Witness Pseudorandom Functions for Vector Commitments and Applications

Rishabh Bhadauria
Pedro Branco, IE University
Nico Döttling, CISPA
Sanjam Garg, UC Berkeley
Guru-Vamsi Policharla, Commonware

Abstract

A witness pseudorandom function (WPRF) is a PRF which has an additional mode of public evaluation. Given the public key, it can be evaluated publicly if one provides a valid NP witness for that input, while the output remains pseudorandom to anyone without such a witness. WPRFs are powerful objects, and general-purpose constructions are currently only known from assumptions that imply indistinguishability obfuscation.

In this work, we construct a WPRF for a specific language related to the Libert-Yung vector commitment (TCC 2010). More specifically, public evaluation of the WPRF on an input is possible if a valid local opening for that input is provided. Our construction relies only on standard assumptions on pairing groups and is fully black-box.

We further show that this primitive enables us to solve several open problems in the study of communication-efficient secure computation:

- **Rate-1 Laconic Oblivious Transfer.** We construct a laconic oblivious transfer protocol with total communication complexity $2k + \mathsf{poly}(\lambda)$ for k executions. Previously, achieving this efficiency required non-falsifiable assumptions such as evasive LWE (Wee, CRYPTO'24).
- **Near-optimal Laconic Private Set Intersection.** We construct laconic private set intersection for which the amortized communication complexity approaches λ bits per element in the sender's set (assuming each set element is represented by λ bits) and is independent of the receiver's set. This is within a constant factor of the information-theoretic lower bound.
- **Rate-1 Batch Registration-Based Encryption.** We construct a registration-based encryption scheme in which, when the encryptor sends multiple messages to multiple receivers, the ciphertext overhead is only two group elements.

1 Introduction

Pseudorandom functions are ubiquitous objects in cryptography. They mimic a random oracle by producing outputs that appear random to any efficient adversary without the secret key. Witness pseudorandom functions (WPRF), introduced by Zhandry [Zha16], are a flavor of PRFs in which public evaluation of the PRF is possible if one has a valid witness for some NP statement. In more detail, in a WPRF for an NP relation $\mathcal{R}$, a PRF key K is generated together with a public key pk. Then, evaluation of the PRF can be done in two different ways via a private algorithm $\mathsf{EvalStmt}$ and a public one $\mathsf{EvalWit}$.

- $\mathsf{EvalStmt}$: given a private key K, one can compute the PRF output $F(K, x)$ on any input x.
- $\mathsf{EvalWit}$: given the public key pk, one can compute the PRF output $F(K, x)$ if a valid witness w is also provided such that $\mathcal{R}(w, x) = 1$.

In terms of security, we want pseudorandomness to hold for false statements.

Witness pseudorandom functions (WPRFs) are powerful cryptographic objects. They appear to be strictly stronger than witness encryption (WE) [GGSW13], and can be used to construct advanced primitives such as non-interactive key exchange [Zha16]. At present, however, general-purpose constructions of WPRFs or WE are only known under assumptions that imply indistinguishability obfuscation (iO) [BGI+01, GGH+13]. There have been attempts to build WE from new non-standard lattice-based assumptions [Tsa22, VWW22, BDJ+25]; however, at the moment, there is evidence that these assumptions might be insecure [BÜW25, DJM+25, AMYY25, HJL25].

For *special-purpose languages*, the situation is more encouraging. In the case of WE, several restricted instantiations have been achieved, leading to important applications. Examples include constructions of hash proof systems (HPS) [CS02], which led to CCA-secure encryption; identity-based encryption [BF01]; and hash with encryption [CDG+17, DG17], which led to applications in laconic cryptography.

In this work, we push forward the study of WPRFs for special-purpose languages related to vector commitments that can be constructed from standard cryptographic assumptions, i.e., without relying on iO, and explore their applications.

Vector commitments. The focus of this work is to build WPRFs for languages related to *vector commitments*. A vector commitment [CF13] is a special type of commitment scheme with the following properties:

- It allows to hash a large database $\mathsf{DB} \in \{0,1\}^N$ using a hashing key hk into a small digest $h \in \{0,1\}^\lambda$ with size independent of N.
- It allows to reveal a single position $i \in [N]$ of DB by revealing a *short* proof π_i, that is, we require that $|\pi_i| = \mathsf{poly}(\lambda, \log N)$.
- It should provide the usual binding and hiding properties of a commitment scheme.

Vector commitments have been instrumental in building more complex cryptographic primitives with succinct communication. At the moment, we know how to build them using collision-resistant hash functions via Merkle trees, but we also know algebraic constructions for them from, e.g., factoring-based assumptions [CF13], pairings [LY10, KZG10, CF13] or lattice-based assumptions [PPS21, WW23, dCP23].

1.1 Witness PRF for Vector Commitments

In this work, we focus on a specific pairing-based vector commitment, namely the Libert-Yung commitment scheme [LY10], and show how to build a WPRF for it. Specifically, the relation $\mathcal{R}_{\mathsf{hk},h}$ for which we build the WPRF is parameterized by a hashing key hk and a digest h, and checks the following:

You can evaluate the PRF on the statement (y, i)
iff
there is a valid local opening π_i *for* y *on position* i *w.r.t. the digest* h.

A savvy reader might notice that, as described, this language does not contain false statements: since h is compressing, there will always exist a valid local opening for any tuple (y, i). In other words, because there are no false statements, we cannot guarantee pseudorandomness of the WPRF on any input. One possible resolution is to use a *somewhere statistically binding* hash function [HW15], which guarantees that for at least one index (or a small set of indices), the statement can be made false. In this case, we could hope to guarantee pseudorandomness only with respect to these statistically binding indices. However, if we want to guarantee pseudorandomness for a large set of indices, then we would lose compactness of our construction.

Instead, we take a different approach. The crucial point is that, if we assume the digest h is well-formed and computed from a database DB, any valid local opening for $(1 - \mathsf{DB}_i)$ is *computationally hard* to find as the underlying commitment scheme is computationally binding. Hence, we define security with respect to

the *honest* database DB. Specifically, assume that h is the digest of DB with respect to the hashing key hk, and that π_i is a valid local opening for DB_i. Then

$$\mathsf{EvalStmt}_{\mathsf{hk},h}(K,(\mathsf{DB}_i,i)) = \mathsf{EvalWit}_{\mathsf{hk},h}(\mathsf{pk},(\mathsf{DB}_i,i,\pi_i))$$

whereas $\mathsf{EvalStmt}_{\mathsf{hk},h}(K,(1-\mathsf{DB}_i,i))$ is pseudorandom for anyone not holding the private key K. Crucially, we want this pseudorandomness property to hold for all indices $i \in [N]$ *and* across different digests h computed from different databases DB.[1]

Our construction achieves the security described above assuming standard assumptions on pairing groups, namely the bilinear Diffie-Hellman in the exponent (BDHE) assumption. Moreover, our scheme is compact, as both the hash value and the public key consist of just one group element. For practical concrete efficiency, we rely on the random-oracle-based instantiation of the PRF layer discussed below, in which case we can use the same pairing groups as similar prior schemes [GKMR23]; the standard-model RKA-secure instantiation has different concrete tradeoffs.

1.2 Applications

We then show that this new primitive can be used to provide conceptually simple solutions to the following open problems:

1. **Laconic OT.** A rate-1 laconic oblivious transfer protocol, for which the total communication complexity is $2k + \mathsf{poly}(\lambda)$ for k simultaneous laconic OT instances. We additionally show that rate-1 laconic OT can be used to construct an optimal-rate malicious OT in two rounds [BDS24].

2. **Laconic PSI.** A near-optimal laconic private set intersection for a universe $\mathcal{U} = \{0,1\}^\lambda$ with correctness error $2^{-\Omega(\lambda)}$, where the total communication complexity scales only $|Y| \cdot \lambda + \mathsf{poly}(\lambda)$ where Y is the sender's input set.

3. **RBE.** A batch rate-1 registration-based encryption scheme in which, when the encryptor sends multiple messages to multiple receivers, the ciphertext overhead is only two group elements.

We now discuss these applications in more detail.

1.2.1 Rate-1 Laconic Oblivious Transfer

Laconic oblivious transfer (OT) [CDG+17] is a flavor of OT where the receiver starts by hashing a large database DB into a short hash value h (typically independent of the size of the database). Then a sender, with input two bit messages m_0, m_1 and a location i, can send a short message ct to the receiver that allows it to retrieve m_{DB_i}.

Since its introduction, laconic OT has turned out to be a powerful tool and found numerous applications such as in non-interactive secure computation on large inputs [CDG+17], laconic function evaluation of general functions [QWW18], garbled RAM [GOS18] or garbled circuits [GS18].

One important efficiency measure for OT is its communication complexity. This question has been the focus of many recent works [BDGM19, GH19, BBDP22, BDS23, BDS24]; however, all of these works focused on standard OT. For the case of laconic OT, all schemes [CDG+17, DGHM18, DKL+23] require the sender to send a message of size $\mathsf{poly}(\lambda)$, even if its input messages m_0, m_1 are just bits, which leads to poor overall rate. In the batch setting (where the sender inputs several pairs of bit messages and locations), this leads to a sender message of size $k \cdot \mathsf{poly}(\lambda)$ where k is the batch size. The only exception are the recent works of Wee [Wee24] and Abram et al. [AMR25] which present a laconic OT scheme with optimal rate from, respectively, a new non-falsifiable assumption called evasive learning with errors, and the LWE assumption.

[1] In other words, we want the public key pk to be reusable across different databases.

Rate-1 Batch Laconic OT. We show how we can use our new WPRF scheme to build a new laconic bit-OT scheme. The scheme has the following properties.

- In the batch setting, the scheme has an overall optimal rate. That is, the overall communication of the scheme is $2k + \mathsf{poly}(\lambda)$ for a batch of size k. This matches the information-theoretic lower-bound.
- Security relies on the BDHE assumption.

Our laconic OT scheme is the first one to achieve an optimal rate from standard hardness assumptions over groups.

Discussion on rate-1 OT. Our scheme achieves the same communication complexity for OT as recent works [BDGM19, GH19, BBDP22, BDS23]. However, our scheme differs significantly from the approach taken in [BDGM19, GH19, BBDP22, BDS23]. Note that if we think of the insecure OT protocol, we have two options to implement the functionality:

1. The receiver sends a bit b to the sender and gets back m_b. In this case, we have no receiver security.
2. The sender sends both bit messages m_0, m_1. In this case, we have no sender security.

In both cases, we have 2 bits of communication being exchanged. The works of [BDGM19, GH19, BBDP22, BDS23] achieve an optimal-rate OT scheme via the first approach whereas, in this work, we follow the second approach.

Moreover, these schemes achieve communication complexity $2k + o(k) \cdot \mathsf{poly}(\lambda)$. That is, the communication overhead in these protocols actually grows sublinear with k. This limits the practicality of these protocols, as they need to batch a significant number of executions in order for the amortization to kick in. On the other hand, the communication overhead in our scheme is constant (in fact, it is just a single group element).

In terms of assumptions, the schemes of [BDGM19, GH19] rely on the learning with errors (LWE) assumption, whereas [BBDP22, BDS23] rely on a combination of decisional Diffie-Hellman (DDH) and learning parity with noise (LPN). Our protocol relies on standard assumptions on bilinear groups.

Two-round malicious OT with optimal rate. We additionally show how to build two-round malicious OT with optimal rate from rate-1 batch LOT together with laconic CDS [DGGM19] and non-interactive zero-knowledge (NIZKs) proofs [GOS06]. All of these primitives can be constructed from standard pairing assumptions. Recall that, for the case of malicious OT, the lower bound on the communication complexity is actually 3 bits per OT (instead of 2 bits as in the semi-honest setting) [BDS24]. Previously, two-round malicious OT with optimal rate was known to be constructed from either LWE with super-polynomial modulus to noise ratio [DHRW16], group based assumptions (namely DCR and QR) with a large CRS (that is, a CRS which grows with the batch size) that is non-reusable [OSY21], LPN assumption in the random oracle model [BCG+19], or a combination of QR and LPN [BDS24].

1.2.2 Near-Optimal Laconic Private Set Intersection

Laconic private set intersection (PSI) [ABD+21] is a flavor of PSI where the receiver starts by hashing a large set X into a small digest h. Then a sender, with input a set element y, can send a short message to the receiver that allows it to learn y if and only if $y \in X$. Otherwise, if $y \notin X$, then the receiver learns nothing about the set element y. An important feature of laconic PSI is that the total communication complexity of the protocol does not depend on the size of the receiver's set X.

At the moment, there are constructions of laconic PSI from a range of hardness assumptions such as LWE [ABD+21, DKL+23], pairings [ALOS22, GJVL23], DCR, or CDH [ABD+21]. However, all of these schemes introduce a $\mathsf{poly}(\lambda)$ overhead in the size of the sender's message compared to a near communication-optimal insecure protocol.

Communication complexity of set intersection. At this point, let's pause and consider the best possible communication complexity for set intersection.

If we require perfect correctness, the sender must transmit its entire set Y to the receiver.[2] On the other hand, if we are willing to accept computational correctness, the sender can instead send a hash of each element in its set using a suitable statistical or cryptographic hash function. Hence, for a universe $\mathcal{U}$ we can reduce the communication complexity of set intersection by a factor of $\log(1/\varepsilon)/\log(|\mathcal{U}|)$, where ε is the correctness-error we are willing to tolerate [KS92]. Consequently, in PSI the optimal achievable communication complexity is always a function of the correctness-error we are willing to tolerate and unlike for laconic OT there is no single *best-possible* communication complexity.

Near-Optimal laconic PSI. We provide a conceptually simple construction of laconic PSI from our WPRF. Assuming that Y is the sender's input set and each set element is represented by λ bits, and (say) we are willing to accept a correctness error of at most $2^{-\Omega(\lambda)}$, our scheme has an amortized communication complexity approaching λ bits per element in the sender's set. In particular, this is only a constant factor away from the (concretely unknown) best-possible communication complexity of any insecure set intersection protocol which is known to be of order $O(|Y| \cdot \log(1/\varepsilon))$ [KS92].

1.2.3 Batch Rate-1 RBE

The notion of *Registration-Based Encryption* (RBE) [GHMR18, GHM+19] was recently introduced to address one of the main concerns surrounding the notion of identity-based encryption (IBE) [BF01]: The key-escrow problem. We show how to extend our WPRF so that witness evaluation requires not only a valid opening for a specific index and value, but also the corresponding user's secret key. Unlike our main WPRF construction, this RBE application does not use the target-group related-key-secure PRF layer needed for batch pseudorandomness and instead relies only on the underlying bilinear opening mechanism together with a direct BDHE/BDH security argument. This in turn implies an RBE scheme which achieves rate 1 in the batch setting. Specifically, our scheme has the following properties:

- In the batch setting, that is, when the encryptor encrypts several messages to several different receivers, the ciphertext overhead is simply two group elements.
- The size of the public parameters scales linearly with the number of users ℓ.
- Security relies on both the asymmetric bilinear Diffie-Hellman (BDH) and the bilinear Diffie-Hellman in the exponent (BDHE) assumptions.
- It is completely black-box.

A use case for RBE in the batch setting. Consider a streaming service that delivers movies to users. Each user typically watches a different movie, so this is a classic example of batch encryption, where the service encrypts and sends distinct content to multiple users. In this case, each user receives a personalized stream (movie), requiring the encryption of different data for each recipient.[3]

Using standard RBE, the streaming service would need to generate and send k separate ciphertexts to k different users. This means the amount of information the service has to transmit grows proportionally to $k \cdot \mathsf{poly}(\lambda)$. As k increases, this quickly becomes inefficient, especially for large-scale services with millions of users. In turn, using our scheme, we can achieve a communication complexity which grows proportionally to $k + \mathsf{poly}(\lambda)$, which is asymptotically close to optimal.

[2] An alternative approach is for the receiver to send its set to the sender and receive the intersection in return. However, we assume that the receiver's set is significantly larger than the sender's, making this approach inefficient.

[3] This scenario contrasts with broadcast encryption [BGW05], where the same content is sent to all users.

1.3 Additional Related Work

Several works have built WE for special-purpose languages in a variety of different settings. These include classical examples such as hash proof systems [CS02], identity-based encryption [BF01], or WE with applications to multiparty computation [GS17, BL20]. More recently, and closely related to our work, several papers have studied succinctness for WE, measured in terms of the witness size, the statement size, or both, either explicitly [CFK24, FHAS24, ADM+24, GHK+25], or implicitly [CDG+17, DG17, GKMR23, DKL+23, KMW23, GKPW24, CGPP24, CGPW25, BFOQ25, AFP25]. On the application side, Fiore et al.'s *Cuckoo Commitments* [FKdP23] use vector commitments to construct registration-based encryption and key-value map commitments for large spaces.

We stress that our main contribution is not a new algebraic local-opening trick for Libert-Yung commitments. Rather, we show how to upgrade this local-opening relation into a reusable WPRF. This requires proving batch pseudorandomness across many indices and across different digests under a reusable public key, which in our construction is achieved by the additional PRF layer. This focus distinguishes us from the cited WE works, and also from *Cuckoo Commitments*, which uses vector commitments for RBE and key-value map commitments but does not address witness PRFs or batch pseudorandomness. In general, witness PRFs are a stronger primitive than witness encryption, as WE can be obtained directly from a witness PRF (but the converse is not known). In particular, our construction also yields witness encryption for the same family of languages, in fact rate-1 batch witness encryption. Consequently, the (asymptotic) communication complexities achieved in our applications are not known to follow from any existing WE constructions for similar languages. Moreover, WE only requires indistinguishability of encryptions, whereas our setting requires pseudorandomness for a batch of evaluations. Addressing this stronger requirement is precisely why we need new techniques.

2 Technical Outline

Throughout this overview, let $(\mathbb{G}, \mathbb{G}_t, e)$ be a bilinear group of prime order p with a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$. Further, let g be a generator of $\mathbb{G}$. Denote by $[n]$ the set $\{1, \ldots, n\}$.

2.1 Vector Commitments from Pairings.

Our starting point is the vector commitment scheme of Libert and Yung [LY10], which has emerged as a profoundly useful tool in both laconic and registration-based cryptography (e.g., [GKMR23, GJVL23]). This scheme is defined over the symmetric pairing groups. The public parameters pp of this scheme are composed of elements

$$g, g^{\alpha}, \ldots, g^{\alpha^{\ell}}, \textvisiblespace, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell}}.$$

Here we use the symbol $\textvisiblespace$ to emphasize that the $(\ell+1)$-st element in this list, namely $g^{\alpha^{\ell+1}}$, has been omitted.

A commitment to a vector $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_p^\ell$ is computed as

$$h = \prod_{j \in [\ell]} \left(g^{\alpha^j}\right)^{x_j}.$$

To open a position $i \in [\ell]$ of such a commitment, we compute

$$\pi_i = \prod_{j \neq i} \left(g^{\alpha^{\ell+1-i+j}}\right)^{x_j}.$$

We can use π_i to certify that h opens to x_i at position i. For this purpose, the verification equation checks whether

$$e\left(h, g^{\alpha^{\ell+1-i}}\right) = e\left(\pi_i, g\right) \cdot e\left(\left(g^{\alpha^i}\right)^{x_i}, g^{\alpha^{\ell+1-i}}\right). \tag{2.1}$$

The scheme is binding, as a prover capable of providing convincing openings for position i to two different values $x_i \neq x_i'$ can be used to break the ℓ-Diffie-Hellman in the Exponent (ℓ-DHE) problem in $\mathbb{G}$, i.e., it can be used to find the missing group element $g^{\alpha^{\ell+1}}$. [4]

The verification equation (2.1) illustrates the most powerful feature of this commitment scheme: We can shift a *challenge position* i into the *hole* $\ell+1$ by pairing h with $g^{\alpha^{\ell+1-i}}$, that is, it holds that

$$e\left(h, g^{\alpha^{\ell+1-i}}\right) = \prod_{j\in[\ell]} e\left(\left(g^{\alpha^j}\right)^{x_j}, g^{\alpha^{\ell+1-i}}\right) = \prod_{j\in[\ell]} e\left(g,g\right)^{x_j\alpha^{\ell+1+j-i}}$$
$$= \left(\prod_{j\in[\ell]\setminus\{i\}} e(g,g)^{x_j\alpha^{\ell+1+j-i}}\right) \cdot e(g,g)^{x_i\alpha^{\ell+1}}.$$

2.2 Witness PRF for LY Commitments from Pairings

We first introduce the syntax of WPRF for LY commitments. In a WPRF for LY commitments, we have a Setup algorithm that outputs the hash key hk. Then there is a tuple of algorithms Gen, Hash, $\mathsf{EvalWit}$, $\mathsf{EvalStmt}$ such that

- The Gen algorithm takes public parameters and produces a public key pk and a trapdoor td.
- The Hash algorithm takes a large input vector $\boldsymbol{x} \in \Sigma^N$.
- The public $\mathsf{EvalWit}$ algorithm takes the large input vector $\boldsymbol{x}$, the public key pk and an index $i \in [N]$, and outputs a value e_i.
- The private $\mathsf{EvalStmt}$ algorithm takes as input a trapdoor td, a hash value h, an index i and a value $y \in \Sigma$ and outputs a value $d_{i,y}$.

In terms of correctness, we require that if $x_i = y$ and we generate the values $d_{i,y}$ and e_i as described above, then it holds that $e_i = d_{i,y}$. On the other hand, in terms of security, we require that if $y \neq x_i$, then the value $d_{i,y}$ is uniform in the view of a receiver who learns the public key pk. We require security to hold for a *batch* of values $\{d_{i,y_i}\}_{i\in S}$ for a set $S \subseteq [N]$ where all $y_i \neq x_i$, not only one. In other words, we require pk to be reusable.

Basic Scheme We will now describe our basic WPRF scheme. For this instantiation it holds that $\Sigma \subseteq \mathbb{Z}_p$. Let

$$\mathsf{pp} = \left(g, g^{\alpha}, \ldots, g^{\alpha^{\ell}}, _, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell}}\right)$$

be the public parameters for the LY vector commitment. As before, we hash a database $\boldsymbol{x} \in \{0,1\}^{\ell}$ by computing

$$h = \prod_{j\in[\ell]} \left(g^{\alpha^j}\right)^{\boldsymbol{x}_j}$$

and we generate openings π_i via

$$\pi_i = \prod_{j\neq i} \left(g^{\alpha^{\ell+1-i+j}}\right)^{\boldsymbol{x}_j}.$$

The Gen, $\mathsf{EvalStmt}$ and $\mathsf{EvalWit}$ algorithms are given as follows.

- Gen takes the public parameters, chooses $\xi \leftarrow_{\$} \mathbb{Z}_p$ uniformly at random, and outputs $\mathsf{pk} = g^{\xi}$ and $\mathsf{td} = \xi$.

[4] If π_i is a valid opening of h to x_i at position i and π_i' is a valid opening to x_i', it must hold that $\pi_i \cdot \left(g^{\alpha^{\ell+1}}\right)^{x_i} = \pi_i' \cdot \left(g^{\alpha^{\ell+1}}\right)^{x_i'}$ as the pairing e is non-degenerate. But this means we can efficiently compute $g^{\alpha^{\ell+1}} = \left(\pi_i' \cdot \pi_i^{-1}\right)^{1/(x_i - x_i')}$.

- $\mathsf{EvalStmt}$ takes as input a trapdoor $\mathsf{td} = \xi$, a hash value h, and a pair (i, y) consisting of an index i and a value $y \in \Sigma$. It computes and outputs $d_{i,y} = e\left(h, g^{\alpha^{\ell+1-i}}\right)^{\xi} \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{-y\xi}$.
- $\mathsf{EvalWit}$ takes $\mathsf{pk} = g^{\xi}$, $\boldsymbol{x}$ and i. It computes and outputs $e_i = e(\pi_i, \mathsf{pk})$.

Correctness can be established via the verification equation (2.1). If $\boldsymbol{x}_i = y$ it holds that

$$\begin{aligned} d_{i,y} &= e\left(h, g^{\alpha^{\ell+1-i}}\right)^{\xi} \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{-y\xi} \\ &= \left(e(\pi_i, g) \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{\boldsymbol{x}_i}\right)^{\xi} \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{-y\xi} \\ &= e\left(\pi_i, g^{\xi}\right) \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{(\boldsymbol{x}_i - y)\xi} \\ &= e_i \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{(\boldsymbol{x}_i - y)\xi}. \end{aligned}$$

that is, given that $\boldsymbol{x}_i = y$, we have $d_{i,y} = e_i$. Hence the scheme is correct.

Security To argue the security of this scheme, we can appeal to the ℓ-Bilinear Diffie Hellman in the Exponent (ℓ-BDHE) assumption and the Bilinear Diffie Hellman (BDH) assumption. The ℓ-BDHE assumption states that $e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{\xi}$ is computationally indistinguishable from a uniformly random group element in $\mathbb{G}_t$ given $g, g^{\alpha}, \ldots, g^{\alpha^{\ell}}, _, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell}}$ as well as g^{ξ}.

Using the relation $d_{i,y} = e_i \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{(\boldsymbol{x}_i - y)\xi}$, we can also routinely establish security of our scheme *for a single index* $i \in [\ell]$ via the ℓ-BDHE assumption. Concretely, assume that $y \neq \boldsymbol{x}_i$. Given this relation we can simulate $d_{i,y}$ via

$$\begin{aligned} d_{i,y} &= e_i \cdot e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{(\boldsymbol{x}_i - y)\xi} \\ &= e\left(\pi_i, g^{\xi}\right) \cdot \left(e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{\xi}\right)^{\boldsymbol{x}_i - y}. \end{aligned}$$

Consequently, we can embed a given ℓ-BDHE challenge $g, g^{\alpha}, \ldots, g^{\alpha^{\ell}}, _, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell}}, g^{\xi}, z$ where either $z = e(g^{\alpha}, g^{\alpha^{\ell}})^{\xi}$ or $z \leftarrow_{\$} \mathbb{G}_t$ as follows. Again we use the g^{α^j} to define the public parameters pp, and given pp, we compute h and the π_i as in the real scheme. We set the public key to $\mathsf{pk} = g^{\xi}$ and we set $d_{i,y}$ to

$$d_{i,y} = e\left(\pi_i, g^{\xi}\right) \cdot z^{\boldsymbol{x}_i - y}.$$

Clearly, if $z = e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{\xi}$, then as discussed above, we simulate the real security experiment perfectly. On the other hand, if z is chosen uniformly at random, then $d_{i,y}$ is uniform in the view of the semi-honest receiver as $\boldsymbol{x}_i - y \neq 0$.

Batch Security We have thus far only considered security for a single index $i \in [\ell]$. However, our goal is to achieve batch security, i.e., we want to argue that all $d_{1,1-\boldsymbol{x}_1}, \ldots, d_{\ell,1-\boldsymbol{x}_\ell}$ are indistinguishable from independently chosen uniform values. A moment of reflection reveals that our basic scheme above does not satisfy this stronger security notion: Whenever $\boldsymbol{x}_i = \boldsymbol{x}_{i'}$ for $i' \neq i$, it holds that

$$d_{i,1-\boldsymbol{x}_i} \cdot d_{i',1-\boldsymbol{x}_{i'}} = e(\pi_i, \mathsf{pk}) \cdot e(\pi_{i'}, \mathsf{pk}),$$

where the term on the right-hand side is efficiently computable by the receiver. In other words the values $d_{i,1-\boldsymbol{x}_i}$ and $d_{i',1-\boldsymbol{x}_{i'}}$ are correlated. More precisely, the different $d_{i,1-\boldsymbol{x}_i}$ are correlated via the value

$e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{\xi}$ as

$$d_{i,1-\boldsymbol{x}_i} = e_i \cdot \left(e\left(g^{\alpha}, g^{\alpha^{\ell}}\right)^{\xi}\right)^{2\boldsymbol{x}_i - 1}. \tag{2.2}$$

Hence, we need an additional mechanism to break such correlations.

Related-Key Security. Relations such as (2.2) are known as *linear group-induced relations*, and there is a body of work (e.g. [BK03, Luc04, BC10]) studying pseudorandom functions which remain pseudorandom even if the adversary is given access to an augmented oracle which allows him to evaluate the PRF on related keys. Specifically, assume we have a PRF whose keys k are supported on a group $\mathbb{G}$. Then, we allow the adversary to query his oracle with (f, s, x), where $f \in \mathbb{G}$, $s \in \mathbb{Z}_p \setminus \{0\}$ and x is in the domain of the PRF. The real oracle returns $\mathsf{PRF}_{f \cdot k^s}(x)$, i.e., an evaluation of PRF on the related key $f \cdot k^s$. The ideal oracle, on the other hand, responds with $R(f, s, x)$, where R is a truly random function. We say a PRF is secure against related key attacks, or RKA secure if no PPT-adversary can distinguish these two oracles. As discussed in e.g. [BK03], we can realize RKA secure PRFs in the random oracle model.

Now, equipped with an RKA secure PRF whose keys are supported on the group $\mathbb{G}_t$, we can make a final modification to our WPRF scheme which allows us to prove batch security. Specifically, we will use the $d_{i,y}$ and e'_i as keys for PRF to derive our final $d'_{i,y}$ and e'_i

$$d'_{i,y} = \mathsf{PRF}_{d_{i,y}}(h, i) \quad \text{and} \quad e'_i = \mathsf{PRF}_{e_i}(h, i).$$

First note that this transformation preserves correctness: whenever $y = \boldsymbol{x}_i$ it holds that $d_{i,y} = e_i$, as PRF is a function it also holds that

$$d'_{i,y} = \mathsf{PRF}_{d_{i,y}}(h, i) = \mathsf{PRF}_{e_i}(h, i) = e'_i.$$

We can finally argue batch security. The goal of the adversary is to distinguish

$$\begin{aligned} d'_{1,y_1} &= \mathsf{PRF}_{d_{1,y_1}}(h, 1) = \mathsf{PRF}_{e_1 \cdot u^{\boldsymbol{x}_1 - y_1}}(h, 1) \\ &\vdots \\ d'_{\ell,y_\ell} &= \mathsf{PRF}_{d_{i,y_\ell}}(h, \ell) = \mathsf{PRF}_{e_\ell \cdot u^{\boldsymbol{x}_\ell - y_\ell}}(h, \ell) \end{aligned}$$

from uniformly random values and $y_i \neq \boldsymbol{x}_i$. As $u \leftarrow_{\$} \mathbb{G}_t$ is chosen uniformly at random, $\boldsymbol{x}_i - y_i \neq 0$ and PRF is RKA secure, we can replace all the $\mathsf{PRF}_{e_i \cdot u^{\boldsymbol{x}_i - y_i}}(h, i)$ with independent uniformly random values as all the inputs (h, i) are different. This concludes this sketch.

We remark that this argument immediately extends across different hash values h as well, i.e. we can reuse a single public key $\mathsf{pk} = g^{\xi}$ across an unbounded number of hash values h and still argue pseudorandomness. Furthermore, by the security of the RKA-PRF we can let the adversary choose the y_i adaptively by providing him access to an oracle which for $y \neq \boldsymbol{x}_i$ either returns the real $\mathsf{PRF}_{e_i \cdot u^{\boldsymbol{x}_i - y}}(h, i)$ or uniformly random values.

A Scheme in the Standard Model We will briefly sketch that for the case of a binary alphabet our scheme can also be instantiated in the standard model, that is without random oracles. For this purpose, we need to slightly modify the privately and publicly compute the PRF values of the base scheme. Specifically, for $b \in \{0, 1\}$ we will adjust our base scheme by setting $\tilde{d}_{i,b} = d_{i,b}^{1-2b}$ and $\tilde{e}_i = e_i^{1-2\boldsymbol{x}_i}$. It can be seen that the scheme is still correct with a few small modifications, and the $\tilde{d}_{i,b}$ and $\tilde{e}_i$ satisfy the group relation

$$\tilde{d}_{i,1-\boldsymbol{x}_i} = (e'_i)^{-1} \cdot u. \tag{2.3}$$

Bellare and Cash [BC10] show that a variant of the well-known Naor-Reingold (NR) PRF [NR97] is RKA secure against group-induced relations. The NR PRF is defined over a DDH-hard group $\mathbb{G}'$ of order p, its key is a randomly chosen vector from $(\mathbb{Z}_p^*)^k$ (or a sufficiently large subgroup of $\mathbb{Z}_p^*$), its domain is $\{0, 1\}^k$ and its range is $\mathbb{G}'$. Bellare and Cash [BC10] provide a variant of this PRF which is secure against group-induced

relations $\boldsymbol{f} \odot \boldsymbol{k}$, where $\odot$ is the component-wise group operation on $(\mathbb{Z}_p^*)^k$. Bellare and Cash [BC10] provide an instantiation from the k-LIN assumption.

In the case of the Weil- and Tate-Pairings [Jou00, BF01], the target group $\mathbb{G}_t$ is a subgroup of the multiplicative group $\mathbb{F}_{p^t}^*$ of an extension field $\mathbb{F}_{p^t}$. Certain parameter choices (see [Jou00]) admit $t = 1$, in this case p is of the form $s^2 + 1$.

In more common instantiations of pairing groups though the target group is a subgroup of the multiplicative group $\mathbb{F}_{p^2}^*$ of the extension field $\mathbb{F}_{p^2}$ [MOV93]. In this case we can rely on the fact that elements of $\mathbb{F}_{p^2}$ can be represented as matrices in $\mathbb{F}_p^{2\times 2}$ and rely on the 2-LIN-based instantiation of [BC10].

To make our scheme compatible with this larger key space, we have to use a *vector version* of our basic WPRF. For details about this scheme with binary alphabet refer to Section B.

2.3 Batch Laconic OT via WPRF

Recall the basic setting of laconic OT [CDG+17]: A receiver hashes a large database $\mathsf{DB} \in \{0,1\}^\ell$ into a batch hash value h. Given h as well as an index $i \in [\ell]$ and two messages m_0 and m_1, a sender produces a ciphertext c and sends it to the receiver. Now, given DB and c the receiver can recover the message $m_{\mathsf{DB}[i]}$, but learns nothing about $m_{1-\mathsf{DB}[i]}$.

We can easily build a laconic OT scheme from WPRF in the following way:

- The receiver starts by hashing a database DB into a small hash value h using the WPRF hashing algorithm.
- Given the hash value h, the sender i) Computes a public key and trapdoor $(\mathsf{pk}, \mathsf{td})$ using the Gen algorithm of WPRF, and ii) Masks both messages m_0, m_1 using $d_{i,0}$ and $d_{i,1}$ (both obtained by running the $\mathsf{EvalStmt}$ algorithm). It sends pk and the masked values $m_0 \oplus d_{i,0}$ and $m_1 \oplus d_{i,1}$.
- Given the public key pk, the receiver can recover $e_i = d_{i,\mathsf{DB}[i]}$.

Security is guaranteed from the fact that $d_{i,1-\mathsf{DB}[i]}$ is pseudorandom from the perspective of the receiver and, thus, $m_{1-\mathsf{DB}[i]}$ is hidden.

Finally, to obtain a batch LOT, we will batch k laconic OT executions while reusing the same public key pk. Security of the scheme follows from the batch pseudorandomness security of the underlying WPRF. In terms of communication, the sender's message is composed of pk (which has constant size) plus k pairs of masked messages (which have size $2k$).

2.4 Near-Optimal Laconic Private Set Intersection via WPRF

The final application that we will cover in this technical overview is a laconic PSI from WPRF which has near optimal parameters. Our construction combines Bloom-filter inspired ideas with our WPRF.

First, recall that in laconic PSI, a receiver with a large set X starts by hashing it into a small digest h. Then, using h, the sender computes a short message c that encrypts Y. Finally, the receiver learns $Y \cap X$. Assuming that both $X, Y \subseteq \{0,1\}^\lambda$ and $|Y| = k$, in our scheme c is of size $\lambda k + \mathsf{poly}(\lambda)$, which means that the amortized communication per λ-bit set element approaches λ bits. Assuming we want to tolerate a correctness error of at most $2^{-\Omega(\lambda)}$, this is just a constant factor away from optimal [KS92].

For our laconic PSI construction, we need a WPRF for larger alphabets, namely $\mathbb{Z}_p$, instead of binary alphabets, but we note that the construction sketched in Section 2.2 can be easily generalized to support $\mathbb{Z}_p$.

Our scheme can be outlined as follows. Assume we can embed $\{0,1\}^\lambda$ into $\mathbb{Z}_p$, i.e. we assume $X, Y \subseteq \mathbb{Z}_p$. Given a set X of size N, let $\ell = c \cdot N$ for a sufficiently large c. The receiver initializes λ databases $\mathsf{DB}_1, \ldots, \mathsf{DB}_\lambda \in \mathbb{Z}_p^\ell$ and chooses λ pairwise independent hash functions $f_1, \ldots, f_\lambda$ mapping from $\mathbb{Z}_p$ to $[\ell]$. Now the databases $\mathsf{DB}_1, \ldots, \mathsf{DB}_\lambda$ are populated with the elements $x \in X$, in such a way that for every $i \in [\lambda]$ and every $j \in [\ell]$ it holds that $\mathsf{DB}_i[j] = x$ for some $x \in X$ with $f_i(x) = j$, or $\mathsf{DB}_i[j] = 0$ if no such x exists. The receiver now hashes each database DB_i into a WPRF hash h_i and obtains openings $\pi_{i,f_i(x)}$ for $x \in X$.

The receiver now sends the hash values $h_1, \ldots, h_\lambda$ and $f_1, \ldots, f_\lambda$ to the sender. Notice that the receiver's message is succinct, as each h_i and each description of f_i is succinct.

The sender now proceeds as follows. First, it generates a public key and trapdoor pair $(\mathsf{pk}, \mathsf{td})$ for which PRF outputs are just single bits. For each $y \in Y$, it computes the value $d_{i,y} = \mathsf{EvalStmt}(\mathsf{td}, h_i, (f_i(y), y)) \in \{0,1\}$ for every $i \in [\lambda]$ and sets $D_y = (d_{1,y}, \ldots, d_{\lambda,y}) \in \{0,1\}^\lambda$. It then sends pk together with a randomly permuted version $(D_1, \ldots, D_k)$ of $(D_y)_{y \in Y}$ to the receiver.

Now, for each $x \in X$ and every $i \in [\lambda]$ the receiver computes values $e_{i,x} = \mathsf{EvalWit}(\mathsf{pk}, h_i, (f_i(x), \pi_{i,f_i(x)}))$ and sets $E_x = (e_{1,x}, \ldots, e_{\lambda,x}) \in \{0,1\}^\lambda$.

Now the receiver checks for every D_j if there exists an $x \in X$ such that D_j and E_x have hamming distance less than $\lambda/4$, if so it declares x to be in the intersection.

This concludes the description of the protocol.

To achieve receiver security for this protocol, we rely on the statistical hiding of the hash value.[5] Before analyzing correctness, we will start by arguing that this protocol satisfies sender privacy. Hence fix some $y \in Y$ with $y \notin X$. We will argue that the value D_y is pseudorandom in the view of the receiver. Since $y \notin X$, y does not appear in any of the databases DB_i, that is, y is different from every $\mathsf{DB}_i[j]$ for $i \in [\lambda]$ and $j \in [\ell]$. But this means we can immediately appeal to the pseudorandomness property of WPRF and argue that each $d_{i,y}$, and therefore D_y, is pseudorandom in the view of the receiver.

This establishes sender privacy, but also lets us analyze the case $y \notin X$ for the correctness property. As D_y is pseudorandom, it will have hamming distance $\approx \lambda/2$ from every E_x. Specifically, by a standard Chernoff bound the probability that the hamming distance between D_y and any of the E_x is smaller than $\lambda/4$ is negligible. Thus y is not declared to be in the intersection.

On the other hand, if $y \in X$, we can argue as follows. Say that an index $i \in [\ell]$ is collision-free for y, if there is no $x' \in X \setminus \{y\}$ with $f_i(y) = f_i(x')$. It follows instantly that if an index i is collision free for y, then $\mathsf{DB}_i[f_i(y)] = y$. By appealing to pairwise independence of the f_i and a union-bound, we can show that the expected number of collision-free indices for y is at least $(1 - 1/c)\lambda$. Hence, using Chernoff-bound we can argue that the number of collision-free indices is at least $(1 - 2/c)\lambda$, except with negligible probability. By setting $c = 8$, we obtain that y has at least $3/4 \cdot \lambda$ collision-free indices. Now observe that it holds by the correctness of WPRF that for each collision-free index i that $d_{i,y} = e_{i,y}$, i.e. D_y and E_y agree on at least $3/4\lambda$ indices. But this means D_y and E_y have hamming distance less than $\lambda/4$, and thus the receiver will correctly declare y to be in the intersection.

This concludes the outline.

3 Preliminaries

Let λ denote the cryptographic security parameter. We assume that all cryptographic algorithms implicitly take 1^λ as input. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$ there exists λ_0 such that for all $\lambda > \lambda_0$ we have $\mu(\lambda) < \frac{1}{\mathsf{poly}}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function.

Let X and Y be two distributions. Then i) $X \approx_c Y$ means that the distributions are computationally indistinguishable, ii) $X \approx_\varepsilon Y$ means that their statistical distance is at most ε and, iii) $X \approx_s Y$ means that their statistical distance is at most negl in λ. For a finite set S, we denote $x \leftarrow_\$ S$ as the process of sampling x uniformly from the set S.

3.1 Pairing Groups

In this work, we use asymmetric bilinear groups. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three groups of prime order p. We will denote by g_1, g_2 and g_3 their respective generators. Additionally let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be the pairing operation.

[5] This can be achieved by hashing the database together with fresh random coins in a standard way.

Definition 1 (The Bilinear Diffie-Hellman in the Exponent (BDHE) Assumption)**.** *Let* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ *be an asymmetric bilinear group and let* $\ell = \mathsf{poly}(\lambda)$*. We say that the asymmetric BDHE assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$*, if*

$$\left(g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^{\ell}}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^{\ell}}, g_2^{\xi}, g_t^{\xi \cdot \alpha^{\ell+1}}\right)$$
$$\approx_c$$
$$\left(g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^{\ell}}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^{\ell}}, g_2^{\xi}, g_t^{u}\right)$$

where $\alpha, \xi, u \leftarrow_{\$} \mathbb{Z}_p$ *are chosen uniformly random.*

Definition 2 (The Bilinear Diffie-Hellman (BDH) Assumption)**.** *Let* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ *be an asymmetric bilinear group. We say that the asymmetric BDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$*, if*

$$\left(g_1, g_1^{\alpha}, g_1^{\beta}, g_1^{\gamma}, g_2, g_2^{\alpha}, g_2^{\beta}, g_2^{\gamma}, g_t^{\alpha\beta\gamma}\right) \approx_c \left(g_1, g_1^{\alpha}, g_1^{\beta}, g_1^{\gamma}, g_2, g_2^{\alpha}, g_2^{\beta}, g_2^{\gamma}, g_t^{u}\right),$$

where $\alpha, \beta, \gamma, u \leftarrow_{\$} \mathbb{Z}_p$ *are chosen uniformly random.*

We will use the following simple consequence of the BDH assumption.

Lemma 1. *Let* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ *be a bilinear group, and let* $g_1 \in \mathbb{G}_1$ *and* $g_2 \in \mathbb{G}_2$ *be two generators. Let* $z \leftarrow_{\$} \mathbb{G}_t$ *be chosen uniformly random. Define the oracle* $\mathcal{R}_{g_1,g_2,z}()$ *as follows: Upon being queried, it chooses a uniformly random* $\delta \leftarrow_{\$} \mathbb{Z}_p$ *and outputs* $(g_1^{\delta}, g_2^{\delta}, z^{\delta})$*. Further define the oracle* $\mathcal{I}_{g_1,g_2}()$*: Upon being queried it chooses a uniformly random* $\delta \leftarrow_{\$} \mathbb{Z}_p$ *and a uniformly random* $u \leftarrow_{\$} \mathbb{G}_t$ *and outputs* $(g_1^{\delta}, g_2^{\delta}, u)$*.*

Assume that the asymmetric BDH assumption holds in a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$*. Then the oracles* $\mathcal{R}_{g_1,g_2,z}()$ *and* $\mathcal{I}_{g_1,g_2}()$ *are indistinguishable. That is, it holds for any PPT oracle-distinguisher* $\mathcal{D}$ *that*

$$|\Pr[\mathcal{D}^{\mathcal{R}_{g_1,g_2,z}()}(g_1, g_2, z) = 1] - \Pr[\mathcal{D}^{\mathcal{I}_{g_1,g_2}()}(g_1, g_2, z) = 1]| \leq \mathsf{negl}(\lambda),$$

where the probabilities are taken over all random choices. The reduction is tight.

Proof. Let $\mathcal{D}$ be a PPT adversary which distinguishes the two oracles with non-negligible advantage ε. We will construct a PPT-distinguisher $\mathcal{D}'$ with distinguishes the BDH assumption with advantage $\varepsilon - \mathsf{negl}(\lambda)$.

Given a BDH challenge $\left(g_1, g_1^{\alpha}, g_1^{\beta}, g_1^{\gamma}, g_2, g_2^{\alpha}, g_2^{\beta}, g_2^{\gamma}, z\right)$, where either $z = g_t^{\alpha\beta\gamma}$ or $z \leftarrow_{\$} \mathbb{G}_t$ is chosen uniformly random, the distinguisher $\mathcal{D}'$ proceeds as follows. It provides $\left(g_1, g_2, e\left(g_1^{\alpha}, g_2^{\beta}\right) = g_t^{\alpha\beta}\right)$ as explicit input to $\mathcal{D}$ and simulates an oracle $\mathcal{O}()$. Every time $\mathcal{D}$ queries the oracle $\mathcal{O}()$:

- Choose $v, w \leftarrow_{\$} \mathbb{Z}_p$ uniformly at random.
- Set $g_1^{\delta} = \left(g_1^{\gamma}\right)^v \cdot g_1^w$ and $g_2^{\delta} = \left(g_2^{\gamma}\right)^v \cdot g_2^w$
- Set $y = z^v \cdot e\left(g_1^{\alpha}, g_2^{\beta}\right)^w$
- Output $\left(g_1^{\delta}, g_2^{\delta}, y\right)$

First note that the $\delta \in \mathbb{Z}_p$ are indeed well-defined. It holds that

$$\left(g_1^{\gamma}\right)^v \cdot g_1^w = g_1^{v\gamma+w}$$
$$\left(g_2^{\gamma}\right)^v \cdot g_2^w = g_2^{v\gamma+w},$$

i.e. it holds that $\delta = v\gamma + w$. Note that the δ are distributed uniformly as the w are distributed uniformly. Moreover, note that for any non-zero β it holds that $z' = g_t^{\alpha\beta}$ is distributed uniformly (as α is chosen uniformly random).

- In the first case, assume that $z = g_t^{\alpha\beta\gamma}$. Then it holds that

$$y = z^v \cdot e\left(g_1^\alpha, g_2^\beta\right)^w = g_t^{v\alpha\beta\gamma + w\alpha\beta} = g_t^{\alpha\beta\delta} = (z')^\delta .$$

  We can conclude that in this case the samples produced by $\mathcal{O}$ are identically distributed to the samples generated by the oracle $\mathcal{R}_{g_1,g_2,z'}()$

- In the second case, $z \leftarrow_\$ \mathbb{Z}_p$ is chosen uniformly random, i.e. we can write $z = g_t^\mu$ for a uniformly random $\mu \leftarrow_\$ \mathbb{Z}_p$. In this case it holds that

$$y = z^v \cdot e\left(g_1^\alpha, g_2^\beta\right)^w = g_t^{v\mu + w\alpha\beta} =: g_t^\phi .$$

  Consider the matrix

$$\boldsymbol{M} = \begin{pmatrix} \gamma & 1 \\ \mu & \alpha\beta \end{pmatrix}.$$

  The matrix $\boldsymbol{M}$ has determinant $\det(\boldsymbol{M}) = |\alpha\beta\gamma - \mu|$. Note that unless $\mu = \alpha\beta\gamma$, which only happens with probability $1/p = \mathsf{negl}(\lambda)$, it holds that $\det(\boldsymbol{M}) \neq 0$ and hence $\boldsymbol{M}$ has full rank. Given that $\boldsymbol{M}$ has full rank, it follows that the $\phi = v\mu + w\alpha\beta$ and $\delta = v\gamma + w$ are distributed independently uniformly random, as

$$\begin{pmatrix} \delta \\ \phi \end{pmatrix} = \boldsymbol{M} \cdot \begin{pmatrix} v \\ w \end{pmatrix}.$$

  Hence we conclude that in this case the oracle $\mathcal{O}()$ produces the same output distribution as the oracle $\mathcal{I}_{g_1,g_2}()$.

We conclude that $\mathcal{D}'$ distinguishes BDH with non-negligible advantage $\varepsilon - \mathsf{negl}(\lambda)$, which contradicts the hardness of BDH. This concludes the proof. □

3.2 Related-Key-Secure Pseudorandom Functions

A pseudorandom function is called related key secure, if it remains pseudorandom even if the adversary can query it on modified keys.

Definition 3 (Related-Key-Secure Pseudorandom Functions [BK03, BC10])**.** *Let $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$ be an efficiently computable keyed function. Further let $\Phi \subseteq (\{0,1\}^\lambda \to \{0,1\}^\lambda)$ be a class of functions from $\{0,1\}^\lambda$ to $\{0,1\}^\lambda$. We say that F is an RKA-secure PRF with respect to Φ, if it holds for all PPT oracle distinguishers $\mathcal{D}$ that*

$$|\Pr[\mathcal{D}^{F^+(K,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{R^+(\cdot,\cdot)}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where $K \leftarrow_\$ \{0,1\}^\lambda$ and the oracles $F^+(K,\cdot,\cdot)$ and $R^+(\cdot,\cdot)$ are defined as follows. The oracle $F^+(K,\cdot,\cdot)$ takes as input a function $\varphi \in \Phi$ and an input $x \in \{0,1\}^n$ and outputs $F(\varphi(K), x)$. The oracle $R^+(\cdot,\cdot)$ is initialized by choosing a uniformly random key $K' \leftarrow_\$ \{0,1\}^\lambda$ and a uniformly random function $G : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$. On input (φ, x), the oracle returns $G(\varphi(K), x)$.

Note that the random function G in Definition 3 can be simulated efficiently via lazy sampling.

We will make use of the following Theorem of Bellare and Cash [BC10], which provides us with the existence of a PRF secure against group-induced relations from the DDH assumption.

Theorem 1 ([BC10])**.** *Let $\mathbb{G}$ be a group of prime-order p and let $k \geq 0$. Furthermore, let Φ be the class consisting of all functions $\varphi_{\boldsymbol{a}} : (\mathbb{Z}_p^*)^k \to (\mathbb{Z}_p^*)^k$ for all $\boldsymbol{a} \in (\mathbb{Z}_p^*)^k$, defined as follows. The function $\varphi_{\boldsymbol{a}}$ is defined by $\varphi_a(x) = \boldsymbol{a} \odot \boldsymbol{x}$, where $\odot$ is the component-wise product of two vectors in Z_p^k. Assuming the hardness of DDH in $\mathbb{G}$, there exists a pseudorandom function with domain $\{0,1\}^{k-1}$, key-space $(\mathbb{Z}_p^*)^k$ and range $\mathbb{G}$ with RKA security against Φ.*

4 Witness PRF for LY Commitments

We start by defining witness PRFs for vector commitments.

Definition 4 (Witness PRF for vector commitment)**.** *A witness PRF for vector commitments* $\mathsf{WPRF} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{EvalStmt}, \mathsf{EvalWit})$ *is parametrized by alphabets* Σ *and* Σ' *and has the following syntax:*

- *The setup algorithm* $\mathsf{Setup}(1^\lambda, 1^\ell)$ *takes as input the security parameter* 1^λ*, the database-length parameter* 1^ℓ *and outputs public parameters* pp
- *The key generation algorithm* $\mathsf{Gen}(\mathsf{pp})$ *takes as input public parameters* pp *and outputs a pair* $(\mathsf{pk}, \mathsf{td})$ *consisting of a public key* pk *and a trapdoor* td*.*
- *The hashing algorithm* $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$ *takes as input the public parameters* pp *and a database* $\mathsf{DB} \in \Sigma^\ell$ *and outputs a hash value* h *as well as a set of openings* $P = \{\boldsymbol{\pi}_i\}$*.*
- *The* statement *evaluation algorithm* $\mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td}, h, (i, y))$ *takes as input the public parameters* pp*, a trapdoor* td*, a hash value* h *and a pair* (i, y) *consisting of an index* $i \in [\ell]$ *and a value* $y \in \Sigma$ *and outputs a value* $d_i \in \Sigma'$*.*
- *The* witness *evaluation algorithm* $\mathsf{EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (i, \boldsymbol{\pi}_i))$ *takes as input the public parameters* pp*, a public key* pk*, a hash value* h *and a pair* $(i, \boldsymbol{\pi}_i)$ *consisting of an index* $i \in [\ell]$ *and an opening object* $\boldsymbol{\pi}_i$ *and outputs a value* $e_i \in \Sigma'$

We require the following properties:

- ***Succinctness:*** *The hash value* h *is of size* $\mathsf{poly}(\lambda)$*, independent of the database size* ℓ*. Furthermore, the public key* pk *is also of size* $\mathsf{poly}(\lambda)$*, independent of the database size* ℓ*.*
- ***Correctness:*** *Fix a database* $\mathsf{DB} \in \Sigma^\ell$ *and an index* $i \in [\ell]$*. Let* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$*,* $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp})$*,* $(h, P) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB})$ *where* $P = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell)$*,* $e \leftarrow \mathsf{EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (i, \boldsymbol{\pi}_i))$ *and* $d \leftarrow \mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td}, h, (i, \mathsf{DB}[i]))$*. Then it holds that* $e = d$*, except with negligible probability over all random choices.*
- ***Pseudorandomness:*** *Consider the following experiment* $\mathsf{Exp}^b_{\lambda,\ell}(\mathcal{A})$*, where* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *is a PPT adversary.*
  - $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
  - $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
  - *Let* $(\mathsf{DB}^{(1)}, \ldots, \mathsf{DB}^{(q)}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk})$
  - *For* $\eta = 1, \ldots, q$ *set* $(h^{(\eta)}, P^{(\eta)}) \leftarrow \mathsf{Hash}\left(\mathsf{pp}, \mathsf{DB}^{(\eta)}\right)$
  - *If* $b = 0$ *then set* $\mathcal{O} = \mathsf{Real}(\mathsf{td}, \cdot)$ *which, when queried on* $(\eta, i, y) \in [q] \times [\ell] \times \Sigma$*, outputs* $d^{(\eta)}_{i,y} \leftarrow \mathsf{EvalStmt}\left(\mathsf{pp}, \mathsf{td}, h^{(\eta)}, (i, y)\right)$*.*
  - *If* $b = 1$ *then set* $\mathcal{O} = \mathsf{Ideal}(\cdot)$ *which, when queried on* $(\eta, i, y) \in [q] \times [\ell] \times \Sigma$*, outputs* $e^{(\eta)}_i \leftarrow \mathsf{EvalWit}(\mathsf{pp}, \mathsf{pk}, h^{(\eta)}, (i, \boldsymbol{\pi}^{(\eta)}_i))$ *if* $y = \mathsf{DB}^{(\eta)}[i]$*, or a uniformly random* $d^{(\eta)}_{i,y} \leftarrow \Sigma'$ *otherwise.*
  - *Output* $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}\left(\mathsf{st}, \{d^{(\eta)}_i\}_{i \in [\ell], \eta \in [q]}\right)$

  We say the scheme WPRF *has pseudorandomness, if it holds for every PPT adversary* $\mathcal{A}$ *that*

$$|\Pr[\mathsf{Exp}^0_{\lambda,\ell}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}^1_{\lambda,\ell}(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

We define an additional property for our WPRF that requires the hash value to hide the database.

Definition 5 (Statistical hiding)**.** *A WPRF for VC is said to be statistical hiding if for any $\lambda \in \mathbb{N}$, any $\ell = \mathsf{poly}(\lambda)$ and any database $\mathsf{DB} \in \Sigma^\ell$, we require that*

$$\left\{(h, \mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, P) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}) \end{array}\right\} \approx_s \left\{(h, \mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ h \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} \end{array}\right\}.$$

5 Witness PRF for LY commitments from BDHE and Related-Key Security

In this section, we provide a construction of a witness PRF scheme for LY commitments [LY10] from the BDHE assumption and related-key-secure pseudorandom functions for group relations in the target group.

5.1 Construction

Let $k > 0$ be a parameter, called the key parameter. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ be an asymmetric bilinear group of order p, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ is a bilinear map. Let $F : \mathbb{G}_t^k \times [\ell] \to \{0,1\}^\lambda$ be a keyed function. The WPRF scheme $\mathsf{WPRF} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{EvalWit}, \mathsf{EvalStmt})$ with alphabet $\Sigma \subseteq \mathbb{Z}_p$ is given as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$:
  - Choose $\alpha \leftarrow_\$ \mathbb{Z}_p$ uniformly at random.
  - Choose $\gamma_1, \ldots, \gamma_k \leftarrow_\$ \mathbb{Z}_p$ uniformly at random
  - Compute and output the public parameters

$$\mathsf{pp} = \left\{\begin{array}{ccccccccccccc} g_1, & g_1^{\alpha}, & g_1^{\alpha^2}, & \ldots, & g_1^{\alpha^\ell}, & _, & g_1^{\alpha^{\ell+2}}, & \ldots, & g_1^{\alpha^{2\ell}}, & g_2, & g_2^{\alpha^2}, & \ldots, & g_2^{\alpha^\ell} \\ g_1^{\gamma_1}, & g_1^{\gamma_1\alpha}, & g_1^{\gamma_1\alpha^2}, & \ldots, & g_1^{\gamma_1\alpha^\ell}, & _, & g_1^{\gamma_1\alpha^{\ell+2}}, & \ldots, & g_1^{\gamma_1\alpha^{2\ell}}, & g_2^{\gamma_1}, & g_2^{\gamma_1\alpha^2}, & \ldots, & g_2^{\gamma_1\alpha^\ell} \\ & & & & & \vdots & & & & & & & \\ g_1^{\gamma_k}, & g_1^{\gamma_k\alpha}, & g_1^{\gamma_k\alpha^2}, & \ldots, & g_1^{\gamma_k\alpha^\ell}, & _, & g_1^{\gamma_k\alpha^{\ell+2}}, & \ldots, & g_1^{\gamma_k\alpha^{2\ell}}, & g_2^{\gamma_k}, & g_2^{\gamma_k\alpha^2}, & \ldots, & g_2^{\gamma_k\alpha^\ell} \end{array}\right\}$$

- $\mathsf{Gen}(\mathsf{pp})$: Choose $\mathsf{td} = \xi \leftarrow_\$ \mathbb{Z}_p$ uniformly at random. Compute $\mathsf{pk} = g_2^\xi$. Output $(\mathsf{pk}, \mathsf{td})$.
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB} \in \Sigma^\ell; r)$:
  - $r \in \mathbb{Z}_p$ is an optional randomization term, if it is not provided set $r = 0$.
  - Compute

$$h = g_1^r \cdot \prod_{i' \in [\ell]} \left(g_1^{\alpha^{i'}}\right)^{\mathsf{DB}[i']}$$

  - For $i = 1, \ldots, \ell$ and $j = 1, \ldots, k$ compute the openings

$$\pi_{i,j} = \left(g_1^{\gamma_j \alpha^{\ell+1-i}}\right)^r \cdot \prod_{i' \in [\ell] \setminus \{i\}} \left(g_1^{\gamma_j \alpha^{\ell+1+i'-i}}\right)^{\mathsf{DB}[i']}.$$

  - For all $i \in [\ell]$ set $\boldsymbol{\pi}_i = (\pi_{i,j})_{j \in [k]}$.
  - Output h and $P = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell)$
- $\mathsf{EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (i, \boldsymbol{\pi}_i))$
  - Parse $\boldsymbol{\pi}_i = (\pi_{i,1}, \ldots, \pi_{i,k})$
  - For $j \in [k]$ compute $\kappa'_j = e(\pi_{i,j}, \mathsf{pk})$.

- Set $\boldsymbol{\kappa}'_i = (\kappa'_{i,j})_{j\in[k]}$
- and $e_i = F(\boldsymbol{\kappa}'_i, (h, i))$

- $\mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td} = \xi, h, (i, x \in \Sigma))$:
  - For $j = 1, \ldots, k$ compute $\kappa_{i,j} = e\left(h, g_2^{\gamma_j \cdot \alpha^{\ell+1-i}}\right)^{\xi} \cdot e\left(g_1^{\alpha}, g_2^{\gamma_j \cdot \alpha^{\ell}}\right)^{-x\cdot\xi}$.
  - Set $\boldsymbol{\kappa}_i = (\kappa_{i,j})_{j\in[k]}$
  - Compute and output $d_i = F(\boldsymbol{\kappa}_i, (h, i))$.

5.2 Compactness and Efficiency

We note the following compactness properties of the scheme WPRF:

- For databases of size ℓ and a key parameter k, the public parameters pp consist of $(k+1)(3\ell+1)$ group elements
- The public key pk consists of a single group element in $\mathbb{G}_2$.
- The hash value h is a single group element in $\mathbb{G}_1$, the set of openings consists of $k\ell$ group elements in $\mathbb{G}_1$.

Again, we also observe the following efficiency properties of our scheme.

- The Gen algorithm incurs one group operation.
- Hashing incurs ℓ group operations in $\mathbb{G}_1$ to produce the hash value and $k\ell^2$ group operations to produce the openings $\pi_{i,j}$
- The witness evaluation algorithm incurs k pairing operations and one call to F.
- The statement algorithm performs $2k$ pairing operations, as well as a call to F.

Concrete Parameters for the RO Instantiation. The following concrete discussion applies to the random-oracle-based instantiation of F described below. In this case we can choose $k = 1$, which means there is essentially no blow-up in the size of the public parameters and openings and imposes no further constraints on the bilinear group. Our approach then builds on the same assumptions as the work of Glaeser *et al.* [GKMR23], so we can use the same choice of groups they use. For concreteness, we can use the BLS12-381 elliptic curve: each element in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ is represented with 49, 97, and 384 bytes, respectively [GKMR23]. As an example, for a database of size 1GB and setting the key parameter to $k = 1$ we obtain the following concrete parameters:

- **Public parameters:** approximately 2TB.
- **Public key:** 97 bytes.
- **Hash value:** 49 bytes.

5.3 Correctness

We will now establish the correctness of the scheme WPRF.

Lemma 2. *The scheme* WPRF *has perfect correctness.*

Proof. Fix a hash $h = g_1^r \cdot \prod_{i' \in [\ell]} \left(g_1^{\alpha^{i'}}\right)^{\mathsf{DB}[i']}$, $\pi_{i,j} = \left(g_1^{\gamma_j \alpha^{\ell+1-i}}\right)^r \cdot \prod_{i' \in [\ell] \setminus \{i\}} \left(g_1^{\gamma_j \alpha^{\ell+1+i'-i}}\right)^{\mathsf{DB}[i']}$ and $\mathsf{pk} = g_2^{\xi}$, an index $i \in [\ell]$ and $x \in \mathbb{Z}_p$. It holds that

$$\begin{aligned}
\kappa_{i,j} &= e\left(h, g_2^{\gamma_j \alpha^{\ell+1-i}}\right)^{\xi} \cdot e\left(g_1^{\alpha}, g_2^{\gamma_j \alpha^{\ell}}\right)^{-x \cdot \xi} \\
&= e\left(\left(g_1^{\gamma_j \alpha^{\ell+1-i}}\right)^r \cdot \prod_{i' \in [\ell]} \left(g_1^{\gamma_j \alpha^{\ell+1+i'-i}}\right)^{\mathsf{DB}[i']}, g_2^{\xi}\right) \cdot e\left(g_1, g_2\right)^{-x \cdot \xi \cdot \gamma_j \cdot \alpha^{\ell+1}} \\
&= e\left(\pi_{i,j}, \mathsf{pk}\right) \cdot e\left(g_1, g_2\right)^{(\mathsf{DB}[i]-x) \cdot \xi \gamma_j \alpha^{\ell+1}} \\
&= \kappa'_{i,j} \cdot \left(e\left(g_1, g_2\right)^{\xi \gamma_j \alpha^{\ell+1}}\right)^{\mathsf{DB}[i]-x}
\end{aligned}$$

Hence, if $x = \mathsf{DB}[i]$ it holds that $\kappa_{i,j} = \kappa'_{i,j}$ for all $j \in [k]$, i.e. $\boldsymbol{\kappa}_i = \boldsymbol{\kappa}'_i$. Consequently, it also holds that $F(\boldsymbol{\kappa}_i, i) = F(\boldsymbol{\kappa}'_i, i)$, i.e. the scheme is correct. □

5.4 Security

We will now show that the scheme WPRF has pseudorandomness.

Lemma 3. *Given that the BDHE and BDH assumptions hold in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ and F is a pseudorandom function with related key security against linear group-induced relations, then the scheme* WPRF *has pseudorandomness.*

Proof. We have seen in the correctness proof above that $\kappa_{i,j} = \kappa'_{i,j} \cdot \left(e(g_1, g_2)^{\xi \gamma_j \alpha^{\ell+1}}\right)^{\mathsf{DB}[i]-x}$. Consequently, we can simulate $\kappa_{i,j}$ by computing $\kappa'_{i,j} = e(\pi_{i,j}, \mathsf{pk})$ and setting

$$\kappa_{i,j} = e(\pi_{i,j}, \mathsf{pk}) \cdot (e(g_1, g_2)^{\xi \alpha^{\ell+1}})^{(\mathsf{DB}[i]-x)\gamma_j}.$$

Now let $\mathcal{A}$ be a PPT adversary which breaks the pseudorandomness property with non-negligible advantage ε.

Let $\mathsf{DB}^{(1)}, \ldots, \mathsf{DB}^{(q)}$ be the databases provided by $\mathcal{A}_1$, and let $h^{(1)}, \ldots, h^{(q)}$ be their corresponding hashes and $\pi_{i,j}^{(\eta)}$ for $\eta = 1, \ldots, q$ be the corresponding openings. Our proof will proceed in a hybrid argument.

- Hybrid 0: This is the real experiment. However, we will let the EvalStmt compute the keys $\kappa_{i,j}^{(\eta)}$ via $\kappa_{i,j}^{(\eta)} = e(\pi_{i,j}^{(\eta)}, \mathsf{pk}) \cdot z^{(\mathsf{DB}^{(\eta)}[i]-x)\gamma_j}$, where $z = e(g_1, g_2)^{\xi \alpha^{\ell+1}}$. As noted above, computing the $\kappa_{i,j}^{(\eta)}$ in this way is equivalent to computing them the way the EvalStmt algorithm does.
- Hybrid 1: In this hybrid, we replace $z = e(g_1, g_2)^{\xi \alpha^{\ell+1}}$ with a uniformly random group element in $\mathbb{G}_t$. Computational indistinguishability follows from the BDHE assumption. In more detail, given a BDHE instance

  $$g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^{\ell}}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha^2}, \ldots, g_2^{\alpha^{\ell}}$$

  as well as g_2^{ξ} and z, where z is either $g_t^{\xi \cdot \alpha^{\ell+1}}$ or chosen uniformly random in $\mathbb{G}_t$, the reduction $\mathcal{R}_1$ proceeds as follows. It obtains the public parameters pp by choosing the $\gamma_1, \ldots, \gamma_k \leftarrow_{\$} \mathbb{Z}_p$ uniformly at random and computes the $g_1^{\gamma_j \alpha^i} = (g_1^{\alpha^i})^{\gamma_j}$ for $j \in [k]$ and $i \in [2\ell] \setminus \{\ell+1\}$ and $g_2^{\gamma_j \alpha^i} = (g_2^{\alpha^i})^{\gamma_j}$ for $j \in [k]$ and $i \in [\ell]$. Furthermore, it takes $\mathsf{pk} = g_2^{\xi}$ and the value z from the BDHE challenge. Now note that if $z = g_t^{\xi \cdot \alpha^{\ell+1}}$, then $\mathcal{R}_1$'s simulation is identically distributed to hybrid 0. On the other hand, if z is chosen uniformly random, $\mathcal{R}_1$'s simulation is identically distributed to hybrid 1. The claim follows.

- Hybrid 2: In this hybrid, we replace the z^{γ_j} with uniformly random group elements ρ_j for $j \in [k]$. Consequently, setting $\boldsymbol{\kappa}'_i = (\kappa'_{i,j})_{j\in[k]}$ and $\boldsymbol{\rho} = (\rho_j)_{j\in[k]}$, we can compute

$$\boldsymbol{\kappa}_i^{(\eta)} = \boldsymbol{\kappa}'_i \odot \boldsymbol{\rho}^{\mathsf{DB}^{(\eta)}[i]-x}.$$

  Computational indistinguishability of hybrids 1 and 2 follows from the BDH assumption via Lemma 1.

  The reduction $\mathcal{R}_2$ proceeds as follows: It receives g_1, g_2, z as input and obtains $(g_1^{\gamma_j}, g_2^{\gamma_j}, z_j)_{j\in[k]}$ by querying its oracle k times. It chooses $\alpha \leftarrow_\$ \mathbb{Z}_p$ uniformly at random, and obtains the public parameters via computing the $g_1^{\alpha^i}$ (for $i \in [2\ell]\backslash\{\ell+1\}$) and $g_2^{\alpha^i}$ (for $i \in [2\ell]$) and $g_1^{\gamma_j\alpha^i} = (g_1^{\gamma_j})^{\alpha^i}$ as well as $g_2^{\gamma_j\alpha^i} = (g_2^{\gamma_j})^{\alpha^i}$. It further chooses $\xi \leftarrow_\$ \mathbb{Z}_p$ uniformly at random and sets $\mathsf{pk} = g_2^{\xi}$. Finally, for all $j \in [k]$ it sets $\kappa_{i,j} = \kappa'_{i,j} \cdot z_j^{\mathsf{DB}^{(\eta)}[i]-x}$. Now observe that if the z_j are of the form $z_j = g_t^{\gamma_j \cdot u}$, then $\mathcal{R}_2$'s simulation is distributed identically to hybrid 1. On the other hand, if the z_j are chosen uniformly random then $\mathcal{R}_2$'s simulation is distributed identically to hybrid 2. Hence $\mathcal{R}_2$ distinguishes BDH with non-negligible advantage ε, which contradicts the hardness of BDH.

- Hybrid 3: In this hybrid, replace the outputs $d_{i,x}^{(\eta)}$ of the decoding oracle with uniformly random for $x \neq \mathsf{DB}^{(\eta)}[i]$. This is the ideal experiment.

  Computational indistinguishability follows from the RKA-security of F. Specifically, assume that $\mathcal{D}$ distinguishes between hybrid 2 and hybrid 3 with non-negligible advantage ε. We will provide a reduction $\mathcal{R}_3$ such that $\mathcal{R}_3^{\mathcal{D}}$ distinguishes the RKA security experiment with advantage ε. $\mathcal{R}_3$ proceeds as follows. It proceeds identically to hybrid 2, except that it queries its oracle with $(\boldsymbol{\kappa}'_i, (h, i))$ and sets $d_{i,x}^{(\eta)}$ to the output.

  In the first case, $\mathcal{R}_3$'s oracle implements $F^+(\boldsymbol{\rho}, \cdot, \cdot)$ for a uniformly chosen key $\boldsymbol{\rho} \in \mathbb{G}_t^k$. Hence the d_i are of the form $F(\boldsymbol{\kappa}'_i \odot \boldsymbol{\rho}^{\mathsf{DB}^{(\eta)}[i]-x}, (h, i))$. Hence, in this case $\mathcal{R}_3$'s simulation is identically distributed to hybrid 2.

  In the second case, $\mathcal{R}_3$'s oracle implements a uniformly random function R^+. Consequently all the $d_{i,x}^{(\eta)}$ are distributed uniformly random. Hence, in this case $\mathcal{R}_3$'s simulation is distributed identically to hybrid 3. The claim follows.

This concludes the proof. □

We also prove that the scheme is statistically hiding.

Lemma 4. *The scheme above is statistically hiding.*

Proof. For a uniform $r \leftarrow \mathbb{Z}_p$, we have that

$$h = g_1^r \cdot \prod_{i'\in[\ell]} \left(g_1^{\alpha^{i'}}\right)^{\mathsf{DB}[i']} \approx_s g^u$$

where $u \leftarrow \mathbb{Z}_p$ is uniformly random. □

5.5 Instantiations

We will now briefly discuss two instantiations of WPRF, which essentially come down to the choice of the RKA-secure PRF F and influence the key-parameter k.

- If we realize F with a random-oracle-based pseudorandom function, we can choose $k = 1$, which means there is essentially no blow-up in the size of the public parameters and openings. In this case, there are no further constraints on the bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$. This is the instantiation we have in mind in the concrete parameter discussion above.

- For the special case of binary alphabets, we can realize F with the RKA-secure pseudorandom function given in Theorem 1. For this purpose, we also have to make a few modifications to the WPRF protocol which are given in Section B. We further need to choose $2^{k-1} \geq \ell \cdot |\mathbb{G}_t|$, i.e. $k \geq \log(\ell) + \log(|\mathbb{G}_t|) + 1$ to have a sufficiently large domain for F to accommodate pairs (h, i). Furthermore, in this case $\mathbb{G}_t$ must be a subgroup of $\mathbb{Z}_q^*$ for some prime q. Thus, we need an additional DDH-hard group of order q to instantiate F. If $Q = 2q + 1$ is prime, a suitable candidate for $\mathbb{G}$ is the group of quadratic residues $\mathbb{QR}_Q \subseteq \mathbb{Z}_Q^*$, which has order q and is conjectured to be DDH-hard. Accordingly, the concrete BLS12-381 parameter discussion above does not apply to this standard-model instantiation.

6 Rate-1 Batch Laconic OT from Witness PRF

Let $\mathsf{WPRF} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{EvalWit}, \mathsf{EvalStmt})$ be a WPRF for LY commitments. The laconic OT scheme $\mathsf{IOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Rec})$ is given as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Output $\mathsf{pp} \leftarrow \mathsf{WPRF.Setup}(1^\lambda, 1^\ell)$.
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB} \in \{0,1\}^\ell)$:
  - Run $(h, P) \leftarrow \mathsf{WPRF.Hash}(\mathsf{pp}, \mathsf{DB}; r)$ for some uniform $r \leftarrow \{0,1\}^\lambda$ where $P = (\pi_1, \ldots, \pi_\ell)$
  - Output h and a state $\hat{\mathsf{D}} = (h, P)$.
- $\mathsf{Send}(\mathsf{pp}, h, (L_1, \ldots, L_k), (m_{0,i}, m_{1,i})_{i\in[k]})$:
  - Compute $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{WPRF.Gen}(\mathsf{pp})$.
  - For $b \in \{0,1\}$ and $i \in [k]$ compute $d_{b,i} \leftarrow \mathsf{WPRF.EvalStmt}(\mathsf{pp}, \mathsf{td}, h, (L_i, b))$.
  - Output $\mathsf{ct} = (\mathsf{pk}, (m_{0,i} + d_{0,i}, m_{1,i} + d_{1,i})_{i\in[k]})$.
- $\mathsf{Rec}^{\hat{\mathsf{D}}}(\mathsf{pp}, \mathsf{ct}, (L_1, \ldots, L_k))$:
  - Parse ct as $(\mathsf{pk}, (c_{0,i}, c_{1,i})_{i\in[k]})$.
  - Parse $\hat{\mathsf{D}}$ as (h, P) where $P = (\pi_1, \ldots, \pi_\ell)$.
  - For all $i \in [k]$ compute $e_i \leftarrow \mathsf{WPRF.EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (L_i, \pi_{L_i}))$.
  - Output $m_i = c_{\mathsf{DB}[L_i],i} + e_i$ for all $i \in [k]$.

6.1 Compactness, Efficiency and Rate

The scheme has the following compactness and efficiency properties:

- $|h| = \mathsf{poly}(\lambda)$ assuming that the underlying WPRF is compact.
- The Hash algorithm runs in time $|\mathsf{DB}| \cdot \mathsf{poly}(\lambda)$.
- Both Send and Rec run in time $\mathsf{poly}(\lambda)$ assuming that Rec has RAM access to $\hat{\mathsf{D}} = (\pi_1, \ldots, \pi_\ell)$.

Moreover, the sender's message is composed of $\mathsf{ct} = (\mathsf{pk}, (m_{0,i}+d_{0,i}, m_{1,i}+d_{1,i})_{i\in[k]})$ where $|\mathsf{pk}| = \mathsf{poly}(\lambda)$ and $\left|(m_{0,i} + d_{0,i}, m_{1,i} + d_{1,i})_{i\in[k]}\right| = 2k$. Hence, the rate of the sender's message is

$$\rho_{\mathsf{S}} = \frac{\mathsf{poly}(\lambda) + 2k}{k} = 2 + \frac{\mathsf{poly}(\lambda)}{k}$$

which approaches 2 for large enough k.

On the other hand, the rate of the receiver's message is $\rho_{\mathsf{R}} = \frac{\mathsf{poly}(\lambda)}{k} \approx 0$ for large enough k.

Remark 1 (Compactness tradeoff)**.** *We remark that, although the public parameters of our scheme are large (they scale linearly with ℓ) there is a simple trick to reduce the size at the cost of a larger receiver's message. Let ℓ be the size of the receiver's database $\mathsf{DB} = (\mathsf{DB}_1, \ldots, \mathsf{DB}_\mu)$ where $\mu = \sqrt{\ell}$. We generate $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\mu)$. Then the receiver hashes each block individually to produce hash values $h_1, \ldots, h_\mu$. Note that the size of the sender's ciphertext is still rate 2 in this case. This tradeoff allows us to obtain a scheme where $|h|, |\mathsf{pp}| = \sqrt{\ell} \cdot \mathsf{poly}(\lambda)$.*

6.2 Correctness

Lemma 5. *The scheme* IOT *is correct assuming that the underlying* WPRF *is correct.*

Proof. By the correctness of the WPRF we have that

$$e_i = \mathsf{WPRF.EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (L_i, \pi_{L_i})) = \mathsf{WPRF.EvalStmt}(\mathsf{pp}, \mathsf{td}, h, (L_i, \mathsf{DB}[L_i])) = d_{\mathsf{DB}[L_i],i}$$

for all $i \in [k]$. Hence

$$c_{\mathsf{DB}[L_i],i} + e_i = d_{\mathsf{DB}[L_i],i} + m_{\mathsf{DB}[L_i],i} + e_i = m_{\mathsf{DB}[L_i],i}$$

and correctness follows. □

6.3 Security

Lemma 6. *The scheme* IOT *is sender secure assuming that* WPRF *has pseudorandomness.*

Proof. We want to show that there exists a simulator Sim such that for all $\mathsf{DB} \in \{0,1\}^\ell$, all locations $L_1, \ldots, L_k$, and all pairs of messages $(m_{0,i}, m_{1,i})_{i\in[k]}$,

$$(\mathsf{pp}, \mathsf{Send}(\mathsf{pp}, h, (L_1, \ldots, L_k), (m_{0,i}, m_{1,i})_{i\in[k]})) \approx_c (\mathsf{pp}, \mathsf{Sim}(\mathsf{pp}, \mathsf{DB}, (L_1, \ldots, L_k), (m_{\mathsf{DB}[L_i],i})_{i\in[k]})).$$

We start by describing Sim.

$\mathsf{Sim}(\mathsf{pp}, \mathsf{DB}, (L_1, \ldots, L_k), (m_{\mathsf{DB}[L_i],i})_{i\in[k]})$**:**

- Sample $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{WPRF.Gen}(\mathsf{pp})$.
- For all $i \in [k]$, sample $c_{1-\mathsf{DB}[L_i],i} \leftarrow_\$ \{0,1\}$. Additionally, compute $c_{\mathsf{DB}[L_i],i} = m_{\mathsf{DB}[L_i],i} + d_{\mathsf{DB}[L_i],i}$ where $d_{\mathsf{DB}[L_i],i} \leftarrow \mathsf{WPRF.EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (L_i, \pi_{L_i}))$.

Indistinguishability between the real game and the simulated one follows from the following sequence of hybrids.

- Hybrid 0: This is the real experiment where $\mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, (L_1, \ldots, L_k), (m_{0,i}, m_{1,i})_{i\in[k]})$. That is, $\mathsf{ct} = (\mathsf{pk}, (m_{0,i} + d_{0,i}, m_{1,i} + d_{1,i})_{i\in[k]})$ where $d_{b,i} \leftarrow \mathsf{WPRF.EvalStmt}(\mathsf{pp}, \mathsf{td}, h, (L_i, b))$ for $b \in \{0,1\}$ and $i \in [k]$.
- Hybrid 1: In this hybrid, we replace $d_{\mathsf{DB}[L_i],i}$ by $\mathsf{WPRF.EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (L_i, \pi_{L_i}))$. Indistinguishability of hybrids follows from the correctness of WPRF.
- Hybrid 2: In this hybrid, we replace $d_{1-\mathsf{DB}[L_i],i}$ by a uniformly chosen value $u_i \leftarrow \{0,1\}$ for all $i \in [k]$

  Indistinguishability of hybrids follows immediately from the pseudorandomness of the underlying WPRF.
- Hybrid 3: In this hybrid, we replace $c_{1-\mathsf{DB}[L_i],i}$ by a uniform value. The hybrids are statistically indistinguishable.

Note that the last hybrid corresponds to the simulated execution of the protocol. This completes the proof. □

Lemma 7. *The scheme* IOT *is receiver secure assuming that* WPRF *is statistically hiding.*

Proof. The proof follows immediately since for any database $\mathsf{DB} \in \{0,1\}^\ell$ we have that $h \approx_s u$ where $(h, P) \leftarrow \mathsf{WPRF.Hash}(\mathsf{pp}, \mathsf{DB}; r)$ and $u \leftarrow \{0,1\}^\lambda$. □

6.4 Extension against Malicious Senders

We now sketch how the scheme presented above can be easily extended to achieve UC-malicious security against malicious senders without increasing the asymptotic communication complexity. Note that, as soon as the simulator extracts td, then it can extract all the messages from the sender. Hence, in order for the protocol to achieve UC-malicious security against the sender, we require the sender to give a non-interactive zero-knowledge (NIZK) proof that pk is well-formed and that it knows the corresponding trapdoor. Recall that NIZKs are known to exist under standard pairing-based assumptions [GOS06].

Concretely, the sender computes $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{WPRF.Gen}(\mathsf{pp})$ and commits to td using an extractable commitment scheme. It additionally proves using a NIZK scheme that "There is an opening for the commitment such that the opened value td is a valid trapdoor for pk". It sends this proof π to the receiver and $\mathsf{ct} = (\mathsf{pk}, (m_{0,i} + d_{0,i}, m_{1,i} + d_{1,i})_{i\in[k]})$. which is computed exactly as before.

First, note that the proof π is of size $\mathsf{poly}(\lambda)$ as neither pk nor td have size depending on ℓ. Hence, the size of the sender's message is not affected in the asymptotic sense. As for UC-security against malicious senders, first note that the simulator can easily extract td from π and the extractable commitment com. Having td, the simulator can now compute both $d_{0,i}$ and $d_{1,i}$ and recover both $m_{0,i}$ and $m_{1,i}$.

7 Laconic PSI with Near-Optimal Parameters

In this section, we provide a construction of laconic PSI with near-optimal parameters from a witness PRF for LY commitments.

Assume without loss of generality that the PSI universe is $\mathcal{U} = \{0,1\}^\lambda$, and assume further that the size of the receiver's set $X \subseteq \mathcal{U}$ is bounded by $N = \mathsf{poly}(\lambda)$. In the following, let $\mathsf{wgt} : \{0,1\}^\lambda \to \mathbb{N}$ be the Hamming weight function. Let $\ell = 8 \cdot N$. Let $\mathsf{WPRF} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{EvalWit}, \mathsf{EvalStmt})$ be a witness PRF for LY commitments with input domain $\Sigma = \mathcal{U}$ and output domain $\{0,1\}$. Let $\mathcal{F}$ be a family of universal hash functions from $\mathcal{U}$ to Σ.

The laconic PSI scheme $\mathsf{IPSI} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Rec})$ is given as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Output $\mathsf{pp} \leftarrow \mathsf{WPRF.Setup}(1^\lambda, 1^\ell)$.
- $\mathsf{Hash}(\mathsf{pp}, X \subseteq \mathcal{U})$:
  - For $i = 1, \ldots, \lambda$ sample $f_i \leftarrow \mathcal{F}$.
  - Populate databases $\mathsf{DB}_1, \ldots, \mathsf{DB}_\lambda \in \Sigma^\ell$ such that for each $i \in [\lambda]$ and each index $j \in [\ell]$ it either holds that $\mathsf{DB}_i[j] = x$ for some $x \in X$ with $f_i(x) = j$, or $\mathsf{DB}_i[j] = 0$ if no such $x \in X$ exists.
  - For $i = 1, \ldots, \lambda$ run $(h_i, P_i) \leftarrow \mathsf{WPRF.Hash}(\mathsf{pp}, \mathsf{DB}_i; r_i)$ for some uniform $r_i \leftarrow \{0,1\}^\lambda$ where $P_i = (\pi_{i,1}, \ldots, \pi_{i,\ell})$
  - Output $h = (f_i, h_i)_{i\in[\lambda]}$ and a state $\hat{\mathsf{D}} = (P_i)_{i\in[\lambda]}$.
- $\mathsf{Send}(\mathsf{pp}, (f_i, h_i)_{i\in[\lambda]}, Y = \{y_1, \ldots, y_k\})$:
  - Assume that the set Y is randomly shuffled.
  - Compute $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{WPRF.Gen}(\mathsf{pp})$.
  - For $j = 1, \ldots, k$ and $i = 1, \ldots, \lambda$ compute $d_{j,i} \leftarrow \mathsf{WPRF.EvalStmt}(\mathsf{pp}, \mathsf{td}, h_i, (f_i(y_j), y_j))$ and set $D_j = (d_{j,1}, \ldots, d_{j,\lambda}) \in \{0,1\}^\lambda$.
  - Output $\mathsf{ct} = (\mathsf{pk}, (D_j)_{j\in[k]})$.

- $\mathsf{Rec}^{\hat{\mathsf{D}}}(\mathsf{pp}, \mathsf{ct})$:
  - Parse ct as $\mathsf{ct} = (\mathsf{pk}, (D_j)_{j\in[k]})$.
  - Initialize a set $I = \emptyset$.
  - For all $x \in X$
    1. For $i = 1, \ldots, \lambda$ compute $e_{x,i} \leftarrow \mathsf{WPRF.EvalWit}(\mathsf{pp}, \mathsf{pk}, h_i, (f_i(x), \pi_{i,f_i(x)}))$ and set $E_x = (e_{x,1}, \ldots, e_{x,\lambda})$
    2. If there exists a $j \in [k]$ for which $\mathsf{wgt}(E_x \oplus D_j) \leq \lambda/4$, include x in I
  - Output I

7.1 Compactness, Efficiency and Rate

The scheme has the following compactness and efficiency properties:

- $|h| = \mathsf{poly}(\lambda)$ assuming that the underlying WPRF is compact, as each of $h_1, \ldots, h_\lambda$ is compact by the compactness of the witness PRF and the description of the pairwise independent hash functions $f_1, \ldots, f_\lambda$ is also $\mathsf{poly}(\lambda)$.
- The Hash algorithm runs in time $|X| \cdot \mathsf{poly}(\lambda)$.
- The Send algorithm runs in time $\mathsf{poly}(\lambda)$; furthermore, the Rec algorithm runs in time $|X| \cdot \mathsf{poly}(\lambda)$.

The sender's message is composed of $\mathsf{ct} = (\mathsf{pk}, (D_j)_{j\in[k]})$ where $|\mathsf{pk}| = \mathsf{poly}(\lambda)$ and $\left|(D_j)_{j\in[k]}\right| = k \cdot \lambda$. Hence, the rate of the sender's message is

$$\rho_{\mathsf{S}} = \frac{\mathsf{poly}(\lambda) + \lambda k}{k} = \lambda + \frac{\mathsf{poly}(\lambda)}{k}$$

which approaches λ for a large enough k. That is, the amortized bit-complexity per λ-bit sized set element in Y approaches λ.

On the other hand the rate of receiver's message is $\rho_{\mathsf{R}} = \frac{\mathsf{poly}(\lambda)}{k} \approx 0$ for large enough k.

7.2 Correctness and Security

Lemma 8. *The scheme* IPSI *is correct and secure, assuming that the underlying* WPRF *is correct and secure, and that* $\mathcal{F}$ *is a family of pairwise independent hash functions.*

Proof. First, fix an $y_j \in Y$ such that $y_j \in X$. We will show that $\mathsf{wgt}(D_j \oplus E_y) < \lambda/4$, except with negligible probability over the choice of the pairwise independent hash functions $f_1, \ldots, f_\lambda$. Hence let $D_j = (d_{j,1}, \ldots, d_{j,\lambda})$ and $E_y = (e_{y,1}, \ldots, e_{y,\lambda})$.

We will first show that for each index $i \in [\lambda]$, $f_i(y_j)$ does not collide with any $f_i(x')$ for any $x' \in X \setminus \{y_j\}$, except with probability $1/c$. Hence let Z_i be an indicator variable which is 1 if $f_i(y_j)$ does collide with an $f_i(x')$, and otherwise 0. By a union bound and the pairwise independence of f_i it holds that

$$\Pr[Z_i = 1] = \Pr[\exists x' \in X \setminus \{y_j\} : f_i(y_j) = f_i(x')] \leq N \cdot \frac{1}{8N} = 1/8.$$

Hence, in expectation $\lambda/8$ of the Z_i are 1. It thus follows by a Chernoff bound that

$$\Pr[\sum_{i=1}^{\lambda} Z_i \geq \frac{\lambda}{4}] \leq e^{-\lambda/(24)}.$$

Hence, except with negligible probability, less than $\lambda/4$ of the $f_i(y_j)$ collide with other $f_i(x')$ for some $x' \in X \setminus \{y_j\}$.

Let $L \subseteq [\lambda]$ be the set of indices i for which $f_i(y_j)$ is collision free, by the above argument $|L| > 3/4 \cdot \lambda$. By the correctness of WPRF, it holds for every $i \in L$ that $d_{j,i} = e_{y_j,i}$, that is D_j and E_{y_j} agree on more than $3/4 \cdot \lambda$ positions, or equivalently the Hamming distance between D_j and E_{y_j} is less than $\lambda/4$. We conclude that y_j is correctly included in the intersection set I.

On the other hand, if $y_j \notin X$, then it holds by the security of WPRF that each $d_{j,i}$ is pseudorandom, as for every $i \in [\lambda]$ $\mathsf{DB}_i[f_i(y_j)]$ is different from y_j (since $y_j \notin X$). Hence we can replace D_j with a uniformly random bit string of length λ. Now fix any $x \in X$. Then $D_j \oplus E_x$ is also uniform. Hence, by a standard Chernoff bound it holds that $\Pr[\mathsf{wgt}(D_j \oplus E_x) < \lambda/4] \leq e^{-\lambda/4}$, which is negligible. Now, a union bound over all $x \in X$ yields that the probability that for any $x \in X$ $\mathsf{wgt}(D_j \oplus E_x) < \lambda/4$ is at most $N \cdot e^{-\lambda/4}$, which is negligible. Hence, except with negligible probability, y_j is not included in the intersection set I. This also establishes security of IPSI, as we have just shown that the bit string D_j is pseudorandom given that $y_j \notin X$, which in particular implies that we can simulate D_j by choosing a uniformly random string. □

Acknowledgments

Pedro Branco: This work is supported by the Stellar Development Foundation. Nico Döttling: Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Sanjam Garg: This work is supported in part by the AFOSR Award FA9550-24-1-0156, and research grants from the Bakar Fund, Peder Sather Funds, the Stellar Development Foundation, Supra Inc., Category Labs, Input Output Global, and Byzantine Research Inc.

References

[ABD+21] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 94–125. Springer, Cham, November 2021. pages 4, 29

[ADM+24] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. Signature-based witness encryption with compact ciphertext. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 3–31. Springer, Singapore, December 2024. pages 6

[AFP25] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 69–100. Springer, Cham, August 2025. pages 6

[ALOS22] Diego F. Aranha, Chuanwei Lin, Claudio Orlandi, and Mark Simkin. Laconic private set-intersection from pairings. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 111–124. ACM Press, November 2022. pages 4

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor hashing for all circuits. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, page 1875–1886, New York, NY, USA, 2025. Association for Computing Machinery. pages 3

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive LWE: Attacks, variants & obfustopia. Cryptology ePrint Archive, Report 2025/375, 2025. pages 2

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 157–186. Springer, Cham, May / June 2022. pages 3, 4

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Berlin, Heidelberg, August 2010. pages 9, 10, 13

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round ot extension and silent non-interactive secure computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 291–308, New York, NY, USA, 2019. Association for Computing Machinery. pages 4

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437. Springer, Cham, December 2019. pages 3, 4

[BDJ+25] Pedro Branco, Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Pseudorandom obfuscation and applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 663–698, Cham, 2025. Springer Nature Switzerland. pages 2

[BDS23] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. A framework for statistically sender private OT with optimal rate. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 548–576. Springer, Cham, August 2023. pages 3, 4

[BDS24] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. Two-round maliciously-secure oblivious transfer with optimal rate. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 271–300. Springer, Cham, May 2024. pages 3, 4, 42

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001. pages 2, 5, 6, 10

[BFOQ25] Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. In Lujo Bauer and Giancarlo Pellegrino, editors, *USENIX Security 2025*, pages 3457–3476. USENIX Association, August 2025. pages 6

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 1–18, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. pages 2

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Berlin, Heidelberg, August 2005. pages 5

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Berlin, Heidelberg, May 2003. pages 9, 13

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Cham, November 2020. pages 6

[BÜW25] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive lwe assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 418–449, Singapore, 2025. Springer Nature Singapore. pages 2

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. pages 42

[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, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Cham, August 2017. pages 2, 3, 6, 10, 28

[CF13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Berlin, Heidelberg, February / March 2013. pages 2

[CFK24] Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 132–167. Springer, Cham, April 2024. pages 6

[CGPP24] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. pages 6

[CGPW25] Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. In Lujo Bauer and Giancarlo Pellegrino, editors, *USENIX Security 2025*, pages 3477–3495. USENIX Association, August 2025. pages 6

[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, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002. pages 2, 6

[dCP23] Leo de Castro and Chris Peikert. Functional commitments for all functions, with transparent setup and from sis. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 287–320, Cham, 2023. Springer Nature Switzerland. pages 2

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Cham, August 2017. pages 2, 6

[DGGM19] Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In David Zuckerman, editor, *60th FOCS*, pages 661–685. IEEE Computer Society Press, November 2019. pages 4, 38

[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, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Cham, March 2018. pages 3

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Berlin, Heidelberg, August 2016. pages 4

[DJM+25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 73–92. Springer, Cham, August 2025. pages 2

[DKL+23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 417–446. Springer, Cham, April 2023. pages 3, 4, 6

[FHAS24] Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for KZG commitments and efficient laconic OT. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 423–453. Springer, Singapore, December 2024. pages 6

[FKdP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 4-8, 2023, Proceedings*, Lecture Notes in Computer Science. Springer, 2023. pages 6

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990. pages 39

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. pages 2

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. pages 2

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 438–464. Springer, Cham, December 2019. pages 3, 4

[GHK+25] Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable SNARKs and applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 504–539. Springer, Cham, August 2025. pages 6

[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, Part II*, volume 11443 of *LNCS*, pages 63–93. Springer, Cham, April 2019. pages 5

[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, Part I*, volume 11239 of *LNCS*, pages 689–718. Springer, Cham, November 2018. pages 5

[GJVL23] Matthew Green, Abhishek Jain, and Gijs Van Laer. Efficient set membership encryption and applications. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1080–1092, New York, NY, USA, 2023. Association for Computing Machinery. pages 4, 6

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1065–1079. ACM Press, November 2023. pages 3, 6, 16

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 352–386. Springer, Cham, August 2024. pages 6

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Berlin, Heidelberg, May / June 2006. pages 4, 21

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012. pages 39

[GOS18] Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 515–544. Springer, Cham, August 2018. pages 3

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th FOCS*, pages 588–599. IEEE Computer Society Press, October 2017. pages 6

[GS18] Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565. Springer, Cham, April / May 2018. pages 3

[HJL25] Yao-Ching Hsieh, Aayush Jain, and Huijia Lin. Lattice-based post-quantum iO from circular security with random opening assumption (part II: zeroizing attacks against private-coin evasive LWE assumptions). Cryptology ePrint Archive, Paper 2025/390, 2025. pages 2

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015. pages 2

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 3–32. Springer, Cham, October 2021. pages 39

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000. pages 10

[KMW23] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 407–441. Springer, Singapore, December 2023. pages 6

[KS92] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992. pages 5, 10

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. pages 2

[Luc04] Stefan Lucks. Ciphers secure against related-key attacks. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 359–370. Springer, Berlin, Heidelberg, February 2004. pages 9

[LY10] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Berlin, Heidelberg, February 2010. pages 2, 6, 15

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inf. Theory*, 39(5):1639–1646, 1993. pages 10

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997. pages 9

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 678–708. Springer, Cham, October 2021. pages 4

[PPS21] Chris Peikert, Zachary Pepin, and Chad Sharp. Vector and functional commitments from lattices. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 480–511. Springer, Cham, November 2021. pages 2

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Cham, August 2019. pages 39

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, August 2008. pages 42

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018. pages 3

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022. pages 2

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Cham, December 2022. pages 2

[Wee24] Hoeteck Wee. Circuit ABE with $\mathsf{poly}(\mathsf{depth}, \lambda)$-sized ciphertexts and keys from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 178–209. Springer, Cham, August 2024. pages 3

[WW23] Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 385–416. Springer, Cham, April 2023. pages 2

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 421–448. Springer, Berlin, Heidelberg, January 2016. pages 1, 2

A Preliminaries Cont'd

A.1 Laconic Oblivious Transfer

We now define laconic oblivious transfer [CDG+17] in the batch setting.

Definition 6 (Laconic OT)**.** *A laconic oblivious transfer lOT scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda, 1^\ell)$ *takes as input a security parameter* λ *and an integer* $\ell = \mathsf{poly}(\lambda)$*. It outputs public parameters* pp*.*
- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB} \in \{0,1\}^\ell)$ *takes as input public parameters* pp *and a database* $\mathsf{DB} \in \{0,1\}^\ell$*. It outputs a hash value* h *along with a preprocessed database* $\hat{\mathsf{D}}$*.*
- $\mathsf{Send}(\mathsf{pp}, h, (L_1, \ldots, L_k), (m_{0,i}, m_{1,i})_{i\in[k]})$ *takes as input public parameters* pp*, a hash value* h*, locations* $L_1, \ldots, L_k \in [\ell]$ *and pairs of bit messages* $(m_{0,i}, m_{1,i})_{i\in[k]}$*. It outputs a sender's message* ct*.*
- $\mathsf{Rec}^{\hat{\mathsf{D}}}(\mathsf{pp}, \mathsf{ct}, (L_1, \ldots, L_k))$ *takes as input public parameters* pp*, a sender's message* ct *and locations* $L_1, \ldots, L_k \in [\ell]$*, while having RAM access to* $\hat{\mathsf{D}}$*. It outputs bit messages* $m_1, \ldots, m_k$*.*

A laconic OT should fulfill the following properties.

Correctness. *For all databases* $\mathsf{DB} \in \{0,1\}^\ell$*, all tuples of indices* $L_1, \ldots, L_k \in [\ell]$ *and all pairs of messages* $(m_{0,i}, m_{1,i})_{i\in[k]}$ *we have that*

$$\Pr\left[m_i = m_{\mathsf{DB}[L_i],i} \quad \forall i \in [k] : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, (L_1, \ldots, L_k), (m_{0,i}, m_{1,i})_{i\in[k]}) \\ (m_i)_{i\in[k]} \leftarrow \mathsf{Rec}^{\hat{\mathsf{D}}}(\mathsf{pp}, \mathsf{ct}, (L_1, \ldots, L_k)) \end{array}\right] = 1.$$

Compactness and efficiency. *We require that* $|h| = \mathsf{poly}(\lambda)$ *where* poly *is a fixed polynomial. Moreover, the algorithms* Send *and* Rec *should run in time* $\mathsf{poly}(\log|\mathsf{DB}|, \lambda)$*.*

Sender security. *There exists a simulator* Sim *such that for all* $\mathsf{DB} \in \{0,1\}^\ell$*, all locations* $L_1, \ldots, L_k$ *and all pairs of messages* $(m_{0,i}, m_{1,i})_{i\in[k]}$

$$\left\{(\mathsf{pp}, h, \mathsf{ct}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, (L_1, \ldots, L_k), \\ (m_{0,i}, m_{1,i})_{i\in[k]}) \end{array}\right\}$$
$$\approx_c \left\{(\mathsf{pp}, h, \mathsf{ct}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}) \\ \mathsf{ct} \leftarrow \mathsf{Sim}(\mathsf{pp}, \mathsf{DB}, (L_1, \ldots, L_k), \\ (m_{\mathsf{DB}[L_i],i})_{i\in[k]}) \end{array}\right\}.$$

An additional property that one may ask for laconic OT is receiver security.

Definition 7 (Receiver security)**.** *A laconic OT scheme is said to have (statistical) receiver security if for any* $\lambda \in \mathbb{N}$*, any* $\ell = \mathsf{poly}(\lambda)$ *and any two databases* $\mathsf{DB}_0, \mathsf{DB}_1 \in \{0,1\}^\ell$ *we have that*

$$\left\{(h, \mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}_0) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}_0) \end{array}\right\} \approx_s \left\{(h, \mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}_1) \leftarrow \mathsf{Hash}(\mathsf{pp}, \mathsf{DB}_1) \end{array}\right\}.$$

A.2 Laconic Private Set Intersection

We define laconic PSI, which was originally presented in [ABD+21].

Definition 8 (Laconic PSI)**.** *A laconic PSI lPSI scheme consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda, 1^\ell)$ *takes as input a security parameter* λ *and an integer* ℓ*. It outputs public parameters* pp*.*
- $\mathsf{Hash}(\mathsf{pp}, X \subseteq \mathcal{U})$ *takes as input the public parameters* pp *and a set* X*. It outputs a hash value* h *and a state* $\hat{\mathsf{D}}$*.*

- $\mathsf{Send}(\mathsf{pp}, h, Y)$ *takes as input the public parameters* pp*, a hash value* h*, and a set* Y*. It outputs a ciphertext* ct*.*
- $\mathsf{Rec}^{\hat{\mathsf{D}}}(\mathsf{pp}, \mathsf{ct})$ *takes as input the state* $\hat{\mathsf{D}}$ *and a ciphertext* ct*. It outputs a set* I*.*

A lPSI should fulfill the following properties:

- ***Correctness.*** *For all sets* $X, Y \subseteq \mathcal{U}$ *we have that*

$$\Pr\left[X \cap Y \leftarrow \mathsf{Rec}^{\hat{\mathsf{D}}}(\mathsf{pp}, \mathsf{ct}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, X) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, Y) \end{array}\right] = 1.$$

- ***Compactness.*** *We require that* $|h| = \mathsf{poly}(\lambda)$ *where* poly *is a fixed polynomial. Moreover, the algorithms* Send *should run in time* $\mathsf{poly}(\log |X|, |Y|, \lambda)$*.*
- ***Sender security.*** *For all* X *and all* Y_0, Y_1 *such that* $|Y_0| = |Y_1|$ *and* $X \cap Y_0 = X \cap Y_1$ *we have that*

$$\left\{(\mathsf{pp}, h, X, \hat{\mathsf{D}}, \mathsf{ct}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, X) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, Y_0) \end{array}\right\} \approx_c \left\{(\mathsf{pp}, h, X, \hat{\mathsf{D}}, \mathsf{ct}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, X) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, Y_1) \end{array}\right\}.$$

- ***Receiver security.*** *For all* X_0, X_1 *we have that*

$$\left\{(h, \mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, X_0) \end{array}\right\} \approx_s \left\{(h, \mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (h, \hat{\mathsf{D}}) \leftarrow \mathsf{Hash}(\mathsf{pp}, X_1) \end{array}\right\}$$

B Witness PRF with Binary Alphabet from BDHE

In this section, we provide a construction of a witness PRF scheme from the BDHE assumption and related-key-secure pseudorandom functions for group relations in the target group. In this binary variant, the witness object for position i additionally carries the opened bit together with the opening vector.

B.1 Construction

Let $k > 0$ be a parameter, called the key parameter. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ be an asymmetric bilinear group of order p, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ is a bilinear map. Let $F : \mathbb{G}_t^k \times [\ell] \to \{0,1\}^\lambda$ be a keyed function. The WPRF scheme $\mathsf{WPRF} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash}, \mathsf{EvalWit}, \mathsf{EvalStmt})$ with alphabet $\Sigma = \{0,1\}$ is given as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$:
  - Choose $\alpha \leftarrow_\$ \mathbb{Z}_p$ uniformly at random.
  - Choose $\gamma_1, \ldots, \gamma_k \leftarrow_\$ \mathbb{Z}_p$ uniformly at random
  - Compute and output the public parameters

$$\mathsf{pp} = \begin{Bmatrix} g_1, & g_1^{\alpha}, & g_1^{\alpha^2}, & \ldots, & g_1^{\alpha^\ell}, & \lrcorner, & g_1^{\alpha^{\ell+2}}, & \ldots, & g_1^{\alpha^{2\ell}}, & g_2, & g_2^{\alpha^2}, & \ldots, & g_2^{\alpha^\ell} \\ g_1^{\gamma_1}, & g_1^{\gamma_1\alpha}, & g_1^{\gamma_1\alpha^2}, & \ldots, & g_1^{\gamma_1\alpha^\ell}, & \lrcorner, & g_1^{\gamma_1\alpha^{\ell+2}}, & \ldots, & g_1^{\gamma_1\alpha^{2\ell}}, & g_2^{\gamma_1}, & g_2^{\gamma_1\alpha^2}, & \ldots, & g_2^{\gamma_1\alpha^\ell} \\ & & & & & \vdots & & & & & & & \\ g_1^{\gamma_k}, & g_1^{\gamma_k\alpha}, & g_1^{\gamma_k\alpha^2}, & \ldots, & g_1^{\gamma_k\alpha^\ell}, & \lrcorner, & g_1^{\gamma_k\alpha^{\ell+2}}, & \ldots, & g_1^{\gamma_k\alpha^{2\ell}}, & g_2^{\gamma_k}, & g_2^{\gamma_k\alpha^2}, & \ldots, & g_2^{\gamma_k\alpha^\ell} \end{Bmatrix}$$

- $\mathsf{Gen}(\mathsf{pp})$: Choose $\mathsf{td} = \xi \leftarrow_\$ \mathbb{Z}_p$ uniformly at random. Compute $\mathsf{pk} = g_2^\xi$. Output $(\mathsf{pk}, \mathsf{td})$.

- $\mathsf{Hash}(\mathsf{pp}, \mathsf{DB}; r)$:
  - $r \in \mathbb{Z}_p$ is an optional randomization term, if it is not provided set $r = 0$.
  - Compute
$$h = g_1^r \cdot \prod_{i' \in [\ell]} \left(g_1^{\alpha^{i'}}\right)^{\mathsf{DB}[i']}$$
  - For $i = 1, \ldots, \ell$ and $j = 1, \ldots, k$ compute the openings
$$\pi_{i,j} = \left(g_1^{\gamma_j \alpha^{\ell+1-i}}\right)^r \cdot \prod_{i' \in [\ell] \setminus \{i\}} \left(g_1^{\gamma_j \alpha^{\ell+1+i'-i}}\right)^{\mathsf{DB}[i']}.$$
  - For all $i \in [\ell]$ set $\boldsymbol{\pi}_i = \left(\mathsf{DB}[i], (\pi_{i,j})_{j \in [k]}\right)$.
  - Output h and $P = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell)$
- $\mathsf{EvalWit}(\mathsf{pp}, \mathsf{pk}, h, (i, \boldsymbol{\pi}_i))$
  - Parse $\boldsymbol{\pi}_i = (b_i, (\pi_{i,1}, \ldots, \pi_{i,k}))$.
  - For $j \in [k]$ compute $\kappa'_{i,j} = e(\pi_{i,j}, \mathsf{pk})^{1-2b_i}$.
  - Set $\boldsymbol{\kappa}'_i = (\kappa'_{i,j})_{j \in [k]}$
  - and $e_i = F(\boldsymbol{\kappa}'_i, (h, i))$
- $\mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td} = \xi, h, (i, b))$:
  - For $j = 1, \ldots, k$ compute $\kappa_{i,j} = \left(e\left(h, g_2^{\gamma_j \cdot \alpha^{\ell+1-i}}\right)^{\xi} \cdot e\left(g_1^{\alpha}, g_2^{\gamma_j \cdot \alpha^{\ell}}\right)^{-b \cdot \xi}\right)^{1-2b}$.
  - Set $\boldsymbol{\kappa}_i = (\kappa_{i,j})_{j \in [k]}$
  - Compute and output $d_i = F(\boldsymbol{\kappa}_i, (h, i))$.

B.2 Compactness and Efficiency

We note the following compactness properties of the scheme WPRF:

- For databases of size ℓ and a key parameter k, the public parameters pp consist of $(k+1)(3\ell+1)$ group elements
- The public key pk consists of a single group element in $\mathbb{G}_2$.
- The hash value h is a single group element in $\mathbb{G}_1$, the set of openings consists of $k\ell$ group elements in $\mathbb{G}_1$.

Again, we also observe the following efficiency properties of our scheme.

- The Gen algorithm incurs one group operation.
- Hashing incurs ℓ group operations in $\mathbb{G}_1$ to produce the hash value and $k\ell^2$ group operations to produce the openings $\pi_{i,j}$
- The witness evaluation incurs k pairing operations and one call to F.
- The statement evaluation algorithm performs $2k$ pairing operations, as well as a call to F.

B.3 Correctness

We will now establish the correctness of the scheme WPRF.

Lemma 9. *The scheme* WPRF *has perfect correctness.*

Proof. Fix a hash $h = g_1^r \cdot \prod_{i' \in [\ell]} \left(g_1^{\alpha^{i'}}\right)^{\mathsf{DB}[i']}$, $\pi_{i,j} = \left(g_1^{\gamma_j \alpha^{\ell+1-i}}\right)^r \cdot \prod_{i' \in [\ell] \setminus \{i\}} \left(g_1^{\gamma_j \alpha^{\ell+1+i'-i}}\right)^{\mathsf{DB}[i']}$, $\boldsymbol{\pi}_i = \left(\mathsf{DB}[i], (\pi_{i,j})_{j \in [k]}\right)$ and $\mathsf{pk} = g_2^\xi$, an index $i \in [\ell]$ and $b \in \{0,1\}$. It holds that

$$\begin{aligned}
\kappa_{i,j}^{1-2b} &= e\left(h, g_2^{\gamma_j \alpha^{\ell+1-i}}\right)^\xi \cdot e\left(g_1^\alpha, g_2^{\gamma_j \alpha^\ell}\right)^{-b \cdot \xi} \\
&= e\left(\left(g_1^{\gamma_j \alpha^{\ell+1-i}}\right)^r \cdot \prod_{i' \in [\ell]} \left(g_1^{\gamma_j \alpha^{\ell+1+i'-i}}\right)^{\mathsf{DB}[i']}, g_2^\xi\right) \cdot e\left(g_1, g_2\right)^{-b \cdot \xi \cdot \gamma_j \cdot \alpha^{\ell+1}} \\
&= e\left(\pi_{i,j}, \mathsf{pk}\right) \cdot e\left(g_1, g_2\right)^{(\mathsf{DB}[i]-b) \cdot \xi \gamma_j \alpha^{\ell+1}} \\
&= \left(\kappa'_{i,j}\right)^{1-2\mathsf{DB}[i]} \cdot \left(e\left(g_1, g_2\right)^{\xi \gamma_j \alpha^{\ell+1}}\right)^{\mathsf{DB}[i]-b}
\end{aligned}$$

Hence, if $b = \mathsf{DB}[i]$ it holds that $\kappa_{i,j} = \kappa'_{i,j}$ for all $j \in [k]$, i.e. $\boldsymbol{\kappa}_i = \boldsymbol{\kappa}'_i$. Consequently, it also holds that $F(\boldsymbol{\kappa}_i, i) = F(\boldsymbol{\kappa}'_i, i)$, i.e. the scheme is correct. □

B.4 Security

We will now show that the scheme WPRF has pseudorandomness.

Lemma 10. *Given that the BDHE and BDH assumptions hold in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ *and* F *is a pseudorandom function with related key security against group-induced relations, then the scheme* WPRF *has pseudorandomness.*

Proof. We have seen in the correctness proof above that $\kappa_{i,j}^{1-2b} = (\kappa'_{i,j})^{1-2\mathsf{DB}[i]} \cdot (e(g_1,g_2)^{\xi\gamma_j\alpha^{\ell+1}})^{\mathsf{DB}[i]-b}$. Consequently, we can simulate $\kappa_{i,j}$ by computing the $\kappa'_{i,j} = e(\pi_{i,j}, \mathsf{pk})^{1-2\mathsf{DB}[i]}$ and setting

$$\kappa_{i,j} = e(\pi_{i,j}, \mathsf{pk})^{(1-2b)(1-2\mathsf{DB}[i])} \cdot (e(g_1,g_2)^{\xi\alpha^{\ell+1}})^{(\mathsf{DB}[i]-b)(1-2b)\gamma_j}.$$

Now let $\mathcal{A}$ be a PPT adversary which breaks the pseudorandomness property with non-negligible advantage ε.

Let $\mathsf{DB}^{(1)}, \ldots, \mathsf{DB}^{(q)}$ be the databases provided by $\mathcal{A}_1$, and let $h^{(1)}, \ldots, h^{(q)}$ be their corresponding hashes and $\pi_{i,j}^{(\eta)}$ for $\eta = 1, \ldots, q$ be the corresponding openings. Our proof will proceed in a hybrid argument.

- Hybrid 0: This is the real experiment. However, we will let the sender compute the keys $\kappa_{i,j}^{(\eta)}$ via $\kappa_{i,j}^{(\eta)} = e(\pi_{i,j}^{(\eta)}, \mathsf{pk})^{(1-2b)(1-2\mathsf{DB}^{(\eta)}[i])} \cdot z^{(\mathsf{DB}^{(\eta)}[i]-b)(1-2b)\gamma_j}$, where $z = e(g_1,g_2)^{\xi\alpha^{\ell+1}}$. As noted above, computing the $\kappa_{i,j}^{(\eta)}$ in this way is equivalent to computing them the way the $\mathsf{EvalStmt}$ algorithm does.

- Hybrid 1: In this hybrid, we replace $z = e(g_1,g_2)^{\xi\alpha^{\ell+1}}$ with a uniformly random group element in $\mathbb{G}_t$. Computational indistinguishability follows from the BDHE assumption. In more detail, given a BDHE instance

  $$g_1, g_1^\alpha, g_1^{\alpha^2}, \ldots, g_1^{\alpha^\ell}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha^2}, \ldots, g_2^{\alpha^\ell}$$

  as well as g_2^ξ and z, where z is either $g_t^{\xi \cdot \alpha^{\ell+1}}$ or chosen uniformly random in $\mathbb{G}_t$, the reduction $\mathcal{R}_1$ proceeds as follows. It obtains the public parameters pp by choosing the $\gamma_1, \ldots, \gamma_k \leftarrow_\$ \mathbb{Z}_p$ uniformly at random and computes the $g_1^{\gamma_j \alpha^i} = (g_1^{\alpha^i})^{\gamma_j}$ for $j \in [k]$ and $i \in [2\ell] \setminus \{\ell+1\}$ and $g_2^{\gamma_j \alpha^i} = (g_2^{\alpha^i})^{\gamma_j}$ for

$j \in [k]$ and $i \in [\ell]$. Furthermore, it takes $\mathsf{pk} = g_2^{\xi}$ and the value z from the BDHE challenge. Now note that if $z = g_t^{\xi \cdot \alpha^{\ell+1}}$, then $\mathcal{R}_1$'s simulation is identically distributed to hybrid 0. On the other hand, if z is chosen uniformly random, $\mathcal{R}_1$'s simulation is identically distributed to hybrid 1. The claim follows.

- Hybrid 2: In this hybrid, we replace the z^{γ_j} with uniformly random group elements ρ_j for $j \in [k]$. Consequently, setting $\boldsymbol{\kappa}'_i = (\kappa'_{i,j})_{j\in[k]}$ and $\boldsymbol{\rho} = (\rho_j)_{j\in[k]}$, we can compute

$$\boldsymbol{\kappa}_i^{(\eta)} = \left(\kappa'_{i,j} \cdot \rho_j^{(\mathsf{DB}^{(\eta)}[i]-b)(1-2b)}\right)_{j\in[k]}.$$

  Computational indistinguishability of hybrids 1 and 2 follows from the BDH assumption via Lemma 1.

  The reduction $\mathcal{R}_2$ proceeds as follows: It receives g_1, g_2, z as input and obtains $(g_1^{\gamma_j}, g_2^{\gamma_j}, z_j)_{j\in[k]}$ by querying its oracle k times. It chooses $\alpha \leftarrow_\$ \mathbb{Z}_p$ uniformly at random, and obtains the public parameters via computing the $g_1^{\alpha^i}$ (for $i \in [2\ell]\backslash\{\ell+1\}$) and $g_2^{\alpha^i}$ (for $i \in [2\ell]$) and $g_1^{\gamma_j \alpha^i} = (g_1^{\gamma_j})^{\alpha^i}$ as well as $g_2^{\gamma_j \alpha^i} = (g_2^{\gamma_j})^{\alpha^i}$. It further chooses $\xi \leftarrow_\$ \mathbb{Z}_p$ uniformly at random and sets $\mathsf{pk} = g_2^{\xi}$. Finally, for all $j \in [k]$ it sets

$$\kappa_{i,j} = \kappa'_{i,j} \cdot z_j^{(\mathsf{DB}^{(\eta)}[i]-b)(1-2b)}.$$

  Now observe that if the z_j are of the form $z_j = g_t^{\gamma_j \cdot u}$, then $\mathcal{R}_2$'s simulation is distributed identically to hybrid 1. On the other hand, if the z_j are chosen uniformly random then $\mathcal{R}_2$'s simulation is distributed identically to hybrid 2. Hence $\mathcal{R}_2$ distinguishes BDH with non-negligible advantage ε, which contradicts the hardness of BDH.

- Hybrid 3: In this hybrid, we will choose the d_i uniformly random. This is the ideal experiment.

  Computational indistinguishability follows from the RKA-security of F. Specifically, assume that $\mathcal{D}$ distinguishes between hybrid 2 and hybrid 3 with non-negligible advantage ε. We will provide a reduction $\mathcal{R}_3$ such that $\mathcal{R}_3^{\mathcal{D}}$ distinguishes the RKA security experiment with advantage ε. $\mathcal{R}_3$ proceeds as follows. It proceeds identically to hybrid 2, except that it queries its oracle with $(\boldsymbol{\kappa}'_i, (h, i))$ and sets d_i to the output.

  In the first case, $\mathcal{R}_3$'s oracle implements $F^+(\boldsymbol{\rho}, \cdot, \cdot)$ for a uniformly chosen key $\boldsymbol{\rho} \in \mathbb{G}_t^k$. Hence the d_i are of the form $F(\boldsymbol{\kappa}'_i \odot \boldsymbol{\rho}^{(\mathsf{DB}^{(\eta)}[i]-b)(1-2b)}, (h, i))$. Hence, in this case $\mathcal{R}_3$'s simulation is identically distributed to hybrid 2.

  In the second case, $\mathcal{R}_3$'s oracle implements a uniformly random function R^+. Consequently all the d_i are distributed uniformly random. Hence, in this case $\mathcal{R}_3$'s simulation is distributed identically to hybrid 3. The claim follows.

This concludes the proof. □

We also prove that the scheme is statistically hiding.

Lemma 11. *The scheme above is statistically hiding.*

Proof. For a uniform $r \leftarrow \mathbb{Z}_p$, we have that

$$h = g_1^r \cdot \prod_{i'\in[\ell]} \left(g_1^{\alpha^{i'}}\right)^{\mathsf{DB}[i']} \approx_s g^u$$

where $u \leftarrow \mathbb{Z}_p$ is a uniform value. □

C Witness PRF for RBE

In this section, we extend the witness PRF for LY commitments by requiring not just knowledge of an opening but also knowledge of the corresponding user's secret key. The main motivation for considering WPRF for this language is its direct application to registration-based encryption.

Definition 9. *A witness PRF for RBE consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda, 1^\ell)$ *takes as input a security parameter* λ *and an integer* $\ell = \mathsf{poly}(\lambda)$*. It outputs public parameters* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}, i)$ *takes as input public parameters* pp *and an index* $i \in [\ell]$*. It outputs a pair of public and secret keys* $(\mathsf{pk}_i, \mathsf{sk}_i)$ *and hints* $(\mathsf{hint}_j)_{j\in[\ell]\setminus\{i\}}$*.*
- $\mathsf{Hash}((\mathsf{pk}_1, \mathsf{hint}_1), \ldots, (\mathsf{pk}_\ell, \mathsf{hint}_\ell))$ *takes as input pairs of public key and hint* $(\mathsf{pk}_i, \mathsf{hint}_i)_{i\in[\ell]}$*. It outputs a hash value* h *and local openings* $\pi_1, \ldots, \pi_\ell$*.*
- $\mathsf{Gen}(\mathsf{pp})$ *takes as input public parameters* pp*. It outputs a header* c *and a trapdoor* td*.*
- $\mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td}, h, i)$ *takes as input public parameters* pp*, a trapdoor* td*, a hash value* h *and an index* $i \in [\ell]$*. It outputs a key* K_i*.*
- $\mathsf{EvalWit}(\mathsf{pp}, c, (i, \pi_i, \mathsf{sk}_i))$ *takes as input public parameters* pp*, a header* c*, an index* $i \in [\ell]$*, a local opening* π_i *and a secret key* sk_i*. It outputs a key* K*.*

A witness PRF for RBE should fulfill the following properties.

Compactness of public parameters. *We require that* $|h|, |\pi|, |\mathsf{sk}_i| = \mathsf{poly}(\lambda)$ *where* poly *is a fixed polynomial. Moreover, we require the algorithms* Gen*,* $\mathsf{EvalStmt}$ *and* $\mathsf{EvalWit}$ *to run in* $\mathsf{poly}(\log \ell, \lambda)$*.*

Correctness. *For all* $\ell = \mathsf{poly}(\lambda)$ *and all* $i \in [\ell]$ *we have that*

$$\Pr\left[K = K_i : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ (\mathsf{pk}_j, \mathsf{sk}_j, \mathsf{hint}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, j) \quad \forall j \in [\ell] \\ \left(h, (\pi_j)_{j\in[\ell]}\right) \leftarrow \mathsf{Hash}\left(\left(\mathsf{pk}_j, \mathsf{hint}_j\right)_{j\in[\ell]}\right) \\ (c, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}) \\ K_i \leftarrow \mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td}, h, i) \\ K \leftarrow \mathsf{EvalWit}(\mathsf{pp}, c, (i, \pi_i, \mathsf{sk}_i)) \end{array}\right] = 1.$$

Security. *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, consider the following experiment:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$*.*
- $(S, \mathsf{st}, \{\mathsf{pk}_i, \mathsf{hint}_i, r_i\}_{i\notin S}) \leftarrow \mathcal{A}_1(\mathsf{pp})$ *where* S *is a set of target indices of size* k*, and* r_i *are random coins used in the generation of* pk_i
- *For all* $i \notin S$ *check if* $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{hint}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i; r_i)$*.*
- *For all* $j \in S$ *compute* $(\mathsf{pk}_j, \mathsf{sk}_j, \mathsf{hint}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, j; r_j)$ *with fresh random coins* r_j*.*
- *Compute* $(h, \pi_1, \ldots, \pi_\ell) \leftarrow \mathsf{Hash}((\mathsf{pk}_1, \mathsf{hint}_1), \ldots, (\mathsf{pk}_\ell, \mathsf{hint}_\ell))$*.*
- *Sample* $b \leftarrow \{0, 1\}$*.*
- *If* $b = 0$*, then compute* $(c, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp})$ *and* $K_i \leftarrow \mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td}, h, i)$ *for all* $i \in S$*.*
- *If* $b = 1$*, then compute* $(c, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp})$ *and* $K_i \leftarrow \{0, 1\}^\lambda$ *for all* $i \in S$*.*
- $b' \leftarrow \mathcal{A}_2(\mathsf{st}, \{\mathsf{pk}_i, \mathsf{hint}_i, K_i\}_{i\in S})$

We say that a witness PRF for RBE is pseudorandom if for all PPT adversaries $\mathcal{A}$ *we have that* $|\Pr[b = b'] - 1/2| \leq \mathsf{negl}(\lambda)$*.*

C.1 Construction

We now adapt the bilinear core of Section 5 to obtain a witness PRF for RBE. In contrast to the main WPRF construction, this section does not require the related-key-secure PRF used there to obtain batch pseudorandomness.

The witness PRF for RBE $\mathsf{WRBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Hash}, \mathsf{Gen}, \mathsf{EvalStmt}, \mathsf{EvalWit})$ is given as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$:
  - Choose $\alpha \leftarrow_\$ \mathbb{Z}_p$ uniformly at random.
  - Compute and output the public parameters

$$\mathsf{pp} = \left\{ g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^\ell}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha^2}, \ldots, g_2^{\alpha^\ell} \right\}$$

- $\mathsf{KeyGen}(\mathsf{pp}, i)$: Choose $\mathsf{sk}_i = \gamma_i \leftarrow_\$ \mathbb{Z}_p$. Set $\mathsf{pk}_i \leftarrow \left(g_1^{\alpha^i}\right)^{\gamma_i}$ and $\mathsf{hint}_i = \left(\left(g_1^{\alpha^j}\right)^{\gamma_i}\right)_{j \in [\ell] \setminus \{i\}}$. Output $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{hint}_i)$.
- $\mathsf{Hash}((\mathsf{pk}_1, \mathsf{hint}_1), \ldots, (\mathsf{pk}_\ell, \mathsf{hint}_\ell))$:
  - For all $i \in [\ell]$ parse $\mathsf{hint}_i = \left(g_1^{\gamma_i \alpha^j}\right)_{j \in [\ell] \setminus \{i\}}$
  - Compute

$$h = \prod_{i' \in [\ell]} \mathsf{pk}_{i'}$$

  - For $i = 1, \ldots, \ell$ compute the openings $\pi_i = \prod_{i' \in [\ell] \setminus \{i\}} \left(g_1^{\gamma_{i'} \alpha^{\ell+1+i'-i}}\right)$
  - Output h and $\pi_1, \ldots, \pi_\ell$.
- $\mathsf{Gen}(\mathsf{pp})$: Choose $\mathsf{td} = \xi \leftarrow_\$ \mathbb{Z}_p$ uniformly at random. Compute $c_0 = g_2^\xi$ and $c_1 = e\left(g_1^\alpha, g_2^{\alpha^\ell}\right)^\xi$. Set $c = (c_0, c_1)$. Output (c, td).
- $\mathsf{EvalStmt}(\mathsf{pp}, \mathsf{td} = \xi, h, i)$: Compute and output $K_i = e\left(h, g_2^{\alpha^{\ell+1-i}}\right)^\xi$.
- $\mathsf{EvalWit}\,(\mathsf{pp}, c = (c_0, c_1), (i, \pi_i, \mathsf{sk}_i = \gamma_i))$: Compute and output $K_i' = e(\pi_i, c_0) \cdot c_1^{\gamma_i}$.

C.2 Compactness, Efficiency and Rate

The scheme is compact since h and π_i each consist of a single group element, that is, $|h|, |\pi_i| = \mathsf{poly}(\lambda)$. Additionally, the algorithms Gen, $\mathsf{EvalStmt}$ and $\mathsf{EvalWit}$ run in time independent of ℓ.

C.3 Correctness

We will now argue that the scheme WRBE is correct.

Lemma 12. *The scheme* WRBE *has perfect correctness.*

Proof. We will show that the public and private evaluations of the PRF are identical for each identity i. Let

- $h = \prod_{i' \in [\ell]} \mathsf{pk}_{i'} = \prod_{i' \in [\ell]} g_1^{\gamma_{i'} \alpha^{i'}}$
- $\pi_i = \prod_{i' \in [\ell] \setminus \{i\}} g_1^{\gamma_{i'} \alpha^{\ell+1+i'-i}}$

- $c_0 = g_2^{\xi}$ and $c_1 = e\left(g_1, g_2\right)^{\xi\alpha^{\ell+1}}$

It holds that

$$\begin{aligned}
K_i &= e\left(h, g_2^{\alpha^{\ell+1-i}}\right)^{\xi} \\
&= e\left(\prod_{i' \in [\ell]} g_1^{\gamma_{i'}\alpha^{i'}}, g_2^{\alpha^{\ell+1-i}}\right)^{\xi} \\
&= e\left(g_1, g_2\right)^{\xi \cdot \sum_{i' \in [\ell]} \gamma_{i'}\alpha^{\ell+1+i'-i}} \\
&= e\left(g_1, g_2\right)^{\xi \cdot \sum_{i' \in [\ell]\setminus\{i\}} \gamma_{i'}\alpha^{\ell+1+i'-i}} \cdot e\left(g_1, g_2\right)^{\xi\gamma_i \cdot \alpha^{\ell+1}} \\
&= e\left(g_1^{\sum_{i' \in [\ell]\setminus\{i\}} \gamma_{i'}\alpha^{\ell+1+i'-i}}, g_2^{\xi}\right) \cdot c_1^{\gamma_i} \\
&= e\left(\prod_{i' \in [\ell]\setminus\{i\}} g_1^{\gamma_{i'}\alpha^{\ell+1+i'-i}}, c_0\right) \cdot c_1^{\gamma_i} \\
&= e\left(\pi_i, c_0\right) \cdot c_1^{\gamma_i} \\
&= K_i'.
\end{aligned}$$

We conclude that the scheme has perfect correctness. □

C.4 Security

We will now show that the scheme WRBE satisfies pseudorandomness. The proof in this section proceeds directly from the BDHE and BDH assumptions and does not invoke the related-key-security step from the main WPRF construction.

Lemma 13. *Given that the BDHE and BDH assumptions hold in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$*, the scheme* WRBE *has pseudorandomness.*

Proof. Let $\mathcal{A}$ be a PPT-adversary with non-negligible advantage ε against the pseudorandomness experiment. Consider the following sequence of hybrids.

- Hybrid 0: This is the real experiment. However, we compute the keys K_i via the EvalWit algorithm, i.e. we set $K_i = e(\pi_i, c_0) \cdot c_1^{\gamma_i}$, where $c_1 = e(g_1, g_2)^{\xi\alpha^{\ell+1}}$. We can do this because the scheme is perfectly correct.

- Hybrid 1: This hybrid is identical to hybrid 0, except that we choose c_1 uniformly at random. Computational indistinguishability follows from the BDHE assumption. Specifically, given a BDHE challenge

$$\left(g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^{\ell}}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^{\ell}}, g_2^{\xi}, z\right),$$

  where either $z = g_t^{\xi \cdot \alpha^{\ell+1}}$ or z is chosen uniformly random from $\mathbb{G}_t$, the reduction $\mathcal{R}_1$ proceeds as follows. It assembles the public parameters pp from the corresponding elements in the BDHE challenge, sets $c_0 = g_2^{\xi}$ and $c_1 = z$. Otherwise it runs hybrid 0 unmodified.

  Now note that if $z = g_t^{\xi \cdot \alpha^{\ell+1}}$, then $\mathcal{R}_1$'s simulation is distributed identically to hybrid 0. On the other hand, if z is chosen uniformly random, then $\mathcal{R}_1$'s simulation is distributed identically to hybrid 1. The claim follows.

- Hybrid 2: This hybrid is distributed identically to hybrid 1, except that for all $i \in S$ we choose K_i independently uniformly random. This is the ideal experiment.

  Computational indistinguishability follows from the BDH assumption. Specifically, we will again rely on the version of BDH provided in Lemma 1. Hence let $(g_1, g_2, z, (g_1^{\gamma_j}, g_2^{\gamma_j})_{j\in[k]}, (z_j)_{j\in[k]})$ be a challenge, where either $z_j = z^{\gamma_j}$ for all $j \in [k]$ for a uniformly random $z \leftarrow_\$ \mathbb{G}_t$ or all z_j are chosen independently uniformly random in $\mathbb{G}_t$. The reduction $\mathcal{R}_2$ proceeds as follows. It first generates the public parameters pp as in hybrid 1, i.e. by choosing $\alpha \leftarrow_\$ \mathbb{Z}_p$ uniformly random and setting

$$\mathsf{pp} = \left(g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^\ell}, _, g_1^{\alpha^{\ell+2}}, \ldots, g_1^{\alpha^{2\ell}}, g_2, g_2^{\alpha}, g_2^{\alpha^2}, \ldots, g_2^{\alpha^\ell}\right).$$

  It then runs $\mathcal{A}_1(\mathsf{pp})$ and obtains $(S, \mathsf{st}, \{\mathsf{pk}_i, \mathsf{hint}_i, r_i\}_{i\notin S})$, where we write $S = \{i_1, \ldots, i_k\}$. For each $i \notin S$, it runs $\mathsf{KeyGen}(\mathsf{pp}, i; r_i)$ to recover $(\mathsf{pk}_i, \mathsf{sk}_i = \gamma_i, \mathsf{hint}_i)$. For each target index $i_j \in S$, it sets

$$\mathsf{pk}_{i_j} = \left(g_1^{\gamma_j}\right)^{\alpha^{i_j}} \quad \text{and} \quad \mathsf{hint}_{i_j} = \left(\left(g_1^{\gamma_j}\right)^{\alpha^m}\right)_{m\in[\ell]\setminus\{i_j\}}.$$

  Using these values it computes $(h, \pi_1, \ldots, \pi_\ell) \leftarrow \mathsf{Hash}\left((\mathsf{pk}_1, \mathsf{hint}_1), \ldots, (\mathsf{pk}_\ell, \mathsf{hint}_\ell)\right)$. Moreover, it sets $c_0 = g_2^{\xi}$ for a $\xi \leftarrow_\$ \mathbb{Z}_p$ chosen uniformly random and, for every target index $i_j \in S$, it sets

$$K_{i_j} = e(\pi_{i_j}, c_0) \cdot z_j.$$

  Otherwise, it proceeds as hybrid 1.

  First assume that it holds that $z_j = z^{\gamma_j}$ for all $j \in [k]$ where $z \leftarrow_\$ \mathbb{G}_t$ is chosen uniformly random. Then it follows that $\mathcal{R}_2$'s simulation is distributed identically to hybrid 1: for every target index $i_j \in S$, the public key pk_{i_j} and hint hint_{i_j} are distributed exactly as in KeyGen, and the corresponding challenge key satisfies

$$K_{i_j} = e(\pi_{i_j}, c_0) \cdot z^{\gamma_j},$$

  which is exactly the hybrid-1 distribution with $c_1 = z$. On the other hand, if all z_j are chosen independently uniformly random, then all keys K_{i_j} for $i_j \in S$ are independently uniformly random as well. It follows that in this case $\mathcal{R}_2$'s simulation is distributed identically to hybrid 2.

This concludes the proof. □

C.5 Tradeoffs and Rate-1 RBE

We now discuss a few interesting properties of our scheme.

Rate-1 batch RBE. Having a witness PRF for RBE as described above immediately implies RBE, since the algorithms are identical. To see that the resulting RBE scheme has rate-1 ciphertexts, note that the encryptor sends only a single ciphertext $c = (c_0, c_1) \leftarrow \mathsf{Gen}(\mathsf{pp})$. This allows multiple receivers to derive different shared keys. Assume that the encryptor wants to send k different messages to k different receivers $i_1, \ldots, i_k$. It simply derives the shared keys $K_{i_1}, \ldots, K_{i_k}$ for these receivers and then sends, e.g., $\mathsf{ct} = (c, m_1 + \mathsf{LSB}(K_{i_1}), \ldots, m_k + \mathsf{LSB}(K_{i_k}))$, where LSB outputs the least significant bit. The rate of the ciphertext is

$$\rho_{\mathsf{ct}} = \frac{|c| + k}{k} = 1 + \frac{\mathsf{poly}(\lambda)}{k}$$

which approaches 1 for large enough k.

Reducing the size of the public parameters. Similarly to the case of laconic OT, we can apply the same trick as in Remark 1 to obtain an RBE with public parameters and hash value growing with $\sqrt{\ell}$.

D Malicious OT with Optimal Rate

In this section, we show that rate-1 LOT with malicious sender security implies a two-round malicious OT with optimal rate. First, we define laconic conditional disclosure of secrets as in [DGGM19].

Definition 10 (Laconic CDS)**.** *A laconic conditional disclosure of secrets (LCDS) scheme for an NP language $\mathcal{L}$ (with relation $\mathcal{R}$) consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input a security parameter λ. It outputs public parameters* pp.
- $\mathsf{Hash}(\mathsf{pp}, x, w)$ *takes as input public parameters* pp*, a statement x and a witness w. It outputs a hash value h and a state* st.
- $\mathsf{Send}(\mathsf{pp}, h, x, m)$ *takes as input public parameters* pp*, a hash value h, a statement x, and a message m. It outputs a sender's message* ct.
- $\mathsf{Rec}(\mathsf{pp}, \mathsf{ct}, \mathsf{st})$ *takes as input public parameters* pp*, a sender's message* ct *and a state* st*. It outputs a message m.*

An LCDS fulfills the following properties.

Correctness. *For all $\lambda \in \mathbb{N}$, all pairs (x, w) such that $\mathcal{R}(x, w) = 1$ and all messages $m \in \{0,1\}$ we have that*

$$\Pr\left[m \leftarrow \mathsf{Rec}(\mathsf{pp}, \mathsf{ct}, \mathsf{st}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (h, \mathsf{st}) \leftarrow \mathsf{Hash}(\mathsf{pp}, x, w) \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, x, m) \end{array}\right] = 1.$$

Message indistinguishability. *For all $\lambda \in \mathbb{N}$, all $x \notin \mathcal{L}$ and all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have that*

$$\left|\Pr\left[b \leftarrow \mathcal{A}_2(\mathsf{st}, \mathsf{ct}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (h, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ b \leftarrow \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{pp}, h, x, m_b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

Receiver simulation. *For all $\lambda \in \mathbb{N}$, all pairs (x, w) such that $\mathcal{R}(x, w) = 1$, there exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for all PPT adversaries $\mathcal{A}$ we have that*

$$\left|\begin{array}{c} \Pr\left[b \leftarrow \mathcal{A}_1(\mathsf{pp}, h)) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (h, \mathsf{st}) \leftarrow \mathsf{Hash}(\mathsf{pp}, x, w) \end{array}\right] - \\ \Pr\left[b \leftarrow \mathcal{A}_1(\mathsf{pp}, h)) : \begin{array}{c} (\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{Sim}_1(1^\lambda) \\ h \leftarrow \mathsf{Sim}_2(\mathsf{pp}, x, \mathsf{td}) \end{array}\right] \end{array}\right| \leq \mathsf{negl}(\lambda)$$

Efficiency. *We require that i) h is of size $\mathsf{poly}(\lambda)$, ii) The* Send *runs in time $\mathsf{poly}(\log|w|, |x|, |m|)$, and iii) The* ct *output by the* Send *algorithm is of size $\mathsf{poly}(\log|w|, \log|x|, m)$. We note that laconic CDS with this efficiency was presented in [DGGM19].*

We will also need an extractable commitment scheme.

Definition 11 (Commitment scheme)**.** *An extractable commitment scheme is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open})$ *such that:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input a security parameter λ and outputs a* crs.
- $\mathsf{Com}(\mathsf{crs}, m)$ *takes as input a* crs *and a message m. It outputs a commitment* com *and an opening γ.*
- $\mathsf{Open}(\mathsf{com}, m, \gamma)$ *takes as input a commitment* com*, a message m and an opening γ. It outputs a bit b.*

An extractable commitment scheme fulfills the following properties.

Correctness. *For all $\lambda \in \mathbb{N}$, all messages m we have that*

$$\Pr\left[1 \leftarrow \mathsf{Open}(\mathsf{com}, m, \gamma) : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{com}, \gamma) \leftarrow \mathsf{Com}(\mathsf{crs}, m) \end{array}\right] = 1.$$

Extractability. *There exists a pair of algorithms $(\mathsf{ExtSetup}, \mathsf{Ext})$ such that for all $\lambda \in \mathbb{N}$ we have that*

$$\{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{ExtSetup}(1^\lambda)\} \approx_c \{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\}$$

and for all PPT adversaries $\mathcal{A}$ we have that

$$\Pr\left[m \neq m' \wedge 1 \leftarrow \mathsf{Open}(\mathsf{com}, m, \gamma) : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{ExtSetup}(1^\lambda) \\ (\mathsf{com}, m, \gamma) \leftarrow \mathcal{A}(\mathsf{crs}) \\ m' \leftarrow \mathsf{Ext}(\mathsf{com}, \mathsf{td}) \end{array}\right] = \mathsf{negl}(\lambda).$$

Hiding. *For all $\lambda \in \mathbb{N}$ and all pairs of message (m, m') we have that*

$$\left\{(\mathsf{crs}, \mathsf{com}) : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{com}, \gamma) \leftarrow \mathsf{Com}(\mathsf{crs}, m) \end{array}\right\} \approx_c \left\{(\mathsf{crs}, \mathsf{com}') : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{com}', \gamma') \leftarrow \mathsf{Com}(\mathsf{crs}, m') \end{array}\right\}.$$

Our last building block are non-interactive zero-knowledge proof systems.

Definition 12 (Non-interactive zero-knowledge)**.** *A non-interactive zero-knowledge (NIZK) argument of knowledge (AoK) for an* NP *language $\mathcal{L}$, with corresponding instance-witness relation $\mathcal{R} : \mathcal{X} \times \mathcal{W} \to \{0,1\}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ such that:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter 1^λ and outputs a common reference string* crs*.*
- $\mathsf{Prove}(\mathsf{crs}, x, w)$ *takes as input a common reference string* crs*, a statement $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$. It outputs a proof π.*
- $\mathsf{Verify}(\mathsf{crs}, x, \pi)$ *takes as input a common reference string* crs*, a statement $x \in \mathcal{X}$ and a proof π. It outputs a bit $b \in \{0,1\}$.*

A NIZK-AoK satisfies the following properties:

- **Completeness.** For every $x \in \mathcal{X}$ and $w \in \mathcal{W}$ such that $\mathcal{R}(x, w) = 1$, we have:

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

- **Argument of Knowledge.** There exists a PPT extractor $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$ such that for every non-uniform PPT prover Prove^*, we have:

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \wedge \mathcal{R}(x, w) = 0 : \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Ext}_1(1^\lambda), \\ (x, \pi) \leftarrow \mathsf{Prove}^*(\mathsf{crs}), \\ w \leftarrow \mathsf{Ext}_2(\mathsf{td}, x, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- **Zero-knowledge.** There exists a PPT simulator Sim such that for any $x \in \mathcal{X}$ and $w \in \mathcal{W}$ such that $\mathcal{R}(x, w) = 1$, we have:

$$\left\{(\mathsf{crs}, \pi) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\right\} \approx_c \left\{(\mathsf{crs}, \pi) : (\mathsf{crs}, \pi) \leftarrow \mathsf{Sim}(1^\lambda, x)\right\}.$$

  If the two distributions described above are *statistically* indistinguishable then we say that the NIZK achieves statistical zero-knowledge.

NIZK argument of knowledge can be constructed from a NIZK and an extractable commitment. Specifically, we commit to the witness using an extractable commitment and prove that the committed value is a valid witness. This construction satisfies both zero-knowledge and argument of knowledge properties. NIZK arguments are known from factoring [FLS90], or from bilinear maps [GOS12], or from LWE [PS19], or sub-exponential DDH [JJ21].

D.1 Construction

We now show how to generically build a malicious OT scheme with optimal rate using the following ingredients.

Ingredients. We will use the following ingredients:

- A rate-1 LOT scheme $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Rec})$ with UC-malicious security against senders.
- A PRF $\mathsf{PRF}_K : \{0,1\}^\lambda \to \{0,1\}$.
- A PRG $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\mathsf{poly}(\lambda)+2\ell}$.
- An extractable commitment scheme $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open})$.
- A laconic CDS scheme $\mathsf{LCDS} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Rec})$ for the language

$$\mathcal{L} = \left\{ (h_{\mathsf{LOT}}, \mathsf{hk}, \mathsf{DB}', \mathsf{com}) : \exists (K, \gamma) \text{ s.t. } \begin{array}{c} 1 \leftarrow \mathsf{Com.Open}(\mathsf{com}, K, \gamma) \\ h_{\mathsf{LOT}} \leftarrow \mathsf{LOT.Hash}\left(\mathsf{hk}, \mathsf{DB}' + \mathsf{PRF}_K\left([\ell]\right)\right) \end{array} \right\}$$

  where $\mathsf{PRF}_K([\ell]) = (\mathsf{PRF}_K(1), \ldots, \mathsf{PRF}_K(\ell))$ and $\mathsf{DB}' \in \{0,1\}^\ell$.
- A NIZK $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for the language

$$\mathcal{L}_{\mathsf{LCDS}} = \{(\mathsf{ct}_{\mathsf{LCDS}}, \mathsf{pp}_{\mathsf{LCDS}}, h_{\mathsf{LCDS}}, h) : \exists S \text{ s.t. } \mathsf{ct}_{\mathsf{LCDS}} \leftarrow \mathsf{LCDS.Send}(\mathsf{pp}_{\mathsf{LCDS}}, h_{\mathsf{LCDS}}, h, S)\}$$

  which guarantees that a certain laconic CDS sender's message is well-formed.

Construction. The OT scheme $\mathsf{OT} = (\mathsf{Setup}, \mathsf{Rec}_1, \mathsf{Send}, \mathsf{Rec}_2)$ is given as follows.

$\mathsf{Setup}(1^\lambda)$**:**

- Run $\mathsf{pp}_{\mathsf{LOT}} \leftarrow \mathsf{LOT.Setup}(1^\lambda, 1^\ell)$, $\mathsf{pp}_{\mathsf{CDS}} \leftarrow \mathsf{LCDS.Setup}(1^\lambda)$, $\mathsf{crs} \leftarrow \mathsf{CS.Setup}(1^\lambda)$ and $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
- Output $\mathsf{pp} = (\mathsf{pp}_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{CDS}}, \mathsf{crs}, \mathsf{crs}_{\mathsf{NIZK}})$.

$\mathsf{Rec}_1(\mathsf{pp}, (b_1, \ldots, b_\ell))$**:**

- Parse pp as $(\mathsf{pp}_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{CDS}}, \mathsf{crs}, \mathsf{crs}_{\mathsf{NIZK}})$.
- Sample a PRF key $K \leftarrow \{0,1\}^\lambda$ and compute $(\mathsf{com}, \gamma) \leftarrow \mathsf{CS.Com}(\mathsf{crs}, K)$.
- Set $\mathbf{c} = \mathbf{b} + \mathsf{PRF}_K([\ell])$.
- Compute $(h_{\mathsf{LOT}}, \hat{\mathbf{b}}) \leftarrow \mathsf{LOT.Hash}(\mathsf{pp}_{\mathsf{LOT}}, \mathbf{b})$, where $\mathbf{b} = (b_1, \ldots, b_\ell)$.
- Compute $(h_{\mathsf{CDS}}, \mathsf{st}_{\mathsf{CDS}}) \leftarrow \mathsf{LCDS.Hash}(\mathsf{pp}_{\mathsf{CDS}}, (h_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{LOT}}, \mathbf{c}, \mathsf{com}), (K, \gamma))$.
- Output $\mathsf{ot}_1 = (h_{\mathsf{LOT}}, h_{\mathsf{CDS}}, \mathsf{com}, \mathbf{c})$ and $\mathsf{st} = (\hat{\mathbf{b}}, \mathsf{st}_{\mathsf{CDS}})$.

$\mathsf{Send}(\mathsf{pp}, \mathsf{ot}_1, (m_{0,i}, m_{1,i})_{i\in[\ell]})$**:**

- Parse pp as $(\mathsf{pp}_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{CDS}}, \mathsf{crs}, \mathsf{crs}_{\mathsf{NIZK}})$ and ot_1 as $(h_{\mathsf{LOT}}, h_{\mathsf{CDS}}, \mathsf{com}, \mathbf{c})$.
- Compute $\mathsf{ct}_{\mathsf{LOT}} \leftarrow \mathsf{LOT.Send}(\mathsf{pp}_{\mathsf{LOT}}, h_{\mathsf{LOT}}, [\ell], (m_{0,i}, m_{1,i})_{i\in[\ell]})$.
- Sample a PRG seed $S \leftarrow \{0,1\}^\lambda$ and set $\mathsf{ct} = \mathsf{ct}_{\mathsf{LOT}} + \mathsf{PRG}(S)$.
- Compute $\mathsf{ct}_{\mathsf{LCDS}} \leftarrow \mathsf{LCDS.Send}(\mathsf{pp}_{\mathsf{LCDS}}, h_{\mathsf{LCDS}}, (h_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{LOT}}, \mathbf{c}, \mathsf{com}), S)$. Compute $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, x, w)$ where $x = (\mathsf{ct}_{\mathsf{LCDS}}, \mathsf{pp}_{\mathsf{LCDS}}, h_{\mathsf{LCDS}}, h)$ and $w = S$.
- Output $\mathsf{ot}_2 = (\mathsf{ct}, \mathsf{ct}_{\mathsf{LCDS}}, \pi)$.

$\mathsf{Rec}_2(\mathsf{ot}_2, \mathsf{st})$:

- Parse ot_2 as $(\mathsf{ct}, \mathsf{ct}_{\mathsf{LCDS}})$ and st as $(\hat{\mathbf{b}}, \mathsf{st}_{\mathsf{CDS}})$.
- Check if $1 \leftarrow \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, x, \pi)$.
- Retrieve $S \leftarrow \mathsf{LCDS.Rec}(\mathsf{pp}_{\mathsf{LCDS}}, \mathsf{ct}_{\mathsf{LCDS}}, \mathsf{st}_{\mathsf{CDS}})$.
- Compute $\mathsf{ct}_{\mathsf{LOT}} = \mathsf{ct} - \mathsf{PRG}(S)$ and $(m_i)_{i\in[\ell]} \leftarrow \mathsf{LOT.Rec}^{\hat{\mathbf{b}}}(\mathsf{pp}_{\mathsf{LOT}}, \mathsf{ct}_{\mathsf{LOT}}, [\ell])$.
- Output $(m_i)_{i\in[\ell]}$.

D.2 Correctness and Efficiency

We first prove correctness of the construction.

Lemma 14 (Correctness)**.** *The scheme* OT *is correct.*

Proof. Let $\mathsf{ot}_2 = (\mathsf{ct}, \mathsf{ct}_{\mathsf{LCDS}}, \pi)$ be the message sent by the sender. Assuming that the NIZK is complete, we have that $1 \leftarrow \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, x, \pi)$. Hence, the receiver does not abort in this step. Moreover, assuming that the underlying LCDS is correct, we have that $S \leftarrow \mathsf{LCDS.Rec}(\mathsf{pp}_{\mathsf{LCDS}}, \mathsf{ct}_{\mathsf{LCDS}}, \mathsf{st}_{\mathsf{CDS}})$ and hence $\mathsf{ct}_{\mathsf{LOT}} = \mathsf{ct} - \mathsf{PRG}(S)$. Thus, by the correctness of the underlying LOT, we have that $(m_{b_i,i})_{i\in[\ell]} \leftarrow \mathsf{LOT.Rec}^{\hat{\mathbf{b}}}(\mathsf{pp}_{\mathsf{LOT}}, \mathsf{ct}_{\mathsf{LOT}}, [\ell])$. □

We now analyze the rate of our scheme starting with the size of the receiver's message.

Receiver's message. The receiver's message is composed of $\mathsf{ot}_1 = (h_{\mathsf{LOT}}, h_{\mathsf{CDS}}, \mathsf{com}, \mathbf{c})$ where

- $|h_{\mathsf{LOT}}| = \mathsf{poly}(\lambda)$.
- $|h_{\mathsf{LCDS}}| = \mathsf{poly}(\lambda)$.
- $|\mathsf{com}| = \mathsf{poly}(\lambda)$ as this is a commitment to $K \in \{0,1\}^\lambda$.
- $|\mathbf{c}| = \ell$.

Hence, the upload rate ρ_{up} of the scheme is

$$\rho_{\mathsf{up}} = \frac{|\mathsf{ot}_1|}{\ell} = 1 + \frac{\mathsf{poly}(\lambda)}{\ell}$$

which approaches 1 for large enough ℓ.

Sender's message. The sender's message is composed of $\mathsf{ot}_2 = (\mathsf{ct}, \mathsf{ct}_{\mathsf{LCDS}}, \pi)$ where

- $|\mathsf{ct}| = |\mathsf{ct}_{\mathsf{LOT}}| = \mathsf{poly}(\lambda) + 2\ell$ since the underlying LOT scheme is rate-1.
- $|\mathsf{ct}_{\mathsf{LCDS}}| = \mathsf{poly}(\lambda)$ since it only scales with the size of $S \in \{0,1\}^\lambda$.
- $|\pi| = \mathsf{poly}(\lambda)$ since it only scales with the size of the circuit $\mathsf{LCDS.Send}$.

Hence, the download rate ρ_{down} of the scheme is

$$\rho_{\mathsf{down}} = \frac{|\mathsf{ot}_2|}{2\ell} = 2 + \frac{\mathsf{poly}(\lambda)}{\ell}$$

which approaches 2 for large enough ℓ.

D.3 Security

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 an 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 13. *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 define the OT ideal functionality as in [PVW08, BDS24].

OT functionality. A (batch) OT scheme should implement the following functionality.

- **Receiver's input:** A string of bits $\mathbf{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$.
- **Sender's input:** Pairs of messages $(\mathbf{m}_0, \mathbf{m}_1) \in (\{0,1\}^k)^2$, where $\mathbf{m}_0 = (m_{0,1}, \ldots, m_{0,k})$ and $\mathbf{m}_1 = (m_{1,1}, \ldots, m_{1,k})$
- **Receiver's output:** The string $\mathbf{m_b} = (m_{b_1,1}, \ldots, m_{b_k,k})$.

We now show security of our scheme. We start by showing security against malicious senders.

Lemma 15 (Security against malicious senders)**.** *The scheme OT is UC-secure against malicious senders.*

Proof. Recall that the underlying LOT scheme is UC-malicious secure against senders. Let $\mathsf{Sim}_{\mathsf{LOT}}$ be the corresponding simulator. Additionally, let $\mathsf{Sim}_{\mathsf{LCDS}}$ be the simulator of the laconic CDS. We will start by describing the simulator $\mathsf{Sim}_{\mathsf{S}}$ against malicious senders.

$\mathsf{Sim}_{\mathsf{S}}(1^\lambda)$:

- Set the $\mathsf{pp} = (\mathsf{pp}_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{LCDS}}, \mathsf{crs}, \mathsf{crs}_{\mathsf{NIZK}})$ where i) both $\mathsf{pp}_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{LCDS}}$ are in simulation mode, and ii) $\mathsf{crs}_{\mathsf{NIZK}}$ is in extraction mode.
- Use $\mathsf{Sim}_{\mathsf{LOT}}(1^\lambda)$ to simulate h_{LOT}. Use $\mathsf{Sim}_{\mathsf{LCDS}}$ to simulate h_{LCDS}. Compute $\mathsf{com} \leftarrow \mathsf{CS.Com}(\mathsf{crs}, 0)$ and $\mathbf{c} \leftarrow \{0,1\}^\ell$.
- Upon receiving a message $\mathsf{ot}_2 = (\mathsf{ct}, \mathsf{ct}_{\mathsf{LCDS}}, \pi)$ from the sender, first extract S from π via the NIZK extractor. Then compute $\mathsf{ct}_{\mathsf{LOT}} = \mathsf{ct} - \mathsf{PRG}(S)$ and use the $\mathsf{Sim}_{\mathsf{LOT}}$ extractor to extract messages $(m_{0,i}, m_{1,i})_{i\in[\ell]}$.
- Send $(m_{0,i}, m_{1,i})_{i\in[\ell]}$ to the ideal functionality.

We now show that the simulation is indistinguishable from the real-world execution of the protocol. The proof follows the following sequence of hybrids.

- Hybrid 0: This is the real experiment.
- Hybrid 1: In this hybrid, we switch the $\mathsf{crs}_{\mathsf{NIZK}}$ to extraction mode and extract S.

  Indistinguishability of hybrids follows from the argument of knowledge property of the underlying NIZK.
- Hybrid 2: In this hybrid, we replace the $\mathsf{pp}_{\mathsf{LCDS}}$ to simulation mode and simulate h_{LCDS}.

  Indistinguishability of hybrids follows from the receiver simulation of the underlying LCDS scheme.

- Hybrid 3: In this hybrid, we replace the com by a commitment of 0.

  Indistinguishability of hybrids follows from the hiding property of the commitment scheme.

- Hybrid 4: In this hybrid, we replace $\mathbf{c}$ by a uniform string.

  Indistinguishability of hybrids follows from the pseudorandomness of the PRF.

- Hybrid 5: In this hybrid, we replace the $\mathsf{pp}_{\mathsf{LOT}}$ to simulation mode and simulate h_{LOT}. Additionally, we extract $(m_{0,i}, m_{1,i})_{i\in[\ell]}$ using this simulator.

  Indistinguishability of hybrids follows from the UC-malicious security against senders of the underlying LOT scheme.

This completes the proof against malicious senders. □

We will now focus on security against malicious receivers.

Lemma 16 (Security against malicious receivers)**.** *The scheme* OT *is UC-secure against malicious receivers.*

Proof. We start by presenting the simulator $\mathsf{Sim}_{\mathsf{R}}$.

$\mathsf{Sim}_{\mathsf{R}}$:

- Set the $\mathsf{pp} = (\mathsf{pp}_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{LCDS}}, \mathsf{crs}, \mathsf{crs}_{\mathsf{NIZK}})$ where i) crs is in simulation mode and ii) $\mathsf{crs}_{\mathsf{NIZK}}$ is in simulation mode.
- Upon receiving $\mathsf{ot}_1 = (h_{\mathsf{LOT}}, h_{\mathsf{CDS}}, \mathsf{com}, \mathbf{c})$ from the receiver do the following: i) Extract a PRF key K from com and check if $h_{\mathsf{LOT}} \leftarrow \mathsf{LOT.Hash}(\mathsf{pp}_{\mathsf{LOT}}, \mathbf{c} - \mathsf{PRF}_K([\ell]))$ (if this step fails, jump to D.3, ii) Compute $\mathbf{b} = \mathbf{c} - \mathsf{PRF}_K([\ell])$.
- If step i) above fails, then send $\mathsf{ct} \leftarrow \{0,1\}^*$, $\mathsf{ct}_{\mathsf{LCDS}} \leftarrow \mathsf{LCDS.Send}(\mathsf{pp}_{\mathsf{LCDS}}, h_{\mathsf{LCDS}}, h, 0)$ and a simulated proof π.
- Otherwise, send $\mathbf{b}$ to the ideal functionality and get back $m_{b_i,i}$.
- Compute ot_2 exactly as in the real scheme except that

$$\mathsf{ct}_{\mathsf{LOT}} \leftarrow \mathsf{LOT.Send}(\mathsf{pp}_{\mathsf{LOT}}, h_{\mathsf{LOT}}, [\ell], (m_{b_i,i}, m_{b_i,i})_{i\in[\ell]}).$$

We now show that both the simulated and the real-world executions are indistinguishable.

- Hybrid 0: This is the real execution of the protocol.
- Hybrid 1: In this hybrid, we set crs in the extraction mode and extract K using the trapdoor.

  Indistinguishability of hybrids follows from the mode indistinguishability of the underlying commitment scheme.

- Hybrid 2: In this hybrid, we simulate π using the simulator of the underlying NIZK.

  Indistinguishability of hybrids follows from the zero-knowledge of the NIZK.

- Hybrid 3: In this hybrid, we extract K from com using the extraction trapdoor. If $h_{\mathsf{LOT}} \neq \mathsf{LOT.Hash}(\mathsf{pp}_{\mathsf{LOT}}, \mathbf{c} - \mathsf{PRF}_K([\ell]))$ then set $\mathsf{ct}_{\mathsf{LCDS}} \leftarrow \mathsf{LCDS.Send}(\mathsf{pp}_{\mathsf{LCDS}}, h_{\mathsf{LCDS}}, h, 0)$.

  Indistinguishability of hybrids follows from the security of the underlying LCDS since $(h_{\mathsf{LOT}}, \mathsf{pp}_{\mathsf{LOT}}, \mathbf{c}, \mathsf{com}) \notin \mathcal{L}$.

- Hybrid 4: In this hybrid, if $h_{\mathsf{LOT}} \neq \mathsf{LOT.Hash}(\mathsf{pp}_{\mathsf{LOT}}, \mathbf{c} - \mathsf{PRF}_K([\ell]))$ then set $\mathsf{ct} \leftarrow \{0,1\}^*$.

  Indistinguishability of hybrids follows from the pseudorandomness of the underlying PRG.

- Hybrid 5: In this hybrid, we set $\mathsf{ct}_{\mathsf{LOT}} \leftarrow \mathsf{LOT.Send}(\mathsf{pp}_{\mathsf{LOT}}, h_{\mathsf{LOT}}, [\ell], (m_{b_i,i}, m_{b_i,i})_{i\in[\ell]})$. Indistinguishability of hybrids follows from the semi-honest security of the LOT scheme.

This completes the proof against malicious receivers. □