Laconic Private Set Intersection and Applications

Navid Alamati[1], Pedro Branco[2], Nico Döttling[3], Sanjam Garg[1,4], Mohammad Hajiabadi[5], and Sihang Pu[3]

[1]UC Berkeley
[2]IT, IST - University of Lisbon
[3]Helmholtz Center for Information Security (CISPA)
[4]NTT Research
[5]University of Waterloo

Abstract

Consider a server with a *large* set S of strings $\{x_1, x_2 \ldots, x_N\}$ that would like to publish a *small* hash h of its set S such that any client with a string y can send the server a *short* message allowing it to learn y if $y \in S$ and nothing otherwise. In this work, we study this problem of two-round private set intersection (PSI) with low (asymptotically optimal) communication cost, or what we call *laconic* private set intersection (ℓPSI) and its extensions. This problem is inspired by the recent general frameworks for laconic cryptography [Cho et al. CRYPTO 2017, Quach et al. FOCS'18].

We start by showing the first feasibility result for realizing ℓPSI based on the CDH assumption, or LWE with polynomial noise-to-modulus ratio. However, these feasibility results use expensive non-black-box cryptographic techniques leading to significant inefficiency. Next, with the goal of avoiding these inefficient techniques, we give a construction of ℓPSI schemes making only black-box use of cryptographic functions. Our construction is secure against semi-honest receivers, malicious senders and reusable in the sense that the receiver's message can be reused across any number of executions of the protocol. The scheme is secure under the ϕ-hiding, decisional composite residuosity and subgroup decision assumptions.

Finally, we show natural applications of ℓPSI to realizing a semantically-secure encryption scheme that supports detection of encrypted messages belonging to a set of "illegal" messages (e.g., an illegal video) circulating online. Over the past few years, significant effort has gone into realizing laconic cryptographic protocols. Nonetheless, our work provides the first black-box constructions of such protocols for a natural application setting.

1 Introduction

Laconic cryptography [CDG+17, QWW18, DGI+19, DGGM19] is an emerging paradigm which enables realizing cryptographic tasks with asymptotically-optimal communication in just two messages. In this setting, the receiver has a potentially large input, and the size of her protocol message only depends on the security parameter, and not her input size. The second message, sent by the sender, may grow with the size of the sender's input, but should be independent of the receiver's input size.

The pioneering work of [CDG+17] introduced the notion of laconic oblivious transfer (laconic OT), which allows a receiver with a large input $D \in \{0,1\}^n$ to send a short hash digest h of her input D. Next, a sender with an input $(i \in [n], m_0, m_1)$, sends a short message ots to the receiver, enabling the receiver to learn $m_{D[i]}$, and nothing more. We require (a) the sizes of h and ots be $\mathsf{poly}(\log(n), \lambda)$, where λ is the security parameter; (b) the sender's computation time be $\mathsf{poly}(\log(n), \lambda)$ and (c) and receiver's second-phase computation time be $\mathsf{poly}(\log(n), \lambda)$.

The notion of laconic OT, and the techniques built around it, have led to breakthrough results in the last few years, which, among others, include the first construction of identity-based encryption from CDH [DG17b, DG17a, BLSV18, DGHM18], and two-round MPC protocols from minimal assumptions [GS17, GS18, BL18].

Laconism beyond OT? Motivated by the developments enabled by laconic OT, it is natural to ask whether we can push the boundary further, realizing laconism for richer functionalities. Laconic OT by itself does not seem to be sufficient for this task (at least generically). Specifically, the general laconic OT+garbled circuit based approach for a function $f(\cdot,\cdot)$ results in protocols in which the size of the sender's protocol message grows with the receiver's input size.

The work of Quach, Wee and Wichs [QWW18] shows how to realize laconic cryptography for general functionalities using LWE. However, two significant issues remain. Firstly, it is not clear whether we can achieve laconism from other assumptions, for functionalities beyond OT. As mentioned above, research in laconic OT has led to several breakthrough feasibility results, motivating the need for developing techniques that can be realized using wider assumptions and for richer functionalities. Secondly, existing constructions of laconic primitives are non-black-box, leading to inefficient constructions. Addressing the above shortcomings, our goals are twofold: (1) Feasibility: Can we realize laconic primitives beyond OT from assumptions other than LWE? and (2) Black-boxness: Can we make the constructions black-box?

Black-box techniques. We use the notion "black-box" techniques in the sense that the construction should not use an explicit circuit-level description of cryptographic primitives. In this sense, we think of constructions which e.g., compute cryptographic primitives inside garbled circuits (as previous laconic OT constructions) or use general purpose NIZK proofs (which express statements in terms of NP-complete languages) as "non-black-box" techniques.

Laconic PSI. We make the first progress toward the above two goals with respect to a non-trivial functionality: Laconic Private Set Intersection (ℓPSI) and its family. Private set intersection (PSI) is a cryptographic primitive that allows two parties to learn the intersection of their input sets and nothing else. Because of its usefulness and versatility, this cryptographic primitive has been extensively studied in numerous settings throughout the years (see e.g., [KS05, PSZ14, HV17, RR17, KMP+17, PSWW18] and references therein).

Laconic PSI allows a receiver to send a short digest of its large data set, which in turn can be used by a sender to compute a PSI second round message. We require that the total communication complexity as well as the sender's running time to be independent of the receiver's input size.

1.1 Our Results

As our first result, we give a generic construction of laconic PSI from a primitive called anonymous hash encryption, which in turn can be realized from CDH/LWE [DG17b, DG17a, BLSV18]. Our construction builds on the Merkle-tree garbled circuit based approach of [DG17b, DG17a, BLSV18, GHMR18, GHM+19, GV20], showing how to use garbled circuits to perform binary search on a set of sorted values. Prior to our work there did not exist any construction of a laconic primitive from CDH beyond OT. We also obtain an LWE instantiation with polynomial modulus to noise ration, improving the subexponential ratio of [QWW18].

The above construction is non-black-box caused by the use of garbled circuits. As our second contribution, we achieve a black-box construction of laconic PSI from the ϕ-hiding assumption.

Both constructions above are only semi-honest secure, and can be made malicious (UC) secure by using Non-Interactive Zero Knowledge (NIZK).[1] However, the eventual protocol will be non-black-box. To enhance applicability, we show how to make our second construction secure against malicious senders, and semi-honest receivers in the CRS model, by additionally assuming decisional composite residuosity (DCR) and subgroup

[1]Note that in the laconic setting we cannot prove malicious security against a receiver since it is information-theoretically impossible to extract its input. Thus, since the NIZK will only be computed by the sender, the protocol will remain laconic.

decision assumptions. We term this notion *reusable malicious* laconic PSI, meaning the receiver's message may be re-used.[2]

Applications. We show an application of laconic PSI in realizing a primitive that we dub self-detecting encryption. A self-detecting encryption acts like a normal public-key encryption with a key difference that it is possible to detect whether the underlying message of a given ciphertext belongs to a database of special (e.g., "illegal") messages. This can be determined just by knowing the database values, as opposed to the system's secret key. Such encryption systems provide a feature for detecting the presence of illegal contents, without compromising the privacy of legal messages. There has only been a limited number of proposals for this task so far, and all of them use heavy tools (e.g., FHE) for this purpose (see [Gre19] for more details). We formally define this notion, and show how to realize it using laconic PSI.

In a self-detecting encryption, an authority (e.g., a government entity or a delegated NGO) publishes a small hash value of a (possibly large) database of special messages such that a user can encrypt a message using the system's public key and the hash value. If the message belongs to the database, then the authority can detect it; else, the message remains hidden to the authority. We require that the size of the hash and the encryption running time to be independent of the database size.

We note that attribute-based encryption does not provide a solution to the above problem, because either the authority should reveal its database to a master-key generator, or it should be the master-key generator itself – both of which defeat our security purposes.

Additional new results: Labeled laconic PSI and malicious laconic OT (LOT). We extend our laconic PSI techniques to build a reusable *labeled* laconic PSI. Labeled PSI [JL10, CHLR18] is a flavor of PSI, where the sender holds a *label* ℓ_i associated with each set element x_i, and the receiver will learn the labels corresponding to the intersection elements. Labeled PSI has several practical applications (e.g., private web service queries [CHLR18]).

Moreover, we show how to use our techniques to realize the first construction of a reusable LOT secure against malicious senders and semi-honest receivers.

DV-NIZK range proofs for DJ ciphertexts. As a building block for our laconic PSI protocol, we propose a DV-NIZK range proof scheme for Damgård Jurik (DJ) ciphertexts, which may be of independent interest. Our DV-NIZK has statistical simulation soundness and computational zero-knowledge given that the subgroup decision (SD) assumption holds [BGN05, GOS06].

Such range proofs can also be constructed in the random oracle model (ROM) via the Fiat-Shamir transform (e.g., [DJ01, BBC+18, BBB+18, TBM+20]), which might yield the best efficiency. As our LPSI construction is modular, this can be done independently of the remaining results in the paper. The goal of our DV-NIZK is to provide an efficient standard model construction which we see as a reasonable middle ground between feasibility from the weakest assumption (at the cost of unrealistic efficiency) and practical efficiency (at the cost of relying on strong heuristic assumptions such as the ROM).

1.2 Previous Work

Laconic PSI can be seen as a particular case of unbalanced PSI. Protocols for unbalanced PSI were presented in [ADT11, RA18, CLR17, CHLR18]. The protocol of [RA18] achieves linear communication complexity on the receiver's set size in the *pre-processing* model. The protocols of [CLR17, CHLR18] rely on somewhat homomorphic encryption (SWHE) and proceed in two rounds. However, the communication complexity scales with the size of the receiver's set (and logarithmic with the size of the sender's set), in contrast with our protocol whose communication complexity scales with the sender's set size.

[2] We use the word reusability only in conjunction with malicious security, since in the semi-honest setting, reusability is satisfied by default.

Comparison with [ADT11]. Ateniese et al. in [ADT11] proposed a semi-honest size-hiding PSI protocol[3] inspired by RSA accumulators that achieves communication complexity independent of the receiver's set size. However, we emphasize that their scheme does not fit the framework of laconic cryptography since it requires the sender to know the factorization of a CRS modulus N. Thus, either it requires pre-processing (giving a designated secret key to the sender), or it requires three rounds in the CRS model. In contrast, laconic cryptography requires (a) two rounds and (b) no pre-processing (i.e., neither party receives a secret key correlated with the CRS). Both (a) and (b) are crucially used in applications of laconic cryptography. Specifically, these restrictions prevent use of [ADT11] in settings with multiple senders, an aspect that has been critical for laconic cryptography applications. Finally, we remark that the security of [ADT11] relies on random oracles, whereas we prove security in the standard model and achieve a substantially stronger security notion without resorting to heavy generic tools.

All of above constructions are just secure against semi-honest adversaries, except for [CHLR18] which achieves security against a malicious receiver.

1.3 Open Problems

The main open question is to realize laconic cryptography for functionalities richer than PSI. A second question is to build laconic PSI in a black-box way from assumptions not involving ϕ-hiding (e.g., pairings alone).

In this work, we build DV-NIZK for proving equality of plaintexts across different encryption schemes, namely between the DJ [DJ01] and the BGN [BGN05, GOS06] encryption schemes. This scheme opens the door to new applications since it allows us to extend the capabilities of GS/GOS proof systems [GOS06, GS08] to non-pairing-based primitives with additional properties (in our case to the DJ cryptosystem). We believe that these ideas will have applications beyond range proofs, e.g., one can think of further uses of structure preserving cryptography, so we leave this as an open problem for future works.

2 Technical Overview

2.1 Semi-Honest PSI from CDH/LWE

Our protocol uses hash encryption and garbled circuits, building on [DG17b, BLSV18, GHMR18], while introducing new techniques. A hash-encryption scheme allows one to encrypt a message m to the output h of a hash function by specifying an index/bit (i, b) (denoted $\mathsf{HEnc}(h, (i, b), m)$), so that knowledge of a consistent pre-image value z allows for decryption ($\mathsf{Hash}(z) = h$ and $z_i = b$) while having semantic security against inconsistent pre-image values (i.e., against z where $\mathsf{Hash}(z) = h$ but $z_i = \bar{b}$).[4]

In all discussion below we assume the sender's and receiver's elements are in $\{0,1\}^\lambda$ and that the output of Hash also has λ bits.

Receiver's set size is 2. We first assume the receiver has only two elements $S_\mathsf{R} = \{\mathsf{id}_1, \mathsf{id}_2\}$ and the sender has a single element id. The receiver sends $\mathsf{hr}_\mathsf{root} := \mathsf{Hash}(\mathsf{id}_1, \mathsf{id}_2)$. Consider a circuit $\mathsf{F}[\mathsf{id}]$, with id hardwired, which on input $(\mathsf{id}', \mathsf{id}'')$ outputs id if $\mathsf{id} \in \{\mathsf{id}', \mathsf{id}''\}$; else, $\perp$. The sender garbles $\mathsf{F}[\mathsf{id}]$ to get $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}\})$ and sends $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \{\mathsf{ct}_{i,b}\})$, where $\mathsf{ct}_{i,b} := \mathsf{HEnc}(\mathsf{hr}_\mathsf{root}, \mathsf{lb}_{i,b}, (i, b)))$. The receiver who has the pre-image $z := (\mathsf{id}_1, \mathsf{id}_2)$ can retrieve only the labels lb_{i,z_i}, and the rest will be hidden. Thus, by garbled circuit security the receiver will only learn the output of $\mathsf{F}[\mathsf{id}](\mathsf{id}_1, \mathsf{id}_2)$, as desired.

Moving beyond $|S_\mathsf{R}| = 2$. Suppose the receiver has four elements $S_\mathsf{R} = \{\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_3, \mathsf{id}_4\}$ in ascending order. The receiver Merkle-hashes all these values and sends $\mathsf{hr}_\mathsf{root}$, the root hash. Let h_1 and h_2 be the two hash values at level one (i.e., $h_1 = \mathsf{Hash}(\mathsf{id}_1, \mathsf{id}_2)$). If the sender knows the value of, say, h_1, he may hash-encrypt $\{\mathsf{lb}_{i,b}\}$ (defined in the previous paragraph) under h_1, so that the receiver can only open the labels

[3] Such schemes were also studied in [IP07, LNO13, HW15].

[4] Enc also takes as input a public parameter pp, which we ignore here.

that correspond to the bits of $z = (\mathsf{id}_1, \mathsf{id}_2)$, revealing the value of $\mathsf{F}[\mathsf{id}](\mathsf{id}_1, \mathsf{id}_2)$. However, h_1 is statistically hidden given $\mathsf{hr}_{\mathsf{root}}$. Thus, we use the idea of deferred evaluation [DG17b, CDG+17, DG17a, BLSV18], delegating the task of hash-encrypting $\{\mathsf{lb}_{i,b}\}$ to the receiver herself, via garbled circuits.

In essence, we want the receiver to be able to compute the hash encryption of $\{\mathsf{lb}_{i,b}\}$ wrt either h_1 or h_2 (depending on whether $\mathsf{id} \leq \mathsf{id}_2$ or not), but not both; because obtaining both hash encryptions will allow the receiver to open both labels $\mathsf{lb}_{i,0}$ and $\mathsf{lb}_{i,1}$ for some indices i (because $(\mathsf{id}_1, \mathsf{id}_2) \neq (\mathsf{id}_3, \mathsf{id}_4)$), destroying garbled circuit security. Thus, the sender has to make sure that the receiver will be able to obtain only either of the above hash encryptions, the one whose sub-tree sandwiches id. To enable this, we perform binary search.

Performing binary search. We handle the above difficulty by performing *binary search* using ideas developed in the context of registration-based encryption [GHMR18]. The hash of each node is now computed as the hash of the concatenation of its left child's hash, right child's hash, and the largest identity under its left child. For example, the hash root is $\mathsf{hr}_{\mathsf{root}} = \mathsf{Hash}(h_1, h_2, \mathsf{id}_2)$, where h_1 and h_2 are the hash values of the two nodes in the first level, and in turn $h_1 = \mathsf{Hash}(\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$. Now let id be the sender's element, and change $\mathsf{F}[\mathsf{id}]$ to be a circuit that on input $(\mathsf{id}', \mathsf{id}'', *)$ outputs id if $\mathsf{id} \in \{\mathsf{id}', \mathsf{id}''\}$, else $\perp$. Letting $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}\})$ be the garbling of $\mathsf{F}[\mathsf{id}]$, consider a circuit $\mathsf{G}[\mathsf{id}, \{\mathsf{lb}_{i,b}\}]$ which on input (h, h', id') outputs a hash-encryption of $\mathsf{lb}_{i,b}$ either under h or under h', depending on whether $\mathsf{id} \leq \mathsf{id}'$ or $\mathsf{id} > \mathsf{id}'$. Let $(\widetilde{\mathsf{C}}', \{\mathsf{lb}'\}_{i,b})$ be the garbling of $\mathsf{G}[\mathsf{id}, \{\mathsf{lb}_{i,b}\}]$, let $\{\mathsf{ct}_{i,b}\}$ be hash encryption of $\{\mathsf{lb}'_{i,b}\}$ wrt $\mathsf{hr}_{\mathsf{root}}$, and return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \widetilde{\mathsf{C}}', \{\mathsf{ct}_{i,b}\})$. Using the pre-image $z := (h_1, h_2, \mathsf{id}_2)$ of $\mathsf{hr}_{\mathsf{root}}$, the receiver can retrieve the labels $\{\mathsf{lb}'_{i,z[i]}\}$, allowing to compute $\mathsf{G}[\mathsf{id}, \{\mathsf{lb}_{i,b}\}](h_1, h_2, \mathsf{id}_2)$, which will produce a hash encryption $\{\mathsf{ct}'_{i,b}\}$ of $\{\mathsf{lb}_{i,b}\}$ under either h_1 or h_2, depending on whether $\mathsf{id} \leq \mathsf{id}_2$, or not. For concreteness, suppose $\mathsf{id} \leq \mathsf{id}_2$, meaning that $\{\mathsf{ct}'_{i,b}\}$ are formed under h_1, and so the pre-image $z' = (\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$ of h_1 will lead to $\{\mathsf{lb}_{i,z'_i}\}$, which along with $\widetilde{\mathsf{C}}_0$ will reveal the value of $\mathsf{F}[\mathsf{id}](\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$. Of course, the receiver in *a priori* does not know whether $\{\mathsf{ct}'_{i,b}\}$ are encryptions under h_1 or h_2, so the receiver should try decrypting wrt both, and see which one succeeds.

Are we done? Unfortunately, when arguing security, a subtle issue emerges. Suppose a hash-encryption ciphertext reveals its hash value (e.g., the hash is appended to the ciphertext). Then, the ciphertexts $\{\mathsf{ct}'_{i,b}\}$ will reveal whether they were encrypted under h_1 or h_2; equivalently, whether $\mathsf{id} \leq \mathsf{id}_2$ or $\mathsf{id} > \mathsf{id}_2$. We cannot allow this information to be leaked if $\mathsf{id} \notin S_{\mathsf{R}}$. To fix this issue we assume the hash-encryption scheme is *anonymous*, meaning that, roughly, a random ciphertext leaks no information about the underlying hash value. This property was defined in [BLSV18] for achieving anonymous IBE. The use of anonymous hash encryption does not resolve the issue completely yet. For concreteness, suppose $\mathsf{id} < \mathsf{id}_1$. This means that $\{\mathsf{ct}'_{i,b}\}$ is encrypted under h_1, and so by decrypting $\{\mathsf{ct}'_{i,b}\}$ using $z' = (\mathsf{id}_1, \mathsf{id}_2, \mathsf{id}_1)$, the receiver will obtain meaningful labels, evaluating the garbled circuit $\widetilde{\mathsf{C}}_0$ to $\perp$ (rightly so, because $\mathsf{id} \notin S_{\mathsf{R}}$). On the other hand, if the receiver tries decrypting $\{\mathsf{ct}'_{i,b}\}$ using $z'' = (\mathsf{id}_3, \mathsf{id}_4, \mathsf{id}_3)$ which is not a pre-image of h_1, then the resulting labels will be meaningless, evaluating $\widetilde{\mathsf{C}}_0$ to junk. This leaks which path is the right binary search path, giving information about id. To fix this issue, we change the circuit F so that if $\mathsf{id} \notin S_{\mathsf{R}}$, then decryption along any path will result in a random value. Specifically, sample two random values r and r', let $\mathsf{F}[\mathsf{id}, r, r'](\mathsf{id}', \mathsf{id}'', *)$ return r if $\mathsf{id} \notin \{\mathsf{id}', \mathsf{id}''\}$ and r' otherwise. We will also include r in the clear in psi_2. Now the receiver can check decryption along which path (if any) yields r; in which case, the receiver can determine the intersection identity. To argue security, if we use anonymous garbled circuits [BLSV18], then we can argue if $\mathsf{id} \notin S_{\mathsf{R}}$, then psi_2 is pseudorandom to the receiver. Arguing this formally (especially for the general case) is non-trivial, requiring a delicate formulation of hybrids.

Receiver's security? The receiver's hash $\mathsf{hr}_{\mathsf{root}}$ is computed deterministically from S_{R}, so it cannot be secure. But this is easy to fix: On the leaf level we append the identities with random values, and only then will perform the Merkel hash.

2.2 Reusable Laconic PSI

We now outline our techniques for obtaining laconic PSI in a black-box way, for both semi-honest and malicious cases.

A semi-honestly secure protocol Our starting point is a recent construction of a *one-way function with encryption* from the ϕ-hiding assumption due to Goyal, Vusirikala and Waters [GVW20], and we remark that similar *accumulator-style* ideas were used before to construct PSI [ADT11]. Since the protocol of [GVW20] is "almost" a PSI protocol, we will directly describe the underlying semi-honestly secure PSI based. Assume for a moment that both the receiver's input S_{R} and the sender's input S_{S} are subsets of a polynomially-sized universe $\mathcal{U} = \{1, \ldots, \ell\}$. We will later remove this size-restriction on $\mathcal{U}$. We have a common reference string crs which is composed of an RSA modulus $N = PQ$, a uniformly random generator $g \in \mathbb{Z}_N^*$ and pairwise distinct primes $p_1, \ldots, p_\ell$.

For the sake of simplicity, we will assume in this outline that the sender's input set S_{S} is a singleton set $\{w\} \subseteq \mathcal{U}$. The actual protocol will be obtained by running the protocol we will now sketch for every element in the sender's input set. The protocol commences as follows: The receiver first *hashes* its input set into

$$h = g^{r \prod_{i \in S_{\mathsf{R}}} p_i},$$

where r is chosen a uniformly chosen random from $[N]$ (and thus $r \bmod \phi(N)$ is statistically close to uniform). The receiver then sends h to the sender.

The sender, whose input is $S_{\mathsf{S}} = \{w\}$, chooses a uniformly random value $\rho \leftarrow_{\$} [N]$ and a uniformly random seed s for a suitable randomness extractor Ext, and computes the values $f \leftarrow g^{\rho p_w}$ and $R \leftarrow \mathsf{Ext}(s, h^\rho)$. It sends s, f and R to the receiver.

The receiver, upon receiving f and R, will check for all elements $i \in S_{\mathsf{R}}$ whether it holds that $R_i \stackrel{?}{=} R$, for $R_i \leftarrow \mathsf{Ext}(s, f^{r \cdot \prod_{j \in S_{\mathsf{R}} \setminus \{i\}} p_j})$. If it finds such an i, it outputs $\{i\}$ as the intersection of S_{R} and S_{S}. Correctness of this protocol follows routinely[5]. by noting that if $w \in S_{\mathsf{R}}$ then

$$f^{r \cdot \prod_{j \in S_{\mathsf{R}} \setminus \{w\}} p_j} = g^{\rho \cdot r \cdot \prod_{j \in S_{\mathsf{R}}} p_j} = h^\rho.$$

Also, note that this scheme is laconic, as the size of the messages exchanged by the parties is independent of the size of the set S_{R}.

Arguing security against a semi-honest sender is also routine, as h is in fact statistically close to a uniformly random group element in $\mathbb{Z}_N^*$. Proving security against a semi-honest receiver is a bit more involved and proceeds via the following hybrid modifications. Let $S_{\mathsf{S}} = \{w\}$ be the sender's input such that $w \notin S_{\mathsf{R}}$. In the first hybrid, we will choose the modulus N such that p_w divides $\phi(N)$; under the ϕ-hiding assumption this change will go unnoticed. Now, via a standard lossiness-argument, we have that $f = g^{\rho p_w}$ loses information about g^ρ, i.e., g^ρ has high min-entropy given f. This means that $h^\rho = g^{\rho r \cdot \prod_{i \in S_{\mathsf{R}}} p_i}$ has also high min-entropy as $w \notin S_{\mathsf{R}}$ and thus p_w does not divide $r \cdot \prod_{i \in S_{\mathsf{R}}} p_i$ (w.o.p). Consequently, as h^ρ has high min-entropy conditioned on f, in the next hybrid change we can replace $R = \mathsf{Ext}(s, h^\rho)$ with a uniformly random value, incurring only a negligible statistical distance via the extraction property of Ext. In the next hybrid change, we can switch the modulus N back to normal mode, i.e., such that p_w does not divide $\phi(N)$. But now $f = g^{\rho p_w}$ is statistically close to uniform in $\mathbb{Z}_N^*$. Thus, in the last hybrid change we can replace f with a uniformly random value in $\mathbb{Z}_N^*$ and get that the view of the receiver is independent of w, as required.

For the case that the sender's input S_{S} contains more than a single element, we mount a hybrid argument repeating the above modifications for each element of S_{S} not in the receiver's set S_{R}.

Large universes The above protocol has the drawback that the size of the common reference string crs depends linearly on the size of the universe $\mathcal{U}$, which is highly undesirable. There is a standard way of overcoming this issue: Instead of explicitly listing all the primes p_i in crs, we will describe them implicitly

[5] We will not further discuss the small correctness-error of this protocol as our final protocol will not suffer from this defect

via a pseudorandom function (PRF).[6] For this purpose, we need a PRF which maps into the set of primes of a certain size. This can e.g. be achieved by using rejection sampling: we first sample $y \leftarrow F_k(x|i)$ (starting with $i = 1$) and check if y is a prime number. If it is, we output y; else, we increment i until a prime is hit. Under standard number-theoretic assumptions, this process finds a prime after a logarithmic number of steps. One small issue is that, in the above security proof, we need to replace one of the primes with a prime provided by the ϕ-hiding experiment. We resolve this issue by making the PRF programmable in one point, e.g., by setting $F_{k,k'}(x|i) = F'_k(x|i) \oplus k_i$ for a PRF F', $k' = (k_1, \ldots, k_\xi)$ and a suitable choice of ξ.

A first attempt at malicious sender security Our protocol thus far, however, offers no security against a malicious sender. The main issue is that a corrupted sender may choose the values f and R arbitrarily, and further, there is no mechanism for a simulator against a malicious to extract the senders input w. Of course, this protocol can be made secure against malicious senders by letting the sender prove via a general purpose NIZK proof that it follows the semi-honest protocol correctly. This however would necessitate to make non-black-box use of our semi-honest laconic PSI protocol, contrary to our goal of achieving a fully black-box protocol.

Re-inspecting the above protocol, we have not made full use of the fact that the extracted string R is uniformly random. Our first idea to make the sender extractable is to make better use of R. Instead of sending R in the plain, we will use R as random coins for a public key encryption (PKE) scheme to encrypt the sender's input w. More concretely, we will modify the above protocol as follows. We include a public key pk of a PKE scheme in the common reference string crs and, instead of having the sender include R in the plain in its message to the receiver, it will include a ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, i; R)$. We also need to modify the procedure of the receiver. The receiver will recover R_i as before, but will now use R_i to *re-encrypt* the index i, that is, for each $i \in S_\mathsf{R}$ it will compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, i; R_i)$.

First notice that, as a side bonus, this modification makes our laconic PSI scheme perfectly correct, given that the PKE scheme is perfectly correct, as now ct_i uniquely specifies the element i.

In terms of security, we first observe that this modification does not harm security against a semi-honest receiver given that the PKE scheme is IND-CPA secure. In the above sketch of a security proof, we have argued that, if w is not in the set S_R, then R is uniformly random from the view of the receiver. This means now that ct is a freshly encrypted ciphertext, using fresh random coins (independent of ρ). Moreover, we can use IND-CPA security of the PKE to replace ct with an encryption of 0, and then continue as above to argue security against a semi-honest receiver.

To establish security against a malicious sender, we would like to argue as follows. The simulator can now generate the public key pk in crs together with a secret key sk. Given a message (s, f, ct) by a malicious sender, the simulator can recover the set element w by decrypting the ciphertext ct using sk. At a first glance this seems to provide us with security against malicious senders. And indeed, the simulator will recover all elements for which the receiver would have declared to be in the intersection. There is a grave issue however: The simulator has no means of detecting whether the honest receiver would actually have succeeded in re-encrypting the index i. In other words, the malicious sender can make the simulator *false positives*, such that the simulator declares an element i to be in the intersection, whereas an honest receiver would not have.

Switch groups, extract everything! We need to enable the simulator against a corrupted sender to check whether the honest receiver would have succeeded in re-encrypting w. Indeed, if the simulator had a way of recovering ρ from the group element $f = g^{\rho p_w}$, it could just compute $R \leftarrow \mathsf{Ext}(s, h^\rho)$ on its own and check if the re-encryption test of the receiver succeeds. Our approach to enable this is to replace $\mathbb{Z}_N^*$ by a group in which we can efficiently recover ρp_w from $g^{\rho p_w}$ using a trapdoor.

We briefly recall some facts about the Damgård-Jurik cryptosystem [DJ01]. The group $\mathbb{Z}^*_{N^{\xi+1}}$ contains a cyclic subgroup $\mathbb{NR}_N$ of order $\phi(N)$. Now let $g_0 \in \mathbb{NR}_N$ be a generator of $\mathbb{NR}_N$. Then we can generate the entire group $\mathbb{Z}^*_{N^{\xi+1}}$ by g_0 and $1 + N$, i.e. we can write every $h \in \mathbb{Z}^*_{N^{\xi+1}}$ as $h = g_0^t \cdot (1 + N)^m$ for some

[6] We remark that we use a PRF, not because we want uniform outputs, but to implicitly define the set of primes. A similar trick was used in [BGI16].

$t \in \mathbb{Z}_{\phi(N)}$ and $m \in \mathbb{Z}_{N^\xi}$. Furthermore, we can efficiently compute discrete logarithms relative to $1+N$, i.e. if $h = (1+N)^m$ for an $m \in \mathbb{Z}_{N^\xi}$, then we can efficiently compute m from h. Finally, the decisional composite residue (DCR) assumption in $\mathbb{Z}^*_{N^{\xi+1}}$ states that a random element in $\mathbb{NR}_N$ is indistinguishable from a random element in $\mathbb{Z}^*_{N^{\xi+1}}$. It follows that $g_1 = g_0^{t_1}$ and $g_2 = g_0^{t_2} \cdot (1+N)$ (for uniformly random $t_1, t_2 \leftarrow_\$ \mathbb{Z}_{\phi(N)}$) are computationally indistinguishable. Moreover, if $h = g_2^t$ for a $t < N^{\xi-1}$, we can efficiently compute t from h using $\phi(N)$ as a trapdoor by first computing

$$h^{\phi(N)} = g_2^{t\cdot\phi(N)} = \underbrace{g_0^{t\phi(N)}}_{=1} \cdot (1+N)^{t\cdot\phi(N)} = (1+N)^{t\cdot\phi(N)},$$

from which we can efficiently compute $t \cdot \phi(N)$ (as $t \cdot \phi(N) < N^\xi$) and thus t.

Given this, we will now make the following additional modification to our PSI protocol. Instead of choosing the element g in the common reference string crs to be a random generator of $\mathbb{Z}^*_N$, we choose g to be a random generator of $\mathbb{NR}_N$, where $\mathbb{NR}_N$ is the subgroup of order $\phi(N)$ in $\mathbb{Z}^*_{N^{\xi+1}}$ (for a sufficiently large but constant ξ). Our first observation is that this does not affect the security proof in the case of a semi-honest receiver, since $\mathbb{NR}_N$ is still a cyclic group of order $\phi(N)$ and the above argument using the ϕ-hiding assumption works analogously in this group.

Assume for a moment we had a mechanism which ensures that the group element f in the sender's message is of the form $f = g^a$ for an $a < N^{\xi-1}$. We can then argue security against a malicious sender as follows: First we make a hybrid change and choose the element g in the common reference string like g_2 above, i.e. we choose $g = g_0^t(1+N)$; under the DCR assumption this change goes unnoticed. Now, given that $f = g^a$ for an $a < N^{\xi-1}$ and using $\phi(N)$ as a trapdoor, the simulator can efficiently compute a from f as described above. Since it can also recover the index w from the ciphertext ct as described above, it can now check if a is of the form $a = \rho \cdot p_w$. If so, it recovers ρ and performs the same re-encryption test for ct which the real receiver would perform. This makes the simulation indistinguishable from the real experiment.

2.3 DV-NIZK Range Proofs for DJ Ciphertexts

The final component which is missing to make the above argument succeed is a mechanism which ensures that the group element f is indeed of the form $f = g^a$ for a *small* a. For the sake of generality, we will make the following discussion for general DJ-ciphertexts, that is, ciphertexts of the form $c = h^t \cdot (1+N)^a$ (where $h = g_1^z$ is the public key). If we can show that such a ciphertext encrypts a small value a, proving that $f = g^a$ and $c = h^t \cdot (1+N)^a$ for the same a can be efficiently proven via a standard hash-proof system (HPS) [CS02].

First, we observe that, to show that $c = h^t \cdot (1+N)^a$ encrypts a value $a < 2^k$ for some parameter k, it suffices to prove that some ciphertexts $c_0, \ldots, c_{k-1}$ encrypt *bits* $b_1, \ldots, b_{k-1}$. Assume for now we had a DV-NIZK protocol Π to prove that the ciphertexts $c_0, \ldots, c_{k-1}$ all just encrypt bits. The prover can convince the verifier as follows that c encrypts a value $a < 2^k$. First the prover encrypts bit b_i in a ciphertext c_i and sets $c' = \prod_{i=0}^{k-1} c_i^{2^i}$ (it is not hard to see that c' encrypts a). Now, the prover uses Π to to convince the verifier that $c_0, \ldots, c_{k-1}$ indeed encrypt bits. Furthermore, it can use a standard HPS to prove that c and c' indeed encrypt the same value. Zero-knowledge follows routinely. To see that this protocol is sound, observe that if the c_i indeed encrypt bits, then c' must encrypt a value bounded by 2^k.

A DV-NIZK proof system for ciphertext equality across different encryption schemes Alas, we do not know of a black-box DV-NIZK which proves that DJ ciphertexts encrypt bits. However, for the pairing-based Boneh-Goh-Nissim (BGN) cryptosystem [BGN05], such a proof system was constructed by Groth, Ostrovsky and Sahai [GOS06]. Consequently, if we could prove in a black box way that a BGN ciphertext encrypts the same value as DJ ciphertext we would be done.

Recall that, in the BGN cryptosystem, public keys are of the form (G, H), where G and H a generators of subgroups of a composite-order pairing group $\mathbb{G}$. BGN ciphertexts are of the form $C = G^m H^r$, where m is the encrypted message and r are random coins.

Our final contribution is a DV-NIZK proof system which allows us to prove that a DJ ciphertext and a BGN ciphertext encrypt the same value.

To simplify the description of our prove system, assume we have BGN public keys $(G, H_1), \ldots, (G, H_\ell)$, i.e. each key sharing the same G but having fresh and random H_i, and an element H_0. Furthermore, assume that we have DJ public keys $h_1, \ldots, h_\ell$, and an element h_0. We will assume that both sequences of keys are in a public setup, together with the elements H_0, h_0.

Suppose further that we have BGN ciphertexts $C_1, \ldots, C_\ell$, where $C_i = G^{m_i} H_i^r$, i.e., all ciphertexts use the same random coins r but encrypt possibly different bits m_i.[7] As mentioned above, using the NIZK scheme from [GOS06], we can prove that the ciphertexts $C_i = G^{m_i} H_i^r$ are indeed well-formed and that $m_i \in \{0, 1\}$. Moreover, we have $C_0 = H_0^r$, which can be proven well-formed using a standard hash proof system (HPS) [CS02].

Assume further that we are given DJ ciphertexts $c_1, \ldots, c_\ell$, where $c_i = h_i^t \cdot (1+N)^{m'_i}$, i.e., again the ciphertexts share the same random coins t.[8] Moreover, assume that we have a value h_0^r exactly as above. We want to prove that it holds for all $i \in [\ell]$ that $m_i = m'_i$. Our DV-NIZK proof system for equality of BGN and DJ ciphertexts now proceeds roughly as follows:

- The verifier starts by sampling a uniformly random binary string $\sigma \leftarrow_\$ \{0,1\}^\ell$ and computes $F = H_0^A \prod H_i^{\sigma_i} \in \mathbb{G}$ and $f = h_0^\alpha \prod h_i^{\sigma_i} \in Z^*_{N^{\xi+1}}$, for uniformly random values A, α. It sends $\mathsf{crs} = (F, f)$ to the prover and keeps σ as the designated-verifier key.
- The prover is given ciphertexts $C_1, \ldots, C_\ell$ and $c_1, \ldots, c_\ell$ with $C_i = G^{m^i} H_i^r$ and $c_i = h_i^t (1+N)^{m_i}$, and the values $C_0 = H_0^r$ and $c_0 = h_0^t$. It computes $K = F^r G^\tau$ and $k = f^t (1+N)^\tau$ where τ is sampled according to a distribution which is wide enough to drown the m_i, but short enough such that it is bounded by N. The proof π is consists of (K, k).
- The verifier, given the proof $\pi = (K, k)$, computes the discrete log y (in base $(1+N)$) of $k^{-1} c_0^\alpha \prod_{i=1}^\ell c_i^{\sigma_i}$ and checks if $G^y = K^{-1} C_0^A \prod_{i=1}^\ell C_i^{\sigma_i}$.

For completeness, note that

$$\begin{aligned} k^{-1} c_0^\alpha \prod c_i^{\sigma_i} &= \left(h_0^\alpha \prod h_i^{\sigma_i}\right)^{-t} (1+N)^{-\tau} \left(h_0^t\right)^\alpha \prod \left(h_i^t (1+N)^{m_i}\right)^{\sigma_i} \\ &= (1+N)^{\sum \sigma_i m_i - \tau}, \end{aligned}$$

from which the verifier can recover $y = \sum \sigma_i m_i - \tau$. Moreover

$$L = K^{-1} C_0^A \prod C_i^{\sigma_i} = \left(H_0^A \prod H_i^{\sigma_i}\right)^{-r} G^{-\tau} (H_0^r)^A \prod (H_i^r G^{m_i})^{\sigma_i} = G^{\sum \sigma_i m_i - \tau}$$

and thus $G^y = L$.

The zero-knowledge property can be established by noting that the term τ statistically drowns $\sum_i \sigma_i m_i$.

To prove *reusable statistical soundness* (or simulation soundness), we argue as follows. First note that σ is statistically hidden, given $F = H_0^A \prod H_i^{\sigma_i}$ and $f = h_0^\alpha \prod h_i^{\sigma_i}$, by the uniform values A, α. We need to show that if there is an index i for which $m_i \neq, m'_i$, then the verifier will reject with high probability, irrespective of the (adversarial) choices of τ, τ' (which are not necessarily short)[9]. It follows from the above description that the verifier accepts a proof if the condition

$$\sum \sigma_{i,j} m_i - \tau_j \mod n = \left(\sum \sigma_{i,j} m'_i - \tau'_j \bmod N^\xi\right) \mod n$$

is satisfied, where n is the order of the subgroup of $\mathbb{G}$ generated by G. In the main body we will show that, given that $n > N^\xi$, this condition will be violated with probability $\approx 1/2$ if the there exists an index i for which $m_i \neq m'_i$. By repeating the protocol λ times, we achieve negligible soundness error.

[7] Via a standard rerandomization argument we can show that reusing the same random coins across different keys does not harm CPA security.

[8] Same as above.

[9] We assume that the verifier rejects if it fails to compute the discrete logarithm of $k^{-1} \prod d_i^{\sigma_i}$.

2.4 Labeled Laconic PSI and Laconic OT

Our laconic PSI construction can be easily extended into a labeled laconic PSI, in which the receiver also learns labels associated with set elements in the intersection. To achieve this, we simply use an extractor with an output size twice as large: the first half is used as above to perform the re-encryption step; the other half is used as an one-time pad to encrypt the corresponding label. It is easy to see that the receiver can only recover the labels for the elements within the intersection, since the security proof follows the same blueprint as before.

We also build an LOT using the same ideas as above. The receiver commits to a database $D \in \{0,1\}^\Gamma$ by computing $h = g_0^{r \prod_{i=1}^{\Gamma} e_{i,D_i}} \bmod N^{\xi+1}$, where each prime $e_{i,b}$ is the output of a PRF (just as before). The sender computes $f_j = g_0^{\rho_j e_{L,j}}, F_j = g_1^{\rho_j e_{L,j}}(1+N)^{\rho_j e_{L,j}}$ for each $j \in \{0,1\}$, together with a range proof. Moreover, he encrypts each message as $\mathsf{ct}_j = k_j \oplus m_j$ where $k_j \leftarrow \mathsf{Ext}(s_j, h^{\rho_j})$. Again, security follows the same reasoning as above. Our LOT protocol is the first one to provide security against a malicious sender while incurring in communication complexity independent of the size of D.

3 Preliminaries

The acronym PPT denotes "probabilistic polynomial time". Throughout this work, λ denotes the security parameter. By $\mathsf{negl}(\lambda)$, we denote a negligible function in λ, that is, a function that vanishes faster than any inverse polynomial in λ.

Let $n \in \mathbb{N}$. Then, $[n]$ denotes the set $\{1, \ldots, n\}$. If $\mathcal{A}$ is an algorithm, we denote by $y \leftarrow \mathcal{A}(x)$ the output y after running $\mathcal{A}$ on input x. If S is a (finite) set, we denote by $x \leftarrow\!\!\$\, S$ the experiment of sampling uniformly at random an element x from S. If D is a distribution over S, we denote by $x \leftarrow\!\!\$\, D$ the element x sampled from S according to D. We say that D is B-bounded if for every $x \leftarrow\!\!\$\, D$, we have $|x| < B$, except with negligible probability. If D_0, D_1 are two distributions, we say that D_0 is statistically indistinguishable from D_1, denoted by $D_0 \approx_\varepsilon D_1$, if no unbounded adversary can distinguish both distributions except with probability ε.

Throughout this work, ϕ will denote the Euler's totient function.

Let $\Phi_{\mathbb{Z},\beta}$ be the distribution that outputs a uniformly chosen value in $\mathbb{Z}$ from the interval $[-\beta, \beta]$. We call *shifted rectangle* to this distribution [AIK11]. The following lemma states that we can *drown* (i.e., statistically hide) a value using a sample from a much wider $\Phi_{\mathbb{Z},\beta}$ distribution.

Lemma 1 (Drowning [AIK11])**.** *Let $B_0 \in \mathbb{N}$ and $\beta \in \mathbb{Z}$ and let $e_0 \in [-B_0, B_0]$. Let $e_1 \leftarrow\!\!\$\, \Phi_{\mathbb{Z},\beta}$. If $B_0/s = \mathsf{negl}(\lambda)$ then $e_1 \approx_{\mathsf{negl}(\lambda)} e_0 + e_1$.*

3.1 Cryptographic Primitives

Here, we present the cryptographic primitives which are meaningful to this work, as well as their security properties.

3.1.1 Strong Extractors

Extractor allow to extract randomness from sources with a certain min-entropy.

Definition 1 (Strong Extractor)**.** *A (k,ε)-strong extractor $\mathsf{Ext} : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is a deterministic algorithm with domain $\mathcal{X}$, seed space $\mathcal{S}$ and range $\mathcal{Y}$ with the following property: For every distribution X with support $\mathcal{X}$ and min-entropy at least k,*

$$(s, \mathsf{Ext}(s,x)) \approx_\varepsilon (s, y)$$

where $x \leftarrow\!\!\$\, X$ and $y \leftarrow\!\!\$\, \mathcal{Y}$.

3.1.2 Public-Key Encryption

We recall the classical definition of public-key encryption (PKE).

Definition 2 (Public-Key Encryption)**.** *A* Public-Key Encryption *(PKE) scheme is defined by the following algorithms:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input a security parameter. It outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Enc}(\mathsf{pk}, m)$ *takes as input a public key* pk *and a message* $m \in \{0,1\}^*$*. It outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *takes as input a secret keys* sk *and a ciphertext* ct*. It outputs a a message* m *or bot* $\perp$*.*

We require the usual correctness and IND-CPA properties for a PKE.

- ***Correctness:*** *We say that a PKE is* correct *if*

$$\Pr\left[m \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) : \ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \ \right] = 1.$$

- ***IND-CPA security:*** *For any PPT adversary* $\mathcal{A}$*, we require that*

$$\Pr\left[b \leftarrow \mathcal{A}(\mathsf{ct}, \mathsf{st}) : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda);\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk}) \\ b \leftarrow\!\!\$\, \{0,1\};\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

3.1.3 Programmable Pseudorandom Function

Pseudorandom functions (PRF) are ubiquitous objects in cryptography. We present the definition of PRF in the following.

Definition 3 (Pseudorandom Function)**.** *A* Pseudorandom Function *(PRF) is defined by a keyed function* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *such that, for any PPT adversary* $\mathcal{A}$

$$|\Pr[1 \leftarrow \mathcal{A}(y, x) : y \leftarrow \mathsf{PRF}(k, x)] - \Pr[1 \leftarrow \mathcal{A}(y, x) : y \leftarrow f(x)]| \leq \mathsf{negl}(\lambda)$$

for any $x \in \mathcal{X}$*, where* $f : \mathcal{X} \to \mathcal{Y}$ *is a uniformly chosen random function and the key* k *is sampled uniformly at random from* $\mathcal{K}$*.*

A programmable PRF allows the simulator to program the output of a PRF on several inputs at key generation time.

Definition 4 (Programmable PRF [KMP+17])**.** *A programmable PRF (PPRF) is composed by the following algorithms:*

- $k = (k', \mathsf{hint}) \leftarrow \mathsf{KeyGen}(1^\lambda, (x, y))$ *takes as input a security parameter and a pair of points* $(x, y) \in \mathcal{X} \times \mathcal{Y}$*. It outputs a key* k' *and a hint* hint*.*
- $y \leftarrow \mathsf{PPRF}(k, x)$ *takes as input a key* $k \in \mathcal{K}$ *and a value* $x \in \mathcal{X}$*. It outputs* $y \in \mathcal{Y}$*.*

Correctness of the PPRF states that $y \leftarrow \mathsf{PPRF}(k, x)$ for the programmed point (x, y). Security roughly states that it is hard for the adversary to guess the point x which was programmed even given the hint (see [KMP+17]).

An example. Let $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \{0,1\}^\ell$ and $\mathsf{Primes}(\ell)$ be the primes of length ℓ. In this work, we use a programmable PRF $\mathsf{PPRF} : \mathcal{K} \times (\mathcal{X} \times \mathbb{Z}) \to \mathsf{Primes}(\ell)$ in which the key (and the hint) is of the form $K = (k, k' = (k'_1, \ldots, k'_\xi)) \in \mathcal{K} \times \{0,1\}^{\ell\xi}$ and where the output of an element $x \in \mathcal{X}$ is computed as: i) Start by initializing $i = 1$. ii) Compute $y = \mathsf{PPRF}(k, (x, i)) \oplus k'_i$. iii) Output y, if it is a prime number; else, set $i = i + 1$ and return to step ii); repeat until $i = \xi$. It is easy to see that, under standard number-theoretic assumptions, the process described above outputs a prime number after $\mathcal{O}(\log 2^\ell)$ steps (e.g., [FT14]). If we set $\xi \in \mathcal{O}((\log 2^\ell)^2)$, a direct calculation yields that the probability of not existing any $i \in [\xi]$ such that $\mathsf{PPRF}(k, (x, i)) \oplus k'_i$ is not a prime is negligible in ℓ.

In order to program the output of PPRF at some input x, we first sample a prime number p and a index i from a suitable distribution.[10] Then, we set $k'_i = p \oplus \mathsf{PPRF}(k, (x, i))$. Finally, we choose k'_j, for all $j < i$, uniformly at random such that $\mathsf{PPRF}(k, (x, j)) \oplus k'_j$ is not a prime number. All other k'_j, for $j > i$ are chosen uniformly at random. Such a procedure will succeed with non-negligible probability.

This is a special case of the PPRF designed in [KMP+17] and it is easy to see that, if the PPRF is programmed on a pair of points $(x, y) \in \mathcal{X} \times \mathsf{Primes}(\ell)$ where $y \leftarrow\!\!\$\, \mathsf{Primes}(\ell)$, then it is hard for any PPT adversary $\mathcal{A}$ to guess the programmed point x.

A remark. We slightly overload the notation and denote k as the PPRF key (which is composed by a PRF key k' and a hint hint as in Definition 4). We do this because, in our case, the hint (when it is a uniformly random value) reveals nothing about the programmed value [KMP+17]. That is, we will use the notation $K \leftarrow \mathsf{KeyGen}(1^\lambda, (x, y))$ where $K = (k, k' = \mathsf{hint})$.

3.1.4 Designated-Verifier Non-Interactive Zero-Knowledge

NIZK is a cryptographic primitive that allows a prover to prove that it holds a witness for a certain NP statement to a verifier in just one message. In the designated-verifier setting, only a designated party can verify the validity of proofs. This is in contrast with standard NIZK where the verification algorithm can be run by any party.

Let $\mathcal{Z}$ be the set of statements and $\mathcal{W}$ be the set of witnesses. Let $\mathcal{L}$ be a NP language with relation $\mathcal{R}$ such that $z \in \mathcal{L}$ if there is a $w \in \mathcal{W}$ such that $\mathcal{R}(z, w) = 1$.

Definition 5 (DV-NIZK)**.** *Let $\mathcal{L}$ be a NP language. A* Designated-Verifier Non-Interactive Zero-Knowledge *(DV-NIZK) for language $\mathcal{L}$ is composed by the following algorithms:*

- $\mathsf{GenCRS}_\mathcal{L}(1^\lambda)$ *takes as input a security parameter. It outputs a common reference string* crs *together with the corresponding trapdoor* td*.*
- $\mathsf{Prove}_\mathcal{L}(\mathsf{crs}, x, w)$ *takes as input a common reference string* crs*, a statement x and a witness w. It outputs a proof π.*
- $\mathsf{Verify}_\mathcal{L}(\mathsf{td}, x, \pi)$ *takes as input a common reference string* crs*, a trapdoor* td*, a statement x and a proof π. It outputs a bit $b \in \{0,1\}$.*

A DV-NIZK should fulfill the following properties: completeness, soundness and honest-verifier zero-knowledge.

- ***Completeness:*** *A DV-NIZK is* correct *if for all pairs (x, w) such that $\mathcal{R}(x, w) = 1$,*

$$\Pr\left[1 \leftarrow \mathsf{Verify}_\mathcal{L}(\mathsf{td}, x, \pi) : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_\mathcal{L}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}_\mathcal{L}(\mathsf{crs}, x, w)\end{array}\right] = 1.$$

[10] The index i is sampled from the distribution of the number of uniform samples we need to perform in order to find a prime number. Such a distribution can be easily simulated by just running a prime sampler with true randomness and output i (the number of trials until success) instead of the prime.

- ***Statistical Simulation Soundness:*** *A DV-NIZK is* statistical simulation sound *if for all computationally unbounded adversaries $\mathcal{A}$ and all $x \notin \mathcal{L}$,*

$$\Pr\left[1 \leftarrow \mathsf{Verify}_{\mathcal{L}}(\mathsf{td}, x, \pi) : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^\lambda) \\ \pi \leftarrow \mathcal{A}^{\mathsf{Verify}_{\mathcal{L}}(\mathsf{td},\cdot,\cdot)}(\mathsf{crs}, x)\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Remark that, in the statistical setting, selective soundness is equivalent to adaptive soundness.

- ***Zero-knowledge:*** *A DV-NIZK is said to be* zero-knowledge *if for all adversaries $\mathcal{A}$ there is an simulator* Sim *such that*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}, x, \pi) : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}_{\mathcal{L}}(\mathsf{crs}, x, w)\end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}, x, \pi) : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{GenCRS}_{\mathcal{L}}(1^\lambda) \\ \pi \leftarrow \mathsf{Sim}_{\mathcal{L}}(\mathsf{td}, x)\end{array}\right]\right| \leq \mathsf{negl}(\lambda)\,.$$

When $\mathcal{A}$ is computationally bounded, we say that zero-knowledge holds computationally. When $\mathcal{A}$ is computationally unbounded, if its advantage is negligible in the security parameter, we say that zero-knowledge holds statistically while if its advantage is zero, then zero-knowledge holds perfectly.

Range Proof Systems for DJ Ciphertexts. In this work, we construct a range proof system for DJ ciphertexts. That is, we build a DV-NIZK scheme that allows the prover to prove that a given DJ ciphertext ct encrypts a message $m \in [-B, B]$ for some public $B \in \mathbb{Z}$.

Such a scheme can be constructed in the random oracle model (ROM) using the Fiat-Shamir transform (e.g., [DJ01, BBC+18, BBB+18, TBM+20] just to name a few). However, we focus on efficient range proofs in the standard model in this work.

3.2 Hardness Assumptions

We start by introducing some notation. Let $\mathsf{Primes}(\kappa)$ denote the set of prime numbers of bit-length κ. Let

$$\mathsf{RSA}(\lambda) = \{N : N = PQ \text{ and } P, Q \in \mathsf{Primes}(\lambda/2) \text{ and } \gcd(P-1, Q-1) = 2\}$$

and

$$\mathsf{RSA}_e(\lambda) = \{N : e | \phi(N)\}$$

for any $e \leq 2^\lambda$.

Definition 6 (Phi-Hiding)**.** *The* phi-hiding *assumption, denoted as ϕ-hiding, states that for all $\varepsilon > 0$ and $3 < e < 2^{\lambda/4-\varepsilon}$ and all PPT adversaries $\mathcal{A}$, we have that*

$$|\Pr[1 \leftarrow \mathcal{A}(N, e) : N \leftarrow_\$ \mathsf{RSA}(\lambda)] - \Pr[1 \leftarrow \mathcal{A}(N, e) : N \leftarrow_\$ \mathsf{RSA}_e(\lambda)]| \leq \mathsf{negl}(\lambda)\,.$$

Let $N = PQ$ for $P, Q \in \mathsf{Primes}$ and consider the multiplicative group $\mathbb{Z}^*_{N^{\xi+1}}$ where ξ is a fixed non-negative integer. Recall that $\mathbb{Z}^*_{N^{\xi+1}}$ can be written as the product of two subgroups $\mathbb{H}_N \times \mathbb{NR}_N$ where $\mathbb{H}_N = \{(1+N)^i : i \in [N^\xi]\}$ and $\mathbb{NR}_N = \{x^{N^\xi} : x \in \mathbb{Z}^*_N\}$ (the group of N^ξ-residues) which has order $\phi(N)$. Given $(1+N)^m \bmod N^{\xi+1}$, there is a polynomial-time algorithm that allows to recover m [DJ01]

The following lemma is straightforwardly adapted from [GVW20].

Lemma 2 ([GVW20])**.** *Assume that the ϕ-hiding assumption holds. Let* Ext *be a $(\kappa - 1, \mathsf{negl}(\lambda))$-strong extractor. For every admissible stateful PPT adversary $\mathcal{A}$ and for all λ, κ such that $\lambda \geq 5\kappa$, we have that*

$$\left|\Pr\left[b \leftarrow \mathcal{A}(y_b) : \begin{array}{c} N \leftarrow_\$ \mathsf{RSA}(\lambda);\ s \leftarrow_\$ \{0,1\}^\lambda \\ e \leftarrow_\$ \mathsf{Primes}(\kappa);\ g \leftarrow \mathbb{NR}_N \\ G \leftarrow \mathcal{A}(N, s, e, g);\ b \leftarrow_\$ \{0,1\} \\ y_0 \leftarrow \mathsf{Ext}(s, g^{Ge^{-1}} \bmod N^{\xi+1});\ y_1 \leftarrow_\$ \mathcal{Y}\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where an admissible adversary is one that outputs G such that e does not divide G.

In this work, we also make use of the DCR assumption which we define in the following. We present the DCR assumption as a subgroup indistinguishability assumption [BG10].

Definition 7 (Decisional Composite Residuosity)**.** *Let $N = \mathsf{RSA}(\lambda)$ and let $\xi \geq 0$ be a fixed integer. The* decisional composite residuosity *(DCR) assumption states that for all PPT adversaries $\mathcal{A}$,*

$$|\Pr[1 \leftarrow \mathcal{A}(N, x) : x \leftarrow_{\$} \mathbb{Z}^*_{N^{\xi+1}}] - \Pr[1 \leftarrow \mathcal{A}(N, x) : x \leftarrow_{\$} \mathbb{NR}_N]| \leq \mathsf{negl}(\lambda)\,.$$

Corollary 1. *Assume that the DCR assumption holds. Then for all PPT adversaries $\mathcal{A}$,*

$$\left| \begin{array}{c} \Pr[1 \leftarrow \mathcal{A}(N, x) : x \leftarrow_{\$} \mathbb{NR}_N] - \\ \Pr\left[1 \leftarrow \mathcal{A}(N, x) : \begin{array}{c} x' \leftarrow_{\$} \mathbb{NR}_N \\ x = x'(1+N)^{-1} \bmod N^{\xi+1} \end{array}\right] \end{array} \right| \leq \mathsf{negl}(\lambda)\,.$$

sketch. It is natural to see that we can switch the group for x and x' from $\mathbb{NR}_N$ to $\mathbb{Z}^*_{N^{\xi+1}}$ unnoticeable under DCR assumptions, then $x' = y(1+N)^i \bmod N^{\xi+1}$ for some $y \in \mathbb{NR}_N$ and $i \in \mathbb{Z}_{N^\xi}$, thus $x \leftarrow \mathbb{Z}^*_{N^{\xi+1}}$ and $x'(1+N)^{-1}$ have the same distribution. □

We also use Boneh-Goh-Nissim (BGN) cryptosystem [BGN05] in our range proofs. Thus, its underlying subgroup decision assumption is rephrased as follows for completeness.

Let $\mathcal{G}$ be an algorithm that takes a security parameter as input and outputs $\mathsf{val} := (p, q, \mathbb{G}, \mathbb{G}_1, e)$ such that p, q are primes, $n = pq$ and $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order n and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. Let $q\mathbb{G}$ be the subgroup of $\mathbb{G}$ of order q.

Definition 8 (Subgroup Decision [GOS06])**.** *Let $\mathcal{G}$ be an algorithm that takes a security parameter as input and outputs $\mathsf{val} := (p, q, \mathbb{G}, \mathbb{G}_1, e)$ such that p, q are primes, $n = pq$ and $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order n and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. Let $q\mathbb{G}$ be the subgroup of $\mathbb{G}$ of order q. The* subgroup decision *(SD) assumption holds for generator $\mathcal{G}$ states that for all PPT adversaries $\mathcal{A}$,*

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, G, H) : \begin{array}{c} \mathsf{val} \leftarrow \mathcal{G}(1^k), \quad n = pq \\ G, H \leftarrow \mathbb{G}_{gen} \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, G, H) : \begin{array}{c} \mathsf{val} \leftarrow \mathcal{G}(1^k), \quad n = pq \\ G \leftarrow \mathbb{G}_{gen}, \quad H \leftarrow q\mathbb{G} \setminus \{1\} \end{array}\right] \end{array} \right| \leq \mathsf{negl}(\lambda)\,.$$

We also present the definitions of the computational Diffie-Hellman (CDH) and learning with errors (LWE) assumptions.

Definition 9 (Computational Diffie-Hellman)**.** *Let $\mathcal{G}(\lambda)$ be an algorithm that outputs $(\mathbb{G}, p, g)$ where $\mathbb{G}$ be a group of prime order p and g a generator of the group. The CDH assumption holds for generator $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[g^{a_1 a_2} \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^{a_1}, g^{a_2}) : \begin{array}{c} (\mathbb{G}, p, g) \leftarrow \mathcal{G}(\lambda) \\ a_1, a_2 \leftarrow_{\$} \mathbb{Z}_p \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Definition 10 (Learning with Errors)**.** *Let $q, k \in \mathbb{N}$ where $k \in \mathsf{poly}(\lambda)$, $\mathbf{A} \in \mathbb{Z}_q^{k \times n}$ and $\beta \in \mathbb{R}$. For any $n = \mathsf{poly}(k \log q)$, the LWE assumption holds if for every PPT algorithm $\mathcal{A}$ we have*

$$|\Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e})] - \Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{y})]| \leq \mathsf{negl}(\lambda)$$

for $\mathbf{s} \leftarrow_{\$} \{0,1\}^k$, $\mathbf{e} \leftarrow_{\$} D_{\mathbb{Z}^n, \beta}$ and $\mathbf{y} \leftarrow_{\$} \{0,1\}^n$, where $D_{\mathbb{Z}^n, \beta}$ is some error distribution.

3.3 Laconic Private Set Intersection

Laconic Private Set Intersection. An ℓPSI is a two-round protocol that implements a PSI functionality and has special compactness properties.

Definition 11. *A ℓPSI scheme* $\mathsf{LPSI} = (\mathsf{GenCRS}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *is defined as follows:*

- $\mathsf{GenCRS}(1^\lambda)$*: Takes as input a security parameter* 1^λ*, and outputs a common reference string* crs*.*
- $\mathsf{R}_1(\mathsf{crs}, S_R)$*: Takes as input a* crs *and a set* S_R*. It outputs a first PSI message* psi_1 *and a state* st*.*
- $\mathsf{S}(\mathsf{crs}, S_S, \mathsf{psi}_1)$*: Takes as input a* crs*, a set* S_S *and a first PSI message* psi_1*. It outputs a second PSI message* psi_2*.*
- $\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$*: Takes as input a* crs*, a state* st *and a second message* psi_2*. It outputs a set* $\mathcal{I}$*.*

We require the following properties.

- ***Correctness****: The protocol satisfies PSI correctness in the standard sense.*
- ***Efficiency Requirements.*** *There exists a fixed polynomial* poly *such that the length of* psi_1 *and the running time of* S *are at most* $\mathsf{poly}(\lambda, \log |S_R|)$*.*

For malicious security, we work in the standard UC-framework [Can01] that allows us to prove security of protocols under arbitrary composition with other protocols. Let $\mathcal{F}$ be a functionality, π a protocol that implements $\mathcal{F}$ and $\mathcal{E}$ be a environment, an entity that oversees the execution of the protocol in both the real and the ideal worlds. Let $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$ be a random variable that represents the output of $\mathcal{E}$ after the execution of $\mathcal{F}$ with adversary Sim. Similarly, let $\mathsf{REAL}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{E}}$ be a random variable that represents the output of $\mathcal{E}$ after the execution of π with adversary $\mathcal{A}$ and with access to the functionality $\mathcal{G}$.

Definition 12. *A protocol* π UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model *if for every PPT adversary* $\mathcal{A}$ *there is a PPT simulator* Sim *such that for all PPT environments* $\mathcal{E}$*, the distributions* $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$ *and* $\mathsf{REAL}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{E}}$ *are computationally indistinguishable.*

We present the (reusable) PSI ideal functionality.

Reusable PSI functionality. The functionality $\mathcal{F}_{\mathsf{rPSI}}$ is parametrized by a universe $\mathcal{U}$ and works as follows:

- **Setup phase.** R sends $(\mathsf{sid}, S_\mathsf{R})$ to $\mathcal{F}_{\mathsf{rPSI}}$ where $S_\mathsf{R} \subseteq \mathcal{U}$. It ignores future messages from R with the same sid.
- **Send phase.** S sends $(\mathsf{sid}, i, S_\mathsf{S} \subseteq \mathcal{U})$ to $\mathcal{F}_{\mathsf{rPSI}}$. $\mathcal{F}_{\mathsf{rPSI}}$ sends $(\mathsf{sid}, i, S_\mathsf{R} \cap S_\mathsf{S})$ to R. It ignores future messages from S with the same sid and $i \in \mathbb{N}$.

4 Semi-Honest Laconic Private Set Intersection from CDH/LWE

In this section we show how to realize semi-honest ℓPSI from CDH/LWE. Our construction is non-black-box, making use of garbled circuits. This leads to the first feasibility result based on CDH, and an alternative LWE construction to that of [QWW18].

Our construction makes use of hash encryption schemes in conjunction with garbled circuits, which we review below.

Definition 13 (Hash Encryption [DG17b, BLSV18])**.** *A hash encryption scheme* $\mathsf{HE} = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{HEnc}, \mathsf{HDec})$ *is defined as follows.*

- $\mathsf{HGen}(1^\lambda, n)$*: Takes as input a security parameter* 1^λ *and an input size* n *and outputs a hash key* pp*.*
- $\mathsf{Hash}(\mathsf{pp}, z)$*: Takes as input a hash key* pp *and* $z \in \{0,1\}^n$*, and deterministically outputs* $h \in \{0,1\}^\lambda$*.*
- $\mathsf{HEnc}(\mathsf{pp}, h, \{m_{i,b}\}_{i\in[n],b\in\{0,1\}}; \{r_{i,b}\})$*: Takes as input a hash key* pp*, a hash output* h*, messages* $\{m_{i,b}\}$ *and randomness* $\{r_{i,b}\}$*, and outputs* $\{\mathsf{cth}_{i,b}\}_{i\in[n],b\in\{0,1\}}$*. We write it shortly as* $\{\mathsf{cth}_{i,b}\}$*. Overloading notation, each ciphertext* $\mathsf{cth}_{i,b}$ *is computed as* $\mathsf{cth}_{i,b} = \mathsf{HEnc}(\mathsf{pp}, h, m_{i,b}, (i,b); r_{i,b})$*.*

- $\mathsf{HDec}(z, \{\mathsf{cth}_{i,b}\})$: *Takes as input a hash input* z *and* $\{\mathsf{cth}_{i,b}\}$ *and outputs* n *messages* $(m_1, \ldots, m_n)$.

We require correctness meaning that for the variables above, $(m_1, \ldots, m_n) = (m_{1,z[1]}, \ldots, m_{n,z[n]})$. *We define two notions of security.*

- ***Semantic Security****: Given* $z \in \{0,1\}^n$, *no adversary can distinguish between encryptions of messages made to indices* $(i, \bar{z}_i)$. *For any PPT* $\mathcal{A}$, *sampling* $\mathsf{pp} \leftarrow_\$ \mathsf{HGen}(1^\lambda, n)$, *if* $(z, \{m_{i,b}\}, \{m'_{i,b}\}) \leftarrow_\$ \mathcal{A}(\mathsf{pp})$ *and if* $m_{i,z[i]} = m'_{i,z[i]}$ *for all* $i \in [n]$, *then* $\mathcal{A}$ *cannot distinguish between* $\mathsf{HEnc}(\mathsf{pp}, h, \{m_{i,b}\})$ *and* $\mathsf{HEnc}(\mathsf{pp}, y, \{m'_{i,b}\})$, *where* $h := \mathsf{Hash}(\mathsf{pp}, z)$.
- ***Anonymous Semantic Security****: For a random* $\{m_{i,b}\}$ *with equal rows (i.e.,* $m_{i,0} = m_{i,1}$*), the output of* $\mathsf{HEnc}(\mathsf{pp}, h, \{m_{i,b}\})$ *is pseudorandom even in the presence of the hash input. Formally, for any* $z \in \{0,1\}^n$, *sampling* $\mathsf{pp} \leftarrow_\$ \mathsf{HGen}(1^\lambda, n)$, $h := \mathsf{Hash}(\mathsf{pp}, z)$, *and sampling* $\{m_{i,b}\}$ *uniformly at random with the same rows, then* $\boldsymbol{v} := (\mathsf{pp}, z, \mathsf{HEnc}(\mathsf{pp}, h, \{m_{i,b}\}))$ *is indistinguishable from another tuple in which we replace the hash-encryption component of* $\boldsymbol{v}$ *with a random string.*

We have the following results from [BLSV18, GGH19].

Lemma 3. *Assuming CDH/LWE there exists anonymous hash encryption schemes, where* $n = 3\lambda$ *(i.e.,* $\mathsf{Hash}(\mathsf{pp}, \cdot)\colon \{0,1\}^{3\lambda} \mapsto \{0,1\}^\lambda$*).*[11] *Moreover, the hash function* Hash *satisfies robustness in the following sense: for any input distribution on* z *which samples at least* 2λ *bits of* z *uniformly at random,* $(\mathsf{pp}, \mathsf{Hash}(\mathsf{pp}, z))$ *and* (pp, u) *are statistically close, where* $\mathsf{pp} \leftarrow_\$ \mathsf{HGen}(1^\lambda, 3\lambda)$ *and* $u \leftarrow_\$ \{0,1\}^\lambda$.

We also review the notion of garbled circuits and the anonymous property, as defined in [BLSV18].

Definition 14 (Garbled Circuits)**.** *A garbling scheme for a class of circuits* $\{\mathsf{C}\colon \{0,1\}^n \mapsto \{0,1\}^m\}$ *consists of* $(\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$ *satisfying the following.*

- ***Correctness****: for all* $\mathsf{C} \in \mathcal{C}$, $\mathsf{m} \in \{0,1\}^n$, $\Pr[\mathsf{Eval}(\tilde{\mathsf{C}}, \{\mathsf{lb}_{i,\mathsf{m}[i]}\}) = \mathsf{C}(\mathsf{m})] = 1$, *where* $(\tilde{\mathsf{C}}, \{\mathsf{lb}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(1^\lambda, \mathsf{C})$.
- ***Simulation Security****: For any* $\mathsf{C} \in \mathcal{C}$ *and* $\mathsf{m} \in \{0,1\}^n$*:* $(\tilde{\mathsf{C}}, \{\mathsf{lb}_{i,\mathsf{m}[i]}\}) \stackrel{c}{\equiv} \mathsf{Sim}(1^\lambda, \mathsf{C}(\mathsf{m}))$, *where* $(\tilde{\mathsf{C}}, \{\mathsf{lb}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(1^\lambda, \mathsf{C})$.
- ***Anonymous Security*** *[BLSV18]: For any* $\mathsf{C} \in \mathcal{C}$, *choosing* $y \leftarrow_\$ \{0,1\}^m$, *the output of* $\mathsf{Sim}(1^\lambda, y)$ *is pseudorandom.*

Lemma 4 ([BLSV18])**.** *Anonymous garbled circuits can be built from one-way functions (OWFs).*

Notation on Hash Encryption. Throughout this section we assume $\mathsf{Hash}(\mathsf{pp}, \cdot)\colon \{0,1\}^n \mapsto \{0,1\}^\lambda$, where $n = 3\lambda$. We use $\{\mathsf{lb}_{i,b}\}$ to define a sequence of pairs of labels, where (throughout this section) $i \in [n]$ and $b \in \{0,1\}$. For $\boldsymbol{r} := \{r_{i,b}\}$ we let $\mathsf{HEnc}(\mathsf{pp}, h, \{\mathsf{lb}_{i,b}\}; \boldsymbol{r})$ denote the ciphertexts $\{\mathsf{cth}_{i,b}\}$, where $\mathsf{cth}_{i,b} = \mathsf{HEnc}(\mathsf{pp}, h, \mathsf{lb}_{i,b}, (i,b); r_{i,b})$. We further overload the notation as follows. We use $\{\mathsf{lb}_i\}$ to denote a sequence of 3λ elements. For $\boldsymbol{r} := \{r_{i,b}\}$ we let $\mathsf{HEnc}(\mathsf{pp}, h, \{\mathsf{lb}_i\}; \boldsymbol{r})$ denote a hash encryption where both plaintext rows are $\{\mathsf{lb}_i\}$; namely, the ciphertexts $\{\mathsf{cth}_{i,b}\}$, where $\mathsf{cth}_{i,b} = \mathsf{HEnc}(\mathsf{pp}, h, \{m_{i,b}\}; r_{i,b})$, where $m_{i,0} = m_{i,1} = \mathsf{lb}_i$, for all i.

Tree Terminology. Throughout this section we work with full binary trees. The depth of a tree is the length of a root-leaf path. We call the leaf level level 0, the level above it level one, and so on. We order the root-leaf paths from left to right; namely, the path from the root to the leftmost leaf node is the first root-leaf path, and the path from the root to the rightmost leaf node is the 2^dth root-leaf path, where d is the depth. Each node has an associated hash value, computed based on values associated to its children. Thus, when representing a root-leaf path, we include both children of each branching intermediate node.

[11] We note that the CDH construction of [BLSV18] satisfies a weaker notion of anonymity, in which only some part of the ciphertext is pseudorandom. But for ease of presentation we keep the notion as is, and remark that our ℓPSI construction works also with respect to that weaker notion.

Circuit $\mathsf{F}[\mathsf{id}, r, r'](\mathsf{id}', x, x')$**:**

- **Hardwired:** target identity id and randomness values r and r'.
- **Operation:** Return

$$y := \begin{cases} r & \mathsf{id} = \mathsf{id}' \\ r' & \text{else} \end{cases}$$

Circuit $\mathsf{V}[\mathsf{pp}, \mathsf{id}, \{\mathsf{lb}_{i,b}\}, \boldsymbol{r}](h_1, h_2, \mathsf{id}')$**:**

- **Hardwired:** Hash public parameter pp, target identity id, labels $\{\mathsf{lb}_{i,b}\}$, randomness $\boldsymbol{r}$.
- **Operation:** Return

$$\mathsf{ct} := \begin{cases} \mathsf{HEnc}(\mathsf{pp}, h_1, \{\mathsf{lb}_{i,b}\}; \boldsymbol{r}) & \mathsf{id} \le \mathsf{id}' \\ \mathsf{HEnc}(\mathsf{pp}, h_2, \{\mathsf{lb}_{i,b}\}; \boldsymbol{r}) & \text{else} \end{cases}$$

Procedure $\mathsf{DecPath}(\mathsf{pth}, \mathsf{psi}_2)$**:**

- **Input:** A leaf-root Path pth and ciphertext $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}^{(d)}\})$.
- **Operation:** Parse $\mathsf{pth} := (\underbrace{(\mathsf{id}, x, x')}_{z_0}, \underbrace{(h_0, h_0', \mathsf{id}_0)}_{z_1}, \ldots, \underbrace{(h_{d-1}, h_{d-1}', \mathsf{id}_{d-1})}_{z_d}, \mathsf{hr}_{\mathsf{root}})$. For $w \in \{d, \ldots, 1\}$:
  1. Let $\{\mathsf{lb}_i^{(w)}\} := \mathsf{HDec}(\mathsf{z}_w, \{\mathsf{cth}_{i,b}^{(w)}\})$.
  2. Set $\{\mathsf{cth}_{i,b}^{(w-1)}\} := \mathsf{Eval}(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\})$.

  Let $\{\mathsf{lb}_i^{(0)}\} := \mathsf{HDec}(z_0, \{\mathsf{cth}_{i,b}^{(0)}\})$. Return $\mathsf{Eval}(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\})$.

Table 1: Circuits F, V and procedure $\mathsf{DecPath}$

Sender's Set Size is One. We assume without loss of generality that the sender holds a single element. For the general case where the sender may have multiple elements, we reuse the first message of the receiver for each element in the sender's set. The overall running time of the sender will only scale with its own set size, and not with the receiver's set size.

Construction 1 (ℓPSI Construction)**.** *We require the following ingredients in our ℓPSI Construction.*

1. *A hash encryption scheme* $\mathsf{HE} = (\mathsf{HGen}, \mathsf{Hash}, \mathsf{HEnc}, \mathsf{HDec})$*, where* $\mathsf{Hash}(\mathsf{pp}, \cdot)\colon \{0,1\}^{3\lambda} \mapsto \{0,1\}^{\lambda}$.
2. *A garbling scheme* $\mathsf{GS} = (\mathsf{Garb}, \mathsf{Eval}, \mathsf{Sim})$.
3. *Circuits* F *and* V*, as well as procedure* $\mathsf{DecPath}$*, defined in Table 1.*

We assume the elements of the receiver and the sender are strings in $\{0,1\}^{\lambda}$*. We refer to each element as an identity. Build* $(\mathsf{GenCRS}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *as follows.*

$\mathsf{GenCRS}(1^{\lambda})$**:** *Return* $\mathsf{crs} \leftarrow_{\$} \mathsf{HGen}(1^{\lambda}, 3\lambda)$.

$\mathsf{R}_1(\mathsf{crs}, S_{\mathsf{R}})$**:** *Assume* $|S_{\mathsf{R}}| = 2^d$*. (With small tweaks the same construction works if* S_{R} *is not a power of two.)*

- *Parse* $\mathsf{crs} := \mathsf{pp}$*. Let* $n := 2^d$*, and sort* $S_{\mathsf{R}} := \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$*, where* $\mathsf{id}_i < \mathsf{id}_{i+1}$ *for all* i*. Populate the leaf node values as follows. For each* $\mathsf{id}_i \in S_{\mathsf{R}}$*, sample* $x_i, x_i' \leftarrow_{\$} \{0,1\}^{\lambda}$*, and let* $h_i^{(0)} := \mathsf{Hash}(\mathsf{pp}, \mathsf{id}_i, x_i, x_i')$*. Set* $\mathsf{H}[v_i^{(0)}] := h_i^{(0)}$ *and* $\mathsf{ID}[v_i^{(0)}] := \mathsf{id}_i$*.*
  1. *For* $w \in [d]$*, populate the values for the nodes at level* w *as follows. Informally, the hash value for each node is the hash of the concatenation of its left child, right child, and the largest identity*

value under its left child. Formally, noting we have 2^{d-w} nodes on level w, for $j \in [2^{d-w}]$, set $h_j^{(w)} := \mathsf{Hash}(\mathsf{pp}, (h_{2j-1}^{(w-1)}, h_{2j}^{(w-1)}, \mathsf{id}_{[j,w]}))$, where $\mathsf{id}_{[j,w]}$ denotes the larges leaf identity under the left child of the current node (i.e., $\mathsf{id}_{[j,w]} = \mathsf{id}_f$, where $f := (2j-1)2^{w-1}$.) Set $\mathsf{H}[v_j^{(w)}] = h_j^{(w)}$ and $\mathsf{ID}[v_j^{(w)}] = \mathsf{id}_{[j,w]}$.

2. *Set $\mathsf{psi}_1 := (d, \mathsf{hr}_{\mathsf{root}})$, where $\mathsf{hr}_{\mathsf{root}} := h_1^{(d)}$ (i.e., the root hash value). Set $\mathsf{st} := (S_{\mathsf{R}}, \{x_i\}, \{x'_i\}, \{v_j^{(w)}\})$ for all values of $i \in [n]$, $w \in \{0, \ldots, d\}$ and $j \in [2^{d-w}]$.*

$\mathsf{S}(\mathsf{crs}, \mathsf{id}, \mathsf{psi}_1)$**:**

- *Parse $\mathsf{psi}_1 := (d, \mathsf{hr}_{\mathsf{root}})$ and $\mathsf{crs} := \mathsf{pp}$. Sample $r, r' \leftarrow_{\$} \{0,1\}^\lambda$ and let $\mathsf{C}_0 := \mathsf{F}[\mathsf{id}, r, r']$ (Table 1). Garble $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}^{(0)}\}) \leftarrow_{\$} \mathsf{Garb}(\mathsf{C}_0)$. For $1 \le w \le d$*

  1. *Sample $\boldsymbol{r}_w$ at random, and let $\mathsf{C}_w := \mathsf{V}[\mathsf{pp}, \mathsf{id}, \{\mathsf{lb}_{i,b}^{(w-1)}\}, \boldsymbol{r}_w]$.*
  2. *Garble $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_{i,b}^{(w)}\}) \leftarrow_{\$} \mathsf{Garb}(\mathsf{C}_w)$.*

- *Let $\{\mathsf{cth}_{i,b}\} \leftarrow_{\$} \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_{i,b}^{(d)}\})$. Return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$.*

$\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$**:**

- *Parse $\mathsf{st} := (S_{\mathsf{R}}, \{x_i\}, \{x'_i\}, \{v_j^{(w)}\})$, $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$ and $S_{\mathsf{R}} := \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$. For $i \in [n]$ let $\mathsf{pth}_i := ((\mathsf{id}_i, x_i, x'_i), \ldots, \mathsf{hr}_{\mathsf{root}})$ be the i'th leaf-root path in the tree, and let*

$$r_i := \mathsf{DecPath}(\mathsf{pth}_i, \widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}).$$

If for a unique index $i \in [n]$, $r_i = r$, then output id_i. Otherwise, output $\perp$.

Theorem 1. *Assuming the hash encryption HE is anonymous and robust (robustness defined in Lemma 3), and that the garbling scheme GS is anonymous, the ℓPSI protocol of Construction 1 provides statistical security for the receiver and semi-honest security for the sender. As a result, such ℓPSI protocols can be realized from CDH/LWE.*

Roadmap for the Proof of Theorem 1. The fact that the protocol provides statistical security for the receiver follows from the robustness of HE. In particular, robustness implies that $h_i^{(0)}$ values statistically hide S_{R}. We can continue this to argue that all the first-level hash values (i.e., $h_i^{(1)}$) also hide S_{R}, and hence, continuing like this, the root hash value $\mathsf{hr}_{\mathsf{root}}$ statistically hides S_{R}.

We now prove that the protocol provides sender security against semi-honest receivers. Let id be the sender's input message, and $S_{\mathsf{R}} := \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ be the receiver's set, where $\mathsf{id}_i < \mathsf{id}_{i+1}$. Assuming $\mathsf{id} \notin S_{\mathsf{R}}$ we will show that the sender's protocol message is pseudorandom in the receiver's point of view. For simplicity suppose $\mathsf{id} < \mathsf{id}_1$; the general case follows via simple changes, which we will illustrate in Remark 1. Let

$$\mathsf{pth} := (\underbrace{(\mathsf{id}_1, x_1, x'_1)}_{z_0}, \underbrace{(h_0, h'_0, \mathsf{id}_0)}_{z_1}, \ldots, \underbrace{(h_{d-1}, h'_{d-1}, \mathsf{id}_{d-1})}_{z_d}, \mathsf{hr}_{\mathsf{root}}) \tag{1}$$

be the leaf-root path from leaf id_1 to the root. Note that $\mathsf{hr}_{\mathsf{root}} = \mathsf{Hash}(\mathsf{pp}, z_d)$, and $h_i = \mathsf{Hash}(\mathsf{pp}, z_i)$ for all $i \in \{0, \ldots, d-1\}$. Noting that $\mathsf{hr}_{\mathsf{root}}$ is the receiver's first-round message, we define the following hybrids for the sender's response message.

Hyb$_0$: The sender's response message psi_2 is formed as in the protocol.

$\mathbf{Hyb}_1$: Sample $r, r' \leftarrow_\$ \{0,1\}^\lambda$. Let $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{F}, r')$. For $1 \le w \le d$

1. Sample $\{\mathsf{cth}_{i,b}^{(w-1)}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{pp}, h_{w-1}, \{\mathsf{lb}_i^{(w-1)}\})$.
2. Let $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}_{i,b}^{(w-1)}\})$.

Let $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\})$. Return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$.

Lemma 5. *Hybrids* $\mathbf{Hyb}_0$ *and* $\mathbf{Hyb}_1$ *are indistinguishable.*

$\mathbf{Hyb}_2$: Sample psi_2 at random.

Lemma 6. *Hybrids* $\mathbf{Hyb}_1$ *and* $\mathbf{Hyb}_2$ *are indistinguishable.*

The above two lemmas establish sender's security; namely — if $\mathsf{id} \notin S_{\mathsf{R}}$, then the sender's message psi_2 is pseudorandom for the receiver. We prove Lemma 5 in Section 4.1 and Lemma 6 in Section 4.2.

4.1 Proof of Lemma 5

In the following, given two hybrids $\mathbf{Hyb}$ and $\mathbf{Hyb}'$, we use the notation $\mathbf{Hyb} \stackrel{c}{\equiv} \mathbf{Hyb}'$ to express that the hybrids are computationally indistinguishable.

We define $d+1$ hybrids between $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$, and prove their indistinguishability.

For $p \in \{0, \ldots, d\}$ we define $\mathbf{Hyb}'_p$ as follows. Under $\mathbf{Hyb}'_p$, we form the first $p+1$ garbled circuits $\widetilde{C}_0, \ldots, \widetilde{C}_p$ and their corresponding labels honestly as in the real game, and we simulate the rest.

$\mathbf{Hyb}'_p$: Let pth be as in Equation 1, and recall that we are assuming $\mathsf{id} < \mathsf{id}_1$. Sample $r, r' \leftarrow_\$ \{0,1\}^\lambda$ and let $\mathsf{C}_0 := \mathsf{F}[\mathsf{id}, r, r']$. Garble $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_{i,b}^{(0)}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_0)$. Let $\{\mathsf{lb}_i^{(0)}\} := \{\mathsf{lb}_{i,z_0[i]}\}$. Do the following:

- For $1 \le w \le p$
  1. Sample $\boldsymbol{r}_w$ at random, and let $\mathsf{C}_w := \mathsf{V}[\mathsf{pp}, \mathsf{id}, \{\mathsf{lb}_{i,b}^{(w-1)}\}, \boldsymbol{r}_w]$.
  2. Garble $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_{i,b}^{(w)}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_w)$.
  3. If $w = p$ (i.e., last step), let $\{\mathsf{lb}_i^{(w)}\} := \{\mathsf{lb}_{i,z_w[i]}^{(w)}\}$.
- For $p+1 \le w \le d$
  1. Sample $\{\mathsf{cth}_{i,b}^{(w-1)}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{pp}, h_{w-1}, \{\mathsf{lb}_i^{(w-1)}\})$.
  2. Let $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}_{i,b}^{(w-1)}\})$.

Let $\{\mathsf{cth}_{i,b}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\})$. Return $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}\}, r)$.

Lemma 7. $\mathbf{Hyb}_0 \stackrel{c}{\equiv} \mathbf{Hyb}'_d$ *and* $\mathbf{Hyb}_1 \stackrel{c}{\equiv} \mathbf{Hyb}'_0$.

Proof. We first show $\mathbf{Hyb}_1 \stackrel{c}{\equiv} \mathbf{Hyb}'_0$. Notice that either hybrid may be simulated just by knowing the value of r and the pair $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\})$. We let $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\})$ and $(\widetilde{\mathsf{C}}', \{\mathsf{lb}'_i\})$ denote the distribution of this pair in $\mathbf{Hyb}_1$ and $\mathbf{Hyb}'_0$, respectively. We have $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{F}, r')$. As for the other pair, letting $\mathsf{C}_0 := \mathsf{F}[\mathsf{id}, r, r']$ for random r, r'

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}_{i,b}\}) \leftarrow_\$ \mathsf{Garb}(\mathsf{C}_0) \tag{2}$$

$$\{\mathsf{lb}'_i\} = \{\mathsf{lb}_{i,z_0[i]}\}, \tag{3}$$

where $z_0 = (\mathsf{id}_1, x_1, x_1')$. By simulation security of garbled circuits

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}_i'\}) \stackrel{c}{\equiv} \mathsf{Sim}(\mathsf{F}, \mathsf{C}_0(z_0)) \tag{4}$$

$$\stackrel{c}{\equiv} \mathsf{Sim}(\mathsf{F}, r'). \tag{5}$$

Thus, $(r, \widetilde{\mathsf{C}}, \{\mathsf{lb}_i\})\mathsf{com}(r, \widetilde{\mathsf{C}}', \{\mathsf{lb}_i'\})$, proving $\mathbf{Hyb}_1 \stackrel{c}{\equiv} \mathbf{Hyb}_0'$.

To prove $\mathbf{Hyb}_0 \stackrel{c}{\equiv} \mathbf{Hyb}_d'$, their only difference lies in how $\{\mathsf{cth}_{i,b}\}$ is sampled: under $\mathbf{Hyb}_0$: $\{\mathsf{cth}_{i,b}\} \leftarrow_{\$} \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_{i,b}^{(d)}\})$, while under $\mathbf{Hyb}_d'$: $\{\mathsf{cth}_{i,b}\} \leftarrow_{\$} \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\})$, where recall that $\{\mathsf{lb}_i^{(d)}\} := \{\mathsf{lb}_{i,z_d[i]}^{(d)}\}$. Since $\mathsf{hr}_{\mathsf{root}} = \mathsf{Hash}(\mathsf{pp}, z_d)$, by security of the hash encryption $\mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\}) \stackrel{c}{\equiv} \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_{i,b}^{(d)}\})$ and the proof is now complete. □

Lemma 8. *For all $p \in \{0, \ldots, d-1\}$, $\mathbf{Hyb}_p' \stackrel{c}{\equiv} \mathbf{Hyb}_{p+1}'$.*

Proof. We will show that the distribution of $(\widetilde{C}_0, \ldots, \widetilde{C}_{p+1}, \{\mathsf{lb}_i^{(p+1)}\})$ is computationally indistinguishable in the two worlds. This will imply the result because the rest of either hybrid may be formed based solely on the above tuple. To argue the above tuple is indistinguishable across the two hybrids, first notice that the distribution of

$$(\widetilde{C}_0, \{\mathsf{lb}_{i,b}^{(0)}\}, \ldots, \widetilde{C}_p, \{\mathsf{lb}_{i,b}^{(p)}\})$$

is formed exactly the same in $\mathbf{Hyb}_p'$ and $\mathbf{Hyb}_{p+1}'$. The only difference between these two hybrids lies in the way in which the pair $(\widetilde{C}_{p+1}, \{\mathsf{lb}_i^{(p+1)}\})$ is sampled. To ease notation, we let $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\})$ and $(\widetilde{\mathsf{C}}', \{\mathsf{lb}_i'\})$ denote the distribution of this pair in $\mathbf{Hyb}_p'$ and $\mathbf{Hyb}_{p+1}'$, respectively. Formally

1. **Under $\mathbf{Hyb}_p'$**: We form

$$\{\mathsf{cth}\} \leftarrow_{\$} \mathsf{HEnc}(\mathsf{pp}, h_p, \{\mathsf{lb}_i^{(p)}\}) \tag{6}$$

$$(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\}) \leftarrow_{\$} \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}\}). \tag{7}$$

2. **Under $\mathbf{Hyb}_{p+1}'$**: We form

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}_{i,b}\}) \leftarrow_{\$} \mathsf{Garb}(\mathsf{C}_{p+1})$$
$$\{\mathsf{lb}_i'\} := \{\mathsf{lb}_{i,z_{p+1}[i]}\},$$

where $\mathsf{C}_{p+1} := \mathsf{V}[\mathsf{pp}, \mathsf{id}, \{\mathsf{lb}_{i,b}^{(p)}\}, \boldsymbol{r}_{p+1}]$ and $z_{p+1} = (h_p, h_p', \mathsf{id}_p)$.

By simulation security of garbled circuits

$$(\widetilde{\mathsf{C}}', \{\mathsf{lb}_i'\}) \stackrel{c}{\equiv} \mathsf{Sim}(\mathsf{V}, \mathsf{C}_{p+1}(z_{p+1})) \tag{8}$$

$$\stackrel{c}{\equiv} \mathsf{Sim}(\mathsf{V}, \mathsf{HEnc}(\mathsf{pp}, h_p, \{\mathsf{lb}_{i,b}^{(p)}\}; \boldsymbol{r}_{p+1})). \tag{9}$$

Notice that in Equation 9 we use the fact $\mathsf{id} < \mathsf{id}_p$, and so by definition of C_{p+1}, its hardwired labels $\{\mathsf{lb}_{i,b}^{(p)}\}$ will be encrypted under h_p.[12] Now, Equation 9 is identical to the right-hand side of Equation 7, and thus $(\widetilde{\mathsf{C}}, \{\mathsf{lb}_i\}) \stackrel{c}{\equiv} (\widetilde{\mathsf{C}}', \{\mathsf{lb}_i'\})$. The proof is now complete. □

[12] This is the place where we use the fact that id is less than all values in S_{R}. In the general case we should change the above distributions accordingly.

4.2 Proof of Lemma 6

We need to show that $\mathsf{psi}_2 := (\widetilde{\mathsf{C}}_0, \ldots, \widetilde{\mathsf{C}}_d, \{\mathsf{cth}_{i,b}^{(d)}\}, r)$ is pseudorandom, where everything is sampled as in $\mathbf{Hyb}_1$. Since $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{F}, r')$ by simulation security of the garbled circuit and Lemma 4 the distribution of $(\widetilde{\mathsf{C}}_0, \{\mathsf{lb}_i^{(0)}\})$ is pseudorandom. Recall that for $1 \leq w \leq d$ we have $\{\mathsf{cth}_{i,b}^{(w-1)}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{pp}, h_w, \{\mathsf{lb}_i^{(w-1)}\})$ and $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\}) \leftarrow_\$ \mathsf{Sim}(\mathsf{V}, \{\mathsf{cth}_{i,b}^{(w-1)}\})$. By Lemma 3 $\{\mathsf{cth}_{i,b}^{(w-1)}\}$ is pseudorandom, and thus by Lemma 4 $(\widetilde{\mathsf{C}}_w, \{\mathsf{lb}_i^{(w)}\}$ is also pseudorandom, for all $0 \leq w \leq d-1$. Finally, since we have $\{\mathsf{cth}_{i,b}^{(d)}\} \leftarrow_\$ \mathsf{HEnc}(\mathsf{pp}, \mathsf{hr}_{\mathsf{root}}, \{\mathsf{lb}_i^{(d)}\})$, by Lemma 3, $\{\mathsf{cth}_{i,b}^{(d)}\}$ is pseudorandom. The proof is now complete. □

Remark 1. *In the proof of security, we assumed* $\mathsf{id} < \mathsf{id}_1$. *For the general case, we just need to change the active path from that in Equation 1 ending in* id_1 *to the path that will end in* id_j, *where* j *is the largest index such that* id *lies between* id_j *and* id_{j+1}. *In case* $\mathsf{id} > \mathsf{id}_n$, *then* $j = n$.

5 Reusable DV-NIZK Range Proofs for DJ Ciphertexts

In this section, we construct a DV-NIZK scheme for ranges of DJ ciphertexts. The main idea of our construction is the following: the prover proves that a BGN ciphertext [BGN05] is within a certain range (this can be done via the protocol of [GOS06]). Then it proves that the DJ and BGN ciphertexts encrypt the same value.

We first recall the required cryptosystems used in this section.

BGN cryptosystem. Recall that the BGN cryptosystem [BGN05] is defined over a group $\mathbb{G}$ of order $n = pq$ for primes p, q. The public key is composed by $(\mathbb{G}, n, G, H)$ where G is a generator of $\mathbb{G}$ and H is an element of order p (let $p\mathbb{G}$ be the subgroup of order p). The public key is composed by $(\mathbb{G}, n, G, H)$ and a ciphertext for a message $m \in \{0, 1\}$ is of the form $C = G^m H^t$ for $t \leftarrow_\$ \mathbb{Z}_n$.

Damgård-Jurik cryptosystem. The Damgård-Jurik (DJ) cryptosystem [DJ01] is defined over $\mathbb{Z}_{N^{\xi+1}}$ where $N \leftarrow_\$ \mathsf{RSA}(\lambda)$. The public key is formed by (N, ξ, g, h) where $g, h \leftarrow_\$ \mathbb{NR}_N$ and $h = g^x$ for $x \leftarrow_\$ \mathbb{Z}_N^*$. A ciphertext has the form (c_1, c_2) where $c_1 = g^t \bmod N^{\xi+1}$ and $c_2 = h^t(1+N)^m \bmod N^{\xi+1}$ for $t \leftarrow_\$ \mathbb{Z}_N$ and $m \in \mathbb{Z}_N$.

5.1 DV-NIZK Schemes for Linear Languages and for BGN Ciphertexts

Let $\mathbb{F}$ be any group and $\mathbf{M} = (m_{i,j})_{i\in[n],j\in[m]}$ be a matrix. For $g \in \mathbb{F}$, we define $g^{\mathbf{M}} = (g^{m_{i,j}})_{i\in[n],j\in[m]}$ A linear language is of the form

$$\mathcal{L} = \{g^{\mathbf{y}} \in \mathbb{F}^m : \exists \mathbf{x} \in \mathbb{F}^n, \mathbf{y} = \mathbf{x}\mathbf{M}\}.$$

The framework of [CS02] (and subseequent works [KW15]) can be instantiated to build a black-box DV-NIZK for any language of this form. The resulting scheme achieves statistical simulation soundness and perfect zero-knowledge.

We now review some specific languages which are of the form described above and for which we can obtain efficient black-box DV-NIZK schemes.

In the following, let $N, n \leftarrow_\$ \mathsf{RSA}(\lambda)$, $\xi \in \mathbb{N}$ and $\mathbb{G}$ be a group of order n (that is, a BGN group).

DV-NIZK for discrete log. First, consider the following language for discrete logs, which is parametrized by $(\mathbb{G}, n, H, N, \xi, h)$

$$\mathcal{DL}_\Delta = \left\{ (H', h) \in \mathbb{G} \times \mathbb{Z}_{N^{\xi+1}} : \exists (X, x) \in \mathbb{Z}_n \times \mathbb{Z}_N \text{ s.t. } \begin{matrix} H' = H^X \\ h' = h^x \bmod N^{\xi+1} \end{matrix} \right\}$$

where $\Delta = (\mathbb{G}, n, H, N, \xi, h)$, $H \in \mathbb{G}$ and $h \in \mathbb{Z}_{N^{\xi+1}}$. The language allows to prove that H' is in the subgroup generated by H and h' is in the same subgroup as h in their respective groups. In [CS02], a reusable DV-NIZK for discrete log languages with statistical soundness is presented.

Lemma 9 ([CS02]). *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{DL}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{DL}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{DL}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{DL}_\Delta})$$

for language $\mathcal{DL}_\Delta$ where $\Delta = (\mathbb{G}, n, H, N, \xi, h)$. The scheme fulfills statistical simulation soundness and perfect zero-knowledge.

DV-NIZK for DJ ciphertexts. First, consider the following language for DJ ciphertexts, which is parametrized by $(\{g_i, h_i\}_i, N, \xi)$

$$\mathcal{DJ}_\Delta = \left\{ \{c_{1,i}, c_{2,i}\}_i \in \mathbb{Z}^{2\ell}_{N^{\xi+1}} : \exists (t, \{m_i\}_i) \in \mathbb{Z}^{\ell+1}_{N^\xi} \text{ s.t. } \begin{array}{c} c_{1,i} = g_i^t \bmod N^{\xi+1} \\ c_{2,i} = h_i^t (1+N)^{m_i} \bmod N^{\xi+1} \end{array} \right\}$$

for $i \in [\ell]$ where $\Delta = (\{g_i, h_i\}_i, N, \xi)$ and $g_i, h_i \in \mathbb{Z}_{N^{\xi+1}}$. The language allows to prove that h is in the same subgroup as g. In [CS02], a reusable DV-NIZK for discrete log languages with statistical simulation soundness is presented.

Lemma 10 ([CS02]). *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{DJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{DJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{DJ}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{DJ}_\Delta})$$

for language $\mathcal{DJ}_\Delta$ where $\Delta = (\{g_i, h_i\}_i, N, \xi)$. The scheme fulfills statistical simulation soundness and perfect zero-knowledge.

DV-NIZK for equality of discrete log. Consider also the following language for the equality of discrete logs.

$$\mathcal{EDL}_\Delta = \left\{ (h_0, h_1) \in \mathbb{Z}_{N^{\xi+1}} : \exists t \in \mathbb{Z}_{N^\xi} \text{ s.t. } \begin{array}{c} g_0^t = h_0 \bmod N^{\xi+1} \\ g_1^t = h_1 \bmod N^{\xi+1} \end{array} \right\}$$

parametrized by $\Delta = (g_0, g_1, N, \xi)$ where $g_0, g_1 \in \mathbb{Z}_{N^{\xi+1}}$. for the equality of discrete logs. Again, the framework of [CS02] can be adapted to obtain an efficient reusable DV-NIZK for this language with statistical simulation soundness.

Lemma 11 ([CS02]). *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{EDL}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{EDL}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{EDL}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{EDL}_\Delta})$$

where $\Delta = (g_0, g_1, N, \xi)$. The scheme fulfills statistical simulation soundness and perfect zero-knowledge.

DV-NIZK for equality of plaintexts. Consider the language for the equality of plaintexts in two different DJ ciphertexts

$$\mathcal{EPDJ}_\Delta = \left\{ \{c_i, d_i\}_i \in \mathbb{Z}^4_{N^{\xi+1}} : \exists (\{t_i\}_i, m) \in \mathbb{Z}^3_{N^\xi} \text{ s.t. } \begin{array}{c} c_1 = g_1^{t_1} \bmod N^{\xi+1} \\ c_2 = h_1^{t_1} (1+N)^m \bmod N^{\xi+1} \\ d_1 = g_2^{t_2} \bmod N^{\xi+1} \\ d_2 = h_2^{t_2} (1+N)^m \bmod N^{\xi+1} \end{array} \right\}$$

for $i = 1, 2$, parametrized by $\Delta = ((g_1, h_1), (g_2, h_2), N, \xi)$ where $(g_1, h_1), (g_2, h_2) \in \mathbb{Z}_{N^{\xi+1}}$ for the equality of plaintexts.

Lemma 12 ([CS02]). *There exists a reusable DV-NIZK*

$$\mathsf{NIZK}_{\mathcal{EPDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{EPDJ}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{EPDJ}_\Delta})$$

where $\Delta = (g, h, N, \xi)$. The scheme fulfills statistical simulation soundness and perfect zero-knowledge.

Range Proofs for BGN Finally, we consider the language of well-formed BGN ciphertexts encrypting a bit.

$$\mathcal{BGN}_\Delta = \left\{ \{C_i\}_i \in \mathbb{G}^\ell : \exists (t, \{m_i\}) \in \mathbb{Z}_n^{\ell+1} \text{ s.t. } \begin{array}{c} m_i \in \{0,1\} \\ C_i = G^{m_i} H_i^t \end{array} \right\}$$

for $i \in [\ell]$, where $\Delta = (\mathbb{G}, n, G, \{H_i\}_{i\in[\ell]})$ and $G, \{H_i\}_{i\in[\ell]} \in \mathbb{G}$.

This is not a linear language and, thus, it cannot be instantiated using the framework of [CS02]. Luckily, the work of [GOS06] presents an efficient scheme for this language.

Lemma 13 ([GOS06]). *There exists a reusable DV-NIZK scheme*[13]

$$\mathsf{NIZK}_{\mathcal{BGN}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{BGN}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{BGN}_\Delta}, \mathsf{NIZK.Verify}_{\mathcal{BGN}_\Delta})$$

for the language $\mathcal{BGN}$. The protocol has perfect simulation soundness and computational zero-knowledge under the SD assumption.

5.2 Equality of Plaintexts in DJ and BGN ciphertexts.

We now show how to prove that a BGN and a DJ ciphertexts encrypt the same value. Consider the following language

$$\mathcal{EQ}_\Delta = \left\{ D_0, h_0, \{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]} : \exists (r, t, \{m_i\}) \text{ s.t. } \begin{array}{c} m_i \in \{0,1\} \\ D_0 = H_0^r \in \mathbb{G} \\ D_i = G^m H_i^r \in \mathbb{G} \\ c_0 = h_0^t \in \mathbb{Z}_{N^{\xi+1}} \\ c_1 = g^t \in \mathbb{Z}_{N^{\xi+1}} \\ c_{2,i} = h_i^t (1+N)^{m_i} \in \mathbb{Z}_{N^{\xi+1}} \end{array} \right\}$$

where $\Delta = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_{i\in[\ell]})$, $G, H_0, \{H_i\}_{i\in[\ell]} \in \mathbb{G}$ and $g, h_0, \{h_i\}_{i\in[\ell]} \in \mathbb{Z}_{N^{\xi+1}}$.

Construction 2. *Let $\ell \in \mathbb{Z}$. Let $\Delta = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_{i\in[\ell]})$ be as above, such that $n > N^{\xi+1}$. Let $\beta \in \mathbb{N}$ such that $\lambda/\beta = \mathsf{negl}(\lambda)$, and $N^\xi/2 > \ell\beta$. We require the following ingredients:*

1. *The scheme of Lemma 13, $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}} = (\mathsf{NIZK.GenCRS}_{\mathcal{BGN}_{\Delta_1}}, \mathsf{NIZK.Prove}_{\mathcal{BGN}_{\Delta_1}}, \mathsf{NIZK.Verify}_{\mathcal{BGN}_{\Delta_1}})$ for some $\Delta_1 = (\mathbb{G}, n, G, \{H_i\}_{i\in[\ell]})$.*
2. *The scheme of Lemma 10, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}} = (\mathsf{NIZK.GenCRS}_{\mathcal{DJ}_{\Delta_2}}, \mathsf{NIZK.Prove}_{\mathcal{DJ}_{\Delta_2}}, \mathsf{NIZK.Verify}_{\mathcal{DJ}_{\Delta_2}})$ for some $\Delta_2 = (\{g, h_i\}_i, N, \xi)$.*
3. *The scheme of Lemma 10, $\mathsf{NIZK}_{\mathcal{DL}_{\Delta_3}} = (\mathsf{NIZK.GenCRS}_{\mathcal{DL}_{\Delta_3}}, \mathsf{NIZK.Prove}_{\mathcal{DL}_{\Delta_3}}, \mathsf{NIZK.Verify}_{\mathcal{DL}_{\Delta_3}})$ for some $\Delta_3 = (\mathbb{G}, n, H, N, \xi, h)$.*

We present the scheme in full detail.

$\mathsf{GenCRS}_{\mathcal{EQ}_\Delta}(1^\lambda)$:

- *Compute* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{BGN}_{\Delta_1}}(1^\lambda)$ *where* $\Delta_1 = (\mathbb{G}, \{G, H_i\}_{i\in[\ell]})$.
- *Compute* $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{DJ}_{\Delta_2}}(1^\lambda)$ *where* $\Delta_2 = (\{g, h_i\}_{i\in[\ell]}, N, \xi)$.
- *Compute* $(\mathsf{crs}_3, \mathsf{td}_3) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{DL}_{\Delta_3}}(1^\lambda)$ *where* $\Delta_3 = (\mathbb{G}, n, H_0, N, \xi, h_0)$.
- *For all $j \in [\lambda]$, do the following:*
  - *Sample $\alpha_j \leftarrow_\$ \mathbb{Z}_N$ and $A_j \leftarrow_\$ \mathbb{Z}_n$. For all $i \in [\ell]$, sample $\sigma_{i,j} \leftarrow_\$ \{0,1\}$. Compute $f_j = h_0^{\alpha_j} \prod_{i=1}^\ell h_i^{\sigma_{i,j}} \bmod N^{\xi+1}$ and $F_j = H_0^{A_j} \prod_{i=1}^\ell H_i^{\sigma_{i,j}}$*
- *Output* $\mathsf{crs} = \left(\{F_j, f_j\}_{j\in[\lambda]}, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{crs}_3\right)$ *and* $\mathsf{td} = \left(\{\alpha_j, A_j, \{\sigma_{i,j}\}_{i\in[\ell]}\}_{j\in[\lambda]}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3\right)$

[13] The scheme presented in [GOS06] is a NIZK scheme and not a DV-NIZK. However, we can view a NIZK as a DV-NIZK where $\mathsf{td} = \perp$.

$\mathsf{Prove}_{\mathcal{EQ}_\Delta}(\mathsf{crs}, x = (D_0, h_0\{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]}), w = (r, t, \{m_{i\in[\ell]}\})) :$

- *Parse* crs *as* $\big(\{F_j, f_j\}_{j\in[\lambda]}, \mathsf{crs}_1, \mathsf{crs}_2, \mathsf{crs}_3\big)$.
- *Compute* $\pi_1 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{BGN}_{\Delta_1}}(\mathsf{crs}_1, x_1, w_1)$ *where* $x_1 = \{D_i\}_{i\in[\ell]}$ *and* $w_1 = (r, \{m_i\}_{i\in[\ell]})$.
- *Compute* $\pi_2 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{DJ}_{\Delta_2}}(\mathsf{crs}_2, x_2, w_2)$ *where* $x_2 = \{c_{1,i}, c_{2,i}\}_{i\in[\ell]}$ *and* $w_2 = (t, \{m_i\}_{i\in[\ell]})$.
- *Compute* $\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{DL}_{\Delta_3}}(\mathsf{crs}_3, x_3, w_3)$ *where* $x_3 = (D_0, c_0)$ *and* $w_3 = (r, t)$.
- *For all* $j \in [\lambda]$*, do the following: Sample* $\tau_j \leftarrow\!\!\$\ \Phi_{\mathbb{Z},\beta}$ *and compute* $a_j = {f_j}^t(1+N)^{\tau_j} \bmod N^{\xi+1}$. *Compute* $K_j = G^{\tau_j} F_j^r$.
- *Output* $\pi = \big(\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3\big)$.

$\mathsf{Verify}_{\mathcal{EQ}_\Delta}(\mathsf{td}, x, \pi) :$

- *Parse* π *as* $\big(\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3\big)$ *and* td *as* $\big(\{\alpha_j, A_j, \{\sigma_{i,j}\}_{i\in[\ell]}\}_{j\in[\lambda]}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3\big)$
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{BGN}_{\Delta_1}}(\mathsf{td}_1, x_1, \pi_1)$ *where* $x_1 = \{D_i\}_{i\in[\ell]}$, *output* 0.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{DJ}_{\Delta_2}}(\mathsf{td}_2, x_2, \pi_2)$ *where* $x_2 = \{c_{1,i}, c_{2.i}\}_{i\in[\ell]}$, *output* 0.
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{DL}_{\Delta_3}}(\mathsf{td}_3, x_3, \pi_3)$ *where* $x_3 = (D_0, c_0)$, *output* 0.
- *For all* $j \in [\lambda]$*, do the following:*
  - *For all* $i \in [\ell]$*, compute* $z_j = a_j^{-1} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} \bmod N^{\xi+1}$. *If there is a* z_j *which is not of the form* $(1+N)^{y_j}$*, output* 0. *Else, recover* y_j.
  - *Compute* $L_j = K_j^{-1} D_0^{A_j} \prod_i D_i^{\sigma_{i,j}}$ *in* $\mathbb{G}$.
- *If there is a* $j \in [\lambda]$ *such that* $G^{y_j} \neq L_j$ *in* $\mathbb{G}$, *output* 0. *Else, output* 1.

Lemma 14. *The scheme presented in Construction 2 is complete.*

Proof. Assume $x \in \mathcal{EQ}_\Delta$. Fix a $j \in [\lambda]$. Then,

$$\begin{aligned} z_j &= a_j^{-1} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_j} \bmod N^{\xi+1} \\ &= \left(h_0^{\alpha_j} \prod_{i=1}^{\ell} h_i^{\sigma_{i,j}}\right)^{-t} (1+N)^{-\tau_j} h_0^{t\alpha_j} \prod_{i=1}^{\ell} h_i^{t\sigma_{i,j}} (1+N)^{m_i\sigma_{i,j}} \bmod N^{\xi+1} \\ &= (1+N)^{\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j} \bmod N^{\xi+1} \end{aligned}$$

from which the verifier can recover $y_j = \sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j$. Moreover, $|y_j| < \ell\beta < N^\xi/2 < n/2$. That is, y_j does not wrap around modulo N^ξ nor modulo n.

In addition, we have

$$\begin{aligned} L_j &= K_j^{-1} D_0^{A_j} \prod_{i=1}^{\ell} D_i^{\sigma_{i,j}} \\ &= \left(H_0^{A_j} \prod H_i^{\sigma_{i,i}}\right)^{-r} G^{-\tau} H_0^{rA_0} \prod_{i=1}^{\ell} (H_i^r G^{m_i})^{\sigma_{i,j}} \\ &= G^{\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j} = G^{y_j} \end{aligned}$$

in $\mathbb{G}$. Thus, the proof is accepted as valid because $y_j \bmod N^\xi = y_j \bmod n$. □ □

Lemma 15. *The scheme presented in Construction 2 has zero-knowledge under the SD assumption.*

Proof. To prove zero-knowledge, we construct a simulator $\mathsf{Sim}_{\mathsf{ZK}}$ that creates transcripts which are indistinguishable from the ones outputted by the protocol.

Let Sim_1, Sim_2 and Sim_3 be the zero-knowledge simulators of the schemes $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{DL}_{\Delta_3}}$,respectively (which exist by Lemma 13, Lemma 10 and Lemma 9) The simulator $\mathsf{Sim}_{\mathsf{ZK}}$ works as follows:

- **CRS generation.** It creates a crs exactly as in the real protocol and keeps $\mathsf{td} = \left(\{\sigma_{i,j}\}_{i\in[\ell],j\in[\lambda]}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3\right)$ to itself.
- Upon receiving an instance $(D_0, h_0\{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]})$, $\mathsf{Sim}_{\mathsf{ZK}}$ first simulates $\pi_1 \leftarrow \mathsf{Sim}_1(\mathsf{td}_1, x_1)$, $\pi_2 \leftarrow \mathsf{Sim}_2(\mathsf{td}_2, x_2)$ and $\pi_3 \leftarrow \mathsf{Sim}_3(\mathsf{td}_3, x_3)$. Then, it repeats the following for every $j \in [\lambda]$: It samples $\tau_j \leftarrow_{\$} \Phi_{\mathbb{Z},\beta}$ and computes $a_j = (1+N)^{\tau_j} c_0^{\alpha_j} \prod c_{2,i}^{\sigma_{i,j}} \bmod N^{\xi+1}$ and $K_j = G^{\tau_j} H_0^{A_j} \prod D_i^{\sigma_{i,j}}$.
- It outputs $\pi = (\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3)$.

We now argue that the distributions of the proofs outputted by $\mathsf{Prove}_{\mathcal{EQ}_\Delta}$ and $\mathsf{Sim}_{\mathsf{ZK}}$ are indistinguishable. By the zero-knowledge property of $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{DL}_{\Delta_3}}$ the proofs π_1, π_2 and π_3 are indistinguishable from the real ones (π_1 is computationally indistinguishable given that the SD assumption holds). So, we just need to analyze the distributions of a_j and K_j.

We start by analyzing the distribution of a_j. First note that,

$$\begin{aligned} a_j^{-1} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} &= c_0^{-\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{-\sigma_{i,j}} (1+N)^{-\tau_j} c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} \bmod N^{\xi+1} \\ &= (1+N)^{-\tau_j} \bmod N^{\xi+1} \end{aligned}$$

To see that a_j is indistinguishable from one created in the real-world, note that by Lemma 1, we have that $\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j \approx_{\mathsf{negl}(\lambda)} \tau_j$, for $\tau_j \leftarrow_{\$} \Phi_{\mathbb{Z},\beta}$ and $m_i, \sigma_{i,j} \in \{0,1\}$, since $\lambda/\beta = \mathsf{negl}(\lambda)$. Hence,

$$(1+N)^{\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j} \approx_{\mathsf{negl}(\lambda)} (1+N)^{\tau_j}$$

and therefore

$$c_0^{\alpha_j} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}} (1+N)^{\tau_j} \approx_{\mathsf{negl}(\lambda)} f_j^t (1+N)^{\sum_{i=1}^{\ell} m_i\sigma_{i,j} - \tau_j}.$$

An identical argument can be used to show that K_j is indistinguishable from the one created in the real protocol. □ □

Lemma 16. *The scheme presented in Construction 2 has statistical simulation soundness.*

Proof. We first show how the simulator $\mathsf{Sim}_{\mathsf{Snd}}$ simulates the $\mathsf{Verify}_{\mathcal{EQ}_\Delta}(\mathsf{td}, \cdot, \cdot)$ oracle to the adversary. Since we are proving *statistical* simulation soundness, our simulator is allowed to run in exponential time.

$\mathsf{Sim}_{\mathsf{Snd}}$ works as follows:

- **CRS generation.** It generates crs by sampling $f_j \leftarrow_{\$} \mathbb{NR}_N$ and $F_j \leftarrow_{\$} p\mathbb{G}$.
- Upon receiving a query to the oracle Verify consisting of a statement $x = (D_0, h_0, \{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]})$ together with a proof π from the adversary, it does the following:
  1. It brute-forces the statement $(D_0, \{D_i\}_{i\in[\ell]})$ to recover the witness $(r, \{m_i\}_{i\in[\ell]})$, and $(c_0, \{c_{1,i}, c_{2,i}\}_{i\in[\ell]})$ to recover $(t, \{m'_i\}_{i\in[\ell]})$.
  2. It parses π as $(\{a_j, K_j\}_{j\in[\lambda]}, \pi_1, \pi_2, \pi_3)$. It brute-forces a_j to recover $\{\bar{t}_j, \tau'_j\}$, and K_j to recover $(\bar{r}_j, \tau_j)$.

3. If there is an index i such that $(c_{1,i}, c_{2,i})$ are not of the form $c_{1,i} = g^{t_i} \bmod N^{\xi+1}$ and $c_{2,i} = h^{t_i}(1+N)^{m'_i} \bmod N^{\xi+1}$ or D_i is not of the form $D_i = G^{m_i} H_i^r$ in $\mathbb{G}$ where $m_i \in \{0,1\}$, it outputs 0.
4. If there is an index i such that a_i is not of the form $f_i^t(1+N))^{\tau'_i} \bmod N^{\xi+1}$ (meaning that $\bar{t}_i \neq t$), it outputs 0. Moreover, if K is not of the form $G^{\tau_j} F_j^r$ in $\mathbb{G}$ (meaning that $\bar{r}_j \neq r$), output 0.
5. If c_0 is not of the form $h^t \bmod N^{\xi+1}$ or if D_0 is not of the form H_0^r, it outputs 0.
6. If there is $i \in [\ell]$ or $j \in [\lambda]$ such that $m_i \neq m'_i$ or $\tau'_j \neq \tau_j$, it outputs 0. Else, it outputs 1

- Upon receiving the challenge proof (x^*, π^*), it performs the same checks as in steps 3, 4 and 5. If the tests pass, it samples $\sigma_{i,j} \leftarrow\!\!\$ \{0,1\}$ and checks if

$$G^{\sum_{i=1}^{\ell} \sigma_{i,j} m'_i - \tau'_j \mod N^{\xi+1}} = G^{\sum_{i=1}^{\ell} \sigma_{i,j} m_i - \tau_j}$$

for every $j \in [\lambda]$. It outputs 0 if the test fails, 1 otherwise.

$\mathbf{Hyb}_0$: This is the real simulation soundness game.

$\mathbf{Hyb}_1$: This game is identical to the previous one except that $\mathsf{Sim}_{\mathsf{Snd}}$ brute-forces the pairs statement/proof (x, π), to recover the witness $(r, \{m_i\}_{i\in[\ell]})$, $(t, \{m'_i\}_{i\in[\ell]})$, and the values $(\tau_j, \bar{r}_j)$ and $\{\bar{t}_j, \tau'_j\}$ from the proof. Finally, it performs the checks in steps 3 and 4.

Claim 1. *Hybrids* $\mathbf{Hyb}_0$ *and* $\mathbf{Hyb}_1$ *are indistinguishable.*

The statistical simulation soundness of the schemes $\mathsf{NIZK}_{\mathcal{BGN}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{DJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{DL}_{\Delta_3}}$ guarantees that D_0, c_0, D_i and $(c_{1,i}, c_{2,i})$ are of the prescribed form, except with negligible probability.

Now fix $j \in [\lambda]$ and assume that $a_j = f_j^{\bar{t}_j}(1+N)^{\tau_j}$. Then, since $a_j^{-1} \prod_{i=1}^{\ell} c_{2,i}^{\sigma_{i,j}}$ must be of the form $(1+N)^{y_j}$, we must have

$$-\bar{t}_j \left(\alpha_j + \sum_{i=1}^{\ell} w_i \sigma_{i,j} \right) + t \left(\alpha_j + \sum_{i=1}^{\ell} w_i \sigma_{i,j} \right) = 0 \mod \phi(N)$$

where $h_0 = h_i^{w_i}$. Thus $\bar{t}_j = t$.

An identical reasoning can be applied to argue that K_j must be of the form $G^{\tau_j} F_j^r$.

$\mathbf{Hyb}_2$. This hybrid is identical to the previous one, except that $\mathsf{Sim}_{\mathsf{Snd}}$ performs the checks in step 6.

Claim 2. *Hybrids* $\mathbf{Hyb}_1$ *and* $\mathbf{Hyb}_2$ *are indistinguishable.*

The adversary $\mathcal{A}$ is able to distinguish both hybrids if there is a proof which is accepted in hybrid $\mathbf{Hyb}_1$ but rejected $\mathbf{Hyb}_2$ (or vice-versa). That is, suppose that $\mathcal{A}$ outputs $(\{m_i, m'_i\}, \{\tau_j, \tau'_j\})$. Fix j. Then the proof is accepted in $\mathbf{Hyb}_1$ if

$$\sum_{i=1}^{\ell} \sigma_{i,j} m_i - \tau_j \bmod n = \left(\sum_{i=1}^{\ell} \sigma_{i,j} m'_i - \tau'_j \bmod N^{\xi+1} \right) \bmod n. \tag{10}$$

Let $e_j = \sum_{i=1}^{\ell} \sigma_{i,j} m_i$ and $d_j = \sum_{i=1}^{\ell} \sigma_{i,j}(m'_i - m_i)$. Then, equation 10 can be rewritten as

$$e_j \mod n = (e_j + d_j - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n.$$

Consider the function $\Gamma_j(z)$ defined as $\Gamma_j(z) = (z - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n$. it is easy to see that $\Gamma_j(z)$ is injective in $\mathbb{Z}_{N^{\xi+1}}$. Let z_1, z_2 be such that

$$\begin{aligned}
\Gamma_j(z_1) &= \Gamma_j(z_2) \\
(z_1 - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n &= (z_2 - \tau'_j \bmod N^{\xi+1}) - \tau_j \bmod n \\
(z_1 - \tau'_j \bmod N^{\xi+1}) &= (z_2 - \tau'_j \bmod N^{\xi+1}) \\
z_1 \bmod N^{\xi+1} &= z_2 - \tau' \bmod N^{\xi+1}
\end{aligned}$$

where the second equivalence holds because $n > N^{\xi+1}$.

Since Γ_j is injective, then we must have

$$\begin{aligned}
e_j \bmod N^{\xi+1} &= e_j + d_j \bmod N^{\xi+1} \\
\sum_{i=1}^{\ell} \sigma_{i,j} m_i \bmod N^{\xi+1} &= \sum_{i=1}^{\ell} \sigma_{i,j} m_i + \sigma_{i,j}(m'_i - m_i) \bmod N^{\xi+1} \\
\sum_{i=1}^{\ell} \sigma_{i,j} m_i \bmod N^{\xi+1} &= \sum_{i=1}^{\ell} \sigma_{i,j} m'_i \bmod N^{\xi+1}
\end{aligned}$$

Assume that there is a index i such that $m_i \neq m'_i$. Then the test will fail with at most $1/2$ probability, for a fixed j. Repeating the process for $j \in [\lambda]$, we get that $m_i = m'_i$, except with negligible probability. Thus $\tau'_j = \tau_j$.

Hybrid $\mathbf{Hyb}_3$. This hybrid is identical to the previous one, except that f_i is chosen uniformly from $\mathbb{NR}_N$ and H is chosen uniformly from $p\mathbb{G}$.

Claim 3. *Hybrids* $\mathbf{Hyb}_2$ *and* $\mathbf{Hyb}_3$ *are indistinguishable.*

Since $\alpha_j \leftarrow_\$ \mathbb{Z}_N$, we can build an hybrid where α_j is sampled from $\mathbb{Z}_N^*$, incurring only in statistical distance. Moreover, since H_0 is a generator of $p\mathbb{G}$, $H_0^{A_j}$ is uniform in $p\mathbb{G}$. The claim follows.

Claim 4. *Let $\mathcal{A}$ be any adversary. For hybrid* $\mathbf{Hyb}_3$*, $\mathcal{A}$ has a negligible advantage.*

Assume that $\mathcal{A}$ outputs (x^*, π^*) where $x^* \notin \mathcal{EQ}_\Delta$. Since $\sigma_{i,j} \leftarrow_\$ \{0,1\}$, then the proof gets accepted if equation 10 is fulfilled. As we have seen before, this happens only with negligible probability. □ □

5.3 DV-NIZK for Range Proofs of DJ Ciphertexts with Equal Discrete Log

Let $N \leftarrow \mathsf{RSA}(\lambda)$ and $\xi \geq 0$ be a fixed integer. In the following, $t \in \mathbb{Z}_{N^\xi}$ are represented in $\{\lceil -N^\xi/2, \ldots, N^\xi/2 \rceil\}$. Consider the following language of ranges:

$$\mathcal{REDJ}_\Delta = \left\{ (c_1, c_2) \in \mathbb{Z}^2_{N^{\xi+1}} : \exists t \in \mathbb{Z}_{N^\xi} \text{ s.t. } \begin{array}{c} t \in [-B, B] \\ c_1 = g^t \bmod N^{\xi+1} \\ c_2 = h^t (1+N)^t \bmod N^{\xi+1} \end{array} \right\}$$

which is parametrized by $\Delta = ((g,h), B, N, \xi)$ where $(g,h) \in \mathbb{Z}_N^*$, $B \in \mathbb{Z}$, N and ξ.

In the following, we present a DV-NIZK scheme for the language above. The main idea is quite simple: The prover outputs BGN ciphertexts D_i encrypting bits m_i and DJ ciphertexts $(c_{1,i}, c_{2,i}$ that encrypt the same values as D_i (we can prove this using the scheme from the previous section). Then, the prover proves that (c_1, c_2) encrypts the same value as $\left(\prod_{i=0}^{\ell} c_{1,i}^{2^i}, \prod_{i=0}^{\ell} c_{2,i}^{2^i}\right)$. Since DJ is linearly-homomorphic, we conclude that (c_1, c_2) encrypts $m = \sum_{i=0}^{\ell} 2^i m_i \leq 2^{\ell-1}$.

Construction 3. *Let $\ell \in \mathbb{N}$ and $B = 2^{\ell-1}$ Let*

- $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}} = (\mathsf{NIZK.GenCRS}_{\mathcal{EQ}_{\Delta_1}}, \mathsf{NIZK.Prove}_{\mathcal{EQ}_{\Delta_1}}, \mathsf{NIZK.Verify}_{\mathcal{EQ}_{\Delta_1}})$ *be the scheme in Construction 2, for some* $\Delta_1 = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_i)$*;*
- $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}} = (\mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_{\Delta_2}}, \mathsf{NIZK.Prove}_{\mathcal{EPDJ}_{\Delta_2}}, \mathsf{NIZK.Verify}_{\mathcal{EPDJ}_{\Delta_2}})$ *be the scheme of Lemma 12, for some* $\Delta_2 = (g, h, N, \xi)$*;*
- $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}} = (\mathsf{NIZK.GenCRS}_{\mathcal{EDL}_{\Delta_3}}, \mathsf{NIZK.Prove}_{\mathcal{EDL}_{\Delta_3}}, \mathsf{NIZK.Verify}_{\mathcal{EDL}_{\Delta_3}})$ *be the scheme of Lemma 11, for some* $\Delta_3 = (g, h, N, \xi)$*.*

We now present the scheme in full detail.

$\mathsf{GenCRS}_{\mathcal{REDJ}_\Delta}(1^\lambda)$:

- *Compute* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EQ}_{\Delta_1}}(1^\lambda)$ *where* $\Delta_1 = (\mathbb{G}, n, G, H_0, \{H_i\}_{i\in[\ell]}, N, \xi, g, h_0, \{h_i\}_{i\in[\ell]})$*.*
- *Compute* $(\mathsf{crs}_{2,i}, \mathsf{td}_{2,i}) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_{\Delta_2}}(1^\lambda)$ *where* $\Delta_{2,i} = ((g,h),(g,h_i),N,\xi)$ *for all* $i \in [\ell]$*.*
- *Compute* $(\mathsf{crs}_{2,0}, \mathsf{td}_{2,0}) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EPDJ}_{\Delta_2}}(1^\lambda)$ *where* $\Delta_{2,0} = ((g,h),(g,h),N,\xi)$*.*
- *Compute* $(\mathsf{crs}_3, \mathsf{td}_3) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{EDL}_{\Delta_3}}(1^\lambda)$ *where* $\Delta_3 = (g, h(1+N), N, \xi)$*.*
- *Output* $\mathsf{crs} = (\mathsf{crs}_1, \mathsf{crs}_{2,0}, \mathsf{crs}_3, \{\mathsf{crs}_{2,i}\}_{i\in[\ell]})$ *and* $\mathsf{td} = (\mathsf{td}_1, \mathsf{td}_{2,0}, \mathsf{td}_3, \{\mathsf{td}_{2,i}\}_{i\in[\ell]})$*.*

$\mathsf{Prove}_{\mathcal{REDJ}_\Delta}(\mathsf{crs}, x = (c_1, c_2), w = t)$:

- *Parse* crs *as* $(\mathsf{crs}_1, \mathsf{crs}_{2,0}, \mathsf{crs}_3, \{\mathsf{crs}_{2,i}\}_{i\in[\ell]})$*.*
- *Let* $(t_1, \ldots, t_\ell)$ *be the bit decomposition of* $t + B/2 \in [0, B]$*. Sample* $r \leftarrow_\$ \mathbb{Z}_n$ *and* $s \leftarrow_\$ \mathbb{Z}_N$*. Compute the ciphertexts* $D_i = G^{t_i} H_i^r$*,* $D_0 = H_0^r$*,* $d_0 = h_0^s \bmod N^{\xi+1}$ *and* $(d_{1,i}, d_{2,i})$ *where* $d_{1,i} = g^s \bmod N^{\xi+1}$ *and* $d_{2,i} = h_i^s (1+N)^{t_i} \bmod N^{\xi+1}$*. Compute the proof* $\pi_1 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{EQ}_{\Delta_1}}(\mathsf{crs}_1, x_1, w_1)$ *where* $x_1 = (D_0, d_0, \{D_i, d_{1,i}, d_{2,i}\}_{i\in[\ell]})$ *and* $w_1 = (r, s, \{t_i\}_{i\in[\ell]})$*.*
- *For* $i \in [\ell]$*, sample* $s_i \leftarrow_\$ \mathbb{Z}_N$*. Compute the ciphertexts* $(c_{1,i}, c_{2,i})$ *where* $c_{1,i} = g^{s_i} \bmod N^{\xi+1}$ *and* $c_{2,i} = h^{s_i}(1+N)^{t_i} \bmod N^{\xi+1}$*. Compute the proofs* $\pi_{2,i} \leftarrow \mathsf{Prove}_{\mathcal{EPDJ}_{\Delta_{2,i}}}(crs_{2,i}, x_{2,i}, w_{2,i})$ *for* $i \in [\ell]$ *where* $x_{2,i} = (c_{1,i}, c_{2,i}, d_{1,i}, d_{2,i})$ *and* $w_{2,i} = (s_i, r, t_i)$*.*
- *Compute new ciphertexts* $(\bar{c}_1, \bar{c}_2)$ *and* (c_1, c_2') *where* $\bar{c}_1 = \prod_{i=1}^{\ell} (c_{1,i})^{2^{i-1}}$*,* $\bar{c}_2 = \prod_{i=1}^{\ell} (c_{2,i})^{2^{i-1}}$ *and* $c_2' = c_2(1+N)^{B/2}$*. Compute the proof* $\pi_{2,0} \leftarrow \mathsf{NIZK.Prove}_{\mathcal{EPDJ}_{\Delta_{2,0}}}(\mathsf{crs}_{2,0}, x_{2,0}, w_{2,0})$ *where* $x_{2,0} = ((c_1, c_2'), (\bar{c}_1, \bar{c}_2))$ *and* $w_{2,0} = (t, \bar{s}, t + B/2)$ *with* $\bar{s} = \sum_{j=1}^{\ell} s_j 2^{j-1}$*.*
- *Compute the proof* $\pi_3 \leftarrow \mathsf{NIZK.Prove}_{\mathcal{EDL}_{\Delta_3}}(\mathsf{crs}_3, x_3, w_3)$ *where* $x_3 = (c_1, c_2)$ *and* $w_3 = t$*.*
- *Output* $\pi = (D_0, d_0, \{D_i, d_{1,i}, d_{2,i}, c_{1,i}, c_{2,i}, \pi_{2,i}\}_{i\in[\ell]}, \pi_1, \pi_{2,0}, \pi_3)$*.*

$\mathsf{Verify}_{\mathcal{REDJ}_\Delta}(\mathsf{td}, x, \pi)$:

- *Parse* π *as* $(D_0, d_0, \{D_i, d_{1,i}, d_{2,i}, c_{1,i}, c_{2,i}, \pi_{2,i}\}_{i\in[\ell]}, \pi_1, \pi_{2,0}, \pi_3)$ *and* td *as* $(\mathsf{td}_1, \mathsf{td}_{2,0}, \mathsf{td}_3, \{\mathsf{td}_{2,i}\}_{i\in[\ell]})$
- *Compute* $\bar{c}_1 = \prod_{i=1}^{\ell} (c_{1,i})^{2^{i-1}}$*,* $\bar{c}_2 = \prod_{i=1}^{\ell} (c_{2,i})^{2^{i-1}}$ *and* $c_2' = c_2(1+N)^{B/2}$*.*
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{EQ}_{\Delta_1}}(\mathsf{td}_1, x_1, \pi_1)$ *where* $x_1 = (D_0, d_0, \{D_i, c_{1,i}, c_{2,i}\}_{i\in[\ell]})$*, output* 0*.*
- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{EPDJ}_{\Delta_{2,i}}}(\mathsf{td}_{2,i}, x_{2,i}, \pi_{2,i})$*, for all* $i \in \{0, \ldots, \ell\}$*, output* 0*.*

- *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{EDL}_{\Delta_1}}(\mathsf{td}_3, x_3, \pi_3)$ *where* $x_3 = (c_1, c_2)$*, output* 0*. Else, output* 1*.*

Lemma 17. *The scheme presented in Construction 3 is complete.*

Proof. Let $(c_1, c_2) \in \mathcal{REDJ}_\Delta$. Then, by the completeness of $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}}$, the proof π_3 is accepted. Now, the ciphertexts $(c_{1,i}, c_{2,i})$ encrypt bits t_i by the completeness of $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$. This means that the ciphertext $(\bar{c}_1, \bar{c}_2)$ encrypts $\bar{t} = \sum_{i=1}^{\ell} 2^{i-1} t_i$. Hence, $\bar{t} \in [0, B]$. By the completeness of $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}}$, (c_1, c_2') encrypts $\bar{t} \in [0, B]$ and, thus, $(c_1, c_2 = c_2'(1+N)^{-B/2})$ encrypts $t \in [-B/2, B/2]$. We conclude that the proof is accepted as valid. □ □

Lemma 18. *The scheme presented in Construction 3 has zero-knowledge under the SD assumption.*

Proof. The proof follows from the fact that the schemes $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}}$ are zero-knowledge (here $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$ has computational zero-knowledge under the SD assumption). □ □

Lemma 19. *The scheme presented in Construction 3 is statistically simulation sound.*

Proof. The proof follows readily from the fact that the schemes $\mathsf{NIZK}_{\mathcal{EQ}_{\Delta_1}}$, $\mathsf{NIZK}_{\mathcal{EPDJ}_{\Delta_2}}$ and $\mathsf{NIZK}_{\mathcal{EDL}_{\Delta_3}}$ are statistically simulation sound and that the DJ scheme is linear homomorphic. That is, if $(c_{1,i}, c_{2,i})$ all encrypt bits, then $(\bar{c}_1, \bar{c}_2)$ where $\bar{c}_1 = \prod_{i=1}^{\ell} (c_{1,i})^{2^{i-1}}$ and $\bar{c}_2 = \prod_{i=1}^{\ell} (c_{2,i})^{2^{i-1}}$ is an encryption of a value smaller than $2^{\ell-1}$. □ □

6 Reusable Laconic Private Set Intersection

In this section, we present a protocol that implements ℓPSI in a black-box fashion. We then prove that the protocol guarantees security against a semi-honest receiver and against a malicious sender. The input sets are subsets of a universe $\mathcal{U}$ of exponential size.

Protocol. We now present the construction for reusable PSI.

Construction 4. *Let* $\mathcal{U}$ *be a universe which contain the input sets of the parties. Let* $\kappa \in \mathbb{Z}$ *such that* $5\kappa \leq \lambda$ *and* $\xi \in \mathbb{N}$*. We require the following ingredients in this construction:*

1. *A PPRF* $\mathsf{PPRF} : \mathcal{K} \times \mathcal{U} \to \mathsf{Primes}(\kappa)$ *which outputs prime numbers.*[14]
2. *A DV-NIZK* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZKVerify}_{\mathcal{REDJ}_\Delta})$ *for the language* $\mathcal{REDJ}_\Delta$ *which is defined in Section 5, for some* $\Delta = ((g_0, g_1), B, N, \xi)$*.*
3. *An IND-CPA PKE scheme* $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$
4. *A* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor* $\mathsf{Ext} : \mathcal{S} \times \mathbb{Z}_{N^{\xi+1}} \to \{0,1\}^\lambda$*.*

We assume that the receiver's set is of size M *and the sender's set is of size* m*, where* $M > m$*. The protocol is composed by the following algorithms:*

$\mathsf{GenCRS}(1^\lambda)$:

- *Sample* $N \leftarrow_\$ \mathsf{RSA}(\lambda)$*, that is,* $N = PQ$ *where* P, Q *are prime numbers. Choose* B *such that* $N^{\xi-1}/2 \geq B > N2^\kappa$*.*
- *Sample a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$*. Additionally, sample a PPRF key* $k \leftarrow_\$ \mathcal{K}$*. Set* $\Delta = ((g_0, g_1), B, N, \xi)$ *where* $(g_0, g_1) \leftarrow_\$ \mathbb{NR}_N$*.*
- *Output* $\mathsf{crs} = (N, \mathsf{pk}, (g_0, g_1), B, k, \Delta)$*.*

[14] We remark that we use a PPRF, not because we want uniform outputs, but to implicitly define the set of primes. A similar trick was used in [BGI16].

$\mathsf{R}_1(\mathsf{crs}, S_\mathsf{R})$:

- *Parse* $\mathsf{crs} := (N, \mathsf{pk}, (g_0, g_1), B, k, \Delta)$, *and* $S_\mathsf{R} := \{\mathsf{id}_i\}_{i\in[M]} \subseteq \mathcal{U}$
- *Compute the prime numbers* $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}_i)$, *for all* $i \in [M]$.
- *Sample* $r \leftarrow_\$ Z_N \setminus \{0\}$ *and compute* $h = g_0^{r\prod_{i\in[M]} p_i} \bmod N^{\xi+1}$.
- *Run* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{RELDJ}_\Delta}(1^\lambda)$.
- *Output* $\mathsf{st} = (r, \mathsf{td}_1)$ *and* $\mathsf{psi}_1 = (h, \mathsf{crs}_1)$.

$\mathsf{S}(\mathsf{crs}, S_\mathsf{S}, \mathsf{psi}_1)$:

- *Parse* $\mathsf{crs} := (N, \mathsf{pk}, (g_0, g_1), B, k, \Delta)$, $\mathsf{psi}_1 := (h, \mathsf{crs}_1)$ *and* $S_\mathsf{S} := \{\mathsf{id}'_i\}_{i\in[m]} \subseteq \mathcal{U}$.
- *For* $i \in [m]$ *do the following:*
  - *Sample* $\rho_i \leftarrow_\$ \mathbb{Z}_N$. *Compute the prime numbers* $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}'_i)$.
  - *Sample an extractor seed* $s_i \leftarrow_\$ \mathcal{S}$ *and compute* $R_i \leftarrow \mathsf{Ext}(s_i, h^{\rho_i} \bmod N^{\xi+1})$
  - *Compute* $f_i = g_0^{\rho_i p_i} \bmod N^{\xi+1}$, $F_i = g_1^{\rho_i p_i}(1+N)^{\rho_i p_i} \bmod N^{\xi+1}$ *and* $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}'_i; R_i)$.
  - *Compute* $\pi_i \leftarrow \mathsf{NIZK.Prove}_{\mathcal{RELDJ}_\Delta}(\mathsf{crs}_1, x_i, w_i)$ *where* $x_i = (f_i, F_i)$ *and* $w_i = \rho_i p_i$.
- *Output* $\mathsf{psi}_2 = \{(f_i, F_i), \mathsf{ct}_i, s_i, \pi_i\}_{i\in[m]}$.

$\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$:

- *Parse* $\mathsf{st} := (r, \mathsf{td}_1)$ *and* $\mathsf{psi}_2 := \{(f_i, F_i), \mathsf{ct}_i, s_i, \pi_i\}_{i\in[m]}$. *Set* $\mathcal{I} = \emptyset$
- *For all* $j \in [m]$ *do the following:*
  - *If* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{RELDJ}_\Delta}(\mathsf{td}_1, x_j, \pi_j)$ *where* $x_j = (f_j, F_j)$, *abort the protocol.*
  - *If there is a* $i \in [M]$ *such that*
$$\mathsf{ct}_j = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}_i; R'_i)$$
where $R'_i \leftarrow \mathsf{Ext}(s_j, f_j^{r_i} \bmod N^{\xi+1})$ *and* $r_i = r \prod_{\ell=1,\ell\neq i}^{M} p_\ell$, *then add the element* id_i *to* $\mathcal{I}$.
- *Output* $\mathcal{I}$.

Communication cost. Here, we analyze the communication cost of the protocol as a function of the input set sizes $|S_\mathsf{S}| = m$ and $|S_\mathsf{R}| = M$ and we omit polynomial factors in the security parameter λ. The first message outputted by R_1 has size $\mathcal{O}(1)$. The second message outputted by S has size $\mathcal{O}(m)$. The overall communication cost is $\mathcal{O}(m)$, that is, it is independent of M.

Analysis. We now analyze the correctness and security of the protocol.

Theorem 2. *The protocol presented in Construction 4 is correct.*

Proof. Let (h, crs_1) be the message sent by R_1 created using the set S_R as input.

Fix an index j such that $b_j \in S_\mathsf{S} \cap S_\mathsf{R}$. Upon receiving $((f_j, F_j), \mathsf{ct}_j, s_j, \pi_j^{\mathcal{RELDJ}})$ from S_S (i.e., the part of psi_2 with respect to b_j).

Since $|\rho_j p_j| < 2^\kappa N < B$, then $1 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{RELDJ}_\Delta}(\mathsf{td}_1, x_j, \pi_j)$ where $x_j = (f_j, F_j)$ except with negligible probability by the completeness of $\mathsf{NIZK}_{\mathcal{RELDJ}_\Delta}$.

Additionally, let $\tilde{r} = r \prod_{i:q_i \neq p_j} q_i$ where $q_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}_i)$ for $\mathsf{id}_i \in S_\mathsf{R}$ and $p_j \leftarrow \mathsf{PPRF}(k, b_j)$. Then

$$\begin{aligned} f_j^{\tilde{r}} \bmod N^{\xi+1} &= g_0^{\rho_j p_j r \prod_{i:q_i \neq p_j} q_i} \bmod N^{\xi+1} \\ &= g_0^{\rho_j r \prod_i q_i} \bmod N^{\xi+1} \\ &= h^{\rho_j} \bmod N^{\xi+1}. \end{aligned}$$

Hence, $R'_j = R_j$ and thus, $\mathsf{ct}_j = \mathsf{PKE.Enc}(\mathsf{pk}, b_j; R'_j)$. Therefore, b_j is added to $\mathcal{I}$. □ □

Theorem 3. *The protocol presented in Construction 4 securely UC-realizes functionality $\mathcal{F}_{\mathsf{rPSI}}$ in the $\mathcal{G}_{\mathsf{CRS}}$-hybrid model against:*

- *a semi-honest receiver given that the ϕ-hiding and the DCR assumptions hold;*
- *a malicious sender, where security holds statistically.*

Proof. We start by proving that the protocol is secure against semi-honest adversaries corrupting the receiver.

Lemma 20. *The protocol is secure against a semi-honest receiver.*

We first show how the simulator Sim_R works. In the following, let $\mathsf{Sim}_{\mathsf{NIZK}}$ be the zero-knowledge simulator from Lemma 18 for the $\mathsf{NIZK}_{\mathcal{RED}\mathcal{J}_\Delta}$ scheme.

1. Sim_R takes the input S_R of R and sends it to the ideal functionality $\mathcal{F}_{\mathsf{rPSI}}$.
2. **CRS generation.** To generate the CRS, Sim behaves as the honest algorithm would do, except that it sets $g_1 = g'_1(1+N)^{-1} \bmod N^{\xi+1}$ where $g'_1 \leftarrow_\$ \mathbb{NR}_N$.
3. The simulator creates the semi-honest receiver's view exactly as in the real protocol and keeps $\mathsf{st} = (r, \mathsf{td}_1)$ to itself.
4. Upon receiving a message $\mathsf{psi}_1 = (h, \mathsf{crs}_1)$ from R and a message $\mathcal{I}$ (of size m', that is, $|\mathcal{I}| = m'$) from the ideal functionality $\mathcal{F}_{\mathsf{rPSI}}$, the simulator does the following:
   - Sample a subset $\mathcal{X}$ of size $m - m'$ from the universe $\mathcal{U}$ and sets $S_\mathcal{S} = \mathcal{I} \cup \mathcal{X}$.
   - For all $i \in \mathcal{I}$, Sim_R computes $((f_i, F_i), \mathsf{ct}_i, s_i, \pi_i)$ as in the real protocol.
   - For all $i \in S_\mathsf{S} \setminus \mathcal{I}$, Sim_R simulates proofs $\pi_i \leftarrow \mathsf{Sim}_{\mathsf{NIZK}}(\mathsf{td}_1, x)$ for $x = (f_i, F_i)$ where $f_i, F_i \leftarrow_\$ \mathbb{NR}_N$. Then, it encrypts $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0; R_i)$ where $R_i \leftarrow \{0,1\}^\lambda$.

To prove indistinguishability between the real protocol and the simulated one, we consider the following sequence of hybrids:

$\mathbf{Hyb}_0$: The is the real protocol.

$\mathbf{Hyb}_1$: This hybrid is identical to the previous one, except that, for $i \in S_\mathsf{S} \setminus \mathcal{I}$, Sim_R simulates the proofs $\pi_i \leftarrow \mathsf{Sim}_{\mathsf{NIZK}}(\mathsf{td}_1, x)$ for $x_i = (f_i, F_i)$.

Claim 5. *Hybrids $\mathbf{Hyb}_0$ and $\mathbf{Hyb}_1$ are statistically indistinguishable.*

The claim above follows directly from the statistical zero-knowledge of the scheme $\mathsf{NIZK}_{\mathcal{RED}\mathcal{J}_\Delta}$.

$\mathbf{Hyb}_2$: This hybrid is identical to the previous one, except that the simulator replaces $g_1 \leftarrow_\$ \mathbb{NR}_N$ by $g_1 = g'_1(1+N)^{-1} \bmod N^{\xi+1}$ where $g'_1 \leftarrow_\$ \mathbb{NR}_N$.

Claim 6. *Assume that the DCR assumption holds. Then hybrids $\mathbf{Hyb}_1$ and $\mathbf{Hyb}_2$ are indistinguishable.*

The claim above follows directly from Lemma 1 which states that it is hard to distinguish $g_1 \leftarrow_\$ \mathbb{NR}_N$ from $g_1 = g'_1(1+N)^{-1} \bmod N^{\xi+1}$ where $g'_1 \leftarrow_\$ \mathbb{NR}_N$, given that the DCR assumption holds.

$\mathbf{Hyb}_{3,\ell}$: This hybrid is identical to the previous one, except that the simulator samples $f_{u\ell}, F_{u\ell} \leftarrow_{\$} \mathbb{NR}_N$ and computes

$$R_{u_\ell} \leftarrow \mathsf{Ext}\left(s, f_{u_\ell}^{r q_{u_\ell}^{-1} \prod_j^M p_j} \bmod N^{\xi+1}\right)$$

where $q_{u\ell} \leftarrow \mathsf{PPRF}(k, x_{u\ell})$ for all $u_\ell \in \{i : x_i \in S_{\mathsf{S}} \setminus \mathcal{I}\}$ and $p_j \leftarrow \mathsf{PPRF}(k, y_j)$ for all $y_j \in S_{\mathsf{R}}$. The hybrid is defined for $\ell = 1, \ldots, m - m'$.

Claim 7. *Hybrids* $\mathbf{Hyb}_2$ *and* $\mathbf{Hyb}_{3,m-m'}$ *are indistinguishable.*

We prove that hybrids $\mathbf{Hyb}_{3,\ell-1}$ and $\mathbf{Hyb}_{3,\ell}$ are indistinguishable for $\ell = 1, \ldots, m - m'$ and where $\mathbf{Hyb}_{3,0} = \mathbf{Hyb}_2$.

First, remark that the distribution of ρ_{u_ℓ} is the uniform distribution over $\mathbb{Z}_N$. Hence, we can build a statistically indistinguishable sequence of hybrids $\mathbf{Hyb}'_2$ where we sample $\rho_{u_\ell} \leftarrow_{\$} \mathbb{Z}_N^*$ incurring only in statistical distance.

Now, since g_0 and $g_0^{p_{u_\ell}}$ are generators of $\mathbb{NR}_N$, then the distribution of $g_0^{\rho_{u_\ell} p_{u_\ell}}$ is identical to $\tilde{f}_i \leftarrow_{\$} \mathbb{NR}_N$, for $\rho_{u_\ell} \leftarrow_{\$} \mathbb{Z}_N^*$.

For p_i sampled using PPRF (for a uniform input $x_{u_\ell} \leftarrow_{\$} \mathcal{U}$), we know that p_{u_ℓ} does not divide $\phi(N)$ and $\rho_{u_\ell} \in \mathbb{Z}_N^*$ if $\rho_{u_\ell} \leftarrow_{\$} \mathbb{Z}_N$, except with negligible probability. We conclude that

$$\tilde{f}_{u_\ell} \bmod N^{\xi+1} \approx_{\mathsf{negl}(\lambda)} g_0^{\rho_{u_\ell} p_{u_\ell}} \bmod N^{\xi+1}$$

where $\tilde{f}_{u_\ell} \leftarrow_{\$} \mathbb{NR}_N$, $g_0 \leftarrow_{\$} \mathbb{NR}_N$, $\rho_{u_\ell} \leftarrow_{\$} Z_N^*$ and $p_{u_\ell} \leftarrow_{\$} \mathsf{Primes}(\kappa)$.

Using a similar argument, we have that

$$F_{u_\ell} \bmod N^{\xi+1} \approx_{\mathsf{negl}(\lambda)} (g_1(1+N))^{\rho_{u_\ell} p_{u_\ell}} \bmod N^{\xi+1}$$

and that for any G

$$\left(f_{u_\ell} \bmod N^{\xi+1}, {f_{u_\ell}}^{G p_{u_\ell}^{-1}} \bmod N^{\xi+1}\right) \approx_{\mathsf{negl}(\lambda)} \left(g_0^{\rho_{u_\ell} p_{u_\ell}} \bmod N^{\xi+1}, {g_0}^{\rho_{u_\ell} G} \bmod N^{\xi+1}\right).$$

$\mathbf{Hyb}_{4,\ell}$: This hybrid is identical to the previous one except that $\mathsf{Sim}_{\mathsf{R}}$ computes $R_{u_\ell} \leftarrow_{\$} \{0,1\}^\lambda$ for all $u_\ell \in \{i : x_i \in S_{\mathsf{S}} \setminus \mathcal{I}\}$. The hybrid is defined for $\ell = 1, \ldots, m - m'$.

Claim 8. *Assume that* Ext *is a* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor and that the* ϕ*-hiding assumption holds. Then hybrids* $\mathbf{Hyb}_3$ *and* $\mathbf{Hyb}_{4,m-m'}$ *are indistinguishable.*

We prove that hybrids $\mathbf{Hyb}_{4,\ell-1}$ and $\mathbf{Hyb}_{4,\ell}$ are indistinguishable by constructing a reduction that contradicts Lemma 2, for $\ell = 1, \ldots, m - m'$ and where $\mathbf{Hyb}_3 = \mathbf{Hyb}_{4,0}$.

Suppose that there is an adversary $\mathcal{A}$ that distinguishes hybrids $\mathbf{Hyb}_{4,\ell-1}$ and $\mathbf{Hyb}_{4,\ell}$. We build an adversary $\mathcal{B}$ that breaks Lemma 2.

$\mathcal{B}$ receives as input (N, s, q, g). It behaves as the simulator in Hybrid $\mathbf{Hyb}_{4,\ell-1}$ except that it sets the modulus in the crs to be N. Additionally, it programs the PPRF such that $q \leftarrow \mathsf{PPRF}(k, x_{u_\ell})$ (this step is done while creating the PPRF key). Upon receiving a message from $\mathcal{A}$ (together with its view), it computes $G = r \prod_i^M p_i$ where $p_i \leftarrow \mathsf{PPRF}(k, x_i)$ for $x_i \in S_{\mathsf{R}}$. It sends G to the challenger and receives $\tilde{z}$. This value $\tilde{z}$ is either equal to $\mathsf{Ext}(s, \tilde{g}^{G/q} \bmod N^{\xi+1})$, if $\gamma = 0$, or it is uniformly chosen, if $\gamma = 1$, where γ is the challenge bit. Now $\mathcal{B}$ sets $f_{u\ell} = \tilde{g}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x_{u\ell}; R_{u\ell})$ and sends $\mathsf{psi}_2 =$ where the u_ℓ-th coordinate is $(f_{u\ell}, F_{u\ell}, \mathsf{ct}_{u\ell}, s_{u\ell}, \pi_{u\ell})$. The adversary outputs a bit b and $\mathcal{B}$ sets b as its guess. It is easy to see that if $\gamma = 0$, then $\mathcal{B}$'s message is indistinguishable from the message of hybrid $\mathbf{Hyb}_{4,\ell-1}$ and if $\gamma = 1$, the it is indistinguishable from the message sent in hybrid $\mathbf{Hyb}_{4,\ell}$.

$\mathbf{Hyb}_{5,\ell}$: This hybrid is identical to the previous one except that $\mathsf{Sim}_{\mathsf{R}}$ encrypts $\mathsf{ct}_{u\ell} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0; R_{u\ell})$ for all for all $u_\ell \in \{i : x_i \in S_{\mathsf{S}} \setminus \mathcal{I}\}$. The hybrid is defined for $\ell = 1, \ldots, m - m'$. Hybrid $\mathbf{Hyb}_{5,m-m'}$ is identical to the simulation.

Claim 9. *Assume that* PKE *is an IND-CPA PKE. Then hybrids* $\mathbf{Hyb}_4$ *and* $\mathbf{Hyb}_{5,m-m'}$ *are indistinguishable.*

The claim follows directly from the IND-CPA property of the underlying PKE. That is, given an adversary $\mathcal{A}$ that distinguishes both hybrids, we can easily build an adversary $\mathcal{B}$ against the IND-CPA property of PKE. This adversary $\mathcal{B}$ simply chooses as messages $m_0 = x_{u_\ell}$ (where $x_{u_\ell} \in S_{\mathsf{S}} \setminus \mathcal{I}$) and $m_1 = 0$. It outputs whatever $\mathcal{A}$ outputs.

Lemma 21. *The protocol is secure against malicious senders.*

We first show how the simulator $\mathsf{Sim}_{\mathsf{S}}$ extracts the sender's input:

1. **CRS generation.** $\mathsf{Sim}_{\mathsf{S}}$ generates the crs following the algorithm GenCRS. It keeps $\phi(N)$ to itself (which can be computed using the prime numbers p, q) and the secret key sk corresponding to pk. It outputs $\mathsf{crs} = (\mathsf{pk}, (g_0, g_1), B, k, \Delta)$
2. $\mathsf{Sim}_{\mathsf{S}}$ samples $h \leftarrow_{\$} \mathbb{NR}_N$ and computes $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{RELDJ}_\Delta}(1^\lambda)$. It sends $\mathsf{psi}_1 = (h, \mathsf{crs}_1)$ to the malicious sender.
3. Whenever $\mathsf{Sim}_{\mathsf{S}}$ receives a message $\mathsf{psi}_2 = \{(f_i, F_i), \mathsf{ct}, s_i, \pi_i\}_{i\in[m]}$ from the sender, the simulator initially sets S_{S} and does the following for all $i \in [m]$:
   - It checks if $1 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{RELDJ}_\Delta}(\mathsf{td}_1, x_j, \pi_j)$ where $x_j = (f_j, F_j)$, and aborts otherwise.
   - It computes $\mathsf{id}'_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}_i)$ and $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}'_i)$. Additionally, it extracts ζ_i by recovering ζ'_i from $(1+N)^{\zeta'_i} = F_i^{\phi(N)}$ and computing $\zeta = \zeta'/\phi(N)$ over the N^ζ. It computes $\rho'_i = \zeta_i/p_i$ over the integers. If $\mathsf{ct}_i = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}'_i; R_i)$ where $R_i = \mathsf{Ext}(s_i, h^{\rho'_i} \bmod N^{\xi+1})$, then it adds id'_i to S_{S}.
4. It sends S_{S} to $\mathcal{F}_{\mathsf{PSI}}$ and halts.

We now show that the simulation is indistinguishable from the real protocol via the following sequence of hybrids.

$\mathbf{Hyb}_0$: This hybrid is the real protocol.

$\mathbf{Hyb}_1$: This hybrid is identical to the previous one, except that $\mathsf{Sim}_{\mathsf{S}}$ keeps $(\phi(N), \mathsf{sk})$ while creating crs. Note that crs is perfectly indistinguishable from a honestly created one.

$\mathbf{Hyb}_2$: This hybrid is identical to the previous one except that the simulator computes the first message (sent by the receiver) as $h \leftarrow \mathbb{Z}_{\phi(N)}$.

Claim 10. *Hybrids* $\mathbf{Hyb}_1$ *and* $\mathbf{Hyb}_2$ *are statistically indistinguishable.*

Since g_0 is a generator of $\mathbb{NR}_N$, the distributions of g^x and $h \leftarrow_{\$} \mathbb{NR}_N$ are identical. It follows that the hybrids are indistinguishable.

$\mathbf{Hyb}_3$: This hybrid is identical to the previous one except that the simulator, instead of checking if there is an index i for which

$$\mathsf{ct}_j = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}_i; R'_i)$$

where $R'_i = \mathsf{Ext}(s_j, f_j^{r_i})$ and $r_i = r \prod_{\ell=1, \ell \neq i}^{M} p_\ell$ (as in the real protocol), it does the checks as in the simulation. That is, it computes $\mathsf{id}'_i \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}_i)$ and $p_i \leftarrow \mathsf{PPRF}(k, \mathsf{id}'_i)$. Additionally, it extracts ζ_i by recovering ζ'_i from $(1+N)^{\zeta'_i} = F_i^{\phi(N)}$ and computing $\zeta = \zeta'/\phi(N)$. It computes $\rho'_i = \zeta_i/p_i$ over the integers. Then, it checks if $\mathsf{ct}_i = \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{id}'_i; R_i)$ where $R_i = \mathsf{Ext}(s_i, h^{\rho'_i})$.

Claim 11. *Hybrids* $\mathbf{Hyb}_2$ *and* $\mathbf{Hyb}_3$ *are indistinguishable given that* PKE *is correct and* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$ *is simulation sound.*

By the simulation soundness of $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta}$, $\zeta_i < N^{\xi-1}/2$. Hence, $\zeta'_i < N^\xi/2$ and thus $\zeta'_i \bmod N^\xi$ is equal to ζ'_i as an integer. Computing $\zeta = \zeta'_i/\phi(N)$ yields $\rho_i p_i$ over $\mathbb{Z}$. Thus $\rho_i = \zeta_i/p_i$ over $\mathbb{Z}$.

Thus, performing the checks in this hybrid has the same outcome as in the real protocol. □ □

Setting the parameters. The value B is such that $N^{\xi-1}/2 \geq B > N2^\kappa$ for $5\kappa \leq \lambda$. Then, it is enough to set $\xi = 3$, so that we can find a B that fulfills the condition.

7 Labeled Laconic PSI and Laconic OT

In this section, we show how we can extend the techniques developed in Section 6 to construct LPSI to obtain new constructions of labeled LPSI and LOT. Both constructions are reusable and secure against malicious senders.

7.1 Reusable Labeled Laconic PSI Secure Against a Malicious Sender

Reusable Labeled PSI functionality. The functionality $\mathcal{F}_{\mathsf{rLPSI}}$ is parametrized by a universe $\mathcal{U}$ and by a universe of labels $\mathcal{L}$ and works as follows:

- **Setup phase.** R sends $(\mathsf{sid}, S_\mathsf{R})$ to $\mathcal{F}_{\mathsf{rLPSI}}$ where $S_\mathsf{R} \subseteq \mathcal{U}$. It ignores future messages from R with the same sid.
- **Send phase.** S sends $(\mathsf{sid}, i, S_{\mathsf{S},\mathsf{lab}} \subseteq \mathcal{U} \times \mathcal{L})$ from S to $\mathcal{F}_{\mathsf{rLPSI}}$. $\mathcal{F}_{\mathsf{rLPSI}}$ sends $(\mathsf{sid}, i, S_{\mathsf{R}\cap S_\mathsf{S},\mathsf{lab}})$ to R, where $S_{\mathsf{R}\cap S_\mathsf{S},\mathsf{lab}} = \{(y, \ell) \in S_{\mathsf{S},\mathsf{lab}} : y \in S_\mathsf{R}\}$. It ignores future messages from S with the same sid and $i \in \mathbb{N}$.

Protocol. We now present the construction for labeled reusable PSI.

Construction 5. *Let* $\mathcal{U}$ *be a universe which contain the input sets of the parties. Let* $\kappa \in \mathbb{Z}$ *such that* $5\kappa \leq \lambda$. *Let*

- $\mathsf{PRF} : \mathcal{K} \times \mathcal{U} \to \mathsf{Primes}(\kappa)$ *be a PRF which outputs prime numbers*
- $\mathcal{REDJ}_\Delta$ *be the language defined in Section 5 and* $\mathsf{NIZK}_{\mathcal{REDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{REDJ}_\Delta}, \mathsf{NIZKVerify}_{\mathcal{REDJ}_\Delta})$ *be a DV-NIZK for the language* $\mathcal{REDJ}_\Delta$, *for some* $\Delta = ((g_0, g_1), B, N, \xi)$.
- $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be an IND-CPA PKE scheme*
- $\mathsf{Ext} : \mathcal{S} \times \mathbb{Z}_{N^{\xi+1}} \to \{0,1\}^{2\lambda}$ *be a* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor.*

We assume that the receiver's set is of size M *and the sender's set is of size* m, *where* $M > m$. *The protocol is composed by the following algorithms:*

GenCRS : *This algorithm is identical to the one described in Construction 4.*

$\mathsf{R}_1(\mathsf{crs}, S_\mathsf{R})$: *This algorithm is identical to the one described in Construction 4.*

$\mathsf{S}(\mathsf{crs}, S_\mathsf{S}, \mathsf{psi}_1)$: *This algorithm is identical to the one described in Construction 4, except that* $(R_i||T_i) \leftarrow \mathsf{Ext}(s, h^{\rho_i} \bmod N^{\xi+1})$. *The string* R_i *is used to encrypt the set element (as in Construction 4) Additionally, compute* $\bar{\mathsf{ct}}_i = T_i \oplus \mathsf{lab}_i$, *where* lab_i *is the corresponding label.*

$\mathsf{R}_2(\mathsf{crs}, \mathsf{st}, \mathsf{psi}_2)$: *This algorithm is identical to the one described in Construction 4, except that whenever* $a_i \in \mathcal{I}$, *compute* $\mathsf{lab}_i = T_i \oplus \bar{\mathsf{ct}}_i$. *Output* $(\mathcal{I}, \{\mathsf{lab}_i\}_{i\in\mathcal{I}})$.

Analysis. We state the theorems that guarantee the required properties for our scheme. We omit the proofs since they are identical to the proofs of Theorems 2 and 3

Theorem 4. *The protocol presented in Construction 5 is correct.*

Theorem 5. *The protocol presented in Construction 5 securely UC-realizes functionality* $\mathcal{F}_{\mathsf{rLPSI}}$ *in the* $\mathcal{G}_{\mathsf{CRS}}$*-hybrid model against:*

- *a semi-honest receiver given that the* ϕ*-hiding and the DCR assumptions hold;*
- *a malicious sender, where security holds statistically.*

7.2 Laconic Oblivious Transfer with Malicious Sender Security

In this section, we present a new laconic oblivious transfer (LOT) scheme which is secure against malicious sender. Besides, it only needs a small CRS and succinct messages for both rounds (as in [GVW20]).

Laconic oblivious transfer ideal functionality. Let $\Gamma = \Gamma(\lambda) \in \mathbb{N}$. The functionality $\mathcal{F}_{\ell\mathsf{OT}}$ works as follows: It receives a database $D \in \{0,1\}^\Gamma$ from R. Upon receiving a message $(i \in \mathbb{N}, m_0, m_1, L \in [\Gamma])$ from the sender S, $\mathcal{F}_{\ell\mathsf{OT}}$ sends (i, m_{D_L}) to R and ignores future messages with the same i from S.

Protocol. We now present the construction for sender-malicious LOT.

Construction 6. *Let* $\Gamma = \Gamma(\lambda)$ *be a polynomial in* λ. *Let Let* $\kappa \in \mathbb{Z}$ *such that* $5\kappa \le \lambda$. *Let*

- $\mathsf{PRF} : \mathcal{K} \times \mathcal{U} \to \mathsf{Primes}(\kappa)$ *be a PRF which outputs prime numbers*
- $\mathcal{RDJ}_\Delta$ *be the language defined in Section 5 and* $\mathsf{NIZK}_{\mathcal{RDJ}_\Delta} = (\mathsf{NIZK.GenCRS}_{\mathcal{RDJ}_\Delta}, \mathsf{NIZK.Prove}_{\mathcal{RDJ}_\Delta}, \mathsf{NIZKVerify}_{\mathcal{RDJ}_\Delta})$ *be a DV-NIZK for the language* $\mathcal{RDJ}_\Delta$, *for some* $\Delta = ((g_0, g_1), B, N, \xi)$.
- $\mathsf{Ext} : \mathcal{S} \times \mathbb{Z}_{N^{\xi+1}} \to \{0,1\}^{2\lambda}$ *be a* $(\kappa - 1, \mathsf{negl}(\lambda))$*-strong extractor.*

$\mathsf{GenCRS}(1^\lambda)$: *This algorithm is identical to the one described in Construction 4, except that it does not create a public key* pk. *It outputs* $\mathsf{crs} = (N, (g_0, g_1), B, k, \Delta)$ *where* $\Delta = ((g_0, g_1), B, N, \xi)$.

$\mathsf{Hash}(\mathsf{crs}, D \in \{0,1\}^\Gamma)$: *It computes* $h = g_0^{r \prod_{i=1}^{\Gamma} e_{i,D_i}} \bmod N^{\xi+1}$, *where* $r \leftarrow_\$ \mathbb{Z}_N$ *and* $e_{i,b} \leftarrow \mathsf{PPRF}(k, 2i+b)$ *for* $i \in [\Gamma]$ *and* $b \in \{0,1\}$, *and computes* $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{NIZK.GenCRS}_{\mathcal{RDJ}_\Delta}(1^\lambda)$. *It outputs* $\mathsf{lot}_1 = (h, \mathsf{crs}_1)$.

$\mathsf{Send}(\mathsf{crs}, \mathsf{lot}_1, m_0, m_1, L)$: *It computes* $f_j = g_0^{\rho_j e_{L,j}}$, $F_j = g_1^{\rho_j e_{L,j}}(1+N))^{\rho_j e_{L,j}}$ *for* $j \in \{0,1\}$ *where* $\rho_j \leftarrow_\$ D_\beta$ *and* $e_{L,j} \leftarrow \mathsf{PPRF}(k, 2L+j)$, *computes* $\mathsf{ct}_j = k_j \oplus m_j$, *where* $k_j \leftarrow \mathsf{Ext}(s_j, h^{\rho_j})$, *computes* $\pi_j \leftarrow \mathsf{NIZK.Prove}_{\mathcal{RDJ}_\Delta}(\mathsf{crs}, x_j, w_j)$ *where* $x_j = (f_j, F_j)$ *and* $w_j = (\rho_j e_{L,j})$ *It outputs* $\mathsf{lot}_2 = (\{f_j, F_j, \mathsf{ct}_j, \pi_j, s_j\}_{j\in\{0,1\}}, L)$.

$\mathsf{Receive}(\mathsf{crs}, \mathsf{lot}_2, \mathsf{st})$: *It aborts if* $0 \leftarrow \mathsf{NIZK.Verify}_{\mathcal{ERDJ}_\Delta}(\mathsf{td}, x_j, \pi_j)$ *where* $x_j = (f_j, F_j)$. *It computes* $k_{D_L} \leftarrow \mathsf{Ext}(s_{D_L}, f_{D_L}^{r\prod_{i\neq L} e_{i,D_i}} \mod N^{\xi+1})$ *, and outputs* $m_{D_L} = \mathsf{ct}_{D_L} \oplus k_{D_L}$.

Analysis. We state the theorems that guarantee the required properties for our scheme.

Theorem 6. *The protocol presented in Construction 6 is correct.*

The proof of correctness essentially follows the same lines as the proof of Theorem 2.

Theorem 7. *The protocol presented in Construction 6 securely UC-realizes functionality* $\mathcal{F}_{\ell\mathsf{OT}}$ *in the* $\mathcal{G}_{\mathsf{CRS}}$*-hybrid model against:*

- *a semi-honest receiver given that the* ϕ*-hiding and the DCR assumptions hold;*
- *a malicious sender, where security holds statistically.*

Proof. The proof of security against a semi-honest receiver is identical to the proof of Lemma 20.

We now sketch how to prove security against a malicious sender. The simulator works analogously to the simulator of Lemma 21, except that, in this case, the simulator knows the prime $e_{L,i}$ for both $i \in \{0,1\}$. Thus, the re-encryption step is not needed anymore since the simulator can easily extract ρ_i, for $i \in \{0,1\}$ by decrypting (f_i, F_i) using $\phi(N)$ (which is well-formed and encrypting a value smaller than N^2 by the soundness of $\mathsf{NIZK}_{\mathcal{ERDJ}_\Delta}$) to recover a value ζ_i'. From this value, it can compute $\rho_i = \zeta_i'/(e_{L,i}\phi(N))$. After recoreving ρ_i, it can compute the keys k_i and extract the messages m_i.

Indistinguishability between the simulated version and the real protocol follows the same blueprint as the proof of Lemma 21. □

8 Self-Detecting Encryption

In this section we define self-detecting encryption, and show how to build it from laconic PSI. We first give a semi-honest definition, and will present the malicious definition in Section 8.1.

Definition 15. *A Self-Detecting Encryption (SDE) scheme is tuple of (randomized) algorithms* $\mathsf{SDE} = (\mathsf{Prm}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Detect})$ *such that:*

- $\mathsf{Prm}(1^\lambda)$*: Takes as input a security parameter* 1^λ*, and outputs a public parameter* pp.
- $\mathsf{Gen}(\mathsf{pp})$*: Takes as input a public parameter* pp*, and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$*: Takes as input a public parameter* pp *and a database* DB*, and outputs a hash value* h *and a private state* st*. We require* $|\mathsf{h}| \leq \mathsf{poly}(\lambda)$*, for a fixed polynomial* poly.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{m})$*: Takes as input a public key* pk*, a hash value* h*, and a message* m*, and outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: Takes as input a secret key* sk *and a ciphertext* ct*, and outputs a message* m *or* $\perp$.
- $\mathsf{Detect}(\mathsf{st}, \mathsf{ct})$*: Takes as input a private state* st *and a ciphertext* ct*, and outputs a message* m *or* $\perp$.

We require the following properties:

- ***Correctness.*** *For any message* m*, letting* $\mathsf{pp} \leftarrow_\$ \mathsf{Prm}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(\mathsf{pp})$*:* $\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m})) \neq \mathsf{m}] \leq \mathsf{negl}(\lambda)$.
- ***Detection.*** *For any* $\mathsf{pp} \in \mathsf{Prm}$*, any* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$*, any database of strings* DB*, and any message* m*, letting* $(\mathsf{h}, \mathsf{st}) \leftarrow_\$ \mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$ *and* $\mathsf{ct} \leftarrow_\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{m})$*, if* $\mathsf{m} \in \mathsf{DB}$ *then* $\mathsf{Detect}(\mathsf{st}, \mathsf{ct}) = \mathsf{m}$.

- ***Efficiency.*** *The size of h and running time of* Enc *are independent of the database size. There exists a polynomial* poly *s.t. for all* $n := n(\lambda)$*, any* $\mathsf{DB} \in \{0,1\}^n$*, letting* $h \leftarrow_\$ \mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$ *and* pp, pk *be as above, then* $|h| \leq \mathsf{poly}(\lambda)$ *and also the running time of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{m})$ *is upper bounded by* $\mathsf{poly}(|\mathsf{m}|, \lambda)$*.*

- ***Database Hiding.*** *For any two databases* $(\mathsf{DB}_0, \mathsf{DB}_1)$ *of equal size, if* $(\mathsf{h}_0, *) \leftarrow_\$ \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}_0)$ *and* $(\mathsf{h}_1, *) \leftarrow_\$ \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}_1)$ *then* h_0 *and* h_1 *are indistinguishable where* $\mathsf{pp} \leftarrow_\$ \mathsf{Gen}(1^\lambda)$*.*

- ***Semantic Security.*** *For any database of strings* DB *and any two messages* $(\mathsf{m}_0, \mathsf{m}_1)$*:* $(\mathsf{pk}, \mathsf{h}, \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_0)) \stackrel{c}{\equiv} (\mathsf{pk}, \mathsf{h}, \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_1))$*, where all the variables are sampled as above.*

- ***Security Against the Authority.*** *For any two messages* $(\mathsf{m}_0, \mathsf{m}_1)$*, if* $\mathsf{m}_0 \notin \mathsf{DB}$ *and* $\mathsf{m}_1 \notin \mathsf{DB}$ *then*

$$\big(\mathsf{pk}, (\mathsf{h}, \mathsf{st}), \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_0)\big) \stackrel{c}{\equiv} \big(\mathsf{pk}, (\mathsf{h}, \mathsf{st}), \mathsf{Enc}(\mathsf{pk}, \mathsf{h}, m_1)\big),$$

  where $\mathsf{pp} \leftarrow_\$ \mathsf{Prm}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(\mathsf{pp})$*, and* $(\mathsf{h}, \mathsf{st}) \leftarrow_\$ \mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$*.*

We now show how to realize self-detecting encryption from semi-honest laconic PSI. Informally, the SDE hash is the receiver's first-round laconic PSI message, and the encryption of a message m consists of a PKE encryption of m as well as a second-round PSI message based on m.

Construction 7. *Let* $\mathsf{PKE} = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be a CPA-secure PKE scheme*[15] *and* $\mathsf{LPSI} = (\mathsf{GenCRS}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *a laconic PSI.*

- $\mathsf{Prm}(1^\lambda)$*: Sample* $\mathsf{crs} \leftarrow_\$ \mathsf{LPSI.GenCRS}(1^\lambda)$*, and let* $\mathsf{pp} := \mathsf{crs}$*.*

- $\mathsf{Gen}(\mathsf{pp})$*: Run* $\mathsf{PKE.Gen}'(1^\lambda)$ *to generate a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*.*

- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$*: Let* h *be the output of the receiver on* DB *and* pp*, i.e.,* $\mathsf{h} \leftarrow_\$ \mathsf{LPSI.R}_1(\mathsf{pp}, \mathsf{DB})$*. In addition, let* st *be the private state of the receiver.*

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{m})$*: Output* $(\mathsf{ct}_1, \mathsf{ct}_2)$*, where* $\mathsf{ct}_1 \leftarrow_\$ \mathsf{PKE.Enc}'(\mathsf{pk}, \mathsf{m})$ *and* $\mathsf{ct}_2 \leftarrow_\$ \mathsf{LPSI.S}(\mathsf{pp}, \{\mathsf{m}\}, \mathsf{h})$*.*

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$*: Output* $\mathsf{PKE.Dec}'(\mathsf{sk}, \mathsf{ct}_1)$*.*

- $\mathsf{Detect}(\mathsf{st}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$*: Output* $\mathsf{R}_2(\mathsf{st}, \mathsf{ct}_2)$*.*

Correctness and efficiency follow immediately.

- **Statistical** database hiding follows from PSI-receiver statistical security.

- Semantic security and security against the authority property of the scheme follows from the CPA security of PKE scheme Π and the sender's security. Observe that if $\mathsf{m} \notin \mathsf{DB}$ then both ct_1 and ct_2 computationally hide the message even in the presence of the private state st of PSI. Specifically, one can argue that ct_1 computationally hides m because of the CPA security of PKE scheme Π, and ct_2 computationally hides m because of the sender's security of laconic PSI. The arguments above can be made formal via a routine hybrid argument, and we omit the details.

8.1 Maliciously Secure Self-Detecting Encryption

Next, we provide a definition of self-detecting encryption in the malicious setting. In this setting, the algorithm Prm provides a trapdoor which allows a server to ensure that the ciphertexts sent on the channel can be verified while ensuring privacy of users. We remark that the trapdoor is only known for the server (and is not included in users secret key).[16] Clearly, as in the semi-honest setting, the authority would only

[15] We proceed with an independent PKE scheme for the sake of simplicity.

[16] Notice that in the malicious setting, there are three entities (user, server, and the authority) with their own secret key/state.

be able to detect illegal content by looking at the ciphertexts communicated through the channel, and no information will be leaked about normal/legal messages.

Specifically, a maliciously secure self-detecting encryption is a tuple of seven (randomized) algorithms $\mathsf{SDE} = (\mathsf{Prm}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Detect}, \mathsf{Verify})$ such that Prm outputs a trapdoor td (along with pp) such that the verification algorithm Verify checks well-formedness of a ciphertext using td as follows:

- $\mathsf{Verify}(\mathsf{td}, \mathsf{ct})$: Takes a trapdoor td and a ciphertext ct and it outputs 1 or 0.

The other five algorithms, namely $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Detect})$, have the same functionality as in the semi-honest setting. We require that SDE should satisfy all the properties of a semi-honest encryption scheme (correctness, efficiency, database hiding, semantic security, and security against the authority), along with the following well-formedness property: for any PPT adversary $\mathcal{A}$ and database DB, if $(\mathsf{pp}, \mathsf{td}) \leftarrow_{\$} \mathsf{Prm}(1^\lambda)$, $(\mathsf{pk}, *) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$, $(\mathsf{h}, *) \leftarrow_{\$} \mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$ then the following holds for any adverserially generated ciphertext $\mathsf{ct} \leftarrow_{\$} \mathcal{A}^{\mathsf{Verify}(\mathsf{td},.)}(\mathsf{pp}, \mathsf{pk}, \mathsf{h})$ with overwhelming probability (where $\mathcal{A}$ has oracle access to the verification algorithm):

- If $\mathsf{Verify}(\mathsf{td}, \mathsf{ct}) = 1$ and $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \in \mathsf{DB}$ then $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{Detect}(\mathsf{st}, \mathsf{ct})$.
- If $\mathsf{Verify}(\mathsf{td}, \mathsf{ct}) = 1$ and $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \notin \mathsf{DB}$ then $\mathsf{Detect}(\mathsf{st}, \mathsf{ct}) = \perp$.

Given a maliciously secure laconic PSI and a DV-NIZK for a specific language, one can construct a maliciously secure SDE following the same blueprint that we provided in the semi-honest setting.

Construction 8. *Let* $\mathsf{PKE} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *be a CPA-secure public-key encryption scheme, and let* $\mathsf{NIZK} = (\mathsf{NIZK.GenCRS}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ *be a DV-NIZK for "message-equality" language (described below).*

- $\mathsf{Prm}(1^\lambda)$*: Sample* $(\mathsf{crs}_N, \mathsf{td}) \leftarrow_{\$} \mathsf{NIZK.GenCRS}(1^\lambda)$ *and* $\mathsf{crs}_L \leftarrow_{\$} \mathsf{LPSI.GenCRS}(1^\lambda)$*, and let* $\mathsf{pp} = (\mathsf{crs}_N, \mathsf{crs}_L)$*.*
- $\mathsf{Gen}(\mathsf{pp})$*: Sample a pair of keys* $(\mathsf{pk}', \mathsf{sk}') \leftarrow_{\$} \mathsf{PKE.Gen}'(1^\lambda)$*. Set* $\mathsf{pk} = (\mathsf{pk}', \mathsf{crs}_N, \mathsf{crs}_L)$ *and* $\mathsf{sk} = \mathsf{sk}'$*.*
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$*: Parse* $\mathsf{pp} = (\mathsf{crs}_N, \mathsf{crs}_L)$*. Output* $(\mathsf{h}, \mathsf{st}) \leftarrow_{\$} \mathsf{LPSI.R}_1(\mathsf{crs}_L, \mathsf{DB})$*.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{h}, \mathsf{m})$*: Parse* $\mathsf{pk} = (\mathsf{pk}', \mathsf{crs}_L, \mathsf{crs})$*. Let* $\mathsf{ct}_1 \leftarrow_{\$} \mathsf{PKE.Enc}'(\mathsf{pk}', \mathsf{m})$ *and* $\mathsf{ct}_2 \leftarrow_{\$} \mathsf{LPSI.S}(\mathsf{crs}_L, \{\mathsf{m}\}, \mathsf{h})$*. Compute a proof for the statement that the messages underlying* ct_1 *and* ct_2 *are equal. Specifically, consider the following language* $\mathcal{L}$ *(parameterized by* Δ*):*

  $$\mathcal{L}_\Delta = \{(\mathsf{ct}_1, \mathsf{ct}_2) : \exists(\mathsf{m}, r, r') \text{ s.t. } \mathsf{ct}_1 = \mathsf{PKE.Enc}'(\mathsf{pk}', \mathsf{m}; r') \wedge \mathsf{ct}_2 = \mathsf{LPSI.S}(\mathsf{crs}_L, \{\mathsf{m}\}, \mathsf{h}; r)\},$$

  where $\Delta = (\mathsf{pk}', \mathsf{crs}_L, \mathsf{h})$*. In addition,* r *and* r' *are the random coins used by* $\mathsf{PKE.Enc}'$ *and* $\mathsf{LPSI.S}$*, respectively. Generate a proof* $\pi \leftarrow_{\$} \mathsf{NIZK.Verify}(\mathsf{crs}_N, \mathsf{ct}_1, \mathsf{ct}_2)$*, and set* $\mathsf{ct}_3 := \pi$*. Finally, publish* $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$ *as the ciphertext.*
- $\mathsf{Verify}(\mathsf{td}, \mathsf{ct})$ *: Run* $\mathsf{NIZK.Verify}$ *on* td *and* ct*, and output the resulting bit.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3))$*: Output* $\mathsf{PKE.Dec}'(\mathsf{sk}', \mathsf{ct}_1)$*.*
- $\mathsf{Detect}(\mathsf{st}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3))$*: Output* $\mathsf{LPSI.R}_2(\mathsf{st}, \mathsf{ct}_2)$*.*

Correctness, efficiency, database hiding, semantic security, and security against the authority of the scheme can be argued in a similar fashion to the semi-honest setting. The additional requirement, namely well-formedness property of the scheme essentially follows from the security of DV-NIZK. Observe that for a maliciously generated ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$, the messages hidden by ct_1 and ct_2 are not equal, and hence the ciphertext ct will be rejected by the verification algorithm of DV-NIZK. We leave a black-box construction of DV-NIZK (for the message-equality language above) from concrete cryptographic assumptions to future work.

Acknowledgment

Pedro Branco thanks the support from DP-PMI and FCT (Portugal) through the grant PD/BD/135181/2017. This work is supported by Security and Quantum Information Group of Instituto de Telecomunicações, by the Fundação para a Ciência e a Tecnologia (FCT) through national funds, by FEDER, COMPETE 2020, and by Regional Operational Program of Lisbon, under UIDB/50008/2020.

Sanjam Garg is supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

Mohammad Hajiabadi is supported in part by NSF CNS Award 2055564.

References

[ADT11] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) size matters: Size-hiding private set intersection. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 156–173, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany. 3, 4, 6

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 120–129, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. 10

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press. 3, 13

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 669–699, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 3, 13

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 14

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 7, 29

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany. 3, 4, 8, 14, 21

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors,

Advances in Cryptology – EUROCRYPT 2018, Part II, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 2

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 2, 4, 5, 15, 16

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 15

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 1, 5

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1223–1237, Toronto, ON, Canada, October 15–19, 2018. ACM Press. 3, 4

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1243–1255, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 3

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. 8, 9, 21, 22, 23

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 2, 5

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 2, 4, 5, 15

[DGGM19] Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In David Zuckerman, editor, *60th Annual Symposium on Foundations of Computer Science*, pages 661–685, Baltimore, MD, USA, November 9–12, 2019. IEEE Computer Society Press. 1

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 3–31, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. 2

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 1

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany. 3, 4, 7, 13, 21

[FT14] Pierre-Alain Fouque and Mehdi Tibouchi. Close to uniform prime number generation with fewer random bits. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014: 41st International Colloquium on Automata, Languages and Programming, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 991–1002, Copenhagen, Denmark, July 8–11, 2014. Springer, Heidelberg, Germany. 12

[GGH19] Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 33–63, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 16

[GHM+19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 63–93, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. 2

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 689–718, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 2, 4, 5

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 3, 4, 8, 9, 14, 21, 23

[Gre19] Matthew Green, 2019. https://blog.cryptographyengineering.com/2019/12/08/on-client-side-media-scanning/. 3

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 4

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 588–599, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press. 2

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 2

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, Lecture Notes in Computer Science, pages 621–651, Santa Barbara, CA, USA, August 16–20, 2020. Springer, Heidelberg, Germany. 2

[GVW20] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. New constructions of hinting PRGs, OWFs with encryption, and more. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, Lecture Notes in Computer Science, pages 527–558, Santa Barbara, CA, USA, August 16–20, 2020. Springer, Heidelberg, Germany. 6, 13, 35

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 175–203, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany. 2

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, January 11–13, 2015. Association for Computing Machinery. 4

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. 4

[JL10] Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 418–435, Amalfi, Italy, September 13–15, 2010. Springer, Heidelberg, Germany. 3

[KMP+17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1257–1272, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 2, 11, 12

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. 2

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 101–128, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 21

[LNO13] Yehuda Lindell, Kobbi Nissim, and Claudio Orlandi. Hiding the input-size in secure two-party computation. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 421–440, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 4

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in*

Cryptology – EUROCRYPT 2018, Part III, volume 10822 of *Lecture Notes in Computer Science*, pages 125–157, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 2

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 797–812, San Diego, CA, USA, August 20–22, 2014. USENIX Association. 2

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 859–870, Paris, France, October 7–9, 2018. IEEE Computer Society Press. 1, 2, 15

[RA18] Amanda C. Davi Resende and Diego F. Aranha. Faster unbalanced private set intersection. In Sarah Meiklejohn and Kazue Sako, editors, *FC 2018: 22nd International Conference on Financial Cryptography and Data Security*, volume 10957 of *Lecture Notes in Computer Science*, pages 203–221, Nieuwpoort, Curaçao, February 26 – March 2, 2018. Springer, Heidelberg, Germany. 3

[RR17] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1229–1242, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 2

[TBM+20] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In *ACM CCS 20: 27th Conference on Computer and Communications Security*, pages 1733–1750. ACM Press, 2020. 3, 13