.3n) + 1) + 3 \cdot (\lambda + 2\log n))n + o(n)$ bits, which gives $275n$ bits for $n = 2^{20}$ and 32-bit items, or $317n$ bits for 64-bit items. In itself, this is a really small communication improvement. However, it has an important consequence: it implies that the Alice-to-Bob communication is now completely dominated by the Bob-to-Alice communication. Concretely, this means that we can easily afford to use a much higher number of bins (which is $1.3n$ currently) if it can allow us to reduce the number of hash functions (which is 3). This brings us to our last optimization.

2.5 Packing Multiple Items per Bin

In this last optimization, our goal is to reduce the number of hash functions used in the Cuckoo hashing protocol, from 3 to 2, by increasing the number of bins to compensate. Unfortunately, this does not work directly: when the number of hash functions is as low as 2, it becomes essentially infeasible to guarantee a 2^{-40} probability of failure, for any reasonable number of bins. This is to be expected given the theoretical failure probability in this setting and the heuristic experiments of [PSZ18]; we further confirmed this using simulations. While one could get away with using a reasonably small stash ($s = 2$ or 3 appears to suffice in our experiments), the cost of handling the stash is high, and nullifies all communication benefits of using two hash functions in the first place.

Generalized Cuckoo Hashing. instead, we use a different approach: we add one degree of freedom to the Cuckoo hashing parameters, *by allowing bins to contain multiple items.* This generalization of Cuckoo hashing is not new: it has been studied in details in several works [DW07, Pan05, W$^+$17], because it comes with a much nicer cache-friendliness than standard Cuckoo hashing (using Cuckoo hashing with d hash functions requires d random memory accesses, which incurs cache misses with high probability; in contrast, allowing up to d items in a given bin only requires retrieving d values from the same memory location, hence increasing the bin load to reduce the number of hash functions is typically a worthwhile tradeoff).

In (d, k)-Cuckoo hashing, n items are mapped to $(1 + \varepsilon) \cdot n$ bins using k hash functions, and each bin is allowed to contain up to d items (standard Cuckoo hashing is $(1, 2)$-Cuckoo hashing, and the usual PSI variant is $(1, 3)$-Cuckoo hashing). Allowing more items per bins significantly improves the efficiency; for example, $(3, 2)$-Cuckoo hashing is known to perform strictly better than $(1, 3)$-Cuckoo hashing in terms of occupancy (i.e., the total number of slots $N = d \cdot (1 + \varepsilon) \cdot n$ which must be used to guarantee a $o(1)$ failure probability). Based on existing analysis of this variant [W$^+$17], it seems reasonable to expect that $(3, 2)$-Cuckoo hashing already achieves a strictly smaller failure probability compared to $(1, 3)$-Cuckoo hashing, with a smaller number of bins.

As in previous Cuckoo hashing schemes, however, proving good concrete bounds is out of reach of current methods. Instead, following the established methodology of [PSZ18], we relied on extensive computer simulations on small values of n (from 256 to 2048) to select parameters, and extrapolated from these results parameters for larger values of n. More precisely, we ran 10^7 experiments with $(3,2)$-Cuckoo hashing for $n \in \{2^8, 2^9, 2^{10}\}$ (we also experimented with 2^{11}, but with a smaller number of experiments) with $c \cdot n$ bins for various values of c. Even for a value as low as $c = 0.65$ and values of n as low as 2^9, our experiments never reported any insertion failure, indicating that the empirical failure probability should already be way below 2^{-20}. Since the theoretical failure probability is known to scale as $O(1/n^\delta)$ for some constant δ with reasonably small constant factors, we extrapolate that for large enough values of n, e.g. $n \geq 2^{18}$, the failure probability should be well below 2^{-40}.

2.6 A Membership BaRK-OPRF

There remains a non-trivial task: to use this improved Cuckoo hashing variant, we need a protocol to handle Cuckoo hashing with up to d items per bins. Intuitively, denoting $\mathbf{x_i} = (x_i^{(1)}, \cdots, x_i^{(d)})$ the d entries of the bin i, we want to construct a new kind of *membership* OPRF, where Bob obtains $F_{\Delta,K_i}(y)$ and Alice obtains the set $F_{\Delta,K_i}(\mathbf{x_i}) = \{F_{\Delta,K_i}(x_i^{(j)})\}_{j\leq d}$. This implies that $F_{\Delta,K_i}(y) \in F_{\Delta,K_i}(\mathbf{x_i})$ if and only if y is equal to any entry of $\mathbf{x_i}$, and $F_{\Delta,K_i}(y)$ looks pseudorandom to Alice otherwise.

Going back to the BaRK-OPRF, recall that for a bin i where Alice placed x_i and Bob placed y_i, Alice computes $H(i, v_i)$ and Bob computes $H(i, K_i - \Delta y_i) = H(i, \Delta \cdot (x_i - y_i) + v_i)$. Here, we view the $x_i - y_i$ term as $P_{x_i}(y_i)$, where $P_{x_i} = X - x_i$ is a degree-1 polynomial with root x_i. This view suggests a natural generalization of this approach, where the P_{x_i} polynomials are replaced by higher degree polynomials. Define $P_{\mathbf{x_i}}$ to be the polynomial $\prod_{j=1}^d (X - x_i^{(j)})$, and let $(c_{j,i})_{0\leq j\leq d-1}$ denote its coefficients: $P_{\mathbf{x_i}}(X) = X^d + \sum_{j=0}^{d-1} c_{j,i} \cdot X^j$. Our new *membership* BaRK-OPRF is a direct generalization of the BaRK-OPRF from Section 2.3, which we sketch below.

Our construction. Let m be the bitlength of Alice's inputs inside the bins, and let $(\mathbf{x_1}, \cdots, \mathbf{x_N})$ be the inputs of Alice in each of the N bins, where the inputs in each bin are viewed as length-d vectors of elements of $\mathbb{F}_{2^m}$. We assume for simplicity that m divides κ, the computational security parameter. Alice and Bob use d sVOLE protocol (e.g. [CRR21]) over the field $\mathbb{F}_{2^\kappa}$, with subfield $\mathbb{F}_{2^m}$, *with the same value* Δ.[5] Let $(\mathbf{u_j}, \mathbf{v_j})_{j\leq d}$ be the outputs of Alice, and $(\Delta, (\mathbf{w_j})_{j\leq d})$ be the output of Bob. Recall that $\mathbf{w_j} = \Delta \cdot \mathbf{u_j} + \mathbf{v_j}$.

For each $\mathbf{x_i}$, let $(c_{0,i}, \cdots, c_{d-1,i})$ be the coefficients of the polynomial $P_{\mathbf{x_i}}$ (omitting the coefficient of X^d, which is always 1). Let $\mathbf{c_j}$ denote the vector $(c_{j,i})_{i\leq N}$ for $j = 0$ to $d-1$. Alice sends $\mathbf{z_j} = \mathbf{c_j} - \mathbf{u_j}$ for $j = 0$ to $d-1$ to Bob, who defines the membership BaRK-OPRF keys to be Δ and $K_i = (k_{j,i})_{0\leq j\leq d-1} = (\Delta \cdot z_{j,i} + w_{j,i})_{0\leq j\leq d-1}$ for $i = 1$ to N. Define the following degree-d polynomial P_{Δ,K_i} over $\mathbb{F}_q$: $P_{\Delta,K_i}(X) = \Delta \cdot X^d + \sum_{j=0}^{d-1} k_{j,i} \cdot X^j$. The OPRF is defined as follows: $F_{\Delta,K_i}(y) = H(i, P_{\Delta,K_i}(y))$ (all operations are over $\mathbb{F}_{2^\kappa}$). Eventually, for each bin i, Alice sets her d tuple of outputs to be $F_{\Delta,K_i}(\mathbf{x_i}) = \{H(i, \sum_{j=0}^{d-1} v_{j,i} \cdot (x_i^{(k)})^j\}_{k\leq d}$. Observe that, since $k_{j,i} = \Delta z_{j,i} + w_{j,i} = \Delta c_{j,i} + v_{j,i}$ for all i, j, we have

$$\begin{aligned} H(i, P_{\Delta,K_i}(y)) &= H\left(i, \Delta \cdot \left(y^d + \sum_{j=0}^{d-1} c_{j,i} y^j\right) + \sum_{j=0}^{d-1} v_{j,i} y^j\right) \\ &= H\left(i, \Delta \cdot P_{\mathbf{x_i}}(y) + \sum_{j=0}^{d-1} v_{j,i} y^j\right). \end{aligned}$$

Therefore, if there exists $k \in \{1, \cdots, d\}$ such that $y = x_i^{(k)}$, we have $P_{\mathbf{x_i}}(y) = 0$, and $H(i, P_{\Delta,K_i}(y)) = H(i, \sum_{j=0}^{d-1} v_{j,i} \cdot (x_i^{(k)})^j) \in F_{\Delta,K_i}(\mathbf{x_i})$. On the other hand, whenever $P_{\mathbf{x_i}}(y) \neq 0$, then the $\Delta \cdot P_{\mathbf{x_i}}(y)$ term in the hash makes the output pseudorandom from the viewpoint of Alice, under the correlation robustness of the hash function.

[5] Note that all known sVOLE protocols allow Bob to choose the value of Δ, hence Bob can enforce the use of the same Δ across all instances.

2.7 Tying up Loose Ends

Using the new construction from the previous Section, together with $(2,3)$-Cuckoo hashing, leads to a total communication of $(0.65 \cdot 3(\ell - \log(0.65n) + 1) + 2 \cdot (\lambda + 2\log n))n + o(n)$ bits (using $N = 0.65n$ bins, with up to 3 items per bin, which we heuristically estimate to fail with probability at most 2^{-40}). For $n = 2^{20}$ and 32 bits items, this gives $188n$ bits of communication, a very significant reduction compared to the $275n$ achieved without this last optimization. We mention a few remaining details:

- In the construction of membership BaRK-OPRF, Alice and Bob need to invoke $d = 3$ length-N sVOLE. In fact, it suffices to invoke a single length-$3N$ sVOLE, and to cut the output in three equal length parts, to obtain the necessary correlation. This means that the concrete cost of distributing the sVOLE seeds remains that of generating a single sVOLE (e.g. $\approx 0.7n$ bits for $n = 2^{20}$).
- In the above, we overlooked an important subtlety: a bin can possibly contain less than d items. In KKRT, this was handled by adding dummy items to empty bins; however, due to our use of permutation-based hashing, we must be careful that dummy items will not collide with some of Bob's inputs; doing so incurs some communication overhead. We use instead a more efficient approach: we let Alice also hide the *degree-d* term of the polynomial defined for each bin. This way, a bin can possibly correspond to a polynomial of degree less than d (viewed as a degree-d polynomial with some zero coefficients) without leaking the actual degree to Bob. With this modification, Alice must now send $d+1$ $\mathbf{u_j}$'s to Bob instead of d. However, observing that the leading coefficient can only be either 0 or 1, the vector $\mathbf{u_d}$ need not be over $\mathbb{F}_{2^m}$: it can be over $\mathbb{F}_2$ instead. This reduces the communication overhead of sending the $\mathbf{u_j}$'s from $(d+1)m \cdot N$ to $(dm+1) \cdot N$, at the (mild) cost of having now to perform two different sVOLE instance, a length-$3N$ sVOLE where the subfield is $\mathbb{F}_{2^m}$, and a length-N sVOLE where it is $\mathbb{F}_2$ (hence, for $n = 2^{20}$, the sVOLE seed distribution overhead now becomes $1.4n$ bits).

3 Preliminaries

Notation. Throughout the paper we use the following notations: we let κ, λ denote the computational and statistical security parameters, respectively. We write $[m]$ to denote a set $\{1, 2, \ldots, m\}$. We typically write $\mathbb{F}_q$ to denote a field with and arbitrary subfield $\mathbb{F}_p$, where p is a prime power and $q = p^r$. We use $\mathcal{R}_\mathsf{p} = \mathbb{F}_\mathsf{p}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ for the ring over the field $\mathbb{F}_p$ where $F(x)$ is some polynomial, and also denote $\mathcal{R}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^\mathsf{t}}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$. For a vector $\mathbf{x}$ we define by x_i its i-th coordinate. Given distribution ensembles $\{X_n\}, \{Y_n\}$, we write $X_n \approx Y_n$ to denote that X_n is computationally indistinguishable to Y_n.

PSI functionality. A private set intersection (PSI) protocol allows two parties to compute the intersection of their input sets while concealing all other information. We typically denote by n the input set sizes. For completeness, the ideal functionalities for PSI (in the semi-honest and in the malicious settings) are given in Appendix A of the Supplementary Material.

Learning parity with noise. Our protocols are built on top of pseudorandom correlation generators (PCGs). State of the art constructions of PCGs rely on various flavors of the learning parity with noise (LPN) assumption. Since we make a black-bow use of PCGs, our work will be essentially oblivious to the underlying assumptions. However, for the sake of completeness, we recall the assumptions which we build upon in Appendix A of the Supplementary Material. We note that, for our second contribution, we build upon the PCG of [BCG+20b]. The latter uses a relatively new flavor of the ring-LPN assumption, over a polynomial ring where the polynomial splits completely; however, in this work, we do *not* need this new flavor, and instead rely solely on the (relatively well-established) standard ring-LPN assumption over a polynomial ring with an irreducible polynomial.

Pseudorandom correlation generators (PCG). Pseudorandom correlations generators have been introduced in a recent line of work [BCGI18,BCG+19b,BCG+19a]. A PCG allows to compress long correlations into short, correlated seeds that can later be locally expanded into pseudorandom instances of the target correlation. Slightly more formally, a PCG for a target correlation C (which samples pairs of long correlated strings (y_0, y_1)) is a pair $(\mathsf{Gen}, \mathsf{Expand})$ of algorithms such that $\mathsf{Gen}(1^\lambda)$ outputs a pair of short, correlated keys $(\mathsf{k}_0, \mathsf{k}_1)$ and $\mathsf{Expand}(\sigma, \mathsf{k}_\sigma)$ outputs a long string $\tilde{y}_\sigma$. Correctness states that $(\tilde{y}_0, \tilde{y}_1)$ are indistinguishable from a random sample from C, while security states that given $\mathsf{k}_{1-\sigma}$, $\tilde{y}_\sigma$ looks like a random sample from C conditioned on satisfying the target correlation with $\mathsf{Expand}(1-\sigma, \mathsf{k}_{1-\sigma})$, for $\sigma = 0, 1$.

A PCG does not in itself provide a protocol to efficiently generate long pseudorandom correlations. To get the latter, one must combine a PCG with a *distributed key generation* protocol, which allows two parties to obliviously run $\mathsf{Gen}(1^\lambda)$ such that each party gets one of the keys. Fortunately, for most PCGs of interest (and in particular, for all PCGs we use in this work), there exists very efficient low-communication distributed setup protocols [BCG+19a, BCG+20b]. Combining a PCG with a distributed setup protocols allows to securely instantiate (with low communication) functionalities that distribute instances of the target correlation. In this work, we will directly rely in a black-box way on such functionalities, and use known protocols to instantiate them. We now expand on the two main functionalities we use in this work.

Subfield Vector-OLE. We described the subfield vector-OLE correlation in the technical overview (see Section 2.2). We represent on Figure 1 the ideal functionality that distributes a subfield VOLE correlation. In our concrete instantiations, we will instantiate this functionality using the efficient protocol of [BCG+19a]. The latter provides a general template which can be instantiated under various flavors of the LPN assumption, and provides a conservative choice under LPN for quasi-cyclic choice. A variant of LPN that leads to a considerably more efficient protocol, when plugged in the template of [BCG+19a], was recently put forth in the work [CRR21] (we note that our communications estimate are oblivious to the underlying variant: only the computational costs depends on the LPN flavor).

PARAMETERS:

- 2 parties, a sender and receiver.
- A finite field $\mathbb{F}_q$ where $q = p^r$, p is a power of prime, r an integer.
- An integer n, the size of the output vector.

FUNCTIONALITY:

- The sender gets 2 random vectors $\mathbf{u} \in \mathbb{F}_p^n, \mathbf{v} \in \mathbb{F}_q^n$.
- The receiver receives $\Delta \in \mathbb{F}_q$ and a vector $\mathbf{w}$ such that $\mathbf{w} = \Delta\mathbf{u} + \mathbf{v} \in \mathbb{F}_q^n$.

Fig. 1. Ideal functionality $\mathcal{F}_{\mathsf{svole}}$ of length n on the field $\mathbb{F}_q$ over the subfield $\mathbb{F}_p$

4 Fast Cuckoo Hashing-Based PSI from Subfield-VOLE

4.1 Correlation Robustness

The security of our OPRF construction reduces to an appropriate correlation robustess assumption, which we state below. Correlation robustness was first introduced in [IKNP03], and later generalized in [KKRT16, PRTY19, KK13, CM20] to a form of *Hamming* correlation robustness.

Our notion of correlation robustness differs from that of KKRT on two aspects. First, because all operations are done over a finite field, we do not need Hamming correlation robustness; our notion is much closer in spirit to the original notion of IKNP. Second, our variant fixes a mistake in the notion used in KKRT: the notion of correlation robustness which they define is vacuous, in that there cannot exist any hash function for which it holds. Technically, this is because the notion must include some appropriate condition to prevent collisions between the hash function inputs, which are missing from the KKRT definition. Stating this condition formally is somewhat tedious if we want to avoid a very large security loss, because we need to specify a notion of collisions happening "with respect to the same index" which is tailored to its use in the OPRF construction. We elaborate on the mistake in

KKRT in Appendix B, and note that the KKRT notion can nevertheless be fixed (and their proof of security adapted) using a Hamming variant of our (more cumbersome) notion below.

Definition 1 ($(\kappa, n, k, \mathbb{F}_p, \mathbb{F}_{p^r})$-Correlation robustness.). *Let κ be a security parameter, p be a prime power, $q = p^r \approx O(2^\kappa)$, $n = \mathsf{poly}(\kappa)$, $k = o(\kappa)$, and H be a hash function: $\{0,1\}^* \times \mathbb{F}_q \to \{0,1\}^v$. Then H is a $(\kappa, n, k, \mathbb{F}_p, \mathbb{F}_{p^r})$-correlation robust if for every $(k_1, \cdots, k_n) \in \{1, \cdots, k\}^n$, any vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_n}$ where $\mathbf{u_i} \in (\mathbb{F}_p \backslash \{0\})^{k_i}$, and any vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ where $\mathbf{v_i} \in \mathbb{F}_{p^r}^{k_i}$ has pairwise distinct entries, the following distribution, induced by the random sampling of $\Delta \leftarrow \mathbb{F}_q$, is pseudorandom (i.e., the advantage of any adversary in distinguishing the distribution from the uniform distribution is negligible in κ):*

$$\begin{aligned}
&\mathsf{H}(1, v_{1,1} - \Delta u_{1,1}), \cdots, \mathsf{H}(1, v_{1,k_1} - \Delta u_{1,k_1})\\
&\mathsf{H}(2, v_{2,1} - \Delta u_{2,1}), \cdots, \mathsf{H}(2, v_{2,k_2} - \Delta u_{2,k_2})\\
&\vdots\\
&\mathsf{H}(n, v_{n,1} - \Delta u_{n,1}), \cdots, \mathsf{H}(n, v_{n,k_n} - \Delta u_{n,k_n})
\end{aligned}$$

4.2 Random Oracle Model Analysis

Definition 1 is relatively complex, and it might not be obvious at first sight why it is a plausible assumption. Nevertheless, we show that it holds unconditionally if H is modeled as a random oracle.

Game 0. For convenience, we reformulate the definition in the form of a game between a Q-query adversary $\mathcal{A}$ and a challenger. The game proceeds as follows:

1. $\mathcal{A}$ chooses vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_n}$ where $\mathbf{u_i} \in (\mathbb{F}_p \backslash \{0\})^{k_i}$, and vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ where the $\mathbf{v_i} \in \mathbb{F}_{p^r}^{k_i}$ have pairwise distinct entries, and submits these sequences to the challenger.
2. The challenger samples randomly $\Delta \leftarrow_r \mathbb{F}_q$ then defines a sequence of strings $z_{i,j} = (i, v_{i,j} - \Delta u_{i,j})_{i \leq n, j \leq k_i} \in \mathbb{F}_q$ and tosses a coin $b \leftarrow_r \{0,1\}$.
   - If $b = 0$, the challenger sends to $\mathcal{A}$ all strings $\mathsf{H}(z_{1,1}), \ldots, \mathsf{H}(z_{n,k_n})$.
   - If $b = 1$, the challenger samples $\sum_{i=1}^n k_i$ uniformly random v-bit strings, and sends them to $\mathcal{A}$.
3. Let $w_{1,1}, \ldots, w_{n,k_n}$ denote the sequence received by $\mathcal{A}$. Then, $\mathcal{A}$ outputs a bit b'. He *wins* if $b = b'$. The advantage of $\mathcal{A}$ is defined by:

$$\mathrm{Adv}(\mathcal{A}, \mathsf{H}) = \Pr(\mathcal{A} \textit{ wins}) - 1/2 = \epsilon$$

The adversary can make up to Q queries to the random oracle at any point during the game.

Game 1. In this game, we add the following check to the original game: in step 2, if there exists an index i such that $v_{i,j_1} - \Delta u_{i,j_1} = v_{i,j_2} - \Delta u_{i,j_2}$ for two distinct indices $j_1, j_2 \leq k_i$, we abort the game. Let E denote this event. Since Δ is sampled uniformly at random from $\mathbb{F}_q$ after receiving the vectors $\mathbf{u_i}$ and $\mathbf{v_i}$, and since $k_i \leq k$ for every i, using a straightforward union bound, E happens with probability at most $k^2 n / 2^\kappa$.

Now, let us analyze the probability that $\mathcal{A}$ wins the game, conditioned on $\neg E$. Because the game did not abort, we know that the strings $z_{i,j}$ are all distinct ($z_{i,j}$ is always different from $z_{i',j}$ for $i' \neq i$ by definition, and the condition guarantees that no collision occurs between $z_{i,j}$'s for the same i). This means that both when $b = 0$ and when $b = 1$, the values $v_{i,j}$ are sampled uniformly and independently at random. Let L denote the list of queries of $\mathcal{A}$. We have:

$$\begin{aligned}
\Pr\big[\mathcal{A} \textit{ wins} | \neg E\big] &= \Pr\big[\mathcal{A} \textit{ wins} \mid \exists i \in [n], j \in [k_i]\ ,\ \ z_{i,j} \in L\big] \cdot \Pr\big[\exists i, j\ :\ z_{i,j} \in L\big]\\
&\quad + \Pr\big[\mathcal{A} \textit{ wins} \mid \nexists i \in [n], j \in [k_i]\ ,\ \ z_{i,j} \in L\big] \cdot \Pr\big[\nexists i, j\ :\ z_{i,j} \in L\big]\\
&\leq 1 \cdot \sum_{i \leq n} \sum_{j \leq k_i} \Pr\big[z_{i,j} \in L\big] + \frac{1}{2} \cdot 1\\
&\leq \frac{1}{2} + \sum_{i \leq n} \sum_{j \leq k_i} \sum_{\ell \leq Q} \Pr\big[z_{i,j} = q_\ell\big] = \frac{1}{2} + \frac{nkQ}{2^\kappa},
\end{aligned}$$

where the last equality holds because the condition $z_{i,j} = q_\ell$, implies $\Delta = (v_{i,j} - q_\ell)/u_{i,j}$, hence $\Pr[z_{i,j} = q_\ell]$ can be bounded by $1/2^\kappa$.

Putting everything together, we have

$$\text{Adv}(\mathcal{A}, \mathsf{H}) \le \frac{nk^2}{2^\kappa} + \frac{nkQ}{2^\kappa} = \frac{nk(Q+k)}{2^\kappa},$$

which concludes the proof.

4.3 A new membership BaRK-OPRF

An OPRF [FIPR05] is a two-party protocol that, given a key k from the sender and an input element x from the receiver, computes and outputs $F_k(x)$ to the receiver. The sender obtains no output and learns no information about x while the receiver learns no information about k.

In this section, we present a new variant of OPRF called *membership* batch, related-key OPRF (mBaRK-OPRF). The purpose behind the construction of mBaRK-OPRF is to handle the case when a general cuckoo hashing has more than one item per bin, namely d items. Our mBaRK-OPRF allows the sender to hold a set of keys $(K_1, K_2, \ldots, K_n)$ such that each key is assigned with a tuple of d input elements of the receiver and then the receiver learns a *relaxed* PRF output on each element in this tuple corresponding with the same key. More formally, denoting $\mathbf{x_i} = (x_i^{(1)}, \cdots, x_i^{(d)})$ consisting of d entries, the sender gets $F_{\Delta,K_i}(y)$ and the receiver obtains a set $\{F_{\Delta,K_i}(x_i^{(j)})\}_{j\in[1,d]}$ such that $F_{\Delta,K_i}(y) \in \{F_{\Delta,K_i}(x_i^{(j)})\}_{j\in[1,d]}$ if and only if y is equal to any entry of $\mathbf{x_i}$, and $F_{\Delta,K_i}(y)$ looks pseudorandom to the sender otherwise.

Our mBaRK-OPRF can be considered as optimization of the KKRT BaRK-OPRF, and its functionality is shown on Figure 2. A PRF F is called a *relaxed* PRF if there is another function $\widetilde{F}$ such that $F(K, x)$ can be efficiently computed from $\widetilde{F}(K, x)$, but pseudorandomness on other points holds even when given relaxed evaluations on several points. mBaRK-OPRF generates a batch of $n = N \cdot d$ instances PRF F with N related keys where each key is corresponding with a tuple of d input elements, allows the receiver learns a *relaxed* PRF output on each input.

PARAMETERS:

- The functionality is parameterized by a relaxed PRF F.
- $\mathbb{F}$ is a finite field. There are 2 parties, a sender (with no input) and a receiver with input set $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{F}^n$ where $n = N \cdot d$; d, N are integers.

FUNCTIONALITY:

- Sample uniformly in random a sequence $(\Delta; K_1, K_2, \ldots, K_N)$ of related keys for F and give it to the sender.
- For each $i \in [1, N]$, give $\{\widetilde{F}_{\Delta,K_i}(x_j)\}_{(i-1)d+1\le j\le id}$ to the receiver.

Fig. 2. Ideal Functionality $\mathcal{F}_{\mathsf{moprf}}$ of batch, related-key mBaRK-OPRF

Main construction. Our mBaRK-OPRF is constructed from a PCG primitive, *subfield* VOLE. Our construction Π_{OPRF} is detailed on Figure 3 and realizes the $\mathcal{F}_{\mathsf{moprf}}$ functionality from Figure 2 in the semi-honest setting. We make use of a correlation robust hash function $\mathsf{H} : \{0,1\}^* \times \mathbb{F}_q \mapsto \{0,1\}^v$. Assume that the receiver inputs the set of $n = N \cdot d$ elements: $X = \{x_1, x_2, \ldots, x_{Nd}\}$ and all of these elements are in $\mathbb{F}_p$. This input sets can be divided into N tuples $\mathbf{x_i}$ of d elements such that the tuple $\mathbf{x_i}$ includes d elements $\{x_{(i-1)d+1}, x_{(i-1)d+2}, \ldots, x_{id}\}$ for $i \in [1, N]$.

First, the sender and the receiver invoke the $\mathcal{F}_{\mathsf{svole}}$ protocol of dimension $N \cdot d$, with their roles reversed, to get a random sVOLE correlation. Specifically, the receiver learns a pair of vectors $(\mathbf{u}, \mathbf{v})$ where $\mathbf{u} \in \mathbb{F}_p^{Nd}$, $\mathbf{v} \in \mathbb{F}_q^{Nd}$, the sender gets $\Delta \in \mathbb{F}_q$ and $\mathbf{w} := \Delta \cdot \mathbf{u} + \mathbf{v}$. We have that, for all $i \in [1, n]$:

$$v_i = w_i - \Delta \cdot u_i \in \mathbb{F}_q$$

At a high level, for each tuple input $\mathbf{x_i}$ of the receiver, the goal is that on any input, the sender can compute the value of a random polynomial associated with $\mathbf{x_i}$ without leaking information. This can be done by hiding the coefficients of this polynomial with the output vector $\mathbf{u}$ of $\mathcal{F}_{\mathsf{svole}}$. Consider the tuple $\mathbf{x_i}$ consisting of d elements $\{x_{(i-1)d+1}, x_{(i-1)d+2}, \ldots, x_{id}\}$, and write its associated polynomial as

$$P_{\mathbf{x_i}}(X) = \prod_{j=1}^{d}(X - x_{(i-1)d+j}) = X^d + \sum_{j=1}^{d} c_{(i-1)d+j} \cdot X^{j-1}$$

where $c_{(i-1)d+j} \in \mathbb{F}_p$ for $i \in [1, N]$, $j \in [1, d]$. Now, the receiver defines $\mathbf{c} := (c_1, c_2, \ldots, c_{Nd})$, and then sends to the sender a vector $\mathbf{z} := \mathbf{c} - \mathbf{u} \in \mathbb{F}_p^{Nd}$. Above, the $u_{(i-1)d+j}$ mask for the coefficient of degree $j-1$ of (the polynomial associated) $\mathbf{x_i}$ where $i \in [1, N]$, $j \in [1, d]$. Indeed, since all positions in $\mathbf{u}$ are distributed uniformly at random in the subfield $\mathbb{F}_p$, the vector $\mathbf{z}$ is a uniformly random over $\mathbb{F}_p^{Nd}$ from the viewpoint of the sender For $i \in [1, N]$, the receiver defines the *relaxed* PRF output $\left(i\ ,\ \mathbf{x_i}\ ,\ \{v_{(i-1)d+k}\}_{1 \le k \le d}\right)$. Then the PRF output on each input of the receiver is computed as

$$F\left(K_i, x_{(i-1)d+j}\right) = \mathsf{H}\left(i\ ,\ \sum_{k=1}^{d} v_{(i-1)d+k} \cdot x_{(i-1)d+j}^{k-1}\right)$$

for $i \in [1, N]$, $j \in [1, d]$. On the other hand, after receiving the vector $\mathbf{z}$, for $i \in [1, N]$, the sender defines the vector $\mathbf{k} := \mathbf{w} + \Delta \cdot \mathbf{z}$ and the set of keys $\left(i\ ,\ \Delta\ ,\ \{k_{(i-1)d+j}\}_{1 \le j \le d}\right)$. As a consequence, for any input $y \in \mathbb{F}_p$, its PRF output is computed as:

$$F\left(K_i, y\right) = \mathsf{H}\left(i\ ,\ \Delta \cdot y^d + \sum_{j=1}^{d} k_{(i-1)d+j} \cdot y^{j-1}\right)$$

for $i \in [1, N]$.

Correctness. Let's consider each related-key K_i ($i \in [1, N]$). Since $\mathbf{z} := \mathbf{c} - \mathbf{u}$ and $\mathbf{w} := \Delta \cdot \mathbf{u} + \mathbf{v}$, for any $i \in [1, N]$, $j \in [1, d]$, we have

$$\begin{aligned}
&\Delta \cdot y^d + \sum_{j=1}^{d} k_{(i-1)d+j} \cdot y^{j-1} \\
&= \Delta \cdot y^d + \sum_{j=1}^{d} \left(w_{(i-1)d+j} + \Delta \cdot z_{(i-1)d+j}\right) \cdot y^{j-1} \\
&= \Delta \cdot y^d + \sum_{j=1}^{d} \left(\Delta \cdot u_{(i-1)d+j} + v_{(i-1)d+j} + \Delta \cdot c_{(i-1)d+j} - \Delta \cdot u_{(i-1)d+j}\right) \cdot y^{j-1} \\
&= \Delta \cdot y^d + \sum_{j=1}^{d} \left(v_{(i-1)d+j} + \Delta \cdot c_{(i-1)d+j}\right) \cdot y^{j-1} \\
&= \Delta \cdot \left(y^d + \sum_{j=1}^{d} c_{(i-1)d+j} \cdot y^{j-1}\right) + \sum_{j=1}^{d} v_{(i-1)d+j} \cdot y^{j-1} \\
&= \Delta \cdot P_{\mathbf{x_i}}(y) + \sum_{j=1}^{d} v_{(i-1)d+j} \cdot y^{j-1}.
\end{aligned}$$

Then, $\quad y = x$ and $x \in \mathbf{x_i} \quad \Longleftrightarrow \quad P_{\mathbf{x_i}}(y) = 0.$

$$\Longrightarrow F\left(K_i, y\right) = \mathsf{H}\left(i\ ,\ \sum_{j=1}^{d} v_{(i-1)d+j} \cdot y^{j-1}\right) = F\left(K_i, x\right),$$

which concludes the proof of correctness.

PARAMETERS:

- Given $\mathbb{F}_p \subseteq \mathbb{F}_q$ $(q = p^r)$, $(\kappa, n, k, \mathbb{F}_p, \mathbb{F}_{p^r})$-Correlation robustness: $\mathsf{H} : \{0,1\}^* \times \mathbb{F}_q \to \{0,1\}^v$.
- The sender has no input and the receiver inputs a set $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathbb{F}_p^n$ where $n = N \cdot d$, N, d are integers.

PROTOCOL:

1. The sender and the receiver invoke to the $\mathcal{F}_{\mathsf{svole}}$ of dimension $N \cdot d$ in the $\mathbb{F}_q$ over the $\mathbb{F}_p$ with the inverse role. The receiver gets two random vectors $\mathbf{u} \in \mathbb{F}_p^n, \mathbf{v} \in \mathbb{F}_q^n$ and the sender receives $\Delta \in \mathbb{F}_q$, $\mathbf{w} := \Delta \mathbf{u} + \mathbf{v} \in \mathbb{F}_q^n$.
2. The receiver divides its input set as N tuples such that the tuple $\mathbf{x_i}$ consisting of d elements $\{x_{(i-1)d+1}, x_{(i-1)d+2}, \dots, x_{id}\}$ for $i \in [1, N]$. The receiver then determines the associated polynomial for each tuple $\mathbf{x_i}$ as
$$P_{\mathbf{x_i}}(X) = \prod_{j=1}^{d}(X - x_{(i-1)d+j}) = X^d + \sum_{j=1}^{d} c_{(i-1)d+j} \cdot X^{j-1}$$
where $c_{(i-1)d+j} \in \mathbb{F}_p$ for $i \in [1, N]$, $j \in [1, d]$.
3. Now, the receiver defines $\mathbf{c} := (c_1, c_2, \dots, c_{Nd})$, and then sends to the sender a vector
$$\mathbf{z} := \mathbf{c} - \mathbf{u} \in \mathbb{F}_p^{Nd}$$
4. For $i \in [1, N]$, the receiver outputs the *relaxed* PRF output
$$\big(i\ ,\ \mathbf{x_i}\ ,\ \{v_{(i-1)d+k}\}_{1 \le k \le d}\big)$$
then computes the PRF value as
$$F\left(K_i, x_{(i-1)d+j}\right) = \mathsf{H}\left(i\ ,\ \sum_{k=1}^{d} v_{(i-1)d+k} \cdot x_{(i-1)d+j}^{k-1}\right)$$
for $i \in [1, N]$, $j \in [1, d]$.
5. The sender defines the vector $\mathbf{k} := \mathbf{w} + \Delta \cdot \mathbf{z}$ and for $i \in [1, N]$ the set of keys
$$\big(i\ ,\ \Delta\ ,\ \{k_{(i-1)d+j}\}_{1 \le j \le d}\big)$$
The sender computes the PRF output on a given input $y \in \mathbb{F}_p$ by below formula
$$F\left(K_i, y\right) = \mathsf{H}\left(i\ ,\ \Delta \cdot y^d + \sum_{j=1}^{d} k_{(i-1)d+j} \cdot y^{j-1}\right)$$
for $i \in [1, N]$.

Fig. 3. Our batch mBaRK-OPRF Π_{moprf} based on $\mathsf{subVOLE}$

4.4 The security of relaxed PRF

The set of keys generated by the ideal functionality of mBaRK-OPRF (figure 2) parametrized with our new PRF are the common key $K := \Delta \in \mathbb{F}_q$, and the set of related-keys $K_i := \big(i\ ,\ \Delta\ ,\ \{k_{(i-1)d+j}\}_{1\le j\le d}\big)$, for $i = 1$ to N. The receiver in our mBaRK-OPRF is allowed to learn $\widetilde{F}(K_i, x_i^{(j)})$ for $j = 1$ to d, defined as

$$\widetilde{F}\,(K_i, y) := \big(i\ ,\ (k_{(i-1)d+j} - \Delta \cdot c_{(i-1)d+j})_{j\le d}\big) = \big(i\ ,\ (v_{(i-1)d+j})_{j\le d}\big)\,,$$

where the $c_{(i-1)d+j}$ are the coefficients of $P_{\mathbf{x_i}}$. Then, the receiver can compute $\chi_i(x) = \sum_{j=1}^{d} v_{(i-1)d+j} \cdot x^{j-1}$ from $\widetilde{F}(K_i, x)$ and set $F(K_i, x) = \mathsf{H}(i, \Delta \cdot P_{\mathbf{x_i}}(x) + \chi_i(x))$, where $\mathsf{H} : \{0,1\}^* \times \mathbb{F}_q \to \{0,1\}^v$ is a correlation robust hash function over the subfield $\mathbb{F}_p \subseteq \mathbb{F}_q$.

We now formulate the security of *relaxed* PRF. The definition is imported from KKRT and adapted to our mBaRK-OPRF.

Definition 2. *Let F be a relaxed PRF with output length v. F has m-*related-key PRF security *if the advantage of any PPT adversary in the following game is negligible:*

1. *The adversary chooses strings $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{F}_p^n$ consisting of N tuple $\mathbf{x_i} = (x_{(i-1)d+1}, x_{(i-1)d+2}, \ldots, x_{id})$, and m pairs $(j_1, y_1), (j_2, y_2), \ldots, (j_m, y_m)$ where $y_i \notin \mathbf{x}_{j_i}$.*
2. *The challenger picks PRF keys $(K; K_1, K_2, \ldots, K_N)$ and $b \leftarrow_r \{0,1\}$.*
   - *If $b = 0$, the challenger outputs $\widetilde{F}(K_i, x_{(i-1)d+j})$ for $i \in [1,N]$, $j \in [1,d]$, and $\{F(K_{j_i}, y_i)\}_{i\in[1,m]}$.*
   - *If $b = 1$, the challenger chooses $z_1, \ldots, z_m \leftarrow \{0,1\}^v$ and outputs the relaxed evaluations $\widetilde{F}(K_i, x_{(i-1)d+j})$ for $i \in [1,N], j \in [1,d]$, and $\{z_i\}_{i\in[1,m]}$.*
3. *The adversary outputs a bit b_0; its advantage is $\Pr[b = b_0] - 1/2$.*

Lemma 3. *Let H be a $(\kappa, m, k, \mathbb{F}_p, \mathbb{F}_{p^r})$-correlation robust hash function (section 1). Then, our relaxed PRF is m-related-key secure.*

Proof. First, we claim that the adversary learns no information about Δ from the outputs of $\widetilde{F}$ on a chosen input set. Indeed, if $x \in \mathbf{x_i}$ then $P_{\mathbf{x_i}}(x) = 0$. Using the equation established in our proof of correctness, we have:

$$\widetilde{F}\,(K_i, x) = \big(i\ ,\ (v_{(i-1)d+j})_{j\le d}\big) \text{ for all } x \in \mathbf{x_i},$$

which are independent of Δ. For any input (j_i, y_i) the adversary knows $P_{\mathbf{x_{j_i}}}(y_i) \in \mathbb{F}_p$, and $\sum_{j=1}^{d} v_{(j_i-1)d+j} \cdot {y_i}^{j-1} \in \mathbb{F}_q$. Notice that $y_i \notin \mathbf{x_{j_i}}$ then $P_{\mathbf{x_{j_i}}}(y_i) \neq 0$. Therefore, the pseudorandomness of the m PRF outputs follows directly from the $(\kappa, m, k, \mathbb{F}_p, \mathbb{F}_{p^r})$-correlation robustness of H.

Theorem 4. *The protocol Π_{moprf} (Figure 3) instantiated with an $(\kappa, m, k, \mathbb{F}_p, \mathbb{F}_{p^r})$-correlation robust hash function, securely realizes the ideal functionality of $\mathcal{F}_{\mathsf{moprf}}$ against a semi-honest adversary in the $\mathcal{F}_{\mathsf{svole}}$ hybrid model.*

Proof. **Corrupted sender.** The simulator Sim interacts with the sender as follows:

1. Sim queries $\mathcal{F}_{\mathsf{moprf}}$ on behalf of the sender, and receives the set of keys for $i \in [1,N]$
$$\big(i\ ,\ \Delta\ ,\ \{k_{(i-1)d+j}\}_{1\le j\le d}\big)\,.$$
Sim aborts if $\Delta = 0$; this happens with probability at most $1/2^\kappa$.
2. Sim simulates $\mathcal{F}_{\mathsf{svole}}$ using the value Δ received from $\mathcal{F}_{\mathsf{moprf}}$, and a uniformly random vector $\mathbf{w}$.
3. Sim defines the vector $\mathsf{k} := \{k_1, k_2, \ldots, k_n\}$ and sets $\mathbf{z} := \Delta^{-1} \cdot (\mathbf{k} - \mathbf{w})$ (recall that $\Delta \neq 0$).
4. On behalf of the receiver, Sim sends the vector $\mathbf{z}$ to the sender. This simulation is statistically indistinguishable from the real execution.

Corrupted receiver. The sender does not send anything to the receiver. Sim simply calls $\mathcal{F}_{\mathsf{moprf}}$ on behalf of the receiver, and receives the relaxed evaluations $\widetilde{F}(K_i, x) = (i, (v_{(i-1)d+j})_{j\le d})$ for $i = 1$ to N and $x \in \mathbf{x_i}$. Sim emulates the role of $\mathcal{F}_{\mathsf{svole}}$ by picking a uniformly random vector $\mathbf{u}$, and setting $\mathbf{v}$ using the $v_{(i-1)d+j}$ values from the relaxed PRF evaluations. This concludes the proof.

4.5 A semi-honest PSI based on subfield VOLE

A variant of mBaRK-OPRF. We propose a variant of our mBaRK-OPRF to deal with the case when the size of each tuple input is not necessarily equal to d. This means that the receiver now can divide the input set to N tuples $\mathbf{x_i}$ and each tuple has less than or equal to d elements. Meanwhile, the sender is not allowed to learn about how many exactly items are in each tuple. This functionality can be obtained from our mBaRK-OPRF plus a small extra cost, i.e, a *subfield* VOLE of length N over the subfield $\mathbb{F}_2$.

The idea is as follows. The receiver's input set has N tuples $\mathbf{x_i} = \{x_{i,1}, x_{i,2}, \ldots, x_{i,j_1}\}$ for $i \in [1, N]$ and $j_i \leq d$. The polynomial associated to each tuple $\{\mathbf{x_i}\}_{i \leq N}$ will be expressed as a polynomial of degree d

$$P_{\mathbf{x_i}}(X) = \prod_{j=1}^{j_i}(X - x_{i,j}) = \sum_{j=j_1+1}^{d+1} c_{i,j} \cdot X^{j-1} + X^{j_i} + \sum_{j=1}^{j_i} c_{i,j} \cdot X^{j-1}$$

where $c_{i,j} \in \mathbb{F}_p$ for $j \in [1, j_i]$ and $c_{i,j} = 0$ for $j \in [j_i + 1, d + 1]$.

As a result, the set of the coefficients of $P_{\mathbf{x_i}}(X) = (c_{i,1}, c_{i,2}, \ldots, c_{i,d+1}\}$. We remark that, compared to the associated polynomial in our original mBaRK-OPRF which has a constant coefficient of degree d of 1, in our variant version this coefficient will equal 0 or 1 since the degree of $P_{\mathbf{x_i}}(X)$ is *less* than or equal to d. So, it requires $(d + 1)$ masks for this polynomial instead of d, but the mask for the coefficient of degree d just needs to be in $\mathbb{F}_2$. For each tuple, we require an additional value $u_i \in \mathbb{F}_2$, so in total we need an additional subfield VOLE of length N over the subfield $\mathbb{F}_2$.

More formally, the sender and receiver invoke a subfield VOLE of length $N \cdot d$ over the subfield $\mathbb{F}_p$ as before (all the notations in figure 3 are reused), and additionally invoke another subfield VOLE instance over the subfield $\mathbb{F}_2$ of length N with an inverse role, while the receiver gets $\mathbf{u}' \in \mathbb{F}_2^N$, and $\mathbf{v}' \in \mathbb{F}_q^N$ the sender holds $\Delta \in \mathbb{F}_q$ (Δ is the same for each time invoking *subfield* VOLE) and $\mathbf{w}' := \Delta \cdot \mathbf{u}' + \mathbf{v}'$. The receiver sends to the sender a vector $\mathbf{z}$, and $\mathbf{z}'$ defined as

$$\mathbf{z} \in \mathbb{F}_p^{N.d} \ , \ \mathbf{z}' \in \mathbb{F}_2^N$$

where $z_{(i-1)d+j} := c_{i,j} - u_{(i-1)d+j}$, $z'_i := c_{i,d+1} - u'_i$ for $i \in [1, N]$, $j \in [1, d]$.

The receiver outputs are computed

$$F(K_i, x_{i,j}) = \mathsf{H}\left(i \ , \ v'_i \cdot x_{i,j}^d + \sum_{k=1}^{d} v_{(i-1)d+k} \cdot x_{i,j}^{k-1}\right)$$

for $i \in [1, N]$, $j \in [1, j_i]$. On the other hand, after receiving the vector $\mathbf{z}$, $\mathbf{z}'$, the sender defines their PRF values as:

$$F(K_i, y) = \mathsf{H}\left(i \ , \ (w'_i + \Delta \cdot z'_i) \cdot y^d + \sum_{j=1}^{d} k_{(i-1)d+j} \cdot y^{j-1}\right)$$

for $i \in [1, N]$.

Main construction of a new PSI. The sender and the receiver have two input sets $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$. Assume that all of these elements have the bit-length ℓ. Our PSI protocol is described on Figure 4; it builds upon the protocol of [KKRT16].

In our protocol, the receiver first uses a (d, k)-cuckoo hashing to map his input set Y to a table with N bins, using k hash functions such that each element is assigned to exactly one bin and each bin contains at most d elements. The k hash functions in (d, k)-generalized Cuckoo hashing are constructed using the permutation-based hashing scheme of [PSSZ15] to reduce the bit-length of the values stored in a bin from ℓ to $\ell - \log n$. Depending on the size of n, we use one of two approaches to handle the bins which are not full (the threshold was chosen empirically to optimize communication).

- If $n \geq 2^{20}$, the variant of our mBaRK-OPRF (using an additional subfield VOLE over $\mathbb{F}_2$) is used; for such sizes, the concrete cost of implementing the additional sVOLE vanishes. Notice that each bin is considered as a tuple consisting of *less* than or equal d elements.

– Otherwise, when $n < 2^{20}$, the receiver adds dummy items of length $\ell - \log n$ to bins such that each bin contains *exactly* d items. To avoid collisions between the dummy items and the elements in the same bin of the sender, we pad an extra bit to all items in the following way: $i||x||b$ where i is the index of hash function corresponding with the stored value x while $b = 1$ if x is a dummy item added and $b = 0$ otherwise.

In both cases, the receiver gets the set of n OPRF values while the sender learns the set of N related-key K_i. The sender uses k hash functions to determine which bins its elements are assigned, i.e, the possible corresponding keys of each element, and then computes PRF values (depending on the size n to formulate the input element for PRF). In total, the sender computes $k \cdot n$ PRF evaluations and sends them (randomly shuffled) to the receiver, who compares it with his OPRF outputs, and outputs the intersection set. To reduce the computational cost in this step, the sender can send separately each set H_i ($i \in [1, k]$) which contains the PRF outputs of each $x \in X$ with the related key $K_{h_i(x)}$. Then for each element, the receiver only needs to search for one set (among k sets H_i) of n items instead of $k \cdot n$.

PARAMETERS:

– A field $\mathbb{F}_q$ with subfield $\mathbb{F}_p \subseteq \mathbb{F}_q$.
– The sender and the receiver have respectively input set $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, all with elements of bit-length ℓ.
– k hash functions $h_1, h_2, \ldots, h_k : \{0,1\}^* \to [N]$ where $N \cdot d > n$ and $d = O(1)$ (see Section 4.6).
– A (d, k)-generalized Cuckoo hashing (GCH) scheme mapping n items to N bins.

PROTOCOL:

1. The receiver uses (d, k)-Cuckoo hashing with k hash functions to map the elements in Y to the table $\mathcal{B}$ consisting of N bins, where each bin has d empty slots.
   Denote the element assigned to the position $j \in [1, j_i]$ of bin $i \in [1, N]$ as $x_{i,j}$ and its stored value as $\mathcal{B}[i][j]$ then $\mathsf{GCH}(x_{i,j}) := \mathcal{B}[i][j]$, where j_i is the number of values stored in bin i.
2. Depending on the size of n, there are two alternatives:
   (a) $n \geq 2^{20}$, the sender and receiver invoke our variant of Π_{moprf} where the receiver uses the input set $Y_{\mathcal{B}} = \{r_{1,1}, r_{1,2}, \ldots, r_{1,j_1}; r_{2,1}, \ldots, r_{2,j_2}; \ldots, r_{N,1}, \ldots, r_{N,j_N}\}$ defined as follows:
      – The tuple of j_i elements $\{r_{i,1}, r_{i,2}, \ldots, r_{i,j_i}\}$ in bin i.
      – $r_{i,j} = (t \parallel \mathcal{B}[i][j])$ for $i \in [1, N], j \leq j_i$, and $t \in [1, k]$ such that $h_t(x_{i,j}) = i$
   (b) $n < 2^{20}$, the sender and receiver directly invoke the Π_{moprf} where the receiver uses the input set $Y_{\mathcal{B}} = \{r_{1,1}, r_{1,2}, \ldots, r_{1,d}; r_{2,1}, \ldots, r_{2,d}; \ldots, r_{N,1}, \ldots, r_{N,d}\}$ defined as follows:
      – The tuple of d elements $\{r_{i,1}, r_{i,2}, \ldots, r_{i,d}\}$ in bin i.
      – $r_{i,j} = (t \parallel \mathcal{B}[i][j] \parallel 1)$ for $i \in [1, N], j \leq j_i$ and $t \in [1, k]$ such that $h_t(x_{i,j}) = i$.
      – $r_{i,j} = (t \parallel \text{dummy value} \parallel 0)$ for $i \in [1, N]$, $j_i < i \leq d$ and $t \leftarrow_r [1, k]$.
3. The receiver obtains n instances *OPRF*:
$$Y' = \{\mathsf{PRF}(K_i\ ,\ r_{i,j}) \mid i \in [1, N]\ ,\ j \leq j_i\}$$
4. The sender uses the k hash functions to map the n element in X to the N bins (each element x mapped to each of the k bins given by the $h_t(x)$). Let x' denote the value stored when mapping x at $h_t(x)$.
5. The sender computes the sets of $k \cdot n$ PRF outputs:
   (a) For $n \geq 2^{20}$: $H_t = \left\{\mathsf{PRF}(K_{b_{t,x}}\ ,\ t \parallel x') \mid x \in X\right\}$ for $t \in [1, k]$
   (b) For $n < 2^{20}$: $H_t = \left\{\mathsf{PRF}(K_{b_{t,x}}\ ,\ t \parallel x' \parallel 1) \mid x \in X\right\}$ for $t \in [1, k]$
   Then the sender randomly permutes and sends each set to the sender.
6. The receiver finds the intersection:
   – if $y \in Y$ is mapped to the position j of bin i by function h_t then check whether $\mathsf{PRF}(k_i, r_{i,j}) \in H_t$ ($r_{i,j}$ is defined depending on n).
   – The sender outputs the intersection set.

Fig. 4. Our new semi-honest PSI protocol

4.6 Parameters

In this section, we discuss concrete parameters used in our new PSI semi-honest protocol. We use a computational security parameter $\kappa = 128$ and a statistical security parameter $\lambda = 40$. The protocol contains several parameters:

– The parameters of th generalized Cuckoo hashing scheme.
– The size of subfield $\mathbb{F}_p$ and field $\mathbb{F}_q$.
– The correlation robust hash function H output length v, e.g, the output length of PRF realized by the mBaRK-OPRF.

The length of OPRF output. We require that no collision occurs among PRF outputs for correctness. Setting the output domain of PRF would be $\{0,1\}^v$ where $v = \lambda + 2\log_2(n)$ guarantees a $2^{-\lambda}$ bound on the collision probability among the two size-n sets. Furthermore, communicating the hashes can be reduced to communicating only $\approx \lambda + \log n$ bits per hash, using a heuristic technique of [TLP$^+$17] that directly leads to an optimization of our PSI protocol.

The size of $\mathbb{F}_p$ and $\mathbb{F}_q$ in mBaRK-OPRF. Assume that the bit-length of all elements in the input sets of the sender and the receiver is ℓ bits. After using permutation-based hashing, each element is mapped to a bin with a stored value in this bin, the bit-length reduces from ℓ to $\ell - \log n$. The input set of mBaRK-OPRF in PSI protocol constructs from stored values concatenating with some extra bits. Then the bit-length of an input element of mBaRK-OPRF is computed as $\ell - \log n + 1$ if $n \geq 2^{20}$ or $\ell - \log n + 2$ otherwise, i.e, the size of $q = 2^{\ell - \log n + 1}$ or $q = 2^{\ell - \log n + 2}$ respectively.

Generalized Cuckoo hashing. We use a (d,k)-general cuckoo hashing scheme without stash. The parameters are chosen such that the failure probability is $2^{-\lambda}$. When $d = 1, k = 3$ these parameters are identical with KKRT except for the number of bins increases slightly to $N = 1.3n$ which is a trade-off to obtain no stash. Even with the higher number of bins, our PSI protocol significantly outperforms KKRT. To minimize the overall communication, we set $k = 2$ to reduce the cost of sending $k \cdot n$ PRF outputs. We used a Python script to simulate randomly assigning n values to $N = c \cdot n$ bins using $(d,2)$-Cuckoo hashing, for several values of d and c, and for $n = 2^9, 2^{10}, 2^{11}, 2^{12}$. For a value of c as low as 0.65, we never observed any insertion failure over 10^7 trials for each values of n (for $n = 2^{12}$, we could only do 10^6 trials), when using $d = 3$ items per bins. For $d = 2$, the failure probability became noticeable already for $c \approx 1$. Based on known theoretical analysis of (d,k)-Cuckoo hashing, the failure probability is known to scale inverse polynomially with n. Therefore, we expect that for reasonably large values of n (e.g. $n \geq 2^{18}$), our parameters should guarantee a failure probability significantly below 2^{-40}.

5 Standard Model Malicious PSI from Subfield-Ring-OLE

In this section, we describe a new PSI protocol, which builds upon a (simple variant of) a pseudorandom correlation generator for the ring-OLE correlation [BCG$^+$20b]. Our protocol enjoys a number of important features: it is in the *standard model*, achieves *malicious security* at essentially no cost, has *low communication* (competitive even with the best maliciously secure PSI protocols in the random oracle model), and reasonable computation (albeit superlinear in n). Furthermore, our protocol can achieve even smaller communication and computation in scenarios where the parties are guaranteed that the set intersection will be large. Our protocol can also be generalized to a powerful notion of *batch non-interactive PSI*, where (after a small logarithmic-cost preprocessing step with each server) a client can broadcast a single encoding of his database, and then obtain the intersection with any of the server databases at any time after receiving a single message from this server. We believe that this functionality itself is of independent interest.

Overview. We rely on the functionality, given on Figure 8, that generates a subfield ring-OLE correlation, over a rings $\mathcal{R}_\mathsf{p} = \mathbb{F}_\mathsf{p}[\mathsf{x}]/\mathsf{F}(\mathsf{x}), \mathcal{R}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^\mathsf{t}}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ where $q = p^r$ for some r, and $F(X)$ is some polynomial of degree $2n + 1$ (more generally, when the two parties have sets of different size n

and m, F will be of degree $n+m+1$). Our protocol makes a single black-box call to this functionality. Consider two parties, a receiver and a sender, where the receiver has a set $A = \{a_1, a_2, \ldots, a_n\} \in \mathbb{F}_p^n$ and the sender has a set $B = \{b_1, b_2, \ldots, b_n\} \in \mathbb{F}_p^n$. Define $p_A := \sum_{i=1}^n (x - a_i) \in \mathcal{R}_p$ and $p_B := \sum_{i=1}^n (x - b_i) \in \mathcal{R}_p$. Let $I := A \cap B$ denote the target output. The protocol computes the common roots of p_A and p_B ,i.e., $\gcd(p_A, p_B)$. To do so, the parties will call the functionality of Figure 8. By carefully revealing some combinations of their output correlation and their input polynomials, the parties achieve the following:

- The sender holds two uniformly random polynomials $r_A^0(x), r_B^0(x)$ of degree n over the ring $\mathcal{R}_q$;
- The receiver gets the polynomial $S(x) = p_A(x) \cdot r_A^0(x) + p_B(x) \cdot r_B^0(x)$.

Using Lemmas 10 and 11, $S(x)$ can be factored as $\gcd(p_A(X), p_B(X) \cdot U(x)$, where with high probability, $U(X)$ has no common root with $p_A(X)$. The receiver will then compute $S(a_i)$ for all $a_i \in A$ (using fast polynomial interpolation) and output $\{a_i \in A \ : \ S(a_i) = 0\}$.

5.1 A Malicious PSI from Subfield Ring-OLE

The functionality $\mathcal{F}_{\mathsf{sole}}$ is represented on Figure 8. Let F be an irreducible polynomial of degree $2n+1$. The receiver and the sender invoke $\mathcal{F}_{\mathsf{sole}}$ over $\mathcal{R}_q$ with subring $\mathcal{R}_p \subseteq \mathcal{R}_q$. The sender receives a pair of uniformly random polynomials $(r_A, r'_B) \in \mathcal{R}_q^2$, each of degree $2n$; let us rewrite r_A as $r_A = r_A^0 + r_A^1 \cdot X^n \in \mathcal{R}_q$ where r_A^0, r_A^1 are of degree at most n. The receiver receives a pair of polynomials, denoted (a', s_A), of degree $2n$ such that $a' \in \mathcal{R}_p$ and $s_A = a' \cdot r_A + r'_B \in \mathcal{R}_q$. Similarly, we rewrite a' as $a' = a'_0 + a'_1 \cdot X^n \in \mathcal{R}_p$, where a'_0 is a random polynomial of degree at most n over the subring $\mathcal{R}_p$. Let us consider the polynomial s_A without reduction to $F(X)$ then

$$\begin{aligned} s_A &= a' \cdot r_A + r'_B = r'_B + (a'_0 + a'_1 \cdot X^n) \cdot \left(r_A^0 + r_A^1 \cdot X^n\right) \\ &= \left(r'_B + a'_0 \cdot r_A^0\right) + \left(a'_0 \cdot r_A^1 . X^n + a'_1 \cdot r_A^0 \cdot X^n + a'_1 \cdot r_A^1 \cdot X^{2n}\right) \\ &= s_A^0 + \left(a'_0 \cdot r_A^1 \cdot X^n + a'_1 \cdot r_A^0 \cdot X^n + a'_1 \cdot r_A^1 \cdot X^{2n}\right) \in \mathbb{F}_q[X] \end{aligned}$$

The receiver computes $t_A := p_A - a'_0 \in \mathcal{R}_p$ and sends (t_A, a'_1) to the sender. Afterwards, the sender samples a random polynomial r_B^0 of degree n over the ring $\mathcal{R}_q$, and computes $s = p_B \cdot r_B^0 \in \mathcal{R}_q$ of degree $2n$. Then, the sender sets

$$s'_B \leftarrow a'_1 \cdot r_A^0 \cdot X^n \bmod F(X) \qquad\qquad s_B \leftarrow t_A \cdot r_A^0 + s - r'_B - s'_B,$$

and sends (s_B, r_A^1) to the receiver. Now, the receiver computes $s'_A = a'_0 \cdot r_A^1 . X^n + a'_1 \cdot r_A^1 \cdot x^{2n} \bmod F(x)$ and defines:

$$\begin{aligned} U = s_A + s_B - s'_A &= (a' \cdot r_A + r'_B \bmod F(X)) + t_A \cdot r_A^0 + s - r'_B - s'_B - s'_A \\ &= \left(r'_B + a'_0 \cdot r_A^0\right) + \left(a'_0 \cdot r_A^1 \cdot X^n + a'_1 \cdot r_A^0 \cdot X^n + a'_1 \cdot r_A^1 \cdot X^{2n} \bmod F(X)\right) \\ &\quad + t_A \cdot r_A^0 + s - r'_B - \left(a'_1 \cdot r_A^0 \cdot X^n \bmod F(X)\right) \\ &\quad - \left(a'_0 \cdot r_A^1 \cdot X^n + a'_1 \cdot r_A^1 \cdot X^{2n} \bmod F(X)\right) \\ &= a'_0 \cdot r_A^0 + t_A \cdot r_A^0 + s = a'_0 \cdot r_A^0 + (p_A - a'_0) \cdot r_A^0 + p_B \cdot r_B^0 \\ &= p_A \cdot r_A^0 + p_B \cdot r_B^0 \in \mathcal{R}_q \end{aligned}$$

where r_A^0, r_B^0 are uniformly distributed degree-n polynomials over the ring $\mathcal{R}_q$ The set of all common roots of p_A and p_B can be deduced from the polynomial U. Indeed, by Lemma 10 and Lemma 11:

$$x \in A \cap B \iff p_A(x) = 0 \wedge p_B(x) = 0 \iff p_A(x) = 0 \wedge U(x) = 0.$$

Theorem 5. *Our PSI protocol (figure 5.1) securely realizes the ideal functionality $\mathcal{F}_{\mathsf{mPSI}}$ (figure 7) with statistical security against malicious adversaries in the $\mathcal{F}_{\mathsf{sole}}$ hybrid model.*

The proof of Theorem 5 appears in Section 5.3.

Efficiency. The main computational cost comes from the computation of fast polynomial interpolation, and multiplications between degree-$2n$ polynomials (as well as a fast multipoint evaluation for the receiver). Regarding communication, the total communication boils down to

- A subfield ring-OLE, which can be implemented with $o(n)$ communication [BCG+20b];
- Two polynomials of degree n in $\mathbb{F}_p[X]$, one polynomial of degree n and one polynomial of degree $2n$ over $\mathbb{F}_q[X]$.

In total, this amounts to $n \cdot (2 \log p + 3 \log q) + o(n)$ bits of communication. Here, the size of the subfield $\mathbb{F}_p$ depends only on the bitsize ℓ of the items in the sets A and B, hence we can set $\log p = \ell$. As we will see in the analysis, $\log q$ must be set to $\log q \approx \lambda + 2 \log n$ to guarantee λ bits of statistical security. This leads to a total communication of $n \cdot (2\ell + 3\lambda + 6 \log n) + o(n)$ bits.

The $o(n)$ term above captures the cost of distributing the PCG seeds of the subfield ring-OLE. Using the maliciously secure seed distribution protocol of [BCG+20b], for $n = 2^{20}$ this amounts to a bit less than 4MB of communications, i.e. about $30n$ bits of communication. For larger values of n, the cost grows logarithmically in n and this term quickly becomes dominated by the other terms.

PARAMETERS:

- Two rings $\mathcal{R}_\mathsf{p} = \mathbb{F}_\mathsf{p}[\mathsf{x}]/\mathsf{F}(\mathsf{x}) \subseteq \mathcal{R}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^\mathsf{t}}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$, where $F(x)$ has degree $2n+1$.
- the receiver and the sender have respective input set $A = \{a_1, a_2, \ldots, a_n\} \subset \mathcal{R}_p$ and $B = \{b_1, b_2, \ldots, b_n\} \subset \mathcal{R}_p$.
- A subfield ring-OLE in the ring $\mathcal{R}_q$ over the subring $\mathcal{R}_p$.

PROTOCOL:

1. The receiver and the sender encode their inputs to $p_A = \sum_{i=1}^n (X - a_i)$, $p_B = \sum_{i=1}^n (X - b_i)$ respectively.
2. The receiver and the sender invoke $\mathcal{F}_\mathsf{sole}$ protocol as a receiver and a sender to generate a subfield ring-OLE over $\mathcal{R}_p, \mathcal{R}_q$. The sender receives two random polynomial (r_A, r'_B) of degree $2n$ while the receiver gets a pair of polynomials (a', s_A) of degree $2n$ such that $a' \in \mathcal{R}_p$ and $s_A = a' \cdot r_A + r'_B$.
3. The receiver decomposes a' as $a' = a'_0 + a'_1 \cdot X^n$ and sends a'_1 to the sender. Similarly, the sender decomposes r_A as $r_A^0 + r_A^1 \cdot X^n$ and sends r_A^1 to the receiver.
4. The receiver sends $t_A = p_A - a'_0 \in \mathcal{R}_p$ to the sender.
5. After receiving t_A, the sender picks randomly a polynomial $r_B^0 \in \mathcal{R}_q$ of degree n, computes $s = r_B^0 \cdot p_B$, $s'_B = a'_1 . r_A^0 \cdot X^n \bmod F(X)$ and $s_B = t_A \cdot r_A^0 + s - r'_B - s'_B$, and sends s_B to the receiver.
6. The receiver compute $s'_A = a'_0 \cdot r_A^1 \cdot X^n + a'_1 \cdot r_A^1 \cdot X^{2n} \bmod F(x)$ and defines the polynomial $U = s_A + s_B - s'_A$. Finally, the receiver outputs the set $I = \{x \in A \mid U(x) = 0\}$.

Fig. 5. Malicious PSI protocol based on OLE

5.2 A Batch Non-Interactive PSI

An important feature of the PCG of [BCG+20b] is that it is *programmable*: when executing the seed distribution protocol, the receiver can ensure that her output (a', s_A) remains *identical* across many instances of the protocol with different parties. This powerful feature implies that, after a logarithmic-cost preprocessing phase where the client (playing the role of the receiver, with input set X) sets up PCG seeds with each of servers (playing the roles of multiple senders , each with input set X_i), the receiver can broadcast the pair $(t_A = p_A - a'_0, a'_1)$ to everyone (it depends solely on a' and her encoded set p_A), which communicates $2n \log p \approx 2\ell n$ bits. This message can be seen as a public encoding of her dataset. Observe that it is almost optimally compact: it is only twice larger than the set of the receiver. Furthermore, the receiver can publish this encoding at any time, even before interacting with any server. Afterwards, any server that executed the distributed seed protocol with the receiver can send a single message (r_A^1, s_B) to the receiver, of length $3n \log q \approx 3(\lambda + 2 \log n)n$, from which the receiver can locally recover $X \cap X_i$. To our knowledge, this batch non-interactive communication pattern was never achieved by any prior proposal; we believe that it can make our protocol appealing in numerous realistic scenarios.

5.3 Proof of Security

In this section, we prove Theorem, restated below:

Theorem 6. *Our PSI protocol (figure 5.1) securely realizes the ideal functionality $\mathcal{F}_{\mathsf{mPSI}}$ (figure 7) with security against malicious adversaries in the $\mathcal{F}_{\mathsf{sole}}$ hybrid model.*

Proof. **Correctness.** We show that the protocol is correct with a probability of at least $1 - n^2/q$. Let us consider the polynomial $U = p_A \cdot r_A^0 + p_B \cdot r_B^0$, where the polynomials r_A^0, r_B^0 of degree n are distributed uniformly at random over the ring $\mathcal{R}_q$ and $p_A, p_B \in \mathcal{R}_p^2$ of degree n are encoding polynomials of the input sets of the sender and the receiver respectively. The protocol is correct iff $I = \{x \in A \mid U(x) = 0\} = A \cap B$. Let denote C be the set of all roots of the polynomial $U(x)$. We show that all the common roots of the polynomials p_A and p_B are the same as those of the polynomials P_A and U. Assume that x is an element in the set $A \cap B$ then:

$$\begin{aligned} x \in A \cap B &\implies p_A(x) = 0 \ \wedge \ p_B(x) = 0 \\ &\implies U(x) = p_A(x) \cdot r_A^0(x) + p_B(x) \cdot r_B^0(x) = 0 \end{aligned}$$

hence $A \cap B \subseteq A \cap C$. Moreover, by Lemma 10, the probability that p_A and r_B^0 share a common root (i.e. $\gcd(p_A, r_B^0) \neq 1$) is at most n^2/q. Then,

$$\begin{aligned} x \in A \cap C &\implies p_A(x) = 0 \wedge U(x) = 0 \\ &\implies p_B(x) \cdot r_B^0(x) = U(x) - p_A(x) \cdot r_A^0(x) = 0 \\ &\implies p_B(x) = 0 \text{ (since } \gcd(p_A, r_B^0) = 1 \text{ w.h.p).} \end{aligned}$$

Therefore $A \cap C \subseteq A \cap B$. Finally we get $A \cap B = A \cap C$ i.e the sender outputs correctly the intersection of their inputs.

Security. We prove security through a sequence of hybrids. We first demonstrate a core property. Given any set S, we denote by $p_S \in \mathcal{R}_p$ the polynomial whose set of roots is S. We have:

$$U = p_A \cdot r_A^0 + p_B \cdot r_B^0 = p_{A \cap B} \cdot \left(p_{A \setminus B} \cdot r_A^0 + p_{B \setminus A} \cdot r_B^0\right).$$

$p_{A \setminus B}, p_{B \setminus A} \in \mathcal{R}_p^2$ are two polynomial of degree at most n and $\gcd(p_{A \setminus B}, p_{B \setminus A}) = 1$. By Lemma 11, the polynomial $p_{A \setminus B} \cdot r_A^0 + p_{B \setminus A} \cdot r_B^0$ is a uniformly random polynomial of degree at most $2n$. Therefore, U conceals all information about $p_{A \setminus B}$ and $p_{B \setminus A}$. This show perfect security against a corrupted sender in the $\mathcal{F}_{\mathsf{sole}}$-hybrid model. More formally:

Security against a corrupted receiver. We first sketch the main idea of simulator.

- Sim plays the role of $\mathcal{F}_{\mathsf{sole}}$. When $\mathcal{A}$ queries $\mathcal{F}_{\mathsf{sole}}$, Sim waits for $\mathcal{A}$ to send a', s_A then Sim computes a'_0, a'_1 such that $a' := a'_0 + a'_1 \cdot X^n$.
- As the sender, Sim generates randomly a polynomial $r_A^1 \in \mathcal{R}_q$ of degree n and sends it to $\mathcal{A}$.
- After receiving $t_A = p_A - a'_0$ from $\mathcal{A}$, Sim extracts p_A.
- Sim defines the set $A = \{x \in \mathbb{F}_p \mid p_A(x) = 0\}$ and sends A to the ideal functionality of $\mathcal{F}_{\mathsf{mPSI}}$ and gets the set $A \cap B$.
- Assume that $|A \cap B| = k \leq n$, Sim defines the polynomials:

$$\begin{aligned} m(x) &:= \left\{ \sum_{i=1}^{k} (x - x_i) \mid x_i \in A \cap B \right\} \\ n(x) &:= \left\{ \sum_{i=1}^{n-k} (x - x_i) \mid x_i \leftarrow \mathbb{F}_q \right\} \\ U(x) &:= m(x) \cdot n(x) \end{aligned}$$

– Since Sim knows a'_0, a'_1, r^1_A then Sim can computes $s'_A = a'_0 \cdot r^1_A \cdot X^n + a'_1 \cdot r^1_A \cdot X^{2n} \bmod F(X)$.
– On behalf of the sender, Sim computes and sends to $\mathcal{A}$ the polynomial $s_B = U - s_A + s'_A$.

We prove that the simulated protocol is indistinguishable from an honest execution through a sequence of hybrids:

– *Hybrid 1:* Same as the real protocol interaction, except that Sim plays the role of $\mathcal{F}_{\mathsf{sole}}$. When $\mathcal{A}$ queries $\mathcal{F}_{\mathsf{sole}}$, Sim waits for $\mathcal{A}$ to send a', s_A where $a' \in \mathcal{R}_p$, $s_A \in \mathcal{R}_q$ are two random polynomials of degree $2n$ and finds $a'_0, a'_1 \in \mathcal{R}_p^2$ such that $a' = a'_0 + a'_1 \cdot X^n$. This hybrid has the same distribution as the honest protocol.
– *Hybrid 2:* Same as Hybrid 1 except that Sim in this hybrid samples himself $r^1_A \in \mathcal{R}_q$ of degree n. Since r^1_A is distributed uniformly at random in the view of $\mathcal{A}$, this hybrid is distributed identically to the previous hybrid.
– *Hybrid 3:* Same as Hybrid 2, but when $\mathcal{A}$ sends $t_A = p_A - a'_0$, Sim extracts $p_A = t_A + a'_0$, defines the set $\tilde{A} = \{x \in \mathbb{F}_p \mid p_A(x) = 0\}$, and sends it to the ideal functionality $\mathcal{F}_{\mathsf{mPSI}}$ to get $\tilde{I} = \tilde{A} \cap B$.
– *Hybrid 4:* Observe that

$$\begin{aligned} U &= s_A + s_B - s'_A = p_A \cdot r^0_A + p_B \cdot r^0_B \\ &= p_{A\cap B} \cdot \left(p_{A\setminus B} \cdot r^0_A + p_{B\setminus A} \cdot r^0_B\right). \end{aligned}$$

Assume that $|\tilde{I}| = |\tilde{A} \cap B| = k \leq n$. Sim in this hybrid defines the polynomials:

$$\begin{aligned} m(x) &:= \left\{ \sum_{i=1}^{k} (x - x_i) \mid x_i \in \tilde{I} \right\} \\ n(x) &:= \left\{ \sum_{i=1}^{n-k} (x - x_i) \mid x_i \leftarrow \mathbb{F}_q \right\} \\ U(x) &:= m(x) \cdot n(x). \end{aligned}$$

Since $p_{A\setminus B} \cdot r^0_A + p_{B\setminus A} \cdot r^0_B$ is uniformly random by Lemma 11, this hybrid is perfectly indistinguishable from the previous hybrid.
– *Hybrid 5:* Same as Hybrid 4 except Sim computes $s_B = U + s'_A - s_A$ and on behalf of the sender sends it to $\mathcal{A}$. Since s_B is random in the view of $\mathcal{A}$, this is distributed exactly as in the previous hybrid. In this hybrid, Sim aborts if there exists a element $y \in B \setminus \tilde{A}$ where $U(y) = 0$. We rewrite

$$U = p_{\tilde{A}\cap B} \cdot \left(p_{\tilde{A}\setminus B} \cdot r^0_A + p_{B\setminus \tilde{A}} \cdot r^0_B\right)$$

Therefore,

$$\{x \in \mathbb{F}_q \mid U(x) = 0\} = \{x \in \tilde{I}\} \cup \{x \in \mathbb{F}_q \mid p_{\tilde{A}\setminus B} \cdot r^0_A + p_{B\setminus \tilde{A}} \cdot r^0_B\}.$$

We show that Sim aborts with negligible probability.
• Case 1: $y \in \tilde{I}$: since we know that $y \in B \setminus \tilde{A}$, this cannot happen.
• Case 2: $y \in \{x \in \mathbb{F}_q \mid p_{\tilde{A}\setminus B} . r^0_A + p_{B\setminus \tilde{A}} \cdot r^0_B\}$. Since $y \in B \setminus \tilde{A} \Leftrightarrow y \in B \setminus \tilde{A} \Leftrightarrow p_{B\setminus \tilde{A}} = 0 \wedge p_{\tilde{A}\setminus B}(y) \neq 0$, we have

$$\begin{aligned} 0 &= p_{\tilde{A}\setminus B}(y) \cdot r^0_A(y) + p_{B\setminus \tilde{A}}(y) \cdot r^0_B(y) = p_{\tilde{A}\setminus B}(y) . r^0_A(y) \\ &\Leftrightarrow r^0_A(y) = 0 \ \wedge \ p_{B\setminus \tilde{A}}(y) = 0 \\ &\implies \gcd(p_{B\setminus \tilde{A}}, r^0_A) \neq 1 \end{aligned}$$

where $r^0_A \in \mathcal{R}_q$ is a polynomial of degree n and is distributed uniformly at random.

By Lemma 10 the probability of $\gcd(p_{B\setminus \tilde{A}}, r^0_A) \neq 1$ is at most n^2/q. Setting $q = \lambda + 2\log n$ suffices to bound the abortion probability by $1/2^\lambda$. Conditioned on not aborting, this hybrid is distributed perfectly as the previous hybrid; hence, no adversary can distinguish between this hybrid and the previous one with probability more than $2^{-\lambda}$.

Security against a corrupt sender. We first sketch the main ideal of the simulator.

- Sim plays the role of $\mathcal{F}_{\mathsf{sole}}$. When $\mathcal{A}$ queries $\mathcal{F}_{\mathsf{sole}}$, Sim waits for $\mathcal{A}$ to send (r_A, r'_B) and extracts r^0_A, r^1_A such that $r^0_A + r^1_A \cdot X^n$.
- Sim simulates the polynomials a'_1, t_A and receives s_B from $\mathcal{A}$. Then Sim computes the polynomial $s'_B = a'_1 \cdot r^0_A \cdot X^n \bmod F(x)$ and obtains $s = s_B + r'_B + s'_B - t_A \cdot r^0_A$.
- Sim defines the set
$$\tilde{B} = \{x \in \mathbb{F}_p \mid s(x) = 0\}$$
Since $s = p_B \cdot r^0_B$ and by Lemma 10, $\gcd(r^0_B, p_A) = 1$ with probability at least $1 - n^2/q$. This leads to $A \cap B = A \cap \tilde{B}$ with high probability.
- Sim sends $\tilde{B}$ to the ideal functionality of $\mathcal{F}_{\mathsf{mPSI}}$.

The proof proceeds through a sequence of hybrids:

- *Hybrid 1:* Same as the real protocol interaction, except Sim plays the role of $\mathcal{F}_{\mathsf{sole}}$. When $\mathcal{A}$ queries $\mathcal{F}_{\mathsf{sole}}$, Sim waits for $\mathcal{A}$ to send (r_A, r'_B) and finds $r^0_A, r^1_A \in \mathcal{R}^2_q$ of degree at most $2n$ such that $r^0_A + r^1_A \cdot X^n$.
- *Hybrid 2:* Same as Hybrid 1 but instead of extracting a'_1 from the sender's output from $\mathcal{F}_{\mathsf{sole}}$ and $t_A := p_A - a'_0$, Sim in this hybrid samples two uniformly random polynomials a'_1, t_A of degree n over $\mathcal{R}_p$ and sends them to $\mathcal{A}$. In the view of $\mathcal{A}$, a'_1, t_A are random so this hybrid has an identical distribution as the previous hybrid.
- *Hybrid 3:* When $\mathcal{A}$ sends s_B to the honest receiver, from a'_1, t_A, r^0_A, r^1_A Sim computes $s := s_B + r'_B + s'_B - t_A \cdot r^0_A$. This hybrid is indistinguishable from the real protocol interaction.
- *Hybrid 4:* Same as Hybrid 3, except we can rewrite the polynomial $s \in \mathcal{R}_q$ as
$$s = s_B + r'_B + s'_B - t_A \cdot r^0_A = r^0_B \cdot p_B$$
This leads to $p_B(x) = 0 \Rightarrow s(x) = 0$. Hence, Sim defines the set $\tilde{B} = \{x \in \mathbb{F}_p \mid s(x) = 0\}$. Simulation in this hybrid fails if the honest receiver holds an $x \in A$ where $s(x) = 0 \,\wedge\, p_B(x) \neq 0$. It is sufficient to show that the probability of this negligible. Indeed,
$$\begin{aligned}\{x \in A \mid s(x) = 0 \,\wedge\, p_B(x) \neq 0\} &\Leftrightarrow \{x \in A \mid r_B(x) = 0\}\\ &\Leftrightarrow \{x \in \mathbb{F}_p \mid p_A(x) = 0 \,\wedge\, r_B(x) = 0\}.\end{aligned}$$
As before, by Lemma 10 the probability of $\gcd(p_A, r_B) \neq 1$ is bounded by $1/2^\lambda$ when $q = \lambda + 2\log n$.
- *Hybrid 5:* Same as hybrid 4, but we change the way for computing the honest receiver's output as the set $A \cap \tilde{B}$. Specifically, in hybrid 4 the receiver's output is computed as
$$I = \{x \in A \mid U(x) = 0\}$$
Where I is equivalent to the set $A \cap \tilde{B}$ up to a negligible failure probability. After defining the set $\tilde{B}$, Sim sends it to the ideal functionality of $\mathcal{F}_{\mathsf{mPSI}}$, and the set $A \cap \tilde{B}$ is delivered to the honest receiver. In this hybrid, Sim makes the honest receiver computes its output as $A \cap \tilde{B}$. This concludes the proof that our PSI protocol is secure against both malicious sender and receiver.

References

ADI$^+$17. B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 223–254. Springer, Heidelberg, August 2017.

BCG$^+$17. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, and M. Orrù. Homomorphic secret sharing: Optimizations and applications. In *ACM CCS 2017*, pages 2105–2122. ACM Press, October / November 2017.

BCG$^+$19a. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCG$^+$19b. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 489–518. Springer, Heidelberg, August 2019.

BCG$^+$20a. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, pages 1069–1080. IEEE Computer Society Press, November 2020.

BCG$^+$20b. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators from ring-LPN. In *CRYPTO 2020, Part II*, *LNCS* 12171, pages 387–416. Springer, Heidelberg, August 2020.

BCGI18. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

BMR20. S. Badrinarayanan, P. Miao, and P. Rindal. Multi-party threshold private set intersection with sublinear communication. Cryptology ePrint Archive, Report 2020/600, 2020. `https://eprint.iacr.org/2020/600`.

CLR17. H. Chen, K. Laine, and P. Rindal. Fast private set intersection from homomorphic encryption. In *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

CM20. M. Chase and P. Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In *CRYPTO 2020, Part III*, *LNCS* 12172, pages 34–63. Springer, Heidelberg, August 2020.

CRR21. G. Couteau, P. Rindal, and S. Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. LNCS, pages 502–534. Springer, Heidelberg, 2021.

DKT10. E. De Cristofaro, J. Kim, and G. Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *ASIACRYPT 2010*, *LNCS* 6477, pages 213–231. Springer, Heidelberg, December 2010.

DMRY09. D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Efficient robust private set intersection. In *ACNS 09*, *LNCS* 5536, pages 125–142. Springer, Heidelberg, June 2009.

DP12. I. Damgård and S. Park. How practical is public-key encryption based on LPN and ring-LPN? Cryptology ePrint Archive, Report 2012/699, 2012. `https://eprint.iacr.org/2012/699`.

DW07. M. Dietzfelbinger and C. Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theoretical Computer Science*, 380(1-2):47–68, 2007.

FHNP16. M. J. Freedman, C. Hazay, K. Nissim, and B. Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, January 2016.

FIPR05. M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography*, pages 303–324, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

FNP04. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004*, *LNCS* 3027, pages 1–19. Springer, Heidelberg, May 2004.

GN19. S. Ghosh and T. Nilges. An algebraic approach to maliciously secure private set intersection. In *EUROCRYPT 2019, Part III*, *LNCS* 11478, pages 154–185. Springer, Heidelberg, May 2019.

GPR$^+$21. G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. Oblivious key-value stores and amplification for private set intersection. LNCS, pages 395–425. Springer, Heidelberg, 2021.

GS19. S. Ghosh and M. Simkin. The communication complexity of threshold private set intersection. In *CRYPTO 2019, Part II*, *LNCS* 11693, pages 3–29. Springer, Heidelberg, August 2019.

Haz15. C. Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. In *TCC 2015, Part II*, *LNCS* 9015, pages 90–120. Springer, Heidelberg, March 2015.

HEK12. Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

HFH99. B. A. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, page 78–86, New York, NY, USA, 1999. Association for Computing Machinery.

HKL$^+$12. S. Heyse, E. Kiltz, V. Lyubashevsky, C. Paar, and K. Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In *FSE 2012*, *LNCS* 7549, pages 346–365. Springer, Heidelberg, March 2012.

HN10. C. Hazay and K. Nissim. Efficient set operations in the presence of malicious adversaries. In *PKC 2010*, *LNCS* 6056, pages 312–331. Springer, Heidelberg, May 2010.

HV17. C. Hazay and M. Venkitasubramaniam. Scalable multi-party private set-intersection. In *PKC 2017, Part I*, *LNCS* 10174, pages 175–203. Springer, Heidelberg, March 2017.

IKNP03. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, *LNCS* 2729, pages 145–161. Springer, Heidelberg, August 2003.

JL10. S. Jarecki and X. Liu. Fast secure computation of set intersection. In *SCN 10*, *LNCS* 6280, pages 418–435. Springer, Heidelberg, September 2010.

KK13. V. Kolesnikov and R. Kumaresan. Improved ot extension for transferring short secrets. In *Advances in Cryptology – CRYPTO 2013*, pages 54–70, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

KKRT16. V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

KRTW19. V. Kolesnikov, M. Rosulek, N. Trieu, and X. Wang. Scalable private set union from symmetric-key techniques. In *ASIACRYPT 2019, Part II*, *LNCS* 11922, pages 636–666. Springer, Heidelberg, December 2019.

KS05. L. Kissner and D. X. Song. Privacy-preserving set operations. In *CRYPTO 2005*, *LNCS* 3621, pages 241–257. Springer, Heidelberg, August 2005.

LP15. H. Lipmaa and K. Pavlyk. Analysis and implementation of an efficient ring-LPN based commitment scheme. In *CANS 15*, LNCS, pages 160–175. Springer, Heidelberg, December 2015.

MB72. R. Moenck and A. Borodin. Fast modular transforms via division. In *Proceedings of the 13th Annual Symposium on Switching and Automata Theory (Swat 1972)*, SWAT '72, page 90–96, USA, 1972. IEEE Computer Society.

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. IEEE, 1986.

OOS17. M. Orrù, E. Orsini, and P. Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In *CT-RSA 2017*, *LNCS* 10159, pages 381–396. Springer, Heidelberg, February 2017.

Pan05. R. Panigrahy. Efficient hashing with lookups in two memory accesses. In *16th SODA*, pages 830–839. ACM-SIAM, January 2005.

PR04. R. Pagh and F. F. Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

PRTY19. B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 401–431. Springer, Heidelberg, August 2019.

PRTY20. B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. PSI from PaXoS: Fast, malicious private set intersection. In *EUROCRYPT 2020, Part II*, *LNCS* 12106, pages 739–767. Springer, Heidelberg, May 2020.

PSSZ15. B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *USENIX Security 2015*, pages 515–530. USENIX Association, August 2015.

PSWW18. B. Pinkas, T. Schneider, C. Weinert, and U. Wieder. Efficient circuit-based PSI via cuckoo hashing. In *EUROCRYPT 2018, Part III*, *LNCS* 10822, pages 125–157. Springer, Heidelberg, April / May 2018.

PSZ14. B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on OT extension. In *USENIX Security 2014*, pages 797–812. USENIX Association, August 2014.

PSZ18. B. Pinkas, T. Schneider, and M. Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

RR17. P. Rindal and M. Rosulek. Malicious-secure private set intersection via dual execution. In *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017.

RS21. P. Rindal and P. Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. LNCS, pages 901–930. Springer, Heidelberg, 2021.

RT21. M. Rosulek and N. Trieu. Compact and malicious private set intersection for small sets. Cryptology ePrint Archive, Report 2021/1159, 2021. `https://eprint.iacr.org/2021/1159`.

SGRR19. P. Schoppmann, A. Gascón, L. Reichert, and M. Raykova. Distributed vector-OLE: Improved constructions and implementation. In *ACM CCS 2019*, pages 1055–1072. ACM Press, November 2019.

TLP$^+$17. S. Tamrakar, J. Liu, A. Paverd, J.-E. Ekberg, B. Pinkas, and N. Asokan. The circle game: Scalable private membership test using trusted hardware. In *ASIACCS 17*, pages 31–44. ACM Press, April 2017.

W$^+$17. U. Wieder et al. Hashing, load balancing and multiple choice. *Foundations and Trends® in Theoretical Computer Science*, 12(3–4):275–379, 2017.

WYKW21. C. Weng, K. Yang, J. Katz, and X. Wang. Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1074–1091. IEEE, 2021.

YWL$^+$20. K. Yang, C. Weng, X. Lan, J. Zhang, and X. Wang. Ferret: Fast extension for correlated OT with small communication. In *ACM CCS 20*, pages 1607–1626. ACM Press, November 2020.

Supplementary Material

A Additional Preliminaries

In this appendix, we provide further preliminaries, including necessary preliminaries for our second construction, described in Section 5.

A.1 Ideal Functionalities and Security Model

The ideal functionality of PSI in the semi-honest and malicious settings are shown on Figure 6 and 7 respectively. The reason for two different functionalities is a standard limitation of all known protocol: in the semi-honest setting, the size of a corrupted party's input set is fixed to n, while in the malicious setting, the functionality allows a malicious party to use a possibly larger set of bounded size $n' > n$. Since PSI is a special case of secure computation, the security analysis of a two-party PSI protocol is performed via the standard simulation paradigm, recalled below.

PARAMETERS:

- An arbitrary field $\mathbb{F}, n \in \mathbb{Z}$.
- There are two parties, a sender with a input set $X \subseteq \mathbb{F}$ and a receiver with a input set $Y \subseteq \mathbb{F}$ where $|X| = |Y| = n$.

FUNCTIONALITY:

- Wait for the input set X of the sender.
- Wait for the input set Y of the receiver.
- The functionality outputs the intersection $X \cap Y$ to the receiver.

Fig. 6. Ideal functionality of semi-honest PSI $\mathcal{F}_{\mathsf{sPSI}}$

PARAMETERS:

- An arbitrary field $\mathbb{F}, n, n'$ public parameters for honest parties and corrupt parties respectively where $n \leq n'$.
- There are two parties, a sender with a input set $X \subseteq \mathbb{F}$ and a receiver with a input set $Y \subseteq \mathbb{F}$.

FUNCTIONALITY:

- Wait for the input set X of the sender then abort if $|X| > n$.
- Wait for the input set Y of the receiver. If $|Y| > n'$ and the receiver is malicious or if $|Y| > n$ and the receiver is honest then abort.
- The functionality outputs the intersection $X \cap Y$ to the receiver.

Fig. 7. Ideal functionality of malicious PSI $\mathcal{F}_{\mathsf{mPSI}}$

Semi-honest security. Let $\mathrm{view}_1^{\Pi}(X, Y)$ and $\mathrm{view}_2^{\Pi}(X, Y)$ be the view of P_1 and P_2 in the protocol Π, $\mathsf{out}^{\Pi}(X, Y)$ be the output of P_2 in the protocol, and $f(X, Y)$ be the output of P_2 from the ideal functionality. The protocol Π is semi-honest secure if there exist PPT simulators Sim_1 and Sim_2 such that for all inputs X, Y,

$$(\mathrm{view}_1^{\Pi}(X, Y), \mathsf{out}^{\Pi}(X, Y)) \approx \mathsf{Sim}_1((1^{\kappa}, X, n), f(X, Y));$$

$$\mathrm{view}_2^{\Pi}(X, Y) \approx \mathsf{Sim}_2((1^{\kappa}, Y, n), f(X, Y))$$

Malicious Security. Let f be a two-party functionality and Π be a secure protocol for computing f. The protocol Π is said to be secure against malicious adversary if for all non-uniform PPT adversary $\mathcal{A}$ in the real model, there exists a non-uniform PPT adversary $\mathcal{S}$ in the deal model satisfying:

$$\text{IDEAL}_{(f,\mathcal{S},i)}(X,Y) \approx \text{REAL}_{(\Pi,\mathcal{A},i)}(X,Y)$$

where $i \in \{1,2\}$ index of corrupted party. $\text{IDEAL}_{(f,\mathcal{S},i)}(X,Y)$ is the output pair of the honest party and the adversary $\mathcal{S}$ in the ideal model, $\text{REAL}_{(\Pi,\mathcal{A},i)}(X,Y)$ is defined as the output pair of the honest party and the adversary $\mathcal{A}$ from the real execution of Π.

A.2 Learning Parity with Noise

We define the LPN assumption over a ring $\mathcal{R}$ with dimension k, number of samples n, w.r.t. a code generation algorithm $\mathbf{C}$, and a noise distribution $\mathcal{D}$:

Definition 7 (Dual LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{k,n}(\mathcal{R})\}_{k,n\in\mathbb{N}}$ denote a family of efficiently sampleable distributions over a ring $\mathcal{R}$, such that for any $k,n \in \mathbb{N}$, $\mathsf{Im}(\mathcal{D}_{k,n}(\mathcal{R})) \subseteq \mathcal{R}^n$. Let $\mathbf{C}$ be a probabilistic code generation algorithm such that $\mathbf{C}(k,n,\mathcal{R})$ outputs a matrix $H \in \mathcal{R}^{k\times n}$. For dimension $k = k(\lambda)$, number of samples (or block length) $n = n(\lambda)$, and ring $\mathcal{R} = \mathcal{R}(\lambda)$, the (dual) $(\mathcal{D},\mathbf{C},\mathcal{R})$-$\mathsf{LPN}(k,n)$ assumption states that*

$$\begin{aligned}\{(H,\mathbf{b}) \mid H \leftarrow_r \mathbf{C}(k,n,\mathcal{R}), \mathbf{e} \leftarrow_r \mathcal{D}_{k,n}(\mathcal{R}), \mathbf{b} \leftarrow H\cdot\mathbf{s}\}\\ \overset{c}{\approx} \{(H,\mathbf{b}) \mid H \leftarrow_r \mathbf{C}(k,n,\mathcal{R}), \mathbf{b} \leftarrow_r \mathcal{R}^n\}.\end{aligned}$$

The dual LPN assumption is also called *syndrome decoding assumption* in the code-based cryptography literature. The dual LPN assumption as written above is equivalent to the *primal* LPN assumption with respect to G (a matrix $G \in \mathcal{R}^{n\times n-k}$ such that $H\cdot G = 0$), which states that $G\cdot\mathbf{s}+\mathbf{e}$ is indistinguishable from random, where $\mathbf{s} \leftarrow_r \mathcal{R}^{n-k}$ and $\mathbf{e} \leftarrow_r \mathcal{D}_{k,n}(\mathcal{R})$; the equivalence follows from the fact that $H\dot(G\cdot\mathbf{s}+\mathbf{e}) = H\cdot\mathbf{e}$.

The standard LPN assumption refers to the case where H is a uniformly random matrix over $\mathbb{F}_2$, and $\mathbf{e}$ is sampled from $\mathsf{Ber}_r(\mathbb{F}_2)$, where r is called the *noise rate*. Other common noise distributions include exact noise (the noise vector $\mathbf{e}$ is a uniformly random weight-rn vector from $\mathbb{F}_2^n$; this is a common choice in concrete LPN-based constructions) and regular noise (the noise vector $\mathbf{e}$ is a concatenation of rn random unit vectors from $\mathbb{F}_2^{1/r}$, widely used in the PCG literature [BCGI18, BCG+19b, BCG+19a]).

Known constructions of subfield-VOLE use various flavors of the dual LPN assumption with regular noise over a finite field. For example, the work of [BCGI18] suggests relying on an LDPC code, while [BCG+19a] uses quasi-cyclic codes, and [CRR21] uses a new family of codes, called Silver codes.

In this section, we recall the Ring-LPN assumption, which was first introduced in [HKL+12] to build efficient authentication protocols. Since then, it has received some attention from the cryptography community [**?**,DP12,LP15,**?**], due to its appealing combination of LPN-like structure, compact parameters, and short runtimes. Below, we also provide a definition of Module-LPN, which generalizes Ring-LPN in the same way that the more well-known Module-LWE generalizes Ring-LWE.

A.3 Ring-LPN

We now define the Ring-LPN assumption, a variant of the dual LPN assumption over polynomial rings, first introduced in [HKL+12] The assumption has been used in multiple works since. Ring-LPN is the natural "ring analog" of LPN, in the same way that ring-LWE is the ring analog of LWE.

Definition 8 (Ring-LPN). *Let $\mathcal{R} = \mathbb{F}[X]/(F(X))$ for some field $\mathbb{F}$ and degree-N polynomial $F(X) \in \mathbb{Z}[X]$, and let $m,t \in \mathbb{N}$. Let HW_t be the distribution over R_p that is obtained via sampling t noise positions $A \leftarrow [0..N)^t$ as well as t payloads $\mathbf{b} \leftarrow \mathbb{Z}_p^t$ uniformly at random, and outputting $e(X) := \sum_{j=0}^{t-1} \mathbf{b}[j]\cdot X^{A[j]}$. The R-$\mathsf{LPN}_{p,q,t}$ problem is hard if for any PPT adversary $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}((a_i, a_i\cdot s+e_i)_{i=1}^m) = 1] - \Pr[\mathcal{A}((a_i,u_i)_{i=1}^m) = 1]| \leq \mathsf{negl}(\lambda)$$

where the probabilities are taken over the random choices of the values $a_1,\ldots,a_m,u_1,\ldots,u_m \leftarrow \mathcal{R}_p$, $s,e_1,\ldots,e_m \leftarrow \mathsf{HW}_t$ and the randomness of $\mathcal{A}$.

A.4 Subfield Ring-OLE

In a recent work [BCG+20b], a new PCG construction was described for the *ring-OLE* correlation. The ring-OLE correlation over a ring $\mathcal{R}_\mathsf{q}$ is the following correlation: $\{((x_0, z_0), (x_1, z_1)) \mid x_0, x_1, z_0 \leftarrow_r \mathcal{R}_\mathsf{q}, z_1 \leftarrow x_0.x_1 - z_0\}$. The main motivation in [BCG+20b] was that, when the ring is a polynomial ring where the polynomial splits fully into n linear factors, such a correlation can be locally converted into n instances of an OLE correlation over a large field, which is very useful for secure computation of arithmetic circuit. We note that our work will actually directly rely on the ring-OLE correlation over a polynomial ring, and we do not need the polynomial to split. This allows to build the necessary PCG from a much more conservative assumption. We note that the work of [BCG+20b] also describes a maliciously secure protocol to distribute the seed of this PCG which, combined with the PCG, leads to a maliciously secure protocol to instantiate the ideal functionality for malicious ring-OLE correlation.

In this work, we rely on a slight variant of the ring-OLE correlation: given the ring $\mathcal{R}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^\mathsf{t}}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ for some polynomial $F(X)$, we consider the *subfield* ring-OLE correlation, where x_0 is instead sampled from the ring $\mathcal{R}_\mathsf{p} = \mathbb{F}_\mathsf{p}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ (that is, the coefficients of x_0 are sampled from the subfield $\mathbb{F}_\mathsf{p}$ instead of the field $\mathbb{F}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^t}$). We represent the corresponding variant of the ideal functionality on Figure 8. We note that the protocol of [BCG+20b] to instantiate the ring-OLE functionality can be adapted to handle the subfield ring-OLE functionality in a straightforward way.

PARAMETERS:

- The security parameter κ, the subring $\mathcal{R}_\mathsf{p} = \mathbb{F}_\mathsf{p}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$, the ring $\mathcal{R}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^\mathsf{t}}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ where $F(x)$ has the degree of n, p is a power prime and t is arbitrary integer.
- There are two parties, a sender and a receiver.

FUNCTIONALITY:

1. If all the parties are honest:
   - Sample uniformly random $x_0 \leftarrow_r \mathcal{R}_p$, $x_1, z_1 \leftarrow_r \mathcal{R}_q$, let $z_0 = x_0 \cdot x_1 + z_1$.
   - Output $(x_0, z_0), (x_1, z_1)$ to the sender and the receiver respectively.
2. One party is corrupt:
   - If the sender is malicious:
     Waiting for the input $x_0 \in \mathcal{R}_p$ and $z_0 \in \mathcal{R}_q$ from the adversary. Sample $x_1 \leftarrow_r \mathcal{R}_q$ and compute $z_1 = x_0 \cdot x_1 - z_0$. Output (x_1, z_1) to the receiver.
   - If the receiver is malicious:
     Waiting for the input $(x_1, z_1) \in \mathcal{R}_\mathsf{p}^2$ from the adversary. Sample $x_0 \leftarrow_r \mathcal{R}_\mathsf{p}$ and compute $z_0 = x_0 \cdot x_1 - z_1$. Output (x_0, z_0) to the sender.

Fig. 8. The ideal functionality $\mathcal{F}_\mathsf{sole}$ of a malicious *subfield-ring* OLE of length n in $\mathcal{R}_q$ over the subring $\mathcal{R}_p$

Theorem 9 ([BCG+20b]). *If the ring-LPN assumption holds over $\mathcal{R} = \mathbb{F}[X]/(F(X))$ where F is a degree-N polynomial, there exists a maliciously secure protocol instantiating the functionality 8 over $\mathcal{R}$, with communication logarithmic in N.*

A.5 Useful Lemmas about Polynomials

Our second protocol will rely on encoding the input sets as polynomials: the set $X = \{x_1, x_2, \dots, x_n\}$ is encoded as the coefficients of $P(X) = \sum_{i=0}^{n}(X - x_i)$. Encoding and decoding can be performed in $O(n^2)$ field operations by Lagrange interpolation and Horner evaluation. For large values of n, the work of [MB72] requires $O(n \log^2 n)$ arithmetic operations where the interpolation and evaluation are reducible to a recursive use of polynomial divisions.

Private set intersection can be reduced to simple arithmetic operations on the polynomial encoding of the sets. This was observed in previous works [KS05,GS19,GN19]. The reduction builds upon simple lemmas, which we state below (we also consider, and prove, a slight generalization of these lemmas).

Lemma 10. *Let $F(X)$ be a degree-n polynomial, $q = p^t$ where p is a prime, $P(x) \in \mathcal{R}_\mathsf{p} = \mathbb{F}_\mathsf{p}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ be an arbitrary polynomial of degree n and $R(x) \in \mathcal{R}_\mathsf{q} = \mathbb{F}_{\mathsf{p}^\mathsf{t}}[\mathsf{x}]/\mathsf{F}(\mathsf{x})$ be a uniformly random polynomial of degree n. Then*

$$\Pr[\gcd(P(x), R(x)) \neq 1] \leq n^2/q.$$

Proof. $\gcd(P(x), R(x)) = 1$ iff $P(x)$ and $R(x)$ share no common root. A random polynomial over $\mathcal{R}_p$ of degree n has at most n roots, which are distributed uniformly; hence, each root of $R(x)$ is equal to a root of $P(x)$ with probability at most $1 - n/q$. Therefore:

$$\begin{aligned}\Pr[\gcd(P(x), R(x)) \neq 1] &= 1 - \Pr[\gcd(P(x), R(x)) = 1] \\ &= 1 - (1 - n/q)^n \leq n^2/q \text{ (union bound).}\end{aligned}$$

Lemma 11 ([BMR20]). *Given $\mathbb{F}_p$ be a finite field of prime order p. Fix any $p = O(\mathsf{poly}(\lambda))$. Let $P(x), Q(x) \in \mathbb{F}_p[x]$ be two arbitrary polynomials of degrees α_1 and α_2 respectively. Let $R_1(x), R_2(x)$ be two polynomials sampled independently and uniformly at random over $\mathbb{F}_p[x]$, of degrees β_1, and β_2 respectively, where $n = \alpha_1 + \beta_1 = \alpha_2 + \beta_2 \leq \alpha_1 + \alpha_2$. Let $S(x) = P(x) \cdot R_1(x) + Q(x) \cdot R_2(x) \in \mathbb{F}_p$. Then $S(x) = \gcd(P(x), Q(x)) \cdot U(x)$, where $U(x)$ is an uniformly random polynomial of degree at most n over $\mathbb{F}_p[x]$.*

B Error in the KKRT Notion of Correlation Robustness

We recall the notion of Hamming correlation robustness from [KKRT16]. The original notion of correlation robustness from [IKNP03] states that $\mathsf{H} : \{0,1\}^k \to \{0,1\}^*$ is *correlation robust* if for a random and independent choice of (polynomial many) strings $s, t_1, \ldots, t_m \in \{0,1\}^k$, the joint distribution $(\mathsf{H}(t_1 \oplus s), \ldots, \mathsf{H}(t_m \oplus s))$ is pseudorandom given $t_1, \ldots, t_m$. The notion was adapted as follows in KKRT:

Definition 12 (Hamming Correlation Robustness). *Let H be a hash function with input length n. Then H is d-Hamming correlation robust if for any strings $z_1, \cdots, z_m \in \{0,1\}^*$, $a_1, \ldots, a_m, b_1, \ldots, b_m \in \{0,1\}^n$ with $\| b_i \|_\mathsf{H} \geq d$ for each $i \in [m]$, the following distribution, induced by random sampling of $s \leftarrow \{0,1\}^n$, is pseudorandom:*

$$(\mathsf{H}(z_1 \parallel a_1 \oplus [b_1.s]), \ldots, \mathsf{H}(z_m \parallel a_m \oplus [b_m.s]).$$

Unfortunately, there is a clear mistake in the above definition: no condition is imposed on the strings z_i, a_i, b_i – in particular, *the definition does not require them to be distinct.* But whenever $(z_i, a_i, b_i) = (z_j, a_j, b_j)$ for two distinct indices (i, j), the hashes are of course equal, hence the distribution cannot be pseudorandom.

The aim of the authors was, perhaps, to state that the. $z_1 \cdots z_m$ should be pairwise distinct: this makes the definition valid (and it becomes easy to show that it holds in the random oracle model). Unfortunately, this notion does not suffice to prove the security of the KKRT protocol. Indeed, the z_i in the KKRT protocol correspond to the indices of the bins where the parties place their inputs. However, one of the parties has to use simple hashing with three hash functions in the KKRT protocol, meaning that a bin can contain up to $\eta \approx 3 \log n / \log\log n$ items. Concretely, this means that up η pairs (a, b) can correspond to the same value z_i.

The "right" definition is a bit more tedious to state: given a bound η on the maximum load of a bin, we require that the z_i are distinct, but for each z_i there can be up to η pairs $(a_{i,j}, b_{i,j}) \in (\{0,1\}^n)^2$ such that for any index i and for distinct $j, k \leq \eta$, $a_{i,j} \neq a_{i,k}$. This is the definition which we use in our construction (with the slight distinction that we do not use a *Hamming* notion of correlation robustness), and it is not too hard to show that this notion holds in the random oracle model (it is a direct adaptation of our ROM proof for our variant of the notion) and suffices to prove the security of the relaxed PRF in [KKRT16].