Laconic Conditional Disclosure of Secrets and Applications

Nico Döttling* Sanjam Garg† Vipul Goyal‡ Giulio Malavolta§

Abstract

In a Conditional Disclosure of Secrets (CDS) a verifier V wants to reveal a message m to a prover P conditioned on the fact that x is an accepting instance of some NP-language $\mathcal{L}$. An honest prover (holding the corresponding witness w) always obtains the message m at the end of the interaction. On the other hand, if $x \notin \mathcal{L}$ we require that no PPT P^* can learn the message m. We introduce *laconic CDS*, a two round CDS protocol with *optimal* computational cost for the verifier V and *optimal* communication cost. More specifically, the verifier's computation and overall communication grows with $\mathsf{poly}(|x|, \lambda, log(T))$, where λ is the security parameter and T is the verification time for checking that $x \in \mathcal{L}$ (given w). We obtain constructions of laconic CDS under standard assumptions, such as CDH or LWE.

Laconic CDS serves as a powerful tool for *maliciousifying* semi-honest protocols while preserving their computational and communication complexities. To substantiate this claim, we consider the setting of non-interactive secure computation: Alice wants to publish a *short* digest corresponding to a private *large* input x on her web page such that (possibly many) Bob, with a private input y, can send a *short* message to Alice allowing her to learn $\mathcal{C}(x, y)$ (where $\mathcal{C}$ is a public circuit). The protocol must be reusable in the sense that Bob can engage in arbitrarily many executions on the same digest. In this context we obtain the following new implications.

(1) *UC Secure Bob-optimized 2PC:* We obtain a UC secure protocol where Bob's computational cost and the communication cost of the protocol grows with $\mathsf{poly}(|x|, |y|, \lambda, d)$, where d is the depth of the computed circuit $\mathcal{C}$.

(2) *Malicious Laconic Function Evaluation:* Next, we move on to the setting where Alice's input x is large. For this case, UC secure protocols must have communication cost growing with $|x|$. Thus, with the goal of achieving better efficiency, we consider a weaker notion of malicious security. For this setting, we obtain a protocol for which Bob's computational cost and the communication cost of the protocol grows with $\mathsf{poly}(|y|, \lambda, d)$, where d is the depth of the computed circuit $\mathcal{C}$.

*CISPA Helmholtz Center for Information Security.

†University of California, Berkeley. Supported in part from AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

‡Carnegie Mellon University. Supported in part by NSF grant 1916939, a gift from Ripple, a JP Morgan Faculty Fellowship, and a Cylab seed funding award.

§Simons Institute for the Theory of Computing. Part of the work done while at Carnegie Mellon University. Supported in part by NSF grant 1916939.

1 Introduction

Consider the following setting: Alice would like to publish a digest h corresponding to her private data x on her web page. Next, Bob (with a private input y) would like to send a *short* message to Alice, so that she learns $\mathcal{C}(x, y)$ for some public circuit $\mathcal{C}$. As is standard in secure computation, Alice and Bob want to achieve this result while keeping their respective inputs hidden from each other. Analogous to Bob's message, Charlie or any other party could reuse h provided by Alice on her web page. This *non-interactive* nature of Alice's message makes this setting highly desirable and has been extensively studied in cryptography [IKO+11]. This primitive is commonly referred to as *non-interactive secure computation*.

Bob-Optimization. Unfortunately, traditional cryptographic techniques for realizing the above task require Bob's computational complexity to grow with $|\mathcal{C}|$ and $|x|$ (in addition to y), which is undesirable in several settings. In particular, since only Alice learns the output of the computation, Bob might find it unfair that he has to perform the computation of $|\mathcal{C}|$. In other words, Bob might be happy to see Alice learn the output of the computation, but may not be willing to take on doing the computation himself. For example, consider the case where Alice's input is her genomic sequence and Bob would like to help Alice to learn how likely she is to have cancer while keeping the logic that he is using to make predictions secret.

Making the problem more challenging, computational constraints on Bob imply additional constraints on the sizes of Alice's digest and Bob's message, which must now also be small. Constructing such Bob-optimized protocols has been the focus of several recent results such as laconic function evaluation [QWW18] and laconic oblivious transfer [CDG+17]. Our setting is inspired by these works. In particular, prior work on laconic function evaluation [QWW18] also considers the problem of non-interactive secure computation, which is computationally optimized for Bob. However, these results are limited to the semi-honest setting and need non-falsifiable assumptions [Kil92, Mic94, Gro10, BCCT13, GGPR13] to upgrade security to the malicious setting. This brings us to the following question:

> *Can we realize maliciously secure Bob-optimized non-interactive secure computation from standard assumptions?*

Bob-Optimization with Large Inputs. Next, we consider the more demanding setting, where Alice's input x itself might be very *large*. As before, Bob might be happy to see Alice learning the output of the computation, but may find reading x himself prohibitively expensive. This could be especially prohibitive if Bob interacts with multiple users, all playing the role of Alice. For example, the FBI (playing as Alice) could hold a huge database of passenger names on the no-fly list. An an airline company (playing as Bob) might want to help the FBI check if any of the passengers on one of its flights is on the list. Here the airline company would prefer to perform computation independent of the FBI's database. This is a special case of the very important and well-studied problem of private set intersection (PSI) [FNP04]. This brings us to the following question:

> *Can we realize maliciously secure Bob-optimized non-interactive secure computation when Alice has a large input from standard assumptions?*

Reusablity of Alice's Digest. We allow multiple Bob's to be able to reuse Alice's digest for multiple computations. In this setting, a corrupt Bob could provide Alice with an ill-formed message and use Alice's output (that it may learn in some way) to cheat Alice or to learn something about Alice's input x. Thus, we aim for security against a malicious Bob that has access to Alice's outputs on its prior interactions.

Reverse Delegation. One can view our setting as a reverse delegation scheme. In a delegation scheme, a user outsources its computation to an untrusted server with the goal of learning the output (possibly while even keeping it private). In contrast, our applications allow a user (Bob) to delegate its computation to the untrusted server (Alice) while also letting it learn the output of the computation, and nothing beyond that.

1.1 Our Results

In this work we introduce the notion of laconic conditional disclosure of secrets (CDS). In a CDS protocol a verifier V wants to reveal a message m to a prover P conditioned on the fact that x is an accepting instance of some NP-languge $\mathcal{L}$. An honest prover (holding the corresponding witness w) always obtains the message m at the end of the interaction. On the other hand if $x \notin \mathcal{L}$ we require that no (possibly corrupted) P^* can learn the message m. CDS can be seen as the two round analog of witness encryption [GGSW13]. The new constraint that we impose is that the interaction has to be *laconic* in the sense that it should not depend on the size of the parties inputs, and in particular on the size of the witness w. Additionally, the verifier computation must be much smaller than recomputing the relation. More specifically, verifier's computation and overall communication grows with $\mathsf{poly}(|x|, \lambda, log(T))$, where λ is the security parameter and T is the verification time for checking that $x \in \mathcal{L}$ (given w).

Next, we obtain constructions of laconic CDS in the common reference string model under standard assumptions, such as CDH or (standard) LWE. Laconic CDS serves as a powerful tool for *maliciousifying* semi-honest protocols while preserving their computational and communication complexities. To demonstrate this power, we show application of laconic CDS to the above-mentioned settings:

(1) *Bob-optimized 2PC:* We obtain a UC secure protocol where Bob's computational cost and the communication cost of the protocol grows with $\mathsf{poly}(|x|, |y|, \lambda, d)$, where d is the depth of the computed circuit C. This protocol achieves UC-security [Can01] and security is based on the LWE assumption.

(1) *Malicious Laconic Function Evaluation:* Next, we consider the more demanding setting where Alice's input x is large. For this case, UC secure protocols must have communication cost growing with $|x|$. Thus, with the goal of achieving better efficiency, we consider a weaker notion of malicious security which we call *context security*. Context security allows us to rewind parts of the execution while at the same time providing some form of composability and subsumes the standard notion of indistinguishability-based security. For this setting, we obtain a protocol for which Bob's computational cost and the communication cost of the protocol grows with $\mathsf{poly}(|y|, \lambda, d)$, where d is the depth of the computed circuit C. The security of this construction is based on LWE as well. This result can be seen as the malicious variant of the recent work on laconic function evaluation [QWW18].

The above applications demonstrate the power of laconic CDS. We see laconic CDS as a natural primitive and expect it to have other applications. As another example, laconic CDS allows a resource-constrained client to delegate the computation of a large circuit to an untrusted worker and to condition the payment of the worker on the fact that the provided output is the correct one. This does not achieve the notion of verifiable computation in the traditional sense, since the client cannot *explicitly* check that the output is correct. However in certain scenarios, e.g., mining in cryptocurrencies, one is interested in rewarding under the condition that *some* computation has been performed. Then miners are free to give the wrong output and not claim the reward. However, there is no clear incentive for doing that.

1.2 Technical Overview

To understand and motivate our techniques it is instructive to discuss a few plausible approaches for constructing laconic CDS and highlight the barriers that they encounter. Below we start with such approaches.

1.2.1 Why Known Techniques Fail

If we were to omit the laconic constraint, then CDS admits a very simple two-round protocol based on two-round oblivious transfer (OT) and Yao's garbled circuits [Yao86]: The prover encodes the binary representation of its witness as the choice bits in several parallel repetitions of the OT. The verifier garbles the circuit that hardcodes the statement x and the message m and returns m if and only if $\mathcal{R}(\cdot, x) = 1$. Then it sends the garbled circuit together with the second message of the OT on input each pair of input label. The prover can locally recover the labels corresponding to its witness and evaluate the circuit learning nothing beyond its output. It is clear that such a solution satisfies our security requirements but falls short in achieving laconic communication since the full circuit encoding of the NP-relation is exchanged between the two parties. At a first glance this seems to be inherent to garbled-circuit based solutions since secure two-party computation with low communication complexity usually requires more sophisticated tools, such as fully-homomorphic encryption [Gen09].

Another plausible angle to attack the problem is to resort to techniques from the field of succinct arguments [Kil92, Mic94]. Then the prover could simply augment a succinct proof with a public key and the verifier would then encrypt the message if the proof correctly verifies. Unfortunately known schemes require at least three rounds of interaction (four without assuming a setup) or rely on non-falsifiable assumptions [Gro10, BCCT13, GGPR13]. Constructing succinct arguments from standard assumptions in less than three rounds seems to hit a roadblock: It has been shown [GW11] that non-interactive succinct arguments cannot be based on falsifiable assumptions, at least in a black-box sense.

1.2.2 Our Solution in a Nutshell

Our starting point is the classical garbled circuit-based solution. A closer look to the source of inefficiency reveals that there are two major challenges to overcome in order to achieve our goal: (1) We need to remove the linear dependency of the OT with respect to the size of the witness and (2) we cannot recompute the full blown NP-relation in the garbled circuit. Our first insight is to bypass the first obstacle using laconic OT [CDG$^+$17]. A laconic OT allows one to hash a long database into a small digest, then the sender can compute the second message of an OT where the choice bit is set to be the value at an arbitrary location of the database. The important message here is that the communication complexity is independent of the size of the database (in our case the witness w). This primitive alone allows us to compress the size of the first message of the CDS. However, laconic OT alone does not buy us anything for the complexity of the second message. Overcoming the second obstacle requires us to borrow techniques from the domain of succinct arguments. The observation here is that checking the validity of an NP-instance does not necessarily require one to recompute the corresponding relation: Probabilistically checkable proofs (PCP) [AS98] encode a witness into a (longer) string such that the membership of the statement can be probabilistically verified by querying a constant amount of bits. Such verification algorithms are inherently erroneous, but the gap can be made negligibly small via standard amplification techniques.

This tool gives us the right leverage for a candidate laconic CDS construction: A prover can now hash the PCP encoding of its witness via a laconic OT and send the corresponding digest to

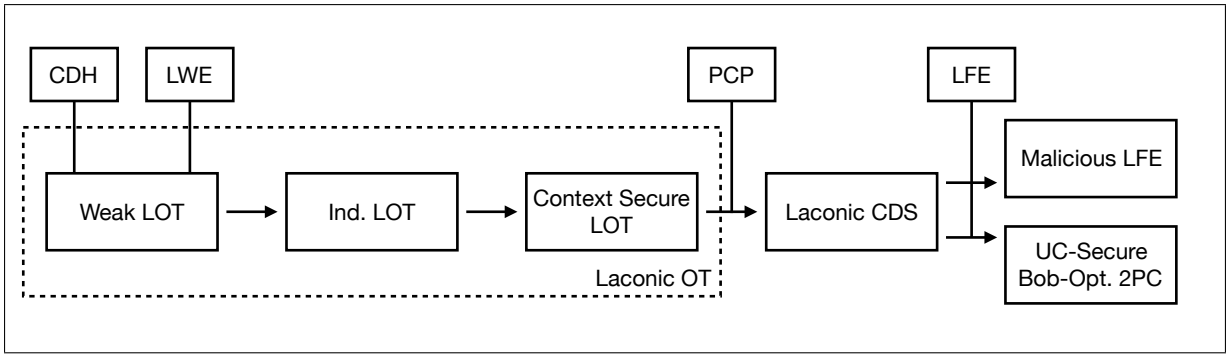

Figure 1: Overview of our Results

the verifier. The latter then samples a few random locations and garbles a circuit that takes as input the bits of the PCP encoding at such locations and returns m if the PCP verifier accepts. Then it sends the garbled circuit in plain and the input labels via the laconic OT. This allows the prover to recover the labels corresponding to the bits of the PCP encoding, evaluate the circuit, and eventually recover the message. This introduces a negligible soundness gap, which does not make a difference in our settings as we rely anyway on computational assumptions. The stretch in the size of the witness is also not a problem since the communication complexity of a laconic OT is independent of the size of the database (in our case the PCP encoding).

While this construction seems to solve all problems at once, a closer look to the building blocks reveals the dependency on a *maliciously secure* laconic OT. Currently, the only known way to construct laconic OT resilient against active attackers is to compile a semi-honest one with succinct arguments. Due to its compressing nature, bypassing the usage of non-falsifiable assumptions in laconic CDS might then appear to be out of reach of current techniques. However, we have now reduced the problem of constructing laconic CDS to that of building laconic OT: In this work we show how to construct maliciously secure laconic OT assuming the hardness of CDH or (standard) LWE with polynomial modulo-to-noise ratio. Most of the technical innovations of this work, and the remainder of this section, are devoted to instantiating malicious laconic OT from standard assumptions. Looking ahead, this will allow us to construct laconic CDS, which in turn will be used as the main technical component to build malicious LFE and UC-secure Bob-optimized 2PC. An outline of our results is given in Figure 1.

1.2.3 Towards Malicious Laconic OT

Before we delve into the actual instantiations of malicious laconic OT we further simplify the problem by lowering the efficiency and security requirements for a laconic OT.

Our starting point is a recent work by Döttling, Garg, Hajiabadi, Masny and Wichs [DGH$^+$19] which constructs maliciously secure oblivious transfer from search assumptions. The main idea in [DGH$^+$19] is to start with a very simple notion of security against malicious receivers and carefully bootstrap this notion to standard simulation based notions. The weakest notion considered in [DGH$^+$19] is called *elementary OT*, and our basic notion of malicious laconic OT extends this notion to the setting of laconic OT. We refer to this primitive as *weak malicious* laconic OT (when it is clear from the context we drop the term malicious). Jumping ahead, we will then show, building on techniques developed in [DGH$^+$19] how to generically upgrade a laconic OT that meets these conditions to a fully efficient and fully secure one. Concretely, we aim for the following guarantees.

(1) Weak Functionality: The sender algorithm no longer takes as input two messages (m_0, m_1) to transfer but instead generates two fresh random strings (k_0, k_1) together with the sender

output c. This can be seen as the analog of key-encapsulation mechanism, where (k_0, k_1) are then used as the session keys to transfer the desired messages.

(2) Weak Security: We require that, given the sender message c, no efficient polynomial-time algorithm can output both k_0 and k_1 at the same time. This does not guarantee, for instance, that an attacker cannot output the first half of k_0 and the second half of k_1. This requirement is in fact identical to the notion of elementary sender security considered in [DGH+19].

(3) Weak Efficiency: We consider a laconic OT where the only efficiency constraint is that the receiver message is 2-to-1 compressing. That is, we do not impose any bound on the size of the setup string and of the sender message, which are potentially as long as the database.

1.2.4 Pitfalls of Known Constructions

A natural question to ask is whether existing constructions of semi-honest laconic OT already satisfies any meaningful notion of security in presence of a corrupted receiver. To exemplify the issues that arise, we briefly recall a simplified version of the hash-encryption construction of [DG17]: The public parameters consist of a matrix of uniformly sampled group elements

$$crs = \begin{pmatrix} g_1^{(0)}, \ldots, g_m^{(0)} \\ g_1^{(1)}, \ldots, g_m^{(1)} \end{pmatrix} \leftarrow_{\$} \mathbb{G}^{2\times m}$$

and the hash of a database D is computed as

$$d = \prod_{i=1}^{m} g_i^{(D[i])}.$$

To generate two keys for a position L, one samples two random random integers (r, s), computes the ciphertext c as the concatenation of the matrices

$$\begin{pmatrix} \left(g_1^{(0)}\right)^r, & \ldots, & \left(g_L^{(0)}\right)^r, & \ldots, & \left(g_m^{(0)}\right)^r \\ \left(g_1^{(1)}\right)^r, & \ldots, & 1, & \ldots, & \left(g_m^{(1)}\right)^r \end{pmatrix}$$

and

$$\begin{pmatrix} \left(g_1^{(0)}\right)^s, & \ldots, & 1, & \ldots, & \left(g_m^{(0)}\right)^s \\ \left(g_1^{(1)}\right)^s, & \ldots, & \left(g_L^{(1)}\right)^s, & \ldots, & \left(g_m^{(1)}\right)^s \end{pmatrix},$$

then sets $k_0 = d^r$ and $k_1 = d^s$. Security stems from the fact that if $D[L] \neq b$, then the receiver has to find a different linear combination of elements that yields the same d in order to compute the corresponding key. This however crucially relies on the fact that the receiver always chooses at least one group element per column of the crs. Consider the following attack where d is set to be $d = g_1^{(0)}$. Then the receiver can recover the keys for both $b = \{0, 1\}$, when encrypting with respect to some $L \neq 1$. This is because the element $\left(g_1^{(0)}\right)^r$ is always given if $L \neq 1$, then k_0 and k_1 are simply the first element of each matrix. This breaks any meaningful notion of security and forces us to rethink even the most basic building block of the primitive.

1.2.5 Construction from CDH

Our approach diverges from prior works and takes a fresh look at the problem. In our scheme the public parameters consist of a vector of group elements sampled uniformly at random

$$crs = (g_1, \ldots, g_m) \leftarrow_{\$} \mathbb{G}^m$$

and the hash of a database D is defined as

$$d = \prod_{i=1}^{m} g_i^{D[i]}.$$

That is the i-th element of the vector is included in the product if $D[i] = 1$ and skipped if $D[i] = 0$. The subtle difference with respect to previous approaches lies in the fact that even setting $d = g_1$ gives us a perfectly valid hash, which corresponds to the pre-image $1\|0^{m-1}$. To generate two session keys (k_0, k_1) for a location L, the sender samples a random integer r and sets the ciphertext to

$$c = \left(g_1^r, \ldots, g_{L-1}^r, g_{L+1}^r, \ldots, g_m^r\right),$$

i.e., all powers are given except for g_L^r. We now want to define k_0 and k_1 in such a way that only $k_{D[i]}$ can be recovered using a pre-image D of d. To this end we set

$$k_0 = d^r \text{ and } k_1 = d^r / g_L^r.$$

Observe that if $D[L] = 0$, then c contains enough information to recompute

$$k_0 = d^r = \prod_{i=1}^{m} g_i^{D[i] \cdot r},$$

since g_L^r would be excluded from the product anyway. On the other hand if $D[L] = 1$, then one can only recompute

$$k_1 = d^r / g_L^r = \prod_{i=1, i \neq L}^{m} g_i^{D[i] \cdot r}.$$

To get an intuition why the scheme is secure for any (possibly maliciously generated) image d, observe that outputting both k_0 and k_1 also reveals $k_0/k_1 = g_L^r$, regardless of the value of d. However, g_L^r was not given as part of the ciphertext and therefore predicting it, given only c and crs, is as hard as solving a random instance of the CDH problem.

1.2.6 Construction from LWE

A CDH-based construction is not entirely satisfactory in terms of assumptions since most of the applications of malicious laconic OT (and consequently of laconic CDS) are only known under the hardness of lattice-related problems. If we were to plug in our machinery we would introduce an additional number-theoretic assumption (i.e., CDH), which is not ideal. An additional reason to look into lattice-based schemes is that current cryptanalytic techniques fail even in the quantum settings, whereas there are polynomial-time quantum algorithms to solve discrete logarithm-related problems. For these reasons, we turn our attention to constructing weak malicious laconic OT from the hardness of the LWE problem. The basic idea of our scheme is conceptually simple, but the noisy nature of the LWE problem introduces some complications. The public parameters consist of a single matrix

$$crs = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$$

where m is the size of the database and n is a parameter that governs the hardness of the LWE problem. Hashing a database $D \in \{0,1\}^m$ (parsed as a column vector) is a simple multiplication

$$\mathbf{A}D = \mathbf{d} \in \mathbb{Z}_q^n$$

that is, we take a linear combination of the columns of $\mathbf{A}$ depending on the bits of D. For a given location L, the sender algorithm samples a column vector $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$ and computes a noisy inner product with each column of $\mathbf{A}$, except for the L-th one. The sender message c consists of the set

$$\left\{\mathbf{s}^T\mathbf{a}_1 + e_1, \ldots, \mathbf{s}^T\mathbf{a}_m + e_m\right\} \setminus \left\{\mathbf{s}^T\mathbf{a}_L + e_L\right\}$$

where the noise vector is sampled from a discrete Gaussian and has small norm. As before we set the keys in such a way that it is easy to compute one, but it is hard to compute both. This is done by setting

$$k_0 = \mathbf{s}^T\mathbf{d} \text{ and } k_1 = -\mathbf{s}^T\mathbf{d} + \mathbf{s}^T\mathbf{a}_L + e_L.$$

It is important to observe that $k_0 + k_1 = \mathbf{s}^T\mathbf{a}_L + e_L$, which is an LWE sample but it is not among those included in c. This suggests that outputting both values of k_0 and k_1 is equivalent to predicting an LWE sample and it is going to be our central leverage to show the security of the scheme. However, the more challenging part is how to ensure that the receiver is able to recover $k_{D[L]}$. In contrast to our previous scheme, the naive strategy fails to recover the key due to the presence of the noise. In fact

$$\sum_{i=1, i\neq L}^{m} c_i \cdot D[i] = k_{D[L]} + \sum_{i=1, i\neq L}^{m} e_i \cdot D[i] = k_{D[L]} + \tilde{e}.$$

To overcome this issue we exploit the fact that $\tilde{e}$ is relatively small when compared to $k_{D[L]}$. We then partition $\mathbb{Z}_q$ in c-many intervals and we set the actual session key to be k_b (as computed above) rounded at the closest multiple of q/c. The key property of this rounding function is that it is resilient to small perturbations, which means that

$$\left\lfloor k_{D[L]} + \tilde{e} \right\rceil_c = \left\lfloor k_{D[L]} \right\rceil_c$$

with high probability, for a small enough $\tilde{e}$. This strategy introduces a few issues, among others the fact that the output of the rounding function is not long enough to argue about unpredictability (i.e., it can be randomly guessed with high enough probability). Fortunately, this issue can be easily circumvented with standard amplification techniques.

1.2.7 Indistinguishability Laconic OT

As discussed before, our notion of security only guarantees that the adversary cannot output both k_0 and k_1 in their entirety but it does not guarantee that there exists a consistent bit b for which the adversary can always guess k_b and never $k_{b\oplus 1}$. For example, it could be possible that for a fixed value of d, the adversary is able to predict k_0 for half of the support of c and k_1 for the complement. To fix this issue, follow the blueprint of [DGH+19] and use techniques developed in the context of hardness amplification of puzzles [CHS05]. The key idea of this step is to amplify the success probability of the adversary via parallel repetitions: By setting the new key to be the concatenation of many independent instances, with high probability at least one of elements will belong to the *wrong* partition of the support of c. Thus with high enough probability, the adversary is forced to simultaneously predict two values (k_0, k_1).

Once this obstacle is surpassed, turning our weak laconic OT into a fully functional one involves rather standard tools: First we turn the search problem into a decision one using

Goldreich-Levin hardcore predicate [GL89]. At this point we can give as input to the sender algorithm any message pair (m_0, m_1) and implement the standard OT functionality by augmenting c with $k_0 \oplus m_0$ and $k_1 \oplus m_1$. It is easy to see that at least one of the two messages must be hidden to the eyes of the adversary as the key $k_{D[L]\oplus 1}$ acts as a one-time pad. At this point, our security definition looks as follows: We define two experiments (Exp_0 and Exp_1) where the adversary is allowed to choose a hash d, two messages (m_0, m_1), and a location L, then the challenger flips a coin b. If $b = 0$ then the ciphertext c is honestly computed via the sender algorithm on input (L, m_0, m_1), otherwise m_0 (m_1 for Exp_1) is replaced with a uniformly sampled message. The guarantee is that the adversary cannot succeed in both experiments with non-negligible probability.

This gives us a more intuitive security notion but it is still not sufficient for constructing laconic CDS. The reason is that we have no idea whether the adversary is going to be able to succeed in Exp_0 or Exp_1, for a given value of L. This information is needed by the simulator when using laconic OT in conjunction with garbled circuits: In a reduction against the security of the garbling scheme we need to know in which position we should plug the challenge label, to correctly simulate the view of the adversary. Even worse, determining whether the adversary is successful in Exp_0 or Exp_1, for all locations L, corresponds to extracting the whole database D! Recall that the hash d of a laconic OT is compressing, therefore its pre-image is not even information-theoretically determined, let alone efficiently computable.

1.2.8 Context-Secure Laconic OT

The final challenge towards constructing malicious laconic OT boils down to extracting the pre-image of any given (possibly maliciously computed) hash. As discussed before, the hash does not even determine the pre-image in an information theoretic sense, so trivial solutions are not applicable. An additional complication is that the protocol is non-interactive, i.e., the receiver outputs a single message and goes offline. Therefore rewinding the receiver also does not seem to help. Of course one could just *assume* the existence of an extractor, which would however place our solution within the category of non-falsifiable assumptions.

We address this problem by leveraging the distinguisher-dependent simulation technique recently developed [JKKR17]. We introduce a new security notion called *context security* to provide a versatile toolkit which allows to deploy distinguisher-dependent simulation in much more complex settings. While on a technical level we use many of the same techniques as [DGH$^+$19], context security allows us to use distinguisher-dependent simulation to its full effect.

We use distinguisher-dependent simulation in the following way to extract the input of a malicious receiver: A careful look to the indistinguishability-based definition shows that, given black-box access to a malicious receiver, we can determine the bit $D[L]$, by approximating the success probability in Exp_0 and Exp_1: For a given location L, the experiment where the adversary is successful must correspond to the value of $D[L]$! Iterating over all locations, we can recover the complete database.

Our main insight is that this technique can be applied to a setting where the overall communication between receiver and sender is too small to information-theoretically specify the input of the receiver. This is in contrast to previous uses of this technique in [JKKR17, DGH$^+$19]. Somewhat counterintuitively, we are able to extract the full pre-image of a short hash in polynomial time without resorting to non-falsifiable assumptions. Our new notion of context *context security* is crafted to precisely characterize the security guarantees we can achieve via distinguisher dependend simulation.

The high level idea of context security is that protocols must remain secure against any context, where a context is defined as a (possibly corrupted) receiver and a distinguisher that takes

as input the sender's message. Security is defined in terms of the existence of an extractor and a simulator: The extractor (with oracle access to the distinguisher) recovers the original input of the receiver and gives enough information to the simulator to simulate the sender message, where the simulator only interacts with the ideal functionality. Additionally, the runtime of the extractor is independent of that of the corrupted receiver (but depends on that of the distinguisher). This enables a meaningful notion of composability since the protocol is required to be remain secure for any context. Since the extractor depends on the distinguisher, context-security does not achieve the same strong guarantees of universal composability (UC) [Can01]. However, in contrast with UC, a protocol can be context-secure even when its transcript does not determine the inputs of the parties in an information theoretic sense. Thus we view context-security as a meaningful relaxation of UC and a versatile tool, instrumental to construct laconic CDS. Finally, we remark that context security implies the standard notion of indistinguishability-based secure computation.

Using ideas developed in [DGH$^+$19] we show that any indistinguishability laconic OT is also context secure. The argument crucially relies on rewinding the distinguisher and measuring its success-probability.

1.2.9 Efficient Laconic OT

Equipped with a context-secure laconic OT, we can show a bootstrapping theorem in the same spirit as [CDG$^+$17]: Given any 2-to-1 compressing context-secure laconic OT we can construct an equally efficient context-secure laconic OT where the size of the hash (and of the common reference string) is independent of the size of the database. The transformation is identical to that of [CDG$^+$17] and works by hashing the database into a Merkle tree and using a chain of garbled circuit to access the leaf corresponding to the location L. However, the argument is fundamentally different since we cannot assume that the hash is honestly computed. The usefulness of the notion of context security is demonstrated by our bootstrapping theorem. We crucially rely on the recursive application of context extractors to extract an entire Merkle-tree. A critical aspect in this step which avoids an exponential blowup is that the runtime of the context-extractor is independent of the computation of the receiver.

1.3 Applications

We now discuss how laconic CDS can be used as a *maliciousifier* for two-round protocols: A two round protocol consists of a message m_1 from the receiver to the sender and a message m_2 from the sender to the receiver. At the end of the interaction, the receiver performs some computation to retrieve the output of the protocol. Using laconic CDS we can turn a semi-honest two-round protocol into a malicious one (for a corrupted receiver) by augmenting m_1 with the first message of a laconic CDS certifying that m_1 is well formed. Then the sender computes the second message of a laconic CDS where the secret is set to m_2. That is, the receiver will learn m_2 if and only if m_1 was well-formed. The new properties that our transformation enables are:

(1) The communication complexity of the malicious protocol is identical to that of the semi-honest one.

(2) The computational complexity of the sender is unchanged.

This comes particularly useful when the input and the computation of the receiver are particularly burdensome. As an example, a laconic CDS allows us to lift non-interactive secure computation for RAM programs [CDG$^+$17] to the malicious settings in a very natural way. Another interesting scenario is when the above transformation is applied to laconic function evaluation

(LFE). In this case we obtain the first maliciously secure Bob-optimized non-interactive secure computation protocol where the communication complexity grows only with the depth of the circuit and with the size of Bob's input. In other words, the malicious protocol is (asymptotically) as efficient as the underlying semi-honest LFE. We can even lift the security to the UC-settings, however at the cost of the communication growing with the size of the receiver's input, which is unavoidable.

Delegating Computation. A less orthodox usage of a laconic CDS is in the context of non-interactive delegation of computation: A client wants to delegate the computation of a large circuit to an untrusted worker and wants some insurance that the output given by the worker is the correct one. The worker performs the computation and obtains the output z, then computes a laconic CDS that certifies that z is indeed the correct output. On input the first message of the laconic CDS and z, the client conditions the payment of the worker on the fact that z is computed correctly. This protocol is not verifiable in the traditional sense since the client cannot explicitly check that z is indeed the correct output. However we can envision scenarios where this is not a limitation. As an example, miners of cryptocurrencies often aggregate in pools, where each peer is rewarded basing on the amount of computation it performed. In this case it is not critical to verify that the output is correct but we are mostly interested in the fact that *some* large circuit has been computed. The miners could of course provide the wrong output and not claim the reward, but it is unclear what would be the incentive to do that.

2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter. We we say that a function *negl* is negligible if it vanishes faster than any polynomial. Given a set S, we denote by $s \leftarrow\!\!\$\, S$ the uniform sampling of an element from S. We say that an algorithm is PPT if it can be implemented by a probabilistic Turing machine running in time polynomial in λ. We recall two useful inequalities.

Theorem 1 (Hoeffding Inequality)**.** *Let $(X_1, \ldots, X_N) \in [0,1]$ be i.i.d. random variable with expectation $\mathsf{E}[\bar{X}]$. Then it holds that*

$$\Pr\left[\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mathsf{E}[\bar{X}]\right| > \delta\right] \leq 2e^{-2N\delta^2}.$$

Lemma 1 (Markov Inequality for Advantages (taken from [DGH+19]))**.** *Let $A(Z)$ and $B(Z)$ be two random variables depending on a random variable Z and potentially additional random choices. Assume that $|\Pr_Z[A(Z)=1] - \Pr_Z[B(Z)=1]| \geq \varepsilon \geq 0$. Then*

$$\Pr_Z[|\Pr[A(Z)=1] - \Pr[B(Z)=1]| \geq \varepsilon/2] \geq \varepsilon/2.$$

Proof. Let $a := \Pr_Z[|\Pr[A(Z)=1] - \Pr[B(Z)=1]| \geq \varepsilon/2]$. We have $\varepsilon \leq a \times 1 + (1-a) \times \varepsilon/2$. Since $0 \leq 1-a \leq 1$, we obtain $\varepsilon \leq a + \varepsilon/2$. The inequality now follows. □

2.1 Probabilistically Checkable Proofs

Probabilistic checkable proofs (PCP) [AS98] are a central tool in complexity theory. The PCP theorem shows that any witness w for an NP-statement can be encoded into a PCP of length $poly(|w|)$ such that it is sufficient to probabilistically test O(1) bits of the encoded witness.

Definition 1 (Probabilistically Checkable Proofs)**.** *A PCP* $(\mathsf{PCPProve}, \mathsf{PCPVerify})$ *for an NP language $\mathcal{L}$ is a tuple of the PPT algorithms defined as follows:*

$\mathsf{PCPProve}(x, w) \rightarrow \pi$: *On input a statement* $x \in \mathcal{L}$ *and the corresponding witness* w*, the proving algorithm returns an encoding* π*.*

$\mathsf{PCPVerify}^{\pi}(x) \rightarrow \{0,1\}$: *On input a statement* x *and with random access to the encoding* π*, the verification algorithm returns a bit* $\{0,1\}$*.*

Correctness is defined canonically.

Definition 2 (Correctness)**.** *A PCP* $(\mathsf{PCPProve}, \mathsf{PCPVerify})$ *is correct if for all* $(x, w) \in \mathcal{R}$ *it holds that*

$$\Pr\left[\mathsf{PCPVerify}^{\pi}(x) = 1 | \pi \leftarrow \mathsf{PCPProve}(x, w)\right] = 1$$

where the probability is taken over the random coins of $\mathsf{PCPVerify}$*.*

The central efficiency measure for a PCP is the number of queries that the verifier issues to the encoding, i.e., the amount of bits that the verifier needs to read before producing his output. This amount (which we denote by q) can be as low as 3 queries. We consider without loss of generality PCPs where the verifier is non-adaptive, i.e., the bits queried do not depend on previous answers. Note that any constant-query adaptive PCP can be turned into a constant-query non-adaptive PCP by querying all possible bits for a given random tape (introducing a constant factor in the query complexity). Another important parameter for a PCP is the amount of random coins needed by the verifier, which is bounded by the PCP theorem to be at most $t = O(log(|\pi|)) = O(log(\lambda))$. We now state the soundness property.

Definition 3 (Soundness)**.** *A PCP* $(\mathsf{PCPProve}, \mathsf{PCPVerify})$ *is sound if for all* $x \notin \mathcal{L}$ *and for all* π *it holds that*

$$\Pr\left[\mathsf{PCPVerify}^{\pi}(x) = 1\right] < 1/3$$

where the probability is taken over the random coins of $\mathsf{PCPVerify}$*.*

It is well known that the success probability can be amplified to a negligible fraction (in the security parameter) by parallel repetition. Additionally we say that a PCP is *witness-extractable* if there exists a PPT algorithm PCPExt and a constant γ such that, on input a PCP encoding π that passes the PCP test with probability at least $(1-\gamma)$, the algorithm outputs a valid witness w for x. This additional property can be obtained for $\gamma = 2/3$ and an explicit construction appears in [Val08].

2.2 Intractable Problems

In the following we introduce some hard problems in cryptography that are going to be useful for our work.

2.2.1 Computational Diffie-Hellman

We recall the search version of the classical Diffie-Hellman problem [DH76]. Let $\mathcal{G}$ be a (prime-order) group generator that takes as input the security parameter and 1^{λ} and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, p is the order of the group, and g is a generator of the group.

Definition 4 (Computational Diffie-Hellman (CDH) assumption)**.** *We say that* $\mathcal{G}$ *satisfies the CDH assumption (or is CDH-hard) if for any PPT adversary* $\mathcal{A}$ *it holds that*

$$\Pr[\mathcal{A}(\mathbb{G}, p, g, g^{a_1}, g^{a_2}) = g^{a_1 a_2}] = negl(\lambda)$$

where $(\mathbb{G}, p, g) \leftarrow\!\!\$\, \mathcal{G}(1^{\lambda})$ *and* $(a_1, a_2) \leftarrow\!\!\$\, \mathbb{Z}_p$*.*

2.2.2 Learning with Errors

The learning with errors (LWE) problem was introduced by Regev [Reg05]. In this work we exclusively use the decisional version, since there exists a well known reduction to the search variant of the problem.

Definition 5 (Learning with Errors (LWE) assumption)**.** *The $LWE_{n,\tilde{n},q,\chi}$ problem, for $(n, \tilde{n}, q) \in \mathbb{N}$ and for a distribution χ supported over $\mathbb{Z}$, is to distinguish between the distributions $(\mathbf{A}, \mathbf{sA} + \mathbf{e} \mod q)$ and $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{n \times \tilde{n}}$, $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow_\$ \chi^{\tilde{n}}$, and $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^{\tilde{n}}$. Then, the $LWE_{n,\tilde{n},q,\chi}$ assumption is that the two distributions are computationally indistinguishable.*

The assumption is consider to hold for any $\tilde{n} = poly(n, log(q))$ and we denote this problem by $\text{LWE}_{n,q,\chi}$. As shown in [Reg05, PRS17], the $\text{LWE}_{n,q,\chi}$ problem with χ being the discrete Gaussian distribution with parameter $\sigma = \alpha q \geq 2\sqrt{n}$ (i.e. the distribution over $\mathbb{Z}$ where the probability of x is proportional to $e^{-\pi(|x|/\sigma)^2}$), is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\alpha)$ in *worst case* dimension n lattices. This is proven using a quantum reduction. Classical reductions (to a slightly different problem) exist as well [Pei09, BLP+13] but with somewhat worse parameters. The best known (classical or quantum) algorithms for these problems run in time $2^{\tilde{O}(n/log(\gamma))}$, and in particular they are conjectured to be intractable for $\gamma = poly(n)$.

A discrete gaussian with parameter αq is $B = \alpha q$ bounded, except with negligible probability. For parameter α the worst-to-average case reduction of [Reg05] gives a worst-case approximation factor of $\tilde{O}(n/\alpha)$ for SIVP. Consequently, in terms of the bound B and the modulus q we get a worst-case approximation factor of $\tilde{O}(nq/B)$ for SIVP.

2.3 Pseudorandom Generator

A pseudorandom generator (PRG) [ILL89] stretches random strings into random-looking strings.

Definition 6 (Pseudorandom Generator)**.** *A PRG $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^m$ is pseudorandom if $m > \lambda$ and for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that*

$$|\Pr[1 \leftarrow \mathcal{A}(r)] - \Pr[1 \leftarrow \mathcal{A}(\mathsf{PRG}(s))]| = negl(\lambda)$$

where $r \leftarrow_\$ \{0,1\}^m$ and $s \leftarrow_\$ \{0,1\}^\lambda$.

2.4 Garbled Circuits

We recall the definition of Yao's garbled circuits [Yao86]. A garbling scheme allows one to obfuscate a circuit and provide enough information to learn a single output and nothing more. Garbled circuits can be constructed assuming the existence of one-way functions only.

Definition 7 (Garbled Circuits)**.** *A garbling scheme is a tuple of PPT algorithms* $(\mathsf{Garble}, \mathsf{Eval})$ *defined as follows:*

- $\mathsf{Garble}(1^\lambda, \mathcal{C}) \to (\tilde{\mathcal{C}}, \{\ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^n)$: *The garbling algorithm takes as input the security parameter 1^λ and the description of a circuit $\mathcal{C}$ and returns a garbled circuit $\tilde{\mathcal{C}}$ and a set of labels $\{\ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^n$ for each input wire of the circuit.*
- $\mathsf{Eval}(\tilde{\mathcal{C}}, \{\ell_i^{(x_i)}\}_{i=1}^n) \to y$: *The evaluation algorithm takes as input a garbled circuit $\tilde{\mathcal{C}}$ and one label per input wire $\ell_i^{(x_i)}$ and returns an output y.*

The definition of correctness is given in the following.

Definition 8 (Correctness)**.** *A garbling scheme* $(\mathsf{Garble}, \mathsf{Eval})$ *is correct if for all* $\lambda \in \mathbb{N}$*, for all polynomial size circuits* $\mathcal{C}$ *with input length* n*, and for all inputs* $x \in \{0,1\}^n$ *it holds that*

$$\Pr\left[\mathcal{C}(x) = \mathsf{Eval}(\tilde{\mathcal{C}}, \{\ell_i^{(x_i)}\}_{i=1}^n) \middle| (\tilde{\mathcal{C}}, \{\ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^n) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})\right] = 1$$

where the probability is taken over the random coins of Garble*.*

Security requires that the evaluation of a circuit reveals nothing beyond its output.

Definition 9 (Security)**.** *A garbling scheme* $(\mathsf{Garble}, \mathsf{Eval})$ *is secure if for all* $\lambda \in \mathbb{N}$ *there exists a PPT algorithm* GCSim *such that for all PPT attacker* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $negl$ *such that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(\tilde{\mathcal{C}}, \{\ell_i^{(x_i)}\}_{i=1}^n, \mathsf{st}) \middle| \begin{array}{l}(\mathcal{C}, x, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\tilde{\mathcal{C}}, \{\ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^n) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})\end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}_2(\tilde{\mathcal{C}}, \{\ell_i\}_{i=1}^n, \mathsf{st}) \middle| \begin{array}{l}(\mathcal{C}, x, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\tilde{\mathcal{C}}, \{\ell_i\}_{i=1}^n) \leftarrow \mathsf{GCSim}(1^\lambda, |\mathcal{C}|, \mathcal{C}(x))\end{array}\right] \right| = negl(\lambda)$$

where the probability is taken over the random coins of $\mathcal{A}_1$*,* Garble*, and* GCSim*.*

2.5 Public Key Encryption

We recall the notion of public-key encryption [GM82].

Definition 10 (Public-Key Encryption)**.** *A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec})$ *is defined as the following tuple of efficient algorithms.*

$\mathsf{PKEKGen}(1^\lambda) \to (sk, pk)$: *On input the security parameter* 1^λ *the key generation algorithm returns a secret key* sk *and a public key* pk*.*

$\mathsf{PKEEnc}(pk, m) \to c$: *On input a public key* pk *and a message* m*, the encryption algorithm returns a ciphertext* c*.*

$\mathsf{PKEDec}(sk, c) \to m$: *On input a secret key* sk *and a ciphertext* c*, the decryption algorithm returns a message* m*.*

We require the scheme to be perfectly correct. It is well known that perfectly correct public-key encryption can be constructed from CDH [ElG84] or LWE [Reg05].

Definition 11 (Correctness)**.** *A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec})$ *is correct if for all* $\lambda \in \mathbb{N}$ *and for all messages* m *it holds that*

$$\Pr\left[m = \mathsf{PKEDec}(sk, c) \middle| \begin{array}{l}(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda) \\ c \leftarrow \mathsf{PKEEnc}(pk, m)\end{array}\right] = 1$$

and the probability is taken over the random coins of $\mathsf{PKEKGen}$ *and* PKEEnc*.*

We recall the definition of semantic security.

Definition 12 (Security)**.** *A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec})$ *is semantically secure if for all* $\lambda \in \mathbb{N}$ *and for all admissible PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $negl$ *such that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(c, \mathsf{st}) \middle| \begin{array}{l}(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathsf{PKEEnc}(pk, m_0)\end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}_2(c, \mathsf{st}) \middle| \begin{array}{l}(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathsf{PKEEnc}(pk, m_1)\end{array}\right] \right| = negl(\lambda)$$

where $\mathcal{A}$ is admissible if m_0 and m_1 have equal length and the probabilities are taken over the random coins of $\mathsf{PKEKGen}$*, $\mathcal{A}_1$, and* PKEEnc*.*

2.6 Non-Interactive Zero-Knowledge

We define non-interactive zero-knowledge (NIZK) [BFM88]. NIZKs have been constructed from trapdoor permutation [FLS90] or, very recently, from LWE [PS19].

Definition 13 (Non-Interactive Zero-Knowledge)**.** *A NIZK* $\mathsf{NIZK} = (\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ *for an NP-language $\mathcal{L}$ with relation $\mathcal{R}$ is defined as the following tuple of algorithms.*

$\mathsf{NIZKSetup}(1^\lambda) \to crs$ *: On input the security parameter 1^λ the setup algorithm returns a common reference string crs.*

$\mathsf{NIZKProve}(crs, x, w) \to \pi$ *: On input the common reference string crs, a statement x and a witness w, the proving algorithm returns a proof π.*

$\mathsf{NIZKVerify}(crs, x, \pi) \to \{0,1\}$ *: On input the common refrence string crs, a statment x, and a proof π, the verifier algorithm returns a bit $\{0,1\}$.*

Correctness is defined in a standard way.

Definition 14 (Correctness)**.** *A NIZK scheme* $\mathsf{NIZK} = (\mathsf{NIZKSetup}, \mathsf{NIZKProveNIZKVerify})$ *is correct if for all $\lambda \in \mathbb{N}$ and for all pairs $(x, w) \in \mathcal{R}$ it holds that*

$$\Pr\left[1 = \mathsf{NIZKVerify}(crs, x, \pi) \middle| \begin{array}{l} crs \leftarrow \mathsf{NIZKSetup}(1^\lambda) \\ \pi \leftarrow \mathsf{NIZKProve}(crs, x, w) \end{array}\right] = 1$$

and the probability is taken over the random coins of $\mathsf{NIZKSetup}$ *and* $\mathsf{NIZKProve}$*.*

We require that proofs for statements not in the language are hard to find.

Definition 15 (Soundness)**.** *A NIZK scheme* $\mathsf{NIZK} = (\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ *is sound if for all $\lambda \in \mathbb{N}$, for all PPT adversaries $\mathcal{A}$, and for all $x \notin \mathcal{L}$, there exists a negligible function negl such that*

$$\Pr\left[1 = \mathsf{NIZKVerify}(crs, x, \pi^*) \middle| \begin{array}{l} crs \leftarrow \mathsf{NIZKSetup}(1^\lambda) \\ \pi^* \leftarrow \mathcal{A}(crs, x) \end{array}\right] = negl(\lambda)$$

and the probability is taken over the random coins of $\mathsf{NIZKSetup}$ *and $\mathcal{A}$.*

We define the classical notion of simulation-based zero-knowledge.

Definition 16 (Zero-Knowledge)**.** *A NIZK scheme* $\mathsf{NIZK} = (\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ *is zero-knowledge if there exists a PPT simulator* $\mathsf{NIZKSim} = (\mathsf{NIZKSim}_1, \mathsf{NIZKSim}_2)$ *such that for all $\lambda \in \mathbb{N}$, all $(x, w) \in \mathcal{R}$, and all PPT adversaries $\mathcal{A}$ it holds that*

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(crs, \pi) \middle| \begin{array}{l} crs \leftarrow \mathsf{NIZKSetup}(1^\lambda) \\ \pi \leftarrow \mathsf{NIZKProve}(crs, x, w) \end{array}\right] - \\ \quad \Pr\left[1 \leftarrow \mathcal{A}(crs, \pi) \middle| \begin{array}{l} (crs, \mathsf{td}) \leftarrow \mathsf{NIZKSim}_1(1^\lambda) \\ \pi \leftarrow \mathsf{NIZKSim}_2(crs, \mathsf{td}, x) \end{array}\right] \end{array} \right| = negl(\lambda)$$

where the probabilities are taken over the random coins of $\mathsf{NIZKSetup}$*,* $\mathsf{NIZKProve}$*,* $\mathsf{NIZKSim}_1$*,* $\mathsf{NIZKSim}_2$*, and $\mathcal{A}$.*

Malicious Designated Verifier NIZK. One can also consider a weakening the standard notion of NIZK that allows a designated verifier (MDV-NIZK) to choose part of the public parameters maliciously and remains secure for polynomially many proofs [LQR+19]. The advantage of this relaxation is that MDV-NIZK schemes are known under a larger class of assumptions, such as CDH [QRW19]. All of the results in this work can use NIZKs and MDV-NIZKs interchangeably.

$\mathcal{F}_{crs}$ interacts with a set of parties $(P_1, \ldots, P_n)$ and an adversary and it is parametrized by a distribution $\mathcal{D}$:

(1) On input (sid, P_i, P_j) from party P_i, sample $crs \leftarrow\!\!\$\, \mathcal{D}$ and send (sid, crs) to P_i and (sid, crs, P_i, P_j) to P_j.

(2) On input (sid, P_i, P_j) from party P_j, check if sid was already used and return (sid, crs) to P_j and the adversary in case; otherwise send nothing.

Figure 2: Ideal Functionality $\mathcal{F}_{crs}$ [CR03].

3 Universal Composability

We give a brief overview of the definition of UC-security [Can01] with static corruption and we establish some terminology. The following assumes familiarity with the basic concepts of UC-security and we refer the reader to [Can01] for a comprehensive discussion on the matter. We denote the environment by $\mathcal{Z}$. For a real protocol Π and an adversary $\mathcal{A}$ we write $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ to denote the ensemble corresponding to the protocol execution. For an ideal functionality $\mathcal{F}$ and an adversary Sim we write $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$ to denote the distribution ensemble of the ideal world execution.

Definition 17 (Security)**.** *A protocol Π UC-realizes a functionality $\mathcal{F}$ if for all PPT adversaries $\mathcal{A}$ corrupting a subset of parties there exists a simulator Sim such that for all environments $\mathcal{Z}$ it holds that the following ensembles*

$$\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$$

are computationally indistinguishable.

Since all of our protocols will be in the common reference string model we define the corresponding functionality in Figure 2.

3.1 Two-Round Oblivious Transfer

In the following we define the notion of UC-secure two-round oblivious transfer (OT). Two-round OT can be realized from a wide range of assumptions, including CDH [DGH+19] and LWE [PVW08]. The syntax is given below and the corresponding ideal functionality is shown in Figure 3.

Definition 18 (Oblivious Transfer)**.** *A two-round string-OT* $\mathsf{OT} = (\mathsf{OTSetup}, \mathsf{OTSend}, \mathsf{OTReceive}, \mathsf{OTDecode})$ *is defined as the following tuple of algorithms.*

$\mathsf{OTSetup}(1^\lambda) \to crs$: *On input the security parameter 1^λ, the generation algorithm returns a common reference string crs.*

$\mathsf{OTReceive}(crs, b) \to (\mathsf{ot}_1, r)$: *On input the common reference string crs and a bit b, the receiver algorithm returns a first message ot_1 and a state r.*

$\mathsf{OTSend}(crs, \mathsf{ot}_1, m_0, m_1) \to \mathsf{ot}_2$: *On input the common reference string crs, a first message ot_1, and a pair of strings $(m_0, m_1) \in \{0,1\}^{2\lambda}$, the sender algorithm returns a second message ot_2.*

$\mathcal{F}_{\mathsf{OT}}$ interacts with an ideal sender S and an ideal receiver R as follows:

(1) On input $(sid, send, m_0, m_1)$ from the sender, store (m_0, m_1).

(2) On input $(sid, receive, b)$ from the receiver, check if some inputs (m_0, m_1) have been recorded for sid. If this is the case, send m_b to the receiver and sid to the adversary; otherwise send nothing.

Figure 3: Ideal Functionality $\mathcal{F}_{\mathsf{OT}}$.

$\mathsf{OTDecode}(crs, \mathsf{ot}_2, r) \to m$: *On input the common reference string crs, a second message ot_2, and the receiver state r, the decoding algorithm returns a message m.*

Correctness is given in the following.

Definition 19 (Correctness)**.** *A string-OT* $\mathsf{OT} = (\mathsf{OTSetup}, \mathsf{OTSend}, \mathsf{OTReceive}, \mathsf{OTDecode})$ *is correct if for all $\lambda \in \mathbb{N}$, for all bits $b \in \{0,1\}$, and any pair of messages $(m_0, m_1) \in \{0,1\}^{2\lambda}$ it holds that*

$$\Pr\left[m_b = \mathsf{OTDecode}(crs, \mathsf{ot}_2, r) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{OTSetup}(1^\lambda) \\ (\mathsf{ot}_1, r) \leftarrow \mathsf{OTReceive}(crs, b) \\ \mathsf{ot}_2 \leftarrow \mathsf{OTSend}(crs, \mathsf{ot}_1, m_0, m_1) \end{array}\right] = 1$$

where the probability is taken over the random coins of $\mathsf{OTSetup}$*,* $\mathsf{OTReceive}$*, and* OTSend*.*

3.2 Two-Round Two-Party Computation

We define two-party computation (2PC) and we focus on the case where the protocol consists of a single message per party. Constructing a UC-secure protocol (with sublinear communication) is going to be one of the main objective of this work. The ideal functionality is given in Figure 4.

Definition 20 (Two-Party Computation)**.** *A 2PC protocol* $\mathsf{2PC} = (\mathsf{2PCSetup}, \mathsf{2PCReceive}, \mathsf{2PCSend}, \mathsf{2PCDecode})$ *is defined with respect to a function f and consists of the following algorithms.*

$\mathsf{2PCSetup}(1^\lambda) \to crs$: *On input the security parameter 1^λ, the generation algorithm returns a common reference string crs.*

$\mathsf{2PCReceive}(crs, x) \to (m_1, r)$: *On input the common reference string crs and an input x, the receiver algorithm returns a first message m_1 and a state r.*

$\mathsf{2PCSend}(crs, m_1, y) \to m_2$: *On input the common reference string crs, a first message m_1, and an input y, the sender algorithm returns a second message m_2.*

$\mathsf{2PCDecode}(crs, m_2, r) \to z$: *On input the common reference string crs, a second message m_2, and the state r, the decoding algorithm returns a message z.*

Correctness is given in the following.

Definition 21 (Correctness)**.** *A 2PC protocol* $\mathsf{2PC} = (\mathsf{2PCSetup}, \mathsf{2PCReceive}, \mathsf{2PCSend}, \mathsf{2PCDecode})$ *is correct if for all $\lambda \in \mathbb{N}$, for all inputs (x, y) it holds that*

$$\Pr\left[f(x,y) = \mathsf{2PCDecode}(crs, m_2, r) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{2PCSetup}(1^\lambda) \\ (m_1, r) \leftarrow \mathsf{2PCReceive}(crs, x) \\ m_2 \leftarrow \mathsf{2PCSend}(crs, m_1, y1) \end{array}\right] = 1$$

where the probability is taken over the random coins of $\mathsf{2PCSetup}$*,* $\mathsf{2PCReceive}$*, and* $\mathsf{2PCSend}$*.*

$\mathcal{F}_{\mathsf{2PC}}$ is parametrized by a function f and interacts with an ideal pair of parties (P_1, P_2) as follows:

(1) On input $(sid, receive, x)$ from the P_1, store x.

(2) On input $(sid, send, y)$ from P_2, check if some input x have been recorded for sid. If this is the case, send $f(x, y)$ to P_1 and sid to the adversary; otherwise send nothing.

Figure 4: Ideal Functionality $\mathcal{F}_{\mathsf{2PC}}$.

4 Laconic Cryptography

In this section we define the main objects of interest of our work. All of the primitives that we consider are in the common reference string (CRS) model.

4.1 Laconic Oblivious Transfer

In the following we introduce laconic oblivious transfer (LOT), the main object of interest of our work [CDG+17]. For presentation purposes, we define two variants of LOT with different efficiency and security requirements. Looking ahead, we will show a generic transformation, thereby simplifying the task of designing a LOT scheme to its simplest flavour.

As discussed by Cho et al. [CDG+17], we can ignore any security requirement on the receiver side: Although some bits of the database could be leaked, any LOT scheme can be generically upgraded to achieve receiver security through a generic transformation, i.e., by running the LOT under a 2PC protocol. In contrast to our work, Cho et al. [CDG+17] also considers the property of updatability for a LOT, which allows one to modify a bit of the database and update the digest in a significantly more efficient way than computing it from scratch. Since it is not relevant for our applications, we omit this property.

4.1.1 Weak Laconic Oblivious Transfer

A weak LOT scheme is identical to the standard LOT [CDG+17] except that the sender algorithm does not take as input two messages but chooses itself two random keys (k_0, k_1). This can be seen as the analogous to key encapsulation for encryption schemes. The syntax is given in the following.

Definition 22 (Weak Laconic Oblivious Transfer)**.** *A weak LOT* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *is defined as the following tuple of algorithms.*

$\mathsf{Setup}(1^\lambda) \to crs$: *On input the security parameter* 1^λ*, the generation algorithm returns a common reference string* crs*.*

$\mathsf{Hash}(crs, D) \to (d, \tilde{D})$: *On input the common reference string* crs *and a database* $D \in \{0,1\}^*$*, the hashing algorithm returns a digest* d *and a state* $\tilde{D}$*.*

$\mathsf{KGen}(crs, d, L) \to (c, k_0, k_1)$: *On input the common reference string* crs*, a digest* d*, and a location* $L \in \mathbb{N}$*, the key generation algorithm returns a ciphertext* c *and a pair of keys* (k_0, k_1)*.*

$\mathsf{Receive}^{\tilde{D}}(crs, c, L) \to m$: *On input the common reference string* crs*, a ciphertext* c*, and a location* $L \in \mathbb{N}$*, the receiver algorithm (with random access to* $\tilde{D}$*) returns a key* k*.*

The definition of correctness is given in the following.

Definition 23 (Correctness)**.** *A weak LOT* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *is correct if for all* $\lambda \in \mathbb{N}$*, for all databases* D *of size polynomial in* λ*, and for all memory locations* $L \in \{1, \ldots, |D|\}$ *it holds that*

$$\Pr\left[k_{D[L]} = \mathsf{Receive}^{\tilde{D}}(crs, c, L) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, \tilde{D}) \leftarrow \mathsf{Hash}(crs, D) \\ (c, k_0, k_1) \leftarrow \mathsf{KGen}(crs, d, L) \end{array}\right] = 1$$

where the probability is taken over the random coins of Setup *and* KGen*.*

We also consider a weaker notion of correctness where the above probability is bounded from below by $1 - 1/poly(\lambda)$, for some polynomial function $poly$. We refer to such notion as $(1/poly(\lambda))$-correctness. A weak LOT is required to satisfy only a rudimental notion of sender security that we define in the following. Loosely speaking, we require that the adversary is not able to predict both keys (k_0, k_1) in their entirety.

Definition 24 (Weak Sender Security)**.** *A weak LOT* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *is weakly sender secure if for all* $\lambda \in \mathbb{N}$ *and for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function negl such that*

$$\Pr\left[(k_0, k_1) = \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (c, k_0, k_1) \leftarrow \mathsf{KGen}(crs, d, L) \end{array}\right] = negl(\lambda)$$

where the probability is taken over the random coins of Setup*,* $\mathcal{A}_1$*, and* KGen*.*

Efficiency. For efficiency, it is only required that the size of the digest d is at most half of that of D, i.e, the Hash function is 2-to-1 compressing. In particular, there is no bound on the efficiency of the algorithms and on the size of the ciphertext, except for being polynomial in the security parameter. We refer to such notion as *semi-efficiency*. Jumping ahead, we will show a generic transformation from any semi-efficient LOT to a full-fledged LOT without additional assumptions.

4.1.2 Indistinguishable Laconic Oblivious Transfer

This version of LOT is equivalent to that introduced in [CDG+17], except that we upgrade the definition of sender security to the malicious settings. Note that we consider without loss of generality schemes that transfer a single bit (bit-LOT), which can be turned into multi-bit schemes (string-LOT) by simply computing multiple ciphertexts over the same digest. The security is preserved by a standard hybrid argument.

Definition 25 (Laconic Oblivious Transfer)**.** *A LOT* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *is defined as the following tuple of algorithms.*

$\mathsf{Setup}(1^\lambda) \to crs$: *On input the security parameter* 1^λ*, the generation algorithm returns a common reference string* crs*.*

$\mathsf{Hash}(crs, D) \to (d, \tilde{D})$: *On input the common reference string* crs *and a database* $D \in \{0,1\}^*$*, the hashing algorithm returns a digest* d *and a state* $\tilde{D}$*.*

$\mathsf{Send}(crs, d, L, m_0, m_1) \to c$: *On input the common reference string* crs*, a digest* d*, a location* $L \in \mathbb{N}$*, and a pair of messages* $(m_0, m_1) \in \{0,1\}^2$*, the sender algorithm returns a ciphertext* c*.*

$\mathsf{Receive}^{\tilde{D}}(crs, c, L) \to m$: *On input the common reference string* crs*, a ciphertext* c*, and a database location* $L \in \mathbb{N}$*, the receiver algorithm (with random access to* $\tilde{D}$*) returns a message* m*.*

The definition of correctness is given in the following.

Definition 26 (Correctness)**.** *A LOT* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *is correct if for all* $\lambda \in \mathbb{N}$*, for all databases* D *of size polynomial in* λ*, for all memory locations* $L \in \{1, \ldots, |D|\}$*, and any pair of messages* $(m_0, m_1) \in \{0,1\}^2$ *it holds that*

$$\Pr\left[m_{D[L]} = \mathsf{Receive}^{\tilde{D}}(crs, c, L) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, \tilde{D}) \leftarrow \mathsf{Hash}(crs, D) \\ c \leftarrow \mathsf{Send}(crs, d, L, m_0, m_1) \end{array}\right] = 1$$

where the probability is taken over the random coins of Setup *and* Send*.*

The fact that the digest of a LOT is compressing, makes even defining sender security a non-trivial task. We put forward the following indistinguishability-based definition, which suffices for our purposes.

Definition 27 (Sender Indistinguishability)**.** *A LOT* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *is sender secure if for all* $\lambda \in \mathbb{N}$*, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, and for all polynomial functions* $poly(\lambda)$ *there exists a negligible function* $negl$ *such that*

$$\Pr\left[\varepsilon_0 > \frac{1}{poly(\lambda)} \text{ and } \varepsilon_1 > \frac{1}{poly(\lambda)}\right] = negl(\lambda)$$

where, for $\beta \in \{0,1\}$*,* ε_β *is defined as*

$$\varepsilon_\beta = \left| \begin{array}{r} \Pr\left[1 \leftarrow \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ c \leftarrow \mathsf{Send}(crs, d, L, m_0, m_1) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ c \leftarrow \mathsf{Send}(crs, d, L, r_0, r_1) \end{array}\right] \end{array} \right|$$

where $r_\beta \leftarrow_\$ \{0,1\}$*,* $r_{1-\beta} = m_{1-\beta}$*, and the probabilities are taken over the random coins of* Setup*,* Send*, and* $\mathcal{A}_1$*.*

Efficiency. Here it is required that the size of the digest d is a fixed polynomial in the security parameter. Furthermore, we also impose a bound on the running time of the Hash algorithm of $|D| \cdot poly(log(|D|), \lambda)$ and on the time complexity of Setup, Encode, and Receive of $poly(log(|D|), \lambda)$. These are the same efficiency requirements of Cho et al. [CDG+17].

4.2 Laconic Function Evaluation

We define laconic function evaluation (LFE), a primitive recently introduced by Quack, Wichs, and Wee [QWW18]. In the same work, they presented a construction assuming the sub-exponential hardness of the LWE problem. The scheme is defined with respect to a class of circuits parametrized by the input and the circuit size, which we denote by $\mathfrak{C}$.

Definition 28 (Laconic Function Evaluation)**.** *A LFE* $\mathsf{LFE} = (\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is defined as the following tuple of algorithms.*

$\mathsf{LFESetup}(1^\lambda) \to crs$: *On input the security parameter* 1^λ*, the generation algorithm returns a common reference string* crs*.*

$\mathsf{LFECompress}(crs, \mathcal{C}) \to (d, r)$: *On input the common reference string* crs *and a circuit* $\mathcal{C}$*, the compression algorithm returns a digest* d *and a decoding information* r*.*

$\mathsf{LFEEnc}(crs, d, x) \to c$: *On input the common reference string* crs*, a digest* d*, and a message* x*, the encryption algorithm returns a ciphertext* c*.*

$\mathsf{LFEDec}(crs, c, r) \to y$: *On input the common reference string* crs*, a ciphertext* c*, and a decoding string* y*, the decoding algorithm returns a message* y*.*

The definition of correctness is given in the following.

Definition 29 (Correctness)**.** *A* $\mathsf{LFE} = (\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is correct if for all* $\lambda \in \mathbb{N}$*, for all circuits* $\mathcal{C} \in \mathfrak{C}$*, and all messages* x *it holds that*

$$\Pr\left[\mathcal{C}(x) = \mathsf{LFEDec}(crs, c, r) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{LFESetup}(1^\lambda) \\ (d, r) \leftarrow \mathsf{LFECompress}(crs, \mathcal{C}) \\ c \leftarrow \mathsf{LFEEnc}(crs, d, x) \end{array}\right] = 1$$

where the probability is taken over the random coins of $\mathsf{LFESetup}$*,* $\mathsf{LFECompress}$*, and* LFEEnc*.*

We require that the encryption of a message x with respect to a compressed circuit $\mathcal{C}$ reveals nothing beyond $\mathcal{C}(x)$. Here we define directly the notion of semi-malicious security, where the adversary is allowed to specify the random coins of for the compression of the circuit.

Definition 30 (Security)**.** *A LFE* $\mathsf{LFE} = (\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is (semi-maliciously) secure if there exists a PPT simulator* LFESim *such that for all admissible PPT attackers* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $negl$ *such that*

$$\left|\begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{LFESetup}(1^\lambda) \\ (x, \mathcal{C}, s, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (d, r) \leftarrow \mathsf{LFECompress}(crs, \mathcal{C}; s) \\ c \leftarrow \mathsf{LFEEnc}(crs, d, x) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{LFESetup}(1^\lambda) \\ (x, \mathcal{C}, s, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (d, r) \leftarrow \mathsf{LFECompress}(crs, \mathcal{C}; s) \\ c \leftarrow \mathsf{LFESim}(crs, d, \mathcal{C}, \mathcal{C}(x)) \end{array}\right] \end{array}\right| = negl(\lambda)$$

where the attacker is admissible if $\mathcal{C} \in \mathfrak{C}$ *and the probability is taken over the random coins of* $\mathsf{LFESetup}$*,* $\mathcal{A}_1$*,* $\mathsf{LFECompress}$*,* LFEEnc *and* LFESim*.*

Function hiding says that the compressed version of a circuit carries no information about the circuit itself.

Definition 31 (Function-Hiding)**.** *A LFE* $\mathsf{LFE} = (\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is function-hiding if there exists a PPT simulator* $\mathsf{LFESim}_{\mathsf{FH}}$ *such that for all admissible PPT attackers* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $negl$ *such that*

$$\left|\begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}_2(d, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{LFESetup}(1^\lambda) \\ (\mathcal{C}, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (d, r) \leftarrow \mathsf{LFECompress}(crs, \mathcal{C}) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}_2(d, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{LFESetup}(1^\lambda) \\ (\mathcal{C}, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ d \leftarrow \mathsf{LFESim}_{\mathsf{FH}}(crs, \mathfrak{C}) \end{array}\right] \end{array}\right| = negl(\lambda)$$

where the attacker is admissible if $\mathcal{C} \in \mathfrak{C}$ *the probability is taken over the random coins of* $\mathsf{LFESetup}$, $\mathcal{A}_1$, *and* $\mathsf{LFESim}_{\mathsf{FH}}$.

Efficiency. The scheme is requires to be laconic in the sense that the size of crs, d, and c must be sublinear in the size of the circuit $\mathcal{C}$. In the scheme constructed in [QWW18], the size of crs and c is independent of the circuit width (but grows with the circuit depth), on the other hand the size of d is bounded by a fixed polynomial in the security parameter. In terms of computation, an LFE is required to be *Bob-optimized*, i.e., the computational complexity of the LFEEnc algorithm depends only on the size of its input y and in particular does not depend on the size of the circuit (except for its depth).

4.3 Laconic Conditional Disclosure of Secrets

Conditional disclosure of secrets (CDS) [AIR01] for a language $\mathcal{L}$ in NP with relation $\mathcal{R}$ is the two-round analog of witness encryption [GGSW13]: Given a statement x and a message m from the sender, the receiver is able to recover m if $x \in \mathcal{L}$, whereas m stays hidden if this is not the case. Furthermore, the witness w for x should be kept secret from the eyes of the sender.

Definition 32 (Conditional Disclosure of Secrets)**.** *A CDS scheme* $\mathsf{CDS} = (\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSend}, \mathsf{CDSDecode})$ *for an NP-langauge* $\mathcal{L}$ *is defined as the following tuple of algorithms.*

$\mathsf{CDSSetup}(1^\lambda) \to crs$: *On input the security parameter* 1^λ*, the generation algorithm returns a common reference string* crs.

$\mathsf{CDSReceive}(crs, x, w) \to (\mathsf{cds}_1, r)$: *On input the common reference string* crs *and a statement-witness pair* (x, w)*, the receiver algorithm returns a first message* cds_1 *and a state* r.

$\mathsf{CDSSend}(crs, x, m, \mathsf{cds}_1) \to \mathsf{cds}_2$: *On input the common reference string* crs*, a statement* x*, a message* $m \in \{0,1\}^*$*, a first message* cds_1*, the sender algorithm returns a second message* cds_2.

$\mathsf{CDSDecode}(crs, \mathsf{cds}_2, r) \to m$: *On input the common reference string* crs*, a second message* cds_2*, and the receiver state* r*, the decoding algorithm returns a message* m.

Correctness requires that the decoding is successful if $x \in \mathcal{L}$.

Definition 33 (Correctness)**.** *A CDS scheme* $\mathsf{CDS} = (\mathsf{CDSSetup}, \mathsf{CDSSend}, \mathsf{CDSReceive}, \mathsf{CDSDecode})$ *is correct if for all* $\lambda \in \mathbb{N}$*, all* $(x, w) \in \mathcal{R}$*, and any message* $m \in \{0,1\}^*$ *it holds that*

$$\Pr\left[m = \mathsf{CDSDecode}(crs, \mathsf{cds}_2, r) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{CDSSetup}(1^\lambda) \\ (\mathsf{cds}_1, r) \leftarrow \mathsf{CDSReceive}(crs, x, w) \\ \mathsf{cds}_2 \leftarrow \mathsf{CDSSend}(crs, x, m, \mathsf{cds}_1) \end{array}\right] = 1$$

where the probability is taken over the random coins of $\mathsf{CDSSetup}$, $\mathsf{CDSReceive}$, *and* $\mathsf{CDSSend}$.

The central requirement for a CDS scheme is that the message is hidden if the given statement is not in the corresponding NP-language.

Definition 34 (Message Indistinguishability)**.** *A CDS scheme* $\mathsf{CDS} = (\mathsf{CDSSetup}, \mathsf{CDSSend}, \mathsf{CDSReceive}, \mathsf{CDSDecode})$ *is message-indistinguishable if for all* $\lambda \in \mathbb{N}$*, for all admissible PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, for all* $x \notin \mathcal{L}$ *there exists a negligible function negl such that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(\mathsf{cds}_2, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{CDSSetup}(1^\lambda) \\ (\mathsf{cds}_1, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ \mathsf{cds}_2 \leftarrow \mathsf{CDSSend}(crs, x, m_0, \mathsf{cds}_1) \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}_2(\mathsf{cds}_2, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{CDSSetup}(1^\lambda) \\ (\mathsf{cds}_1, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ \mathsf{cds}_2 \leftarrow \mathsf{CDSSend}(crs, x, m_1, \mathsf{cds}_1) \end{array}\right] \right| = negl(\lambda)$$

where $\mathcal{A}$ is admissible if m_0 and m_1 have equal length and the probabilities are taken over the random coins of $\mathsf{CDSSetup}$, $\mathsf{CDSSend}$, *and* $\mathcal{A}_1$.

Finally, we require that no information is leaked to the sender by the message of the receiver.

Definition 35 (Receiver Simulation)**.** *A CDS scheme* $\mathsf{CDS} = (\mathsf{CDSSetup}, \mathsf{CDSSend}, \mathsf{CDSReceive}, \mathsf{CDSDecode})$ *is receiver simulatable if there exists a PPT simulator* $\mathsf{CDSSim} = (\mathsf{CDSSim}_1, \mathsf{CDSSim}_2)$ *such that for all $\lambda \in \mathbb{N}$, all $(x, w) \in \mathcal{R}$, and all PPT adversaries $\mathcal{A}$ it holds that*

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(crs, \mathsf{cds}_1) \middle| \begin{array}{l} crs \leftarrow \mathsf{CDSSetup}(1^\lambda) \\ (\mathsf{cds}_1, \cdot) \leftarrow \mathsf{CDSReceive}(crs, x, w) \end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(crs, \mathsf{cds}_1) \middle| \begin{array}{l} (crs, \mathsf{td}) \leftarrow \mathsf{CDSSim}_1(1^\lambda) \\ \mathsf{cds}_1 \leftarrow \mathsf{CDSSim}_2(crs, \mathsf{td}, x) \end{array}\right] \end{array} \right| = negl(\lambda)$$

where the probabilities are taken over the random coins of $\mathsf{CDSSetup}$, $\mathsf{CDSReceive}$, CDSSim_1, CDSSim_2, *and* $\mathcal{A}$.

Efficiency. The standard requirement for a CDS scheme is to run in time polynomial in the security parameter and, possibly, in the size of the statement and of the witness. We say that a CDS is *laconic* if the communication complexity is a fixed polynomial $poly(\lambda)$ and in particular is independent of the size of w (up to a poly-logarithmic factor). This notion can be seen as the equivalent of succinct arguments [Mic94, Kil92] with implicit verification.

5 Context Security

In this Section we provide our definition of context security, which is a UC-inspired security notion [Can01]. The notion is specifically geared towards non-interactive secure computation protocols for which the communication complexity is sublinear in the receiver's input.

UC security ensures that a protocol is secure in any environment or context. However, the way UC-security is formalized makes it necessary that a straight-line simulator is able to extract the inputs of malicious parties, which immediately implies that the communication complexity scales with the size of the inputs of the parties. An important aspect in (standard) UC-security is that the environment is chosen after the simulator. Specialized-Simulator UC [Lin03] allows the simulator to depend on the environment, but not on its random coins. Context security aims for a compromise, while salvaging the original motivation of UC, namely that protocols remain secure in any context.

Unlike other works which relax UC security, e.g., [CLP13, BDH+17], the purpose of context security is *not* to get rid of a trusted setup. Rather, our goal is to obtain meaningful composability guarantees in the malicious setting while allowing for round-optimal protocols with communication complexity that is sublinear in the inputs. This is neither possible with UC/straight-line extraction, nor in the standalone simulation framework: While rewinding does allow to extract large inputs from protocols with small communication, it does not help in the two message setting. Our definition will only consider the single instance setting, but is crafted in a way that multi instance security can be dealt with via standard hybrid arguments. In this sense, we aim for conceptual simplicity rather than generality. We start by formally defining a context.

Definition 36 (Protocol Context)**.** *We say that a PPT machine $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$ is a context for two message protocol* $\Pi = (\mathsf{Setup}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$*, if it has the following syntactic properties: The first stage $\mathcal{Z}_1$ takes as input a common reference string crs (generated by* Setup*) and random coins* r_1 *and outputs a receiver message* rec *and a state* st*. The second phase $\mathcal{Z}_2$ takes as input*

the state st *and random coins* r_2*. The second phase is allowed to make queries* y *to a sender oracle* $\mathcal{O}(y)$*, which are answered by* $\mathsf{S}(crs, \mathsf{rec}, y)$ *(using fresh randomness from* r_2*). In the end the context outputs a bit* b^**. Define* $\mathcal{Z}(1^\lambda)$ *by*

- *Choose random tapes* $\mathsf{r}_1, \mathsf{r}_2$
- *Compute* $crs \leftarrow \mathsf{Setup}(1^\lambda)$
- $(\mathsf{st}, \mathsf{rec}) \leftarrow \mathcal{Z}_1(crs, \mathsf{r}_1)$
- $b^* \leftarrow \mathcal{Z}_2^{\mathsf{S}(crs, \mathsf{rec}, \cdot)}(\mathsf{st}, \mathsf{r}_2)$
- *Output* b^*

We will now provide our definition of context security.

Definition 37. *Let* $\Pi = (\mathsf{Setup}, \mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ *be a two-message protocol realizing a two-party functionality* $\mathcal{F}$*. We say that* Π *is* context-secure *if the following holds for every* Π*-context* $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$*. We require that there exists a context extractor* $\mathsf{Ext}_{\mathcal{Z}}$ *and simulators* $\mathsf{SimSetup}$ *and* Sim *such that the following holds for every* $\delta > 0$*:*

(1) $\mathsf{SimSetup}$ *runs in time polynomial in* λ *(but independent of the run-time of* $\mathcal{Z}_1$ *and* $\mathcal{Z}_2$*) and outputs a common reference string* crs *and a trapdoor* td*.*

(2) $\mathsf{Ext}_{\mathcal{Z}}$ *takes as input* $crs, \mathsf{st}, \mathsf{rec}$*, random coins* r^* *and a parameter* δ *and outputs a value* x^* *and an auxiliary string* aux*.* $\mathsf{Ext}_{\mathcal{Z}}$ *has overhead* $poly(\lambda, p(\lambda)) \cdot T_2$*, where* T_2 *is the overhead of* $\mathcal{Z}_2$ *and the polynomial is independent of* $\mathcal{Z}$*, only depending on* λ*.*

(3) Sim *takes as input* $\mathsf{td}, \mathsf{rec}, \mathsf{aux}$ *and a value* z *and outputs a sender-message* snd*. We require The overhead of* Sim *to be polynomial in the overhead of* λ*, but independent of* $\mathcal{Z}_1$ *and* $\mathcal{Z}_2$*.*

(4) The experiment $\mathcal{EZ}(1^\lambda, \delta)$ *is defined by*

- *Choose random tapes* $\mathsf{r}_1, \mathsf{r}_2$
- *Compute* $(crs, \mathsf{td}) \leftarrow \mathsf{SimSetup}(1^\lambda)$
- $(\mathsf{st}, d) \leftarrow \mathcal{Z}_1(crs, \mathsf{r}_1)$
- $(x^*, \mathsf{aux}) \leftarrow \mathsf{Ext}_{\mathcal{Z}}(crs, \mathsf{st}, \mathsf{rec}, \mathsf{r}^*, \delta)$
- $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}'(\cdot)}(\mathsf{st}, \mathsf{r}_2)$*, where* $\mathcal{O}'(y)$ *computes and outputs* $\mathsf{Sim}(\mathsf{td}, \mathsf{rec}, \mathsf{aux}, \mathcal{F}(x^*, y))$
- *Output* b^*

(5) (Security) It holds for every inverse polynomial $\varepsilon = \varepsilon(\lambda)$ *that*

$$|\Pr[\mathcal{Z}(1^\lambda) = 1] - \Pr[\mathcal{EZ}(1^\lambda, \varepsilon) = 1]| < \varepsilon,$$

except for finitely many λ*.*

Note specifically that we allow the ideal experiment $\mathcal{EZ}$ to depend on the distinguishing advantage ε, which looks unusual at first glance. However, it is precisely this dependence which allows us to prove context security for protocols with sublinear communication complexity. Specifically, this dependence will let us use the distinguisher-dependent simulation technique [JKKR17] to construct extractors $\mathsf{Ext}_{\mathcal{Z}}$. The extractor $\mathsf{Ext}_{\mathcal{Z}}$ has a similar syntax as a knowledge extractor [Nao03], however the main difference is that we will be able to prove this notion under

standard falsifiable assumptions. An important aspect which enables context security to be used in a meaningful way is that $\mathsf{Ext}_{\mathcal{Z}}$ does not get to see r_2.

The runtime requirement for $\mathsf{Ext}_{\mathcal{Z}}$ will make this notion compose benignly, unlike knowledge extractors [Nao03]. More specifically, it will be natural and convenient to define experiments iteratively, and this runtime requirement will ensure that runtimes of the experiments do not explode. An important aspect will be that extractors are not run recursively. Extractors will only be used in the first phase of a modified context, but the extractor itself does not use the first phase. That is, in successive uses of the technique the runtime will only grow additively, but not multiplicatively.

Limitations. We remark that context-security has to be used with care. If a context $(\mathcal{Z}_1, \mathcal{Z}_2)$ is such that $\mathcal{Z}_2$ uses a *long-term secret* generated by $\mathcal{Z}_1$, the extractor also needs to use this secret when simulating $\mathcal{Z}_2$. In particular, this means that in a hybrid proof we cannot make any further modifications to $\mathcal{Z}_1$ after extracting the receiver input, as this could invalidate the extracted input. For example, consider a functionality where the sender provides a (long term) signing key for a signature and the receiver obtains a signature on his input. Moreover, assume that the receiver is also given the verification key of the signature (and can therefore check the validity of signatures). Context-security allows us to extract the inputs of the receiver, but in order to do so the extractor needs to know the signing key. This is problematic if we later want to argue about the unforgeability of the signature, where a reduction is not given the signing key but only access to a signing oracle. Yet, by the observation above the context-extractor cannot extract the signing queries without being given the signing key. Note that this issue does not exist in the setting of UC security.

Other Properties. Finally, a very convenient property of context security is that it immediately implies game-based security notions with an efficient experiment. Specifically, phrase a game as a context $\mathcal{Z}$, apply context security and reason that the advantage in $\mathcal{EZ}$ is 0. Via the way we have defined context security this will immediately imply that also in the original experiment $\mathcal{Z}$ the advantage is negligible. For instance, for non-interactive secure computation (NISC) one can consider the following indistinguishability-based security definition. The malicious receiver first obtains a CRS, outputs a receiver message and two randomized functions F_0 and F_1 for which it holds that on any input x the distributions $F_0(x)$ and $F_1(x)$ are indistinguishable. The experiment flips a bit b and provides the adversary with a sender message which allows the receiver to evaluate F_b on his input. The adversary then has to guess b.

Any context-secure NISC scheme immediately also satisfies this notion. We can phrase the experiment, including the adversary as a context $(\mathcal{Z}_1, \mathcal{Z}_2)$ where $\mathcal{Z}_1$ outputs the receiver message (together with a state), and $\mathcal{Z}_2$ chooses the random bit b and computes the sender message via access to a sender oracle. Now, context-security lets us argue that in the experiment $\mathcal{EZ}$ the actual bit b used by the experiment is hidden from the view of the adversary, as the output of the oracle only depends on $F_b(x)$ (which is indistinguishable from $F_{1-b}(x)$) rather than on F_b.

5.1 Some Useful Functionalities

We define the functionalities corresponding to the primitives of interest of this work, defined in Section 4. Note that all of the functionalities that we consider are single-output, in the sense that only one party learns an output at the end of the execution.

(1) Laconic Oblivious Transfer: The ideal functionality $\mathcal{F}_{\mathsf{LOT}}(D, m_0, m_1, L)$ takes as input a database D, a pair of messages (m_0, m_1) and a location L. It returns $(m_{D[L]}, L)$.

(2) Laconic Function Evaluation: The ideal functionality $\mathcal{F}_{\mathsf{LFE}}(\mathcal{C}, x)$ takes as input a circuit $\mathcal{C}$ and an input x. It returns $\mathcal{C}(x)$.

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

- Sample $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$.
- Sample a uniform vector $(g_1, \ldots, g_{2\lambda}) \leftarrow_\$ \mathbb{G}^{2\lambda}$
- Return $crs = (\mathbb{G}, p, g, g_1, \ldots, g_{2\lambda})$.

$\mathsf{Hash}(crs, D)$:

- Parse crs as $(\mathbb{G}, p, g, g_1, \ldots, g_{2\lambda})$ and $D \in \{0,1\}^{2\lambda}$ as a bitstring.
- Compute $d = \prod_{i=1}^{2\lambda} g_i^{D[i]}$.
- Set $\tilde{D} = (D, d)$ and return $(d, \tilde{D})$.

$\mathsf{KGen}(crs, d, L)$:

- Parse crs as $(\mathbb{G}, p, g, g_1, \ldots, g_{2\lambda})$ and $d \in \mathbb{G}$.
- Sample a uniform $r \leftarrow_\$ \mathbb{Z}_p$.
- Compute the vector $c = (g_1^r, \ldots, g_{2\lambda}^r) \setminus g_L^r$.
- Set $k_0 = d^r$ and $k_1 = (d/g_L)^r$.
- Return (c, k_0, k_1).

$\mathsf{Receive}^{\tilde{D}}(crs, c, L)$:

- Parse $\tilde{D}$ as (D, d) and c as $(h_1, \ldots, h_{2\lambda})$.
- Return $\prod_{i=1, i\neq L}^{2\lambda} h_i^{D[i]}$.

Figure 5: CDH-based weak LOT.

(3) Laconic Conditional Disclosure of Secrets: The ideal functionality $\mathcal{F}_{\mathsf{CDS}}(w, x, m)$ is parametrized by an NP-language $\mathcal{L}$ with relation $\mathcal{R}$ and takes as input a statement x, a witness w, and a message m. The functionality returns m if $\mathcal{R}(w, x) = 1$ and $\perp$ otherwise.

6 Malicious Laconic Oblivious Transfer

In this section we present our constructions from different cryptographic hard problems. We first construct a 2-to-1 compressing weak LOT from the CDH or the LWE assumption, then we show how to generically bootstrap any 2-to-1 compressing weak LOT to a context-secure LOT with an arbitrary compression factor.

6.1 From Computational Diffie-Hellman

We present our scheme for a weak LOT with malicious security assuming the hardness of the CDH problem over a prime-order multiplicative cyclic group $(\mathbb{G}, p, g)$. For conceptual simplicity, we assume that a group element can be represented using λ-many bits, but the scheme can be generalized to arbitrary representations in a natural way. The construction is given in Figure 5.

6.1.1 Description

The public parameters of the scheme consist of a vector of group elements $(g_1, \ldots, g_{2\lambda})$ sampled uniformly at random. The hash of a string D is defined naturally as $d = \prod_{i=1}^{2\lambda} g_i^{D[i]}$, i.e., the i-th group element is included in the product if $D[i] = 1$ and is omitted otherwise. Say that we want to generate keys (k_0, k_1) and a ciphertext c such that c gives the adversary enough information to compute $k_{D[L]}$ but not $k_{D[L]\oplus 1}$. Then c is defined as a linear shift (for a uniform scalar r) of the public parameters $c = (g_1^r, \ldots, g_{2\lambda}^r)$, except that we omit the element g_L^r. Then k_0 is defined as d^r and k_1 as d^r/g_L^r.

We give some intuition why, given c is possible to recompute only one among k_0 and k_1: If $D[L] = 0$, then c contains enough information to recompute $d^r = \prod_{i=1}^{2\lambda} g_i^{D[i]\cdot r}$, since g_L^r would be excluded from the product anyway. On the other hand if $D[L] = 1$, then one can recompute $\prod_{i=1, i\neq L}^{2\lambda} g_i^{D[i]\cdot r} = d^r/g_L^r$, which is exactly the value of k_1. Finally, it is very easy to see that outputting both k_1 and k_0 simultaneously reveals the value of g_L^r, which is as hard as solving the CDH problem.

6.1.2 Analysis

We show that our scheme satisfied correctness.

Theorem 2 (Correctness)**.** *Let $(\mathbb{G}, p, g)$ be a prime-order multiplicative cyclic group. Then the construction $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ as defined in Figure 5 is correct.*

Proof of Theorem 2. The proof consists in the observation that

$$k_{D[L]} = \left(\frac{d}{g_L^{D[L]}}\right)^r = \left(\frac{\prod_{i=1}^{2\lambda} g_i^{D[i]}}{g_L^{D[L]}}\right)^r = \left(\prod_{i=1, i\neq L}^{2\lambda} g_i^{D[i]}\right)^r = \prod_{i=1, i\neq L}^{2\lambda} h_i^{D[i]}$$

which is exactly the output of the receiver. □

We now argue that our scheme satisfies weak sender security.

Theorem 3 (Weak Sender Security)**.** *Let $(\mathbb{G}, p, g)$ be a CDH-hard group. Then the construction $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ as defined in Figure 5 is weakly sender secure.*

Proof of Theorem 3. The theorem is shown with a reduction to the CDH assumption. Assume towards contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that

$$\Pr\left[(k_0, k_1) = \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (c, k_0, k_1) \leftarrow \mathsf{KGen}(crs, d, L) \end{array}\right] = \frac{1}{poly(\lambda)}$$

for some polynomial function *poly*. On input a CDH challenge $(\mathbb{G}, p, g, g^{a_1}, g^{a_2})$, the reduction samples a uniform vector $(x_1, \ldots, x_{2\lambda}) \leftarrow_\$ \mathbb{Z}_p^{2\lambda}$ and an index $\tilde{L} \leftarrow_\$ \{1, \ldots, 2\lambda\}$. Then it sets $crs = (\mathbb{G}, p, g, g^{x_1}, \ldots, g^{x_{L-1}}, g^{a_1}, g^{x_{L+1}}, \ldots, g^{x_{2\lambda}})$ and runs $(d, L, \mathsf{st}) \leftarrow \mathcal{A}_1(crs)$. If $L \neq \tilde{L}$ the reduction aborts, else it sets $c = (g^{a_2\cdot x_1}, \ldots, g^{a_2\cdot x_{L-1}}, g^{a_2\cdot x_{L+1}}, \ldots, g^{a_2\cdot x_{2\lambda}})$. Then it runs $(k_0, k_1) \leftarrow \mathcal{A}_2(c, \mathsf{st})$ and returns k_0/k_1.

The reduction runs in polynomial time and the inputs given to the adversary are identically distributed as those in the original game. Note that if $L = \tilde{L}$ and the adversary correctly guesses both k_0 and k_1, then we have that

$$\frac{k_0}{k_1} = \frac{d^{a_2}}{d^{a_2} \cdot g_L^{-a_2}} = g_L^{a_2} = g^{a_1\cdot a_2}.$$

Since this happens with probability at least $\frac{1}{poly(\lambda)\cdot 2\lambda}$, it contradicts the CDH assumption. □

6.2 From Learning with Errors

In the following we describe our weak LOT scheme assuming the hardness of the LWE problem. Before describing the actual scheme we define some common tools which are instrumental for our purposes.

6.2.1 The Rounding Function

Let $q = c \cdot p$ be an integer modulus. Define the rounding function $\lfloor \cdot \rfloor_c : \mathbb{Z}_q \to \mathbb{Z}_c$ as

$$\lfloor x \rfloor_c = \lfloor \bar{x} \cdot c/q \rfloor \mod c$$

where $\bar{x} \in \mathbb{Z}$ is an arbitrary residue-class representative of $x \in \mathbb{Z}_q$. We are going to use the following lemma about the rounding function $\lfloor \cdot \rfloor_c$, implicitly proven in [BPR12].

Lemma 2. *Let $q = c \cdot p$ be an integer modulus. Let $x \leftarrow\!\!\$\, \mathbb{Z}_q$ be distributed uniformly at random. Then it holds for all $v \in \{-B, \ldots, B\}$ that $\lfloor x + v \rfloor_c = \lfloor x \rfloor_c$, except with probability $(2B+1) \cdot c/q$ over the random choice of x.*

Proof of Lemma 2. Note that there are exactly c multiples of $p = q/c$ in $\mathbb{Z}_q$. Thus, the probability that a uniform $x \in \mathbb{Z}_q$ lands B-close from the nearest multiple of q/c is exactly

$$(2B + 1) \cdot c/q.$$

□

Another property of the rounding function is that it partitions the domain $\mathbb{Z}_q$ in c-many well defined intervals of size exactly $q/c = p$. More concretely, the i-th interval is $[p \cdot (i-1), p \cdot i - 1]$. This fact is going to be useful for our analysis.

6.2.2 Description

Our scheme is parametrized by the following variables:

- m : The size of the database $D \in \{0,1\}^m$.
- n : The dimension of the LWE problem.
- q : The modulus of the LWE problem. For ease of the analysis we are going to assume that q is of the form $c \cdot p$, for some constant c and an arbitrary integer p
- B : The bound on the absolute value of the noise, i.e., $e \in \{-B, \ldots, B\}$.
- c : A constant that parametrizes the function $\lfloor \cdot \rfloor_c$, which is used in our construction.

In the analysis we set constraints on the parameters on demand. In the end of the section we are going to show that such a set of constraints forms a satisfiable system of relations and that the resulting $\mathrm{LWE}_{n,q,\chi}$ problem is still in the regime of parameters for which it is conjectured to be hard. Our scheme is shown in Figure 6.

The scheme follows the same blueprint of the CDH-based construction but the noisy nature of the LWE problem introduces some complications. Consider the following toy example where the public parameters consists of a single matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$: Hashing a database $D \in \{0,1\}^m$ is done by simply taking a linear combination of the columns of $\mathbf{A}$, which results in a compressed column vector $\mathbf{d} = \mathbf{A}D$. Take any location L, the key generation algorithm samples a column vector $\mathbf{s}$ and computes a noisy inner product with each j-th column of $\mathbf{A}$, i.e., $b_j = \mathbf{s}^T \mathbf{a}_j + e_j$,

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

- For all $i \in \{1, \ldots, \lambda\}$: Sample $\mathbf{A}^{(i)} \leftarrow_\$ \mathbb{Z}_q^{n \times m}$.
- Return $crs = (\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\lambda)})$.

$\mathsf{Hash}(crs, D)$:

- Parse crs as $(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\lambda)})$ and $D \in \{0,1\}^m$ as a column vector.
- For all $i \in \{1, \ldots, \lambda\}$: Compute $\mathbf{d}^{(i)} = \mathbf{A}^{(i)} D$.
- Set $d = (\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(\lambda)})$, $\tilde{D} = (D, d)$, and return $(d, \tilde{D})$.

$\mathsf{KGen}(crs, d, L)$:

- Parse crs as $(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\lambda)})$ and d as $(\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(\lambda)})$.
- Sample $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^n$ and parse it as a column vector.
- For all $i \in \{1, \ldots, \lambda\}$:
  - Parse $(\mathbf{a}_1^{(i)}, \ldots, \mathbf{a}_m^{(i)})$ as the columns of $\mathbf{A}^{(i)}$.
  - Sample $(e_1^{(i)}, \ldots, e_m^{(i)}) \leftarrow_\$ \chi^m$.
  - For all $j \in \{1, \ldots, m\} \setminus \{L\}$: Compute $b_j^{(i)} = \mathbf{s}^T \mathbf{a}_j^{(i)} + e_j^{(i)}$.
  - Set $\mathbf{b}^{(i)}$ to be the row vector $(b_1^{(i)}, \ldots, b_{m-1}^{(i)})$.
  - Sample $(t_0^{(i)}, t_1^{(i)}) \leftarrow_\$ \mathbb{Z}_q^2$.
  - Set $k_0^{(i)} = \left\lfloor \mathbf{s}^T \mathbf{d}^{(i)} + t_0^{(i)} \right\rceil_c$ and $k_1^{(i)} = \left\lfloor -\mathbf{s}^T \mathbf{d}^{(i)} + \mathbf{s}^T \mathbf{a}_L^{(i)} + e_L^{(i)} + t_1^{(i)} \right\rceil_c$.
- Set $c = ((\mathbf{b}^{(1)}, t_0^{(1)}, t_1^{(1)}), \ldots, (\mathbf{b}^{(\lambda)}, t_0^{(\lambda)}, t_1^{(\lambda)}))$.
- Set $k_0 = k_0^{(1)} \| \ldots \| k_0^{(\lambda)}$ and $k_1 = k_1^{(1)} \| \ldots \| k_1^{(\lambda)}$.
- Return (c, k_0, k_1).

$\mathsf{Receive}^{\tilde{D}}(crs, c, L)$:

- Parse c as $((\mathbf{b}^{(1)}, t_0^{(1)}, t_1^{(1)}), \ldots, (\mathbf{b}^{(\lambda)}, t_0^{(\lambda)}, t_1^{(\lambda)}))$ and $\bar{D} = D \setminus D[L]$ as a column vector.
- For all $i \in \{1, \ldots, \lambda\}$: Compute $k^{(i)} = \left\lfloor (-1)^{D[L]} \cdot \mathbf{b}^{(i)} \bar{D} + t_{D[L]}^{(i)} \right\rceil_c$.
- Return $k = k^{(1)} \| \ldots \| k^{(\lambda)}$.

Figure 6: LWE-based weak LOT.

except for $j = L$. The keys are then set to $k_0 = \mathbf{s}^T \mathbf{d}$ and $k_1 = -\mathbf{s}^T \mathbf{d} + \mathbf{s}^T \mathbf{a}_L + e_L$. It is important to notice that that $k_0 + k_1 = \mathbf{s}^T \mathbf{a}_L + e_L$, which is an LWE sample but it is unknown to the eyes of the adversary. This suggests that outputting both values of k_0 and k_1 is equivalent to predicting an LWE sample.

As in our previous scheme, given the row vector $\mathbf{b} = (b_1, \ldots, b_m)$ one can reconstruct either k_0 or k_1, depending on the value of $D[L]$, by computing an inner product with $D \setminus D[L]$. However in this case sender cannot recover the exact value of $k_{D[L]}$ but only approximate it, due to the presence of noise in $\mathbf{b}$. To obviate this issue we leverage the fact that the rounding function $\lfloor \cdot \rceil_c$ is resilient to small perturbations of the inputs. With this tool at hand, we set the keys to

be the rounded values k_0 and k_1 (as computed above), which the receiver can recover with very high probability. However there are a few issues to be addressed:

(1) The output of the rounding function consists of a constant amount of bits, so there is just not enough entropy to argue about unpredictability.

(2) The values k_0 and k_1 are not necessarily uniform in $\mathbb{Z}_q$.

We address (1) by simply repeating the protocol in parallel λ-many times. On the other hand (2) can be solved by padding k_b with a random t_b, which is also given to the receiver.

6.2.3 Analysis

Note that setting $m \geq 2 \cdot \lceil log(q) \rceil \cdot n \cdot \lambda$ makes the first message 2-to-1 compressing. We now argue about the (weak) correctness of our scheme.

Theorem 4 (Correctness)**.** *The construction* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *as defined in Figure 6 is* $(1/\lambda^2)$*-correct.*

Proof of Theorem 4. We want to show that for all (D, L)

$$\Pr\left[k_{D[L]} = \mathsf{Receive}^{\tilde{D}}(crs, c, L) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, \tilde{D}) \leftarrow \mathsf{Hash}(crs, D) \\ (c, k_0, k_1) \leftarrow \mathsf{KGen}(crs, d, L) \end{array}\right] \geq 1 - \frac{1}{\lambda^2}.$$

For all $i \in \{1, \ldots, \lambda\}$, we expand expand the product

$$\begin{aligned} \mathbf{b}^{(i)} \bar{D} &= \sum_{j \neq L} D[j] \cdot b_j^{(i)} \\ &= \sum_{j \neq L} D[j] \cdot (\mathbf{s}^T \mathbf{a}_j^{(i)} + e_j^{(i)}) \\ &= \sum_{j \neq L} D[j] \cdot \mathbf{s}^T \mathbf{a}_j^{(i)} + \sum_{j \neq L} D[j] \cdot e_j^{(i)} \\ &= \mathbf{s}^T \left(\sum_{j \neq L} D[j] \cdot \mathbf{a}_j^{(i)} \right) + \sum_{j \neq L} D[j] \cdot e_j^{(i)} \\ &= \mathbf{s}^T \mathbf{d}^{(i)} - D[L] \cdot \mathbf{s}^T \mathbf{a}_L^{(i)} + \sum_{j \neq L} D[j] \cdot e_j^{(i)} \end{aligned}$$

by linearity. Applying this equality for all $i \in \{1, \ldots, \lambda\}$ we obtain

$$
\begin{aligned}
k^{(i)} &= \left\lfloor (-1)^{D[L]} \cdot \mathbf{b}^{(i)} \bar{D} + t^{(i)}_{D[L]} \right\rfloor_c \\
&= \left\lfloor (-1)^{D[L]} \cdot \left(\mathbf{s}^T \mathbf{d}^{(i)} - D[L] \cdot \mathbf{s}^T \mathbf{a}^{(i)}_L + \sum_{j \neq L} D[j] \cdot e^{(i)}_j \right) + t^{(i)}_{D[L]} \right\rfloor_c \\
&= \left\lfloor (-1)^{D[L]} \cdot \left(\mathbf{s}^T \mathbf{d}^{(i)} - D[L] \cdot \mathbf{s}^T \mathbf{a}^{(i)}_L + \sum_{j \neq L} D[j] \cdot e^{(i)}_j \right) + t^{(i)}_{D[L]} + D[L] \cdot e^{(i)}_L - D[L] \cdot e^{(i)}_L \right\rfloor_c \\
&= \left\lfloor (-1)^{D[L]} \cdot \mathbf{s}^T \mathbf{d}^{(i)} + D[L] \cdot \mathbf{s}^T \mathbf{a}^{(i)}_L + (-1)^{D[L]} \cdot \sum_{j=1}^{m} D[j] \cdot e^{(i)}_j + t^{(i)}_{D[L]} + D[L] \cdot e^{(i)}_L \right\rfloor_c \\
&= \left\lfloor (-1)^{D[L]} \cdot \mathbf{s}^T \mathbf{d}^{(i)} + D[L] \left(\cdot \mathbf{s}^T \mathbf{a}^{(i)}_L + e^{(i)}_L \right) + \underbrace{(-1)^{D[L]} \cdot \sum_{j=1}^{m} D[j] \cdot e^{(i)}_j}_{=\tilde{e}} + t^{(i)}_{D[L]} \right\rfloor_c
\end{aligned}
$$

Note that the absolute value of $\tilde{e}$ is bounded by $m \cdot B$. Further note that $t^{(i)}_{D[L]}$ is uniform in $\mathbb{Z}_q$. Thus, by Lemma 2 we have that

$$
\begin{aligned}
k^{(i)} &= \left\lfloor (-1)^{D[L]} \cdot \mathbf{s}^T \mathbf{d}^{(i)} + D[L] \left(\cdot \mathbf{s}^T \mathbf{a}^{(i)}_L + e^{(i)}_L \right) + \tilde{e} + t^{(i)}_{D[L]} \right\rfloor_c \\
&= \left\lfloor (-1)^{D[L]} \cdot \mathbf{s}^T \mathbf{d}^{(i)} + D[L] \left(\cdot \mathbf{s}^T \mathbf{a}^{(i)}_L + e^{(i)}_L \right) + t^{(i)}_{D[L]} \right\rfloor_c \\
&= k^{(i)}_{D[L]}
\end{aligned}
$$

except with probability

$$\frac{(2mB+1) \cdot c}{q}.$$

Setting $log(q) \geq 3log(\lambda) + log(2mBc + c)$ then we have that the above equality holds for all $i \in \{1, \ldots, \lambda\}$ except with probability at most $1/\lambda^3$. Taking a union bound over the λ-many parallel repetitions gives us the desired inequality. □

Finally we show that our scheme satisfies weak sender security We now argue that our scheme satisfies weak sender security.

Theorem 5 (Weak Sender Security)**.** *If the $LWE_{n,q,\chi}$ assumption holds, then the construction* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *as defined in Figure 6 is weakly sender secure.*

Proof of Theorem 5. We show the claim with a reduction against the $\mathrm{LWE}_{n,q,\chi}$ problem. Assume towards contradiction that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that

$$
\Pr \left[(k_0, k_1) = \mathcal{A}_2(c, \mathsf{st}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (c, k_0, k_1) \leftarrow \mathsf{KGen}(crs, d, L) \end{array} \right] = \frac{1}{poly(\lambda)}.
$$

Then we construct a reduction $(\mathcal{R})$ that solves $\mathrm{LWE}_{n,q,\chi}$ as follows. The reduction is given a challenge $(\mathbf{A}, \mathbf{u})$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m \cdot \lambda}$ and $\mathbf{u} \in \mathbb{Z}_q^{m \cdot \lambda}$ and parses $\mathbf{A}$ as the horizontal concatenation of λ-many $\mathbb{Z}_q^{n \times m}$ matrices $(\mathbf{A}^{(i)}, \ldots, \mathbf{A}^{(\lambda)})$. The adversary $\mathcal{A}_1$ is invoked on input

$crs = (\mathbf{A}^{(i)}, \ldots, \mathbf{A}^{(\lambda)})$ and returns a tuple (d, L, st). For each $i \in \{1, \ldots, \lambda\}$, the reduction sets the vector $\mathbf{b}^{(i)}$ to $(\mathbf{u}_{(i-1)m+1}, \ldots, \mathbf{u}_{(i-1)m+m})$ except that it omits the element $\mathbf{u}_{(i-1)m+L}$. Then samples $(t_0^{(i)}, t_1^{(i)}) \leftarrow_\$ \mathbb{Z}_q^2$ uniformly at random. The reduction returns to the adversary $\mathcal{A}_2$ the ciphertext $c = ((\mathbf{b}^{(1)}, t_0^{(1)}, t_1^{(1)}), \ldots, (\mathbf{b}^{(\lambda)}, t_0^{(\lambda)}, t_1^{(\lambda)}))$ along with the state st. The adversary outputs a pair (k_0, k_1) that is parsed by the reduction as the concatenation of two λ-long vectors in $\mathbb{Z}_c$: $k_0 = k_0^{(1)} \| \ldots \| k_0^{(\lambda)}$ and $k_1 = k_1^{(1)} \| \ldots \| k_1^{(\lambda)}$. For all $i \in \{1, \ldots, \lambda\}$, define $R_0^{(i)}$ as the interval of all elements $y_0 \in \mathbb{Z}_q$ such that $\lfloor y_0 \rfloor_c = k_0^{(i)}$ and define $R_1^{(i)}$ analogously. Let $R^{(i)}$ be the set of elements in $z \in \mathbb{Z}_q$ such that there exists a $y_0 \in R_0^{(i)}$ and a $y_1 \in R_1^{(i)}$ such that $y_0 + y_1 = z$. The reduction checks whether $\mathbf{u}_{(i-1)m+L} + t_0^{(i)} + t_1^{(i)} \in R^{(i)}$ and returns 1 if this is the case for all $i \in \{1, \ldots, \lambda\}$. Else it returns 0.

Note that all steps are efficiently computable: In particular, for all $k_b^{(i)} \in \mathbb{Z}_c$ the corresponding interval $R_b^{(i)}$ consists of $[p \cdot k_b^{(i)}, p \cdot (k_b^{(i)} + 1) - 1]$ and consequently also $R^{(i)}$ is trivial to compute. We analyze the probability that the reduction outputs 1 for two separate cases.

(1) Uniform Samples: In this case it is enough to observe that all elements $(\mathbf{u}_L, \ldots, \mathbf{u}_{(\lambda-1)m+L})$ are uniform in $\mathbb{Z}_q$ and completely independent from the view of the adversary. Therefore, for all $i \in \{1, \ldots, \lambda\}$, the probability that $\mathbf{u}_{(i-1)m+L} + t_0^{(i)} + t_1^{(i)} \in R^{(i)}$ is at most

$$\frac{\left|R^{(i)}\right|}{q} = \frac{\left|R_0^{(i)}\right| + \left|R_1^{(i)}\right|}{q} = \frac{2q}{c} \cdot \frac{1}{q} = \frac{2}{c}.$$

Setting $c = 4$ we obtain that the probability that the reduction outputs 1 is at most $1/2^\lambda$.

(2) LWE Samples: Note that in this case the input given to the adversary are identically distributed as an honest run of the protocol where the keys are (implicitly) set to

$$k_0^{(i)} = \left\lfloor \mathbf{s}^T \mathbf{d}^{(i)} + t_0^{(i)} \right\rceil_c \text{ and } k_1^{(i)} = \left\lfloor -\mathbf{s}^T \mathbf{d}^{(i)} + \mathbf{u}_{(i-1)m+L} + t_1^{(i)} \right\rceil_c$$

where $\mathbf{d}^{(i)}$ is part of the digest d given by the adversary. Let guess be the event that the adversary correctly guesses both k_0 and k_1. Then, by the law of total probability we have

$$\begin{aligned}
\Pr[1 \leftarrow \mathcal{R}] &= \Pr[1 \leftarrow \mathcal{R} | \mathsf{guess}] \Pr[\mathsf{guess}] + \Pr[1 \leftarrow \mathcal{R} | \bar{\mathsf{guess}}] \Pr[\bar{\mathsf{guess}}] \\
&\geq \Pr[1 \leftarrow \mathcal{R} | \mathsf{guess}] \Pr[\mathsf{guess}] \\
&= \Pr[1 \leftarrow \mathcal{R} | \mathsf{guess}] \frac{1}{poly(\lambda)}
\end{aligned}$$

by initial hypothesis. Conditioned on the fact that the adversary correctly guesses both k_0 and k_1, then for all $i \in \{1, \ldots, \lambda\}$ we have that

$$\mathbf{s}^T \mathbf{d}^{(i)} + t_0^{(i)} \in R_0^{(i)} \text{ and } -\mathbf{s}^T \mathbf{d}^{(i)} + \mathbf{u}_{(i-1)m+L} + t_1^{(i)} \in R_1^{(i)}$$

by definition, and consequently

$$\mathbf{s}^T \mathbf{d}^{(i)} + t_0^{(i)} - \mathbf{s}^T \mathbf{d}^{(i)} + \mathbf{u}_{(i-1)m+L} + t_1^{(i)} = \mathbf{u}_{(i-1)m+L} + t_1^{(i)} + t_0^{(i)} \in R^{(i)}.$$

Thus the reduction outputs 1 with probability 1. It follows that $\Pr[1 \leftarrow \mathcal{R}] \geq 1/poly(\lambda)$.

The two bounds above show that the probability that the reduction outputs 1 differs by a non-negligible amount depending on whether $(\mathbf{A}, \mathbf{u})$ is an LWE sample or not. This contradicts the $\mathrm{LWE}_{n,q,\chi}$ assumption and concludes our proof. □

$\bar{\mathsf{Setup}}(1^\lambda)$: Return $\mathsf{Setup}(1^\lambda)$.

$\bar{\mathsf{Hash}}(crs, D)$: Return $\mathsf{Hash}(crs, D)$.

$\bar{\mathsf{Send}}(crs, d, L, m_0, m_1)$:

- For all $i \in \{1, \ldots, \lambda\}$: Sample $(c^{(i)}, k_0^{(i)}, k_1^{(i)}) \leftarrow \mathsf{KGen}(crs, d, L)$.
- Set $\tilde{c} = (c^{(1)}, \ldots, c^{(\lambda)})$, $K_0 = (k_0^{(1)}, \ldots, k_0^{(\lambda)})$, and $K_1 = (k_1^{(1)}, \ldots, k_1^{(\lambda)})$.
- Sample two random strings $(t_0, t_1) \leftarrow\!\!\$\ \{0,1\}^{|K_0|+|K_1|}$.
- For all $b \in \{0,1\}$: Compute $C_b = \mathsf{GLEnc}(K_b; t_b) \oplus m_b$.
- Return $c = (\tilde{c}, C_0, C_1, t_0, t_1)$.

$\bar{\mathsf{Receive}}^{\tilde{D}}(crs, c, L)$:

- Parse c as $(\tilde{c}, C_0, C_1, t_0, t_1)$.
- For all $i \in \{1, \ldots, \lambda\}$: Compute $k^{(i)} \leftarrow \mathsf{Receive}^{\tilde{D}}(crs, c^{(i)}, L)$.
- Set $K = (k^{(1)}, \ldots, k^{(\lambda)})$.
- Return $\mathsf{GLEnc}(K; t_{D[L]}) \oplus \tilde{c}$.

Figure 7: From Weak to Full Sender Security.

6.2.4 Parameters

The above analysis fixes the following set of constraints:

- $m \geq 2 \cdot \lceil log(q) \rceil \cdot n \cdot \lambda$
- $log(q) \geq 3log(\lambda) + log(2mBc + c)$
- $c = 4$

Ignoring the constants, the constraint $O(log(q)) \geq O(log(\lambda) + log(m) + log(B))$ introduces a gap polynomial in λ in the modulo-to-noise ratio, since m is polynomially bounded. Note that the circular dependency of constraints (1) and (2) is always satisfied for a large enough q. The parameter n is free and can be set to the regime for which $\mathrm{LWE}_{n,q,\chi}$ is conjectured to be hard.

6.3 Upgrading Laconic Oblivious Transfer

We first upgrade the security and then the efficiency of a LOT, through generic transformations.

6.3.1 From Weak Sender Security to Sender Indistinguishability

We show how to generically upgrade any weak LOT into a LOT with sender indistinguishability. The compiler, shown in Figure 7, heavily relies on the tools introduced by Döttling et al. [DGH+19]: We first ensure that the adversary is not able to produce some digest d that allows him to predict k_0 for some values of c and k_1 for the others. This is done by amplifying the success probability of the adversary up to the point where it is no longer possible to be successful consistently on two different choices for the receiver's bit. Then we turn the search problem into a decision one, using the standard Goldreich-Levin hard-core predicate [GL89]. Here the function $\mathsf{GLEnc}(k; t)$ is defined as $\sum_{i=1}^{|k|} k_i \cdot t_i$, computed over $\mathbb{F}_2$.

The following theorem establishes our claim. The analysis is imported by the work of Döttling et al. [DGH+19] and adapted to our syntax. For completeness, we give the proof for the following theorem in Appendix A.

Theorem 6 (Weak to Full Sender Security)**.** *Let* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *be a weak LOT. Then* $\bar{\mathsf{LOT}} = (\bar{\mathsf{Setup}}, \bar{\mathsf{Hash}}, \bar{\mathsf{KGen}}, \mathsf{Receive})$ *as defined in Figure 7 is a standard LOT.*

6.3.2 From Indistinguishability Security to Context Security

We will now show that any LOT with indistinguishabilty security also suffices context security. To establish this, we will use distinguisher-dependent simulation. Technically, the theorem uses the same ideas of Theorem 6.3 in [DGH+19] and some paragraphs are verbatim from [DGH+19].

Theorem 7. *Assume that* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *satisfies indistinguishability security. Then it also satisfies context security.*

Proof of Theorem 7. We will prove the theorem via several lemmas. In order to do so, we will first provide constructions of the relevant algorithms. Fix a PPT-context $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$ for LOT. We start by defining a hybrid sender algorithm as follows:

$\mathcal{S}_i(crs, d, (x_1^*, \ldots, x_i^*), L, m_0, m_1)$:

- If $L > i$ compute and output $\mathsf{Send}(crs, d, L, m_0, m_1)$
- Otherwise, set $m'_{x_L^*} = m_{x_L^*}$ and $m'_{1-x_L^*} = 0$, compute and output $\mathsf{Send}(crs, d, L, m'_0, m'_1)$.

We will use the following notation. For an efficiently sampleable random variable $T \in \{0,1\}$ we will use the shorthand "Compute an approximation $\tilde{\mu}$ of $\mathsf{E}[T]$ with error δ" to denote the following algorithm which computes a sample average:

- Set $N = \lceil \lambda/\delta^2 \rceil$
- For $j = \{1, \ldots, N\}$ sample $t_j \leftarrow_\$ T$
- Output $\tilde{\mu} \leftarrow_\$ \frac{1}{N} \sum_{j=1}^{N} t_j$

We also define the following bit-extraction algorithm, which extracts an approximation of the i-th bit of the receiver's input.

$\mathsf{Extract}_i(crs, \mathsf{st}, d, \mathsf{r}_1, (x_1^*, \ldots, x_{j-1}^*), \varepsilon')$:

- Compute an approximation $\tilde{\mu}$ of $\mathsf{E}[\mathcal{Z}_2^{\mathcal{S}_{i-1}(crs,d,(x_1^*,\ldots,x_{i-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st})]$ with error $\varepsilon'/4$
- Compute an approximation $\tilde{\mu}_0$ of $\mathsf{E}[\mathcal{Z}_2^{\mathcal{S}_i(crs,d,(x_1^*,\ldots,x_{i-1}^*,0),\cdot,\cdot,\cdot)}(crs, \mathsf{st})]$ with error $\varepsilon'/4$
- Compute an approximation $\tilde{\mu}_1$ of $\mathsf{E}[\mathcal{Z}_2^{\mathcal{S}_i(crs,d,(x_1^*,\ldots,x_{i-1}^*,1),\cdot,\cdot,\cdot)}(crs, \mathsf{st})]$ with error $\varepsilon'/4$.
- Set $\tilde{\delta}_{i,0} \leftarrow |\tilde{\mu}_{i,0} - \tilde{\mu}_i|$
- Set $\tilde{\delta}_{i,1} \leftarrow |\tilde{\mu}_{i,1} - \tilde{\mu}_i|$
- If $\tilde{\delta}_{i,0} > \varepsilon'$ and $\tilde{\delta}_{i,1} > \varepsilon'$ abort and output $\perp$.
- else if $\tilde{\delta}_{i,1} > 2\varepsilon'$ output $x_i^* \leftarrow 0$
- Otherwise output $x_i^* \leftarrow 1$

We will now define the context extractor $\mathsf{Ext}_\mathcal{Z}$ and the simulator Sim. The setup-simulator SimSetup will be identical to the Setup algorithm, thus we will not specify it.

$\mathsf{Ext}_{\mathcal{Z}}(crs, \mathsf{st}, d, \mathsf{r}^*, \varepsilon)$:

- For $j = \{1, \ldots, i\}$:
  - Compute $x_j^* \leftarrow \mathsf{Extract}_j(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{j-1}^*), \varepsilon')$
- Output $x^* \leftarrow (x_1^*, \ldots, x_n^*)$

We will set ε' to a suitable value depending on ε

$\mathsf{Sim}(crs, d, x^*, z = (L, m))$: Set $m'_{x_L^*} = m$ and $m'_{1-x_L^*} = 0$, compute and output $\mathsf{Send}(crs, d, L, m'_0, m'_1)$.

Finally, we will choose $\varepsilon'(\varepsilon) = \varepsilon/n$. This choice of ε' will become meaningful in a moment. Now assume towards contradiction that there exists an inverse polynomial ε such that it holds for infinitely many λ that

$$|\Pr[\mathcal{Z}(1^\lambda) = 1] - \Pr[\mathcal{EZ}(1^\lambda, \varepsilon') = 1]| > \varepsilon,$$

where $\varepsilon' = \varepsilon/n$.

Hybrids. Consider the following sequence of hybrid experiments.

- Hybrid $\mathcal{H}_0$: This experiment is real context $\mathcal{Z}(1^\lambda)$.

For $i = \{1, \ldots, n\}$ define the following sequence of hybrids.

- Hybrid $\mathcal{H}_i$:
  - Choose random tapes $\mathsf{r}_1, \mathsf{r}_2$
  - Compute $crs \leftarrow \mathsf{Setup}(1^\lambda)$
  - Compute $(\mathsf{st}, d, \mathsf{aux}) \leftarrow \mathcal{Z}_1(crs, \mathsf{r}_1)$
  - For $j = \{1, \ldots, i\}$:
    - Compute $x_j^* \leftarrow \mathsf{Extract}_j(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{j-1}^*), \varepsilon')$
  - Compute $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}_i(\cdot,\cdot,\cdot)}(\mathsf{st}, \mathsf{r}_2)$, where $\mathcal{O}_i(L, m_0, m_1)$ computes $\mathcal{S}_i(crs, d, (x_1^*, \ldots, x_i^*), L, m_0, m_1)$.

Notice that it holds that $\mathcal{H}_n$ is identically distributed to $\mathcal{EZ}(1^\lambda, \varepsilon')$. Consequently, it holds by the averaging principle that there must exist an index $i^* \in \{1, \ldots, b\}$ such that

$$|\Pr[\mathcal{H}_{i^*} = 1] - \Pr[\mathcal{H}_{i^*-1} = 1]| > \varepsilon/n.$$

We remark that this dictates our choices of $\varepsilon' = \varepsilon/n$. We will construct an adversary against the indistinguishability security of LOT. The algorithm $\mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \varepsilon')$ computes approximations $\tilde{\delta}_{i^*,0}$ and $\tilde{\delta}_{i^*,1}$ of the advantages

$$\delta_{i^*,0} = |\Pr[\mathcal{Z}_2^{\mathcal{S}_i(crs,d,(x_1^*,\ldots,x_{i-1}^*,0),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1] - \Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1]|$$
$$\delta_{i^*,1} = |\Pr[\mathcal{Z}_2^{\mathcal{S}_i(crs,d,(x_1^*,\ldots,x_{i-1}^*,1),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1] - \Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1]|.$$

We will first establish that the approximations $\tilde{\delta}_{i^*,0}$ and $\tilde{\delta}_{i^*,1}$ are close to $\delta_{i^*,0}$ and $\delta_{i^*,1}$ respectively, except with negligible probability over the coins used to compute the approximations. We establish this by a routine application of the Hoeffding bound.

Lemma 3. *Fix crs, st, d and $x_1^*, \ldots, x_{i^*-1}^*$. Then it holds that*

$$|\tilde{\delta}_{i^*,0} - \delta_{i^*,0}| \leq \varepsilon'$$
$$|\tilde{\delta}_{i^*,1} - \delta_{i^*,1}| \leq \varepsilon'$$

except with probability $2^{-\lambda}$ over the choice of $\mathsf{r}_{\mathsf{Extract},i^*}$.

Proof of Lemma 3. The random variable $\tilde{\mu}_{i^*}$ is the average of $N = \lceil \lambda/\varepsilon'^2 \rceil$ samples of

$$Y = \mathcal{Z}_2^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st}).$$

Analogously, for $b \in \{0,1\}$ the random variables $\tilde{\mu}_{i^*,b}$ is the average of $N = \lceil \lambda/\varepsilon'^2 \rceil$ samples of

$$Y_b = \mathcal{Z}_2^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*,b),\cdot,\cdot,\cdot)}(crs, \mathsf{st}).$$

We can thus write

$$\delta_{i^*,b} = |\mathsf{E}[Y_b] - \mathsf{E}[Y]|.$$

Consequently, it holds by the Hoeffding inequality (Theorem 1) that

$$\Pr[|\tilde{\mu}_{i^*} - \mathsf{E}[Y]| > \varepsilon'/2] \leq 2e^{-2N(\varepsilon'/2)^2} \leq 2e^{-\lambda}$$

and for $b \in \{0,1\}$

$$\Pr[|\tilde{\mu}_{i^*,b} - \mathsf{E}[Y_b]| > \varepsilon'/2] \leq 2e^{-2N(\varepsilon'/2)^2} \leq 2e^{-\lambda}.$$

Given that

$$|\tilde{\mu}_{i^*} - \mathsf{E}[Y]| \leq \varepsilon'/2$$
$$|\tilde{\mu}_{i^*,0} - \mathsf{E}[Y_0]| \leq \varepsilon'/2$$
$$|\tilde{\mu}_{i^*,1} - \mathsf{E}[Y_1]| \leq \varepsilon'/2$$

and using that

$$\tilde{\delta}_{i^*,0} = |\tilde{\mu}_{i^*,0} - \tilde{\mu}_{i^*}|$$
$$\tilde{\delta}_{i^*,1} = |\tilde{\mu}_{i^*,1} - \tilde{\mu}_{i^*}|$$

we get that

$$|\tilde{\delta}_{i^*,0} - \delta_{i^*,0}| \leq \varepsilon'$$

and

$$|\tilde{\delta}_{i^*,1} - \delta_{i^*,1}| \leq \varepsilon'.$$

Consequently, it holds by a union-bound that

$$\Pr\left[|\tilde{\delta}_{i^*,0} - \delta_{i^*,0}| > \varepsilon' \text{ or } |\tilde{\delta}_{i^*,1} - \delta_{i^*,1}| > \varepsilon'\right] \leq 6 \cdot e^{-\lambda} \leq 2^{-\lambda}$$

which concludes the proof. □

Lemma 4. *Assume that $|\Pr[\mathcal{H}_{i^*} = 1] - \Pr[\mathcal{H}_{i^*-1} = 1]| \geq \varepsilon'$ for an $i^* \in [n]$. Then there exists a PPT adversary $\mathcal{B}$ which breaks the indistinguishability sender security of LOT.*

Proof of Lemma 4. First note that $\mathcal{H}_{i^*-1}$ and $\mathcal{H}_{i^*}$ are identical until $x_{i^*}^* \leftarrow \mathsf{Extract}_j(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \varepsilon')$ is computed in $\mathcal{H}_{i^*}$. We will therefore first rephrase $\mathcal{H}_{i^*-1}$ and $\mathcal{H}_{i^*}$. Towards this goal, define an alternate first stage $\mathcal{Z}_1'$ by

$\mathcal{Z}'_1(crs, \mathsf{r}_1)$:

- Compute $(\mathsf{st}, d) \leftarrow \mathcal{Z}'_1(crs, \mathsf{r}_1)$
- For $j = \{1, \ldots, i^* - 1\}$:
  - Compute $x^*_j \leftarrow \mathsf{Extract}_j(crs, \mathsf{st}, d, (x^*_1, \ldots, x^*_{j-1}), \varepsilon')$
- Output $(\mathsf{st}, d, \mathsf{aux}, (x^*_1, \ldots, x^*_{i^*}))$

We can now rephrase $\mathcal{H}_{i^*-1}$ and $\mathcal{H}_{i^*}$ by cutting of the common prefix.

- Hybrid $\mathcal{H}'_{i^*}(crs, \mathsf{st}, d, \mathsf{aux}, (x^*_1, \ldots, x^*_{i^*-1}))$:
  - Compute $(\mathsf{st}, d, \mathsf{aux}, (x^*_1, \ldots, x^*_{i^*-1})) \leftarrow \mathcal{Z}'1(crs, \mathsf{r}_1)$
  - Compute $b^* \leftarrow \mathcal{Z}'^{\mathcal{S}_{i^*-1}(crs,d,(x^*_1,\ldots,x^*_{i^*-1}),\cdot,\cdot,\cdot)}_2(\mathsf{st}, \mathsf{r}_2)$.
- Hybrid $\mathcal{H}'_{i^*}(crs, \mathsf{st}, d, \mathsf{aux}, (x^*_1, \ldots, x^*_{i^*-1}))$:
  - Compute $(\mathsf{st}, d, \mathsf{aux}, (x^*_1, \ldots, x^*_{i^*-1})) \leftarrow \mathcal{Z}'_1(crs, \mathsf{r}_1)$
  - Compute $x^*_{i^*} \leftarrow \mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x^*_1, \ldots, x^*_{i^*-1}), \varepsilon')$
  - Compute $b^* \leftarrow \mathcal{Z}'^{\mathcal{S}_{i^*}(crs,d,(x^*_1,\ldots,x^*_{i^*}),\cdot,\cdot,\cdot)}_2(\mathsf{st}, \mathsf{r}_2)$.

Further, fix crs, st, d, $(x^*_1, \ldots, x^*_{i^*-1})$ and the random tape $\mathsf{r}^*_{i^*}$ used for $\mathsf{Extract}_{i^*}$ and collect them in a variable z. Note that the parameters in inp also determine $x^*_{i^*}$. We will now define three events $\mathsf{GAP}(\mathsf{inp})$, $\mathsf{APPROX}(\mathsf{inp})$ and $\mathsf{GOOD}(\mathsf{inp})$ which only depend on inp.

(1) $\mathsf{GAP}(\mathsf{inp})$ holds, if and only if

$$|\Pr[\mathcal{H}'_{i^*}(\mathsf{inp}) = 1] - \Pr[\mathcal{H}'_{i^*-1}(\mathsf{inp}) = 1]| > 4\varepsilon',$$

where the random choices are over the random coins r_2 of $\mathcal{Z}_2$ and the random choices made by the oracles $\mathcal{S}_{i^*-1}$ and $\mathcal{S}_{i^*}$.

(2) Let $\tilde{\delta}_{i^*,0}$ and $\tilde{\delta}_{i^*,1}$ be the approximated values computed during the execution of $\mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x^*_1, \ldots, x^*_{i^*-1}), \varepsilon')$. $\mathsf{APPROX}(\mathsf{inp})$ holds, if and only if

$$|\tilde{\delta}_{i^*,0} - \delta_{i^*,0}| \leq \varepsilon'$$
$$|\tilde{\delta}_{i^*,1} - \delta_{i^*,1}| \leq \varepsilon'$$

(3) $\mathsf{GOOD}(\mathsf{inp})$ holds if and only if

$$\delta_{i^*,0} > \varepsilon'$$
$$\delta_{i^*,1} > \varepsilon'.$$

We will first elaborate on the events in more detail. The event $\mathsf{GAP}(\mathsf{inp})$ characterizes that for *the same* choice of inp, the hybrids $\mathcal{H}_{i^*}(\mathsf{inp})$ and $\mathcal{H}_{i^*-1}(\mathsf{inp})$ have distance at least $4\varepsilon'$. Notice that the extracted prefix $(\bar{x}_1, \ldots, \bar{x}_{i^*-1})$ is identical in both experiments $\mathcal{H}_{i^*}(\mathsf{inp})$ and $\mathcal{H}_{i^*-1}(\mathsf{inp})$. Consequently, $\mathsf{GAP}(\mathsf{inp})$ immediately implies that none of the $x^*_1, \ldots, x^*_{i^*-1}$ was set to $\perp$, as this would imply that the two experiments are identically distributed. The event $\mathsf{APPROX}(\mathsf{inp})$ ensures that the approximations $\tilde{\delta}_{i^*,0}$ and $\tilde{\delta}_{i^*,1}$ are sufficiently close to the true advantages.

Finally, the event $\mathsf{GOOD}(\mathsf{inp})$ ensures that inp is such that we will be able to mount a successful attack against indistinguishability sender security of LOT. Our first goal will be to show that the event $\mathsf{GOOD}(\mathsf{inp})$ holds with reasonably high probability over the choice of inp. Once this is established, we will construct an adversary $\mathcal{B}$ against the indistinguishability sender security of LOT. Observe that by Lemma 3 it holds that

$$\Pr_{\mathsf{inp}}[\neg\mathsf{APPROX}(\mathsf{inp})] \leq 2^{-\lambda}. \tag{1}$$

As

$$|\Pr_{\mathsf{inp}}[\mathcal{H}_{i^*}(\mathsf{inp}) = 1] - \Pr_{\mathsf{inp}}[\mathcal{H}_{i^*-1}(\mathsf{inp}) = 1]| = |\Pr[\mathcal{H}_{i^*} = 1] - \Pr[\mathcal{H}_{i^*-1} = 1]| \geq 8 \cdot \varepsilon'$$

it holds by the Markov inequality for advantages (Lemma 1) that

$$\Pr_{\mathsf{inp}}[\mathsf{GAP}(\mathsf{inp})] = \Pr_{\mathsf{inp}}[|\Pr[\mathcal{H}_{i^*}(\mathsf{inp}) = 1] - \Pr[\mathcal{H}_{i^*-1}(\mathsf{inp}) = 1]| > 4\varepsilon'] \geq 4\varepsilon'. \tag{2}$$

We will now show that if $\mathsf{GAP}(\mathsf{inp})$ holds, then it must either hold $\mathsf{GOOD}(\mathsf{inp})$ or not $\mathsf{APPROX}(\mathsf{inp})$. We will establish this by showing that $\neg\mathsf{GOOD}(\mathsf{inp})$ and $\mathsf{APPROX}(\mathsf{inp})$ imply $\neg\mathsf{GAP}(\mathsf{inp})$. Thus, fix $\mathsf{inp} = (crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \mathsf{r}_{i^*}^*)$ with $\neg\mathsf{GOOD}(\mathsf{inp})$ and $\mathsf{APPROX}(\mathsf{inp})$. From $\neg\mathsf{GOOD}(\mathsf{inp})$ it follows that there is a $\beta \in \{0, 1\}$ such that

$$|\Pr[\mathcal{Z}_2^{\mathcal{S}_i(crs,d,(x_1^*,\ldots,x_{i-1}^*,\beta),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1] - \Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1]| \leq \varepsilon'.$$

We will now show that under this condition $\mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \varepsilon')$ will be able to identify the correct $x_{i^*}^*$. Observe that since it holds that $\mathsf{APPROX}(\mathsf{inp})$, we get that

$$\tilde{\delta}_{i^*,\beta} \leq \delta_{i^*,\beta} + \varepsilon' \leq 2\varepsilon'.$$

Consequently, $\mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \varepsilon')$ will not output $\perp$. We will distinguish two cases.

Case 1: In this case it holds that

$$\delta_{i^*,1-\beta} \leq 4\varepsilon'.$$

It follows immediately that

$$\delta_{i^*,x_{i^*}^*} \leq 4\varepsilon',$$

regardless which $x_{i^*}^* \in \{0, 1\}$ is chosen.

Case 2: In this case it holds that

$$\delta_{i^*,1-\beta} > 4\varepsilon'.$$

Again since it holds that $\mathsf{APPROX}(\mathsf{inp})$, we get that

$$\tilde{\delta}_{i^*,1-\beta} \geq \delta_{i^*,1-\beta} - \varepsilon' \geq 3\varepsilon' > 2\varepsilon'.$$

Consequently, $\mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \varepsilon')$ will set $\bar{x}_{i^*} \leftarrow \beta$ and again we can conclude

$$\delta_{i^*,x_{i^*}^*} \leq 4\varepsilon',$$

Further observe that since $\mathsf{Extract}_{i^*}(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{i^*-1}^*), \varepsilon')$ will not output $\perp$, the output of $\mathcal{H}_{i^*}(\mathsf{inp})$ is distributed according to ${\mathcal{Z}'_2}^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i^*}^*),\cdot,\cdot,\cdot)}(\mathsf{st}, \mathsf{r}_2)$. We also know that $\mathcal{H}_{i^*-1}(\mathsf{inp})$ is distributed according to ${\mathcal{Z}'_2}^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(\mathsf{st}, \mathsf{r}_2)$. This implies that

$$|\Pr[\mathcal{H}_{i^*}(\mathsf{inp}) = 1] - \Pr[\mathcal{H}_{i^*-1}(\mathsf{inp}) = 1]| = |{\mathcal{Z}'_2}^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i^*}^*),\cdot,\cdot,\cdot)}(\mathsf{st}, \mathsf{r}_2) - {\mathcal{Z}'_2}^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(\mathsf{st}, \mathsf{r}_2)| \leq 4\varepsilon', \quad (3)$$

which in turn implies that $\neg\mathsf{GAP}(\mathsf{inp})$. Thus, we have established that

$$\mathsf{GAP}(\mathsf{inp}) \Rightarrow \mathsf{GOOD}(\mathsf{inp}) \text{ or } \neg\mathsf{APPROX}(\mathsf{inp}). \quad (4)$$

From (2), (4) and (1) we obtain that

$$\begin{aligned} 4\varepsilon' &\leq \Pr[\mathsf{GAP}(\mathsf{inp})] \\ &\leq \Pr[\mathsf{GOOD}(\mathsf{inp}) \text{ or } \neg\mathsf{APPROX}(\mathsf{inp})] \\ &\leq \Pr[\mathsf{GOOD}(\mathsf{inp})] + \Pr[\neg\mathsf{APPROX}(\mathsf{inp})] \\ &\leq \Pr[(\mathsf{GOOD}(\mathsf{inp})] + 2^{-\lambda}, \end{aligned}$$

where the third inequality follows by the union-bound. This implies that

$$\Pr_{\mathsf{inp}}[\mathsf{GOOD}(\mathsf{inp})] \geq 4\varepsilon' - 2^{-\lambda} > \varepsilon'.$$

We are now ready to construct an adversary $\mathcal{B}$ against the indistinguishability sender privacy of LOT. The adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is given as follows. In abuse of notation, we assume that $\mathcal{B}$ is stateful, i.e. the second stage $\mathcal{B}_2$ remembers all variables of the first stage $\mathcal{B}_1$. $\mathcal{B}$ essentially simulates $\mathcal{H}_{i^*-1}$, with the modification that calls to $\mathsf{Send}(crs, d, i^*, \cdot, \cdot)$ of $\mathcal{S}_{i^*-1}$ will be forwarded to $\mathcal{B}$'s oracle.

$\mathcal{B}_1(crs; \mathsf{r}_1)$:

- Compute $(\mathsf{st}, d) \leftarrow \mathcal{Z}'_1(crs, \mathsf{r}_1)$
- For $j = \{1, \ldots, i^* - 1\}$:
  - Compute $x_j^* \leftarrow \mathsf{Extract}_j(crs, \mathsf{st}, d, (x_1^*, \ldots, x_{j-1}^*), \varepsilon')$
- Output (d, i^*)

$\mathcal{B}_2^{\mathcal{O}(\cdot,\cdot)}(\mathsf{st}, \mathsf{r}_2)$:

- Compute and output $b^* \leftarrow {\mathcal{Z}_2}^{\mathcal{S}_{i^*}(\cdot,\cdot,\cdot)}$, where the oracle $\mathcal{S}_{i^*}(L, m_0, m_1)$ is implemented as follows:
  - If $L > i^*$ compute and output $\mathsf{Send}(crs, d, L, m_0, m_1)$
  - If $L < i^*$, set $m'_{x_L^*} = m_{x_L^*}$ and $m'_{1-x_L^*} = 0$, compute and output $\mathsf{Send}(crs, d, L, m'_0, m'_1)$.
  - If $L = i^*$ query and output $\mathcal{O}(m_0, m_1)$.

Now fix crs and r_1. We will distinguish 3 cases.

Case 1: In the first case, the oracle $\mathcal{O}(m_0, m_1)$ computes the function $\mathsf{Send}(crs, d, i^*, m_0, m_1)$. It follows by inspection that in this case the output of $\mathcal{B}$ is distributed according to ${\mathcal{Z}_2}^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st})$.

Case 2: In the second case, the oracle $\mathcal{O}(m_0, m_1)$ computes the function $\mathsf{Send}(crs, d, i^*, m_0, 0)$. It follows by inspection that in this case the output of $\mathcal{B}$ is distributed according to $\mathcal{Z}_2^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i-1}^*,0),\cdot,\cdot,\cdot)}(crs, \mathsf{st})$.

Case 3: In the third case, the oracle $\mathcal{O}(m_0, m_1)$ computes the function $\mathsf{Send}(crs, d, i^*, m_0, 1)$. It follows by inspection that in this case the output of $\mathcal{B}$ is distributed according to $\mathcal{Z}_2^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i-1}^*,1),\cdot,\cdot,\cdot)}(crs, \mathsf{st})$

We conclude that

$$\mathsf{Adv}_{\mathsf{LOT}}^{crs,\mathsf{r},0}(\mathcal{B}) = |\Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i-1}^*,0),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1] - \Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1]|$$
$$\mathsf{Adv}_{\mathsf{LOT}}^{crs,\mathsf{r},1}(\mathcal{B}) = |\Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*}(crs,d,(x_1^*,\ldots,x_{i-1}^*,1),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1] - \Pr[\mathcal{Z}_2^{\mathcal{S}_{i^*-1}(crs,d,(x_1^*,\ldots,x_{i^*-1}^*),\cdot,\cdot,\cdot)}(crs, \mathsf{st}) = 1]|.$$

This implies that

$$\Pr_{crs,\mathsf{r}}\left[\begin{array}{ll} & \mathsf{Adv}_{\mathsf{LOT}}^{crs,\mathsf{r},0}(\mathcal{B}; crs, \mathsf{r}) > \varepsilon' \\ \text{and} & \mathsf{Adv}_{\mathsf{LOT}}^{crs,\mathsf{r},1}(\mathcal{B}; crs, \mathsf{r}) > \varepsilon' \end{array}\right] = \Pr_{crs,\mathsf{r},\mathsf{r}^*}[\mathsf{GOOD}(crs, \mathsf{r}_{\mathcal{A}}, \mathsf{r}_{\mathsf{Extract}})] > \varepsilon',$$

which contradicts the indistinguishability sender privacy of LOT. □

□

6.3.3 Weak to Full Efficiency

The following construction shows that it suffices to consider LOT schemes with 2-to-1 compressing Hash and algorithms Send and $\mathsf{Receive}$ with polynomial running times in the size of the database. A similar statement already appeared in the work of Cho et al. [CDG+17], however their analysis crucially relies on the semi-honest settings and does not directly extend to malicious security. In the following we recall a simplified version of the transformation from [CDG+17]. We assume without loss of generality that the database has $2^d \cdot \lambda$ locations, which can be indexed with strings of the form $b_1\| \ldots \|b_{d-1}\|t$ where the bits b_i define the root-to-leaf path and the string t define the position of the leaf. For ease of exposition we overload the notation for the sender algorithm and we write

$$\mathsf{Send}(crs, d, K) = \mathsf{Send}(crs, d, 1, (K_0^1, K_1^1))\| \ldots \|\mathsf{Send}(crs, d, 2\lambda, (K_0^{2\lambda}, K_1^{2\lambda}))$$

where K is consists of 2λ pairs of λ-bit strings. The same shortcut is used for the receiver algorithm. The construction (given in Figure 10) consists of a chaining of garbled circuits that allows the receiver to traverse the tree downwards via the selected path. Since the sender only knows the root of the tree, the generation of subsequent Send algorithms is delayed by garbling the circuit $\mathcal{C}_{\mathsf{trav}}$ (Figure 9). Once the leaf is reached, the choice of the message can be constrained on the desired bit via the circuit $\mathcal{C}_{\mathsf{read}}$ (Figure 8).

Hardwired Values: (t, m_0, m_1).

Input: $(N_0^1\|N_1^1, \ldots, N_0^{d-1}\|N_{d-1}^1, N^d)$.

- Return $m_{N^d[t]}$

Figure 8: Circuit $\mathcal{C}_{\mathsf{leaf}}[m_0, m_1]$.

The following theorem shows that considering a LOT with 2-to-1 compression factor suffices.

Hardwired Values: (crs, b, K, r).

Input: (N_0, N_1).

- Compute $e \leftarrow \mathsf{Send}(crs, N_b, K; r)$.
- Return e.

Figure 9: Circuit $\mathcal{C}_{\mathsf{trav}}[crs, b, K, r]$.

$\bar{\mathsf{Setup}}(1^\lambda)$: Return $\mathsf{Setup}(1^\lambda)$.

$\bar{\mathsf{Hash}}(crs, D)$:

- Build a Merkle tree $\mathbf{D}$ of D using the function $\mathsf{Hash}(crs, \cdot)$.
- Return $(\rho, \mathbf{D})$, where ρ is the root of $\mathbf{D}$.

$\bar{\mathsf{Send}}(crs, d, L, m_0, m_1)$:

- Parse L as $b_1 \| \ldots \| b_{d-1} \| t$
- For $j = \{d, \ldots, 1\}$:
  - Sample $r_i \leftarrow\!\!\$\, \{0,1\}^\lambda$
  - If $j = 1$: Compute $e_0 \leftarrow \mathsf{Send}(crs, d, K^1)$
  - If $j = d$: Compute $(\tilde{\mathcal{C}}_d, K^d) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{leaf}}[t, m_0, m_1])$
  - Otherwise, compute $(\tilde{\mathcal{C}}_i, K^i) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{trav}}[crs, b_i, K^{i+1}, r_i])$
- Return $c = (e_0, \tilde{\mathcal{C}}_1, \ldots, \tilde{\mathcal{C}}_d)$.

$\bar{\mathsf{Receive}}^{\tilde{D}}(crs, c, L)$:

- Parse c as $(e_0, \tilde{\mathcal{C}}_1, \ldots, \tilde{\mathcal{C}}_d)$ and L as $b_1 \| \ldots \| b_{d-1} \| t$.
- Denote the end note of the path $b_1 b_2 \ldots b_i$ by $\mathbf{D}_{b_1 b_2 \ldots b_i}$.
- For all $i \in \{1, \ldots, d-1\}$, compute:
  - $M^i \leftarrow \mathsf{Receive}(crs, e_{i-1}, \mathbf{D}_{b_1 b_2 \ldots b_{i-1} 0} \| \mathbf{D}_{b_1 b_2 \ldots b_{i-1} 1})$.
  - $e_i \leftarrow \mathsf{Eval}(\mathcal{C}_i, M^i)$.
- Compute $M^d \leftarrow \mathsf{Receive}(crs, e_{d-1}, \mathbf{D}_{b_1 b_2 \ldots b_{d-1} 0} \| \mathbf{D}_{b_1 b_2 \ldots b_{d-1} 1})$.
- Return $m = \mathsf{Eval}(\tilde{\mathcal{C}}_{\mathsf{leaf}}, (M^i, \ldots, M^d))$.

Figure 10: From 2-to-1 to arbitrary compression.

Theorem 8 (Weak to Full Efficiency)**.** *Let* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *be a LOT with* 2*-to-*1 *compression. Then* $\bar{\mathsf{LOT}} = (\bar{\mathsf{Setup}}, \bar{\mathsf{Hash}}, \bar{\mathsf{Send}}, \bar{\mathsf{Receive}})$ *as defined in Figure 10 is LOT with arbitrary compression.*

Proof of Theorem 8. We start with the high-level overview. Our strategy is to associate every node in the Merkle tree with two hybrids. In this sequence of hybrids we start in the root node traverse the tree in such a way that it is ensure that once we reach a node ν, its parent has already been traversed, e.g. via breadth-first search or depth-first search from the root. This

will ensure that once we reach hybrid $\mathcal{H}_\nu$, the LOT message corresponding to the parent of ν has been extracted. In each node ν, we will first switch the garbled circuit to simulation, then extract the digest d_ν at this node.

There are 2^d-1 nodes. Let the sequence of nodes, in the order in which they are interated be $\nu_0, \ldots, \nu_{2^d-2}$. For shorthand, let $\ell = 2^d - 2$. For ease of notation we will leave out random coins in the notation of $\mathcal{Z}_1, \mathcal{Z}_2$ and $\mathsf{Ext}_{\mathcal{Z}}$. We will first provide the hybrids and derive the simulators and the extractor from the last hybrid. Now fix a context $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$.

- Hybrid $\mathcal{H}_0(\delta)$: This is the real experiment. Specifically,
  - $crs \leftarrow \mathsf{Setup}(1^\lambda)$
  - $(\mathsf{st}, d) \leftarrow \mathcal{Z}_1(crs)$
  - Compute and output $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}^{(0)}(\cdot)}(\mathsf{st})$.

  The oracle $\mathcal{O}^{(0)}(L, m_0, m_1)$ implements $\bar{\mathsf{Send}}(crs, d, L, m_0, m_1)$.
- Hybrid $\mathcal{H}_1(\delta)$: In this experiment we will extract d_{ν_1} as a LOT hash (i.e. 2-to-1). First we rephrase $\mathcal{H}_0$ as a context $(\mathcal{Z}_1^{(1)}, \mathcal{Z}_2^{(1)})$ for LOT. We will set $\mathcal{Z}_1^{(1)} = \mathcal{Z}_1$ but interpret $d = d_{\nu_0}$ as a hash value for LOT. Moreover, we interpret $\mathcal{Z}_2^{\mathcal{O}^{(0)}(\cdot)}$ as a single machine $\mathcal{Z}_2^{(1)}$, except for calls by $\mathcal{O}^{(0)}$ to $\mathsf{Send}(crs, d, \cdot)$, which will be implemented by oracle calls. Consequently, $\mathcal{Z}_2^{\mathcal{O}^{(0)}(\cdot)}$ and $\mathcal{Z}_2^{(1)^{\mathsf{Send}(crs,d,\cdot)}}$ compute identical functions. Context security of LOT yields an an extractor $\mathsf{Ext}_{\mathcal{Z}}^{(1)}$ and a simulator Sim, which we will use in the construction of $\mathcal{H}_1$. We can now define $\mathcal{H}_1$ as follows.
  - $crs \leftarrow \mathsf{Setup}(1^\lambda)$
  - $(\mathsf{st}, d) \leftarrow \mathcal{Z}_1(crs)$
  - Let $\nu_{1,l}$ be the left child node of the root node ν_1 and $\nu_{1,r}$ be the right child node of the root node ν_1.
  - $(d_{\nu_{1,l}} \| d_{\nu_{1,r}}, \mathsf{aux}_1) \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{(1)}(crs, \mathsf{st}, d_{\nu_1}, \delta)$
  - Compute and output $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}^{(1)}(\cdot)}(\mathsf{st})$.

  Furthermore, $\mathcal{O}^{(1)}$ is identical to $\mathcal{O}^{(0)}$, except that we replace calls to $\mathsf{Send}(crs, d, K)$ for any K by calls to $\mathsf{Sim}(crs, d, \mathsf{aux}_{root}, K_{d_{\nu_l} \| d_{\nu_r}})$

Consider the following hybrids for $i = \{1, \ldots, 2^d - 1\}$.

- Hybrid $\mathcal{H}_{2i}(\delta)$: $\mathcal{O}^{(2i)}$ is identical to $\mathcal{O}^{(2i-1)}$, except for the following changes. For a given index L let $\nu'_1, \ldots, \nu'_d$ be the root-to-leaf path for L, where ν'_1 is the root node and ν'_l a lead node.

  If $\nu_i = \nu'_d$ (i.e. ν_i is the leaf-node in the path for index L), we replace the instruction $(\tilde{\mathcal{C}}_d, K^d) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{leaf}}[t, m_0, m_1])$ by $(\tilde{\mathcal{C}}_d, K^d) \leftarrow \mathsf{GCSim}(1^\lambda, m_{D[t]})$.

  Otherwise, at node $\nu_i = \nu'_j$ for some j we replace the instruction $(\tilde{\mathcal{C}}_i, K^j) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{trav}}[crs, b_j, K^{j+1}, r_j])$ by $(\tilde{\mathcal{C}}_{\mathsf{trav}}, K^j_{\nu_{i,l} \| \nu_{i,r}}) \leftarrow \mathsf{GCSim}(1^\lambda, \mathsf{Send}(crs, N_{b_j}, K; r))$.
- Hybrid $\mathcal{H}_{2i+1}(\delta)$: In this experiment we will extract d_{ν_i} as a LOT hash (i.e. 2-to-1). First we rephrase $\mathcal{H}_{2i}$ as a context for LOT. That is, we define a context $\mathcal{Z}_1^{(i)}$ which outputs a state st' comprising of a state st and all LOT hash preimages extracted so far. It also outputs the hash d_{ν_i} at node ν_i.

Moreover, we interpret $\mathcal{Z}_2^{\mathcal{O}^{(2i)}(\cdot)}$ as a single machine $\mathcal{Z}_2^{(i)}$, except for calls by $\mathcal{O}^{(2i)}$ to $\mathsf{Send}(crs, d_{\nu_i}, \cdot)$, which will be implemented by oracle calls. Consequently, $\mathcal{Z}_2^{\mathcal{O}^{(2i)}(\cdot)}$ and $\mathcal{Z}_2^{(i)^{\mathsf{Send}(crs,d_{\nu_i},\cdot)}}$ compute identical functions. Context security of LOT yields an an extractor $\mathsf{Ext}_{\mathcal{Z}}^{(i)}$ and a simulator Sim, which we will use in the construction of $\mathcal{H}_{2i+1}$. We can now define $\mathcal{H}_{2i+1}$ as follows.

- $crs \leftarrow \mathsf{Setup}(1^\lambda)$
- $(\mathsf{st}, d) \leftarrow \mathcal{Z}_1(crs)$
- For $j = \{0, \ldots, i\}$:
  - Let $\nu_{j,l}$ be the left child of node ν_j and $\nu_{j,r}$ be the right child of node ν_j.
  - $(d_{\nu_{j,l}} \| d_{\nu_{j,r}}, \mathsf{aux}_j) \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{(j)}(crs, (\mathsf{st}, (d_{\nu_k}, d_{\nu_{k,l}}, d_{\nu_{k,r}}, \mathsf{aux}_k)_{k \le j}), d_{\nu_j}, \delta)$
- Compute and output $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}^{(2i+1)}(\cdot)}(\mathsf{st})$.

Furthermore, $\mathcal{O}^{(2i+1)}$ is identical to $\mathcal{O}^{(2i)}$, except that we replace calls to $\mathsf{Send}(crs, d_{\nu_i}, K)$ for any K by calls to $\mathsf{Sim}(crs, d, \mathsf{aux}_i, K_{d_{\nu_{i,l}} \| d_{\nu_{i,r}}})$

In the last hybrid $\mathcal{H}_{2\ell+1})(\delta)$, the oracle $\mathcal{O}^{2\ell+1}$ does not make any calls to Send, but only calls to Sim. We can therefore now define the extractor $\mathsf{Ext}_{\mathcal{Z}}$ and the simulator $\bar{\mathsf{Sim}}$.

$\mathsf{Ext}_{\mathcal{Z}}(crs, \mathsf{st}, d, \delta)$:

- For $j = \{0, \ldots, \ell\}$:
  - Let $\nu_{j,l}$ be the left child of node ν_j and $\nu_{j,r}$ be the right child of node ν_j.
  - $(d_{\nu_{j,l}} \| d_{\nu_{j,r}}, \mathsf{aux}_j) \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{(j)}(crs, (\mathsf{st}, (d_{\nu_k}, d_{\nu_{k,l}}, d_{\nu_{k,r}}, \mathsf{aux}_k)_{k \le j}), d_{\nu_j}, \delta)$
  - Set $D = (D_1, \ldots, D_{2^d})$
  - Set $\mathsf{aux} = (\mathsf{aux}_j)_j$
  - Output (D, aux)

$\bar{\mathsf{Sim}}(crs, d, \mathsf{aux}, (L, z))$:

- Let $\nu'_1, \ldots, \nu'_d$ be the root-to-leaf path corresponding to L
- Compute $(\tilde{\mathcal{C}}_d, K^d_{d_{\nu'_d}}) \leftarrow \mathsf{GCSim}(1^\lambda, m_{D[t]})$
- For $j = \{d-1, \ldots, 1\}$:
  - Let $\nu_{j,l}$ be the left child of node ν'_j and $\nu_{j,r}$ be the right child of node ν'_j.
  - Compute $(\tilde{\mathcal{C}}_{\mathsf{trav}}, K^j) \leftarrow \mathsf{GCSim}(1^\lambda, \mathsf{Sim}(crs, d, \mathsf{aux}_k, K^{j+1}_{d_{\nu_{j,l}} \| d_{\nu_{j,r}}}))$.
- Compute $e \leftarrow \mathsf{Sim}(crs, d, \mathsf{aux}_k, K^1_{d_{\nu_{0,l}} \| d_{\nu_{0,r}}})$

As there are $2\ell+1$ hybrid steps, we will set $\delta = \varepsilon/(2\ell+1)$. Now assume towards contradiction that there exists an inverse polynomial ε such that it holds for infinitely many 1^λ that

$$|\Pr[\mathcal{Z}(crs) = 1] - \Pr[\mathcal{EZ}(crs, \varepsilon/(2\ell+1))]| > \varepsilon.$$

As $\mathcal{H}_0(\varepsilon/(2\ell+1))$ is identical to $\mathcal{Z}(crs)$ and $\mathcal{H}_{2\ell+1}(\varepsilon/(2\ell+1))$ is identical to $\mathcal{EZ}(crs, \varepsilon/(2\ell+1))$, by the averaging principle there exists an $k \in \{1, \ldots, 2\ell+1\}$ such that

$$|\Pr[\mathcal{H}_k(\varepsilon/(2\ell+1)) = 1] - \Pr[\mathcal{H}_{k-1}(\varepsilon/(2\ell+1))]| > \varepsilon/(2\ell+1).$$

We will now show that this leads to a contradiction for every $k \in \{1, \ldots, 2\ell+1\}$.

Case k = 1: In this case we will show that this leads to a contradiction against the context-security of LOT. For hybrid $\mathcal{H}_1$ we constructed an LOT context $\mathcal{Z}_1 = (\mathcal{Z}_1^{(1)}), \mathcal{Z}_2^{(1)}$ which produces an output that is identically distributed to $\mathcal{H}_0$. $\mathcal{H}_1$ is constructed in a way such that its output is identically distributed to that of $\mathcal{EZ}$. Consequently, it holds that

$$\begin{aligned}|\Pr[\mathcal{Z}_1^{(1)} = 1] - \Pr[\mathcal{EZ}^{(1)}(\varepsilon/(2\ell+1))) = 1]| \\ = |\Pr[\mathcal{H}_1(\varepsilon/(2\ell+1)) = 1] - \Pr[\mathcal{H}_0(\varepsilon/(2\ell+1))]| \\ > \varepsilon/(2\ell+1),\end{aligned}$$

which contradicts the context-security of LOT.

Case k = 2i: For this case we get a contradiction via a routine application of simulation security of the garbling scheme. In more detail, in hybrid $\mathcal{H}_{2i-1}$ there is only one set of input labels at node ν_i which $\mathcal{Z}_2$ can obtain. Consequently, we can apply security of the garbling scheme in a hybrid fashion, i.e. once for each call to the oracle and the contradiction follows.

Case k = 2i + 1: In this case we will show that this leads to a contradiction against the context-security of LOT. We proceed as in the case $k = 1$. More specifically, to define hybrid $\mathcal{H}_{2i+1}$ we have rephrased $\mathcal{H}_{2i}$ as a context $\mathcal{Z}^{(i)} = (\mathcal{Z}_1^{(i)}, \mathcal{Z}_2^{(i)})$ for LOT. By the way we have constructed $\mathcal{H}_{2i+1}$ it holds that the output of $\mathcal{H}_{2i+1}$ is identically distributed to $\mathcal{EZ}^{(i)}$. Consequently, it holds that

$$\begin{aligned}|\Pr[\mathcal{Z}_1^{(i)} = 1] - \Pr[\mathcal{EZ}^{(i)}(\varepsilon/(2\ell+1))) = 1]| \\ = |\Pr[\mathcal{H}_{2i}(\varepsilon/(2\ell+1)) = 1] - \Pr[\mathcal{H}_{2i-1}(\varepsilon/(2\ell+1))]| \\ > \varepsilon/(2\ell+1),\end{aligned}$$

which contradicts the context-security of LOT.

This concludes the proof. □

7 Laconic Conditional Disclosure of Secrets

In the following we show how to construct a laconic CDS given a context-secure LOT. As a corollary, we obtain two concrete instantiations from CDH and LWE.

7.1 Subroutines

Before presenting the description of our scheme, we introduce some subroutines that are going to be useful for presentation purposes. Let $(\mathsf{PCPProve}, \mathsf{PCPVerify})$ be a (non-adaptive) PCP scheme. Since the scheme is non-adaptive it follows that fixing the random tape of the verifier algorithm uniquely determines a set of queries $(L_1, \ldots, L_q)$, where q is some constant that denotes the query complexity of the PCP. Let $\mathcal{O}[\bar{\pi}]$ be an oracle that has hardwired a string $\bar{\pi} \in \{0,1\}^q$. For notational convenience we denote by $\mathsf{PCPVerify}^{\mathcal{O}[\bar{\pi}]}(x; s)$ the execution of the verification algorithm (fixing the random tape to s) where the i-th query is answered with $\bar{\pi}[i]$, regardless of the location asked.

The first circuit that we define is $\mathcal{C}_{\mathsf{PCP}}$ (Figure 11) and has λ-many copies of the PCP verifier hardwired, each executed on a fresh random tape s_i. The circuit takes as input a set of $q \cdot \lambda$ bits, which are aggregated into λ-many q-bits strings $\bar{\pi}_i$, and it tests whether all functions $\mathsf{PCPVerify}^{\mathcal{O}[\bar{\pi}_i]}(x; s_i) = 1$. If this is the case, then it outputs some hardwired message

Hardwired Values: $(m, s_1, \ldots, s_\lambda)$.

Input: $(\pi_1, \ldots, \pi_{q\cdot\lambda})$.

- For all $i \in \{1, \ldots, \lambda\}$:
  - Set $\bar{\pi}_i = \pi_{q(i-1)} \| \ldots \| \pi_{q(i-1)+q}$.
  - Compute $b_i = \mathsf{PCPVerify}^{\mathcal{O}[\bar{\pi}_i]}(x; s_i)$.
- If $\prod_{i=1}^{\lambda} b_i = 1$, then return m.
- Else return $\perp$.

Figure 11: Circuit $\mathcal{C}_{\mathsf{PCP}}[m, s_1, \ldots, s_\lambda]$.

m, otherwise it returns some distinguished string $\perp$. Assume for the moment that the string $\bar{\pi}_i$ is identical to the collection of answers to the queried locations $(L_1, \ldots, L_q)$ of some fixed encoding π, i.e., $\bar{\pi}_i = \pi[L_1] \| \ldots \| \pi[L_q]$. Then the circuit $\mathcal{C}_{\mathsf{PCP}}$ consists of evaluating an amplified version of the PCP verifier.

In Figure 12 we define the circuit $\mathcal{C}_{\mathsf{LOT}}$, which takes as input a digest d and has hardwired a set of locations $(L_1, \ldots, L_{q\cdot\lambda})$ and a set of label pairs $(\ell_i^{(0)}, \ell_i^{(1)})$. The circuit executes several parallel instances of the LOT transferring the label $\ell_i^{(D[L_i])}$, where D is the pre-image of d. To see how the two circuits are connected, one should think of the labels hardwired in $\mathcal{C}_{\mathsf{LOT}}$ as the ones derived from the garbling of $\mathcal{C}_{\mathsf{PCP}}$. This will allow us to run the latter circuit in a consistent way on a very large database (looking ahead, the PCP encoding π) communicating only the bits that we are interested in.

Hardwired Values: $(crs, (L_1, \ell_1^{(0)}, \ell_1^{(1)}), \ldots, (L_{q\cdot\lambda}, \ell_{q\cdot\lambda}^{(0)}, \ell_{q\cdot\lambda}^{(1)}))$.

Input: d.

- For all $i \in \{1, \ldots, q \cdot \lambda\}$: Compute $c_i = \mathsf{Send}(crs, d, L_i, \ell_i^{(0)}, \ell_i^{(1)})$.
- Return $(c_1, \ldots, c_{q\cdot\lambda})$.

Figure 12: Circuit $\mathcal{C}_{\mathsf{LOT}}[crs, \{L_i, \ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^{q\cdot\lambda}]$.

7.2 Description

Our CDS construction $\mathsf{CDS} = (\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSetup}, \mathsf{CDSDecode})$ for an NP-complete langauge $\mathcal{L}$ is built from the following cryptographic objects:

- A maliciously secure string-LOT $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$.
- A UC-secure 2-round string-OT $(\mathsf{OTSetup}, \mathsf{OTReceive}, \mathsf{OTSend}, \mathsf{OTDecode})$.
- A garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.
- A witness-extractable PCP scheme $(\mathsf{PCPProve}, \mathsf{PCPVerify})$.

We give the full description of the laconic CDS scheme in Figure 13. The construction stems from a connection between LOT and PCPs: LOT allows one to hash a large string in such a

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

- Compute $crs_{\mathsf{LOT}} \leftarrow \mathsf{Setup}(1^\lambda)$ and $crs_{\mathsf{OT}} \leftarrow \mathsf{OTSetup}(1^\lambda)$.
- Return $crs = (crs_{\mathsf{OT}}, crs_{\mathsf{LOT}})$.

$\mathsf{CDSReceive}(crs, x, w)$:

- Parse crs as $(crs_{\mathsf{OT}}, crs_{\mathsf{LOT}})$.
- Compute $\pi \leftarrow \mathsf{PCPProve}(x, w)$.
- Compute $(d, \tilde{\pi}) \leftarrow \mathsf{Hash}(crs_{\mathsf{LOT}}, \pi)$.
- For all $i \in \{1, \ldots, |d|\}$: Compute $(\mathsf{ot}_1^{(i)}, r^{(i)}) \leftarrow \mathsf{OTReceive}(crs_{\mathsf{OT}}, d[i])$.
- Return $\mathsf{cds}_1 = (\mathsf{ot}_1^{(1)}, \ldots, \mathsf{ot}_1^{(|d|)})$ and $r = (r^{(1)}, \ldots, r^{(|d|)}, \tilde{\pi})$.

$\mathsf{CDSSend}(crs, x, m, \mathsf{cds}_1)$:

- Parse crs as $(crs_{\mathsf{OT}}, crs_{\mathsf{LOT}})$ and cds_1 as $(\mathsf{ot}_1^{(1)}, \ldots, \mathsf{ot}_1^{(|d|)})$.
- For all $i \in \{1, \ldots, \lambda\}$:
  - Sample a random string $s_i \leftarrow \{0,1\}^t$.
  - Let $(L_{q(i-1)}, \ldots, L_{q(i-1)+q})$ be the queries of the algorithm $\mathsf{PCPVerify}(x; s_i)$.
- Garble $(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \{\ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^{q\cdot\lambda}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{PCP}}[m, s_1, \ldots, s_\lambda])$.
- Garble $(\tilde{\mathcal{C}}_{\mathsf{LOT}}, \{k_i^{(0)}, k_i^{(1)}\}_{i=1}^{|d|}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{LOT}}[crs_{\mathsf{LOT}}, \{L_i, \ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^{q\cdot\lambda}])$.
- For all $i \in \{1, \ldots, |d|\}$: Compute $\mathsf{ot}_2^{(i)} \leftarrow \mathsf{OTSend}(crs_{\mathsf{OT}}, \mathsf{ot}_1^{(i)}, k_i^{(0)}, k_i^{(1)})$.
- Return $\mathsf{cds}_2 = (\tilde{\mathcal{C}}_{\mathsf{PCP}}, \tilde{\mathcal{C}}_{\mathsf{LOT}}, \mathsf{ot}_2^{(1)}, \ldots, \mathsf{ot}_2^{(|d|)}, L_1, \ldots, L_{q\cdot\lambda})$.

$\mathsf{CDSDecode}(crs, \mathsf{cds}_2, r)$:

- Parse cds_2 as $(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \tilde{\mathcal{C}}_{\mathsf{LOT}}, \mathsf{ot}_2^{(1)}, \ldots, \mathsf{ot}_2^{(|d|)}, L_1, \ldots, L_{q\cdot\lambda})$ and r as $(r^{(1)}, \ldots, r^{(|d|)}, \tilde{\pi})$.
- For all $i \in \{1, \ldots, |d|\}$: Compute $k_i \leftarrow \mathsf{OTDecode}(crs_{\mathsf{OT}}, \mathsf{ot}_2^{(i)}, r^{(i)})$.
- Compute $(c_1, \ldots, c_{q\cdot\lambda}) \leftarrow \mathsf{Eval}(\tilde{C}_{\mathsf{LOT}}, \{k_i\}_{i=1}^{|d|})$.
- For all $i \in \{1, \ldots, q \cdot \lambda\}$: Compute $\ell_i \leftarrow \mathsf{Receive}^{\tilde{\pi}}(crs_{\mathsf{LOT}}, c_i, L_i)$.
- Return $\mathsf{Eval}(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \{\ell_i\}_{i=1}^{q\cdot\lambda})$.

Figure 13: Laconic Conditional Disclosure of Secrets.

way that can be accessed at individual point without parsing the full string. On the other hand a PCP consists of an encoded string that can be queried in very few locations to check the validity of a statement. This suggest a very natural strategy to build a laconic CDS: The prover computes the PCP encoding of its witness and hashes it via a LOT scheme. The sender garbles a circuit that internally runs the PCP verifier and returns the secret message if the verifier accepts. The input of the circuit consists of the locations of the PCP encoding queried by the verifier. Computing the LOT Send algorithm on input each pair of input labels makes sure that the prover learns the output of the garbled circuit and nothing more. The probability that a false statement causes the PCP verifier to accept is then decreased arbitrarily using standard amplification techniques.

7.3 Efficiency

To see why the construction is laconic, observe that the first message consists of $|d|$-many parallel string-OT instances (for strings of λ bits) and the size of d is a fixed polynomial in the security parameter, regardless of the size of the database. The second message additionally contains $(q \cdot \lambda)$-many random locations $(L_1, \ldots, L_{q\cdot\lambda})$ and the two garbled circuits $\tilde{\mathcal{C}}_{\mathsf{PCP}}$ and $\tilde{\mathcal{C}}_{\mathsf{LOT}}$. Observe that the size of the circuit $\mathcal{C}_{\mathsf{LOT}}$ depends only on q and λ and q is a constant. On the other hand the size of $\mathcal{C}_{\mathsf{PCP}}$ is dominated by the λ parallel copies of the PCP verifier, whose size is proportional to the size of the statement $|x|$.

All algorithms are clearly PPT. Of particular interest is the computational complexity of the $\mathsf{CDSSend}$ algorithm, which we discuss in the following. The sender algorithm determines a set of locations $(L_1, \ldots L_{q\cdot})$ by running λ copies of the PCP verifier over the statement x. For all of our cases of interest, the size of the statement $|x|$ is going to be bounded by a fixed polynomial in λ. The remainder of the computation of $\mathsf{CDSSend}$ is bounded by some polynomial $poly(\lambda, q, |d|)$, where q is a constant and $|d|$ is also a fixed polynomial in λ. We can conclude that the runtime of $\mathsf{CDSSend}$ is that of a fixed $poly(\lambda)$.

Handling Large Statements. If the statement is not small, then the communication overhead of our laconic CDS can be further reduced with a standard trick: Augment the statement with the hash $H(x) = h$ and define the new language $\mathcal{L}'$ as

$$\mathcal{L}' = \left\{ h \in \{0,1\}^\lambda \;\middle|\; \exists\ (w, x) : \mathcal{R}(w, x) = 1 \wedge H(x) = h \right\}$$

where $\mathcal{R}$ is the original NP-relation, then run the same protocol on $\mathcal{L}'$. The key of the hash is then set to be part of the common reference string. Clearly the sender has to compute the hash at least once, but this can be delegated to an offline phase, prior to the protocol execution.

7.4 Analysis

We show that the scheme is correct.

Theorem 9 (Correctness)**.** *The construction* $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSetup}, \mathsf{CDSDecode})$ *as defined in Figure 13 is correct.*

Proof of Theorem 9. By the correctness of the OT we have that, for all $i \in \{1, \ldots, |d|\}$,

$$k_i^{(d[i])} = \mathsf{OTDecode}(crs_{\mathsf{OT}}, \mathsf{ot}_2^{(i)}, r^{(i)}) = k_i$$

and it follows that

$$\mathsf{Eval}(\tilde{C}_{\mathsf{LOT}}, \{k_i^{(d[i])}\}_{i=1}^{|d|}) = \mathcal{C}_{\mathsf{LOT}}(d) = \{\mathsf{Send}(crs_{\mathsf{LOT}}, d, L_i, \ell_i^{(0)}, \ell_i^{(1)})\}_{i=1}^{q\cdot\lambda} = (c_1, \ldots, c_{q\cdot\lambda})$$

by the correctness of the garbling scheme. For all $i \in \{1, \ldots, q \cdot \lambda\}$, we have

$$\ell_i^{(\pi[i])} = \mathsf{Receive}^{\tilde{\pi}}(crs_{\mathsf{LOT}}, c_i, L_i) = \ell_i$$

by the correctness of LOT. Finally, another invocation of the correctness of the garbling scheme yields

$$\mathsf{Eval}(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \{\ell_i^{(\pi[L_i])}\}_{i=1}^{q\cdot\lambda}) = \mathcal{C}_{\mathsf{PCP}}(\pi[L_1], \ldots, \pi[L_{q\cdot\lambda}]) = m$$

since for all randomness $s \in \{0,1\}^t$ and for all $x \in \mathcal{L}$, we have that

$$\mathsf{PCPVerify}^{\mathcal{O}[\pi[L_1]\|\ldots\|\pi[L_q]]}(x; s) = \mathsf{PCPVerify}^{\pi}(x; s) = 1.$$

where $\pi \leftarrow \mathsf{PCPProve}(x, w)$ and $(L_1, \ldots, L_q)$ is the set of queries of $\mathsf{PCPVerify}(x; s)$. □

We turn to proving that the CDS construction satisfies context security.

Theorem 10 (Context Security)**.** *Let* (Setup, Hash, Send, Receive) *be a context-secure string-LOT, let* (OTSetup, OTReceive, OTSend, OTDecode) *be a UC-secure string-OT, and let* (Garble, Eval) *be a secure garbling scheme. Then the construction* (CDSSetup, CDSReceive, CDSSetup, CDSDecode) *as defined in Figure 13 is context-secure.*

Proof of Theorem 10. Let $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$ be a context for the CDS protocol. We will start by defining a sequence of hybrids.

- Hybrid $\mathcal{H}_0$: This is the real context $\mathcal{Z}(1^\lambda)$. Specifically $\mathcal{H}_0$ computes
  - Compute $crs_{\mathsf{LOT}} \leftarrow \mathsf{Setup}(1^\lambda)$ and $crs_{\mathsf{OT}} \leftarrow \mathsf{OTSetup}(1^\lambda)$.
  - $crs \leftarrow (crs_{\mathsf{OT}}, crs_{\mathsf{LOT}})$.
  - $(\mathsf{st}, \mathsf{cds}_1) \leftarrow \mathcal{Z}_1(crs)$
  - $b^* \leftarrow \mathcal{Z}_2^{\mathsf{CDSSend}(crs,\cdot,\cdot,\mathsf{cds}_1)}(\mathsf{st})$
- Hybrid $\mathcal{H}_1$: This is identical to hybrid $\mathcal{H}_0$, except that we give $\mathcal{Z}_2$ access to an oracle $\mathcal{O}_1$, which makes the following modification to CDSSend. For all $i \in \{1, \ldots, |d|\}$, we simulate the second message of the (standard) OT protocol on input $(k_i^{(0)}, k_i^{(1)})$. Furthermore, we extract the OT-input d of the receiver.
- Hybrid $\mathcal{H}_2$: This is identical to hybrid $\mathcal{H}_1$, except that we give $\mathcal{Z}_2$ access to an oracle $\mathcal{O}_2$, which makes the following modification to $\mathcal{O}_1$. The garbled circuit $\tilde{\mathcal{C}}_{\mathsf{LOT}}$ is computed via the garbled-circuit simulator rather than using the garbling algorithm. That is, instead of computing
$$(\tilde{\mathcal{C}}_{\mathsf{LOT}}, \{k_i^{(0)}, k_i^{(1)}\}_{i=1}^{|d|}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{LOT}}[crs_{\mathsf{LOT}}, \{L_i, \ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^{q\cdot\lambda}])$$
we compute
$$(\tilde{\mathcal{C}}_{\mathsf{LOT}}, \{k_i\}_{i=1}^{|d|}) \leftarrow \mathsf{GCSim}(1^\lambda, |\mathcal{C}_{\mathsf{LOT}}|, (c_1, \ldots, c_{q\cdot\lambda})),$$
where the c_i are computed via $c_i \leftarrow \mathsf{Send}(crs_{\mathsf{LOT}}, d, L_i, \ell_i^{(0)}, \ell_i^{(1)})$.
- Hybrid $\mathcal{H}_3$: First rephrase $\mathcal{H}_2$ as a context for LOT. That is, we represent $\mathcal{H}_2$ as a context $\mathcal{Z}' = (\mathcal{Z}_1', \mathcal{Z}_2')$ where $\mathcal{Z}_1'$ outputs a state st' and a digest d and $\mathcal{Z}_2'$ has oracle access to $\mathsf{Send}(crs_{\mathsf{LOT}}, d, \cdot, \cdot, \cdot)$. Context security of LOT yields an extractor $\mathsf{Ext}'_{\mathcal{Z}}$ and a simulator $\mathsf{Sim}^{\mathsf{LOT}}$. We can now define $\mathcal{H}_3$ as follows.
  - Compute $crs_{\mathsf{LOT}} \leftarrow \mathsf{Setup}(1^\lambda)$ and $crs_{\mathsf{OT}} \leftarrow \mathsf{OTSetup}(1^\lambda)$.
  - $crs \leftarrow (crs_{\mathsf{OT}}, crs_{\mathsf{LOT}})$.
  - $(\mathsf{st}, \mathsf{cds}_1) \leftarrow \mathcal{Z}_1(crs)$
  - Extract the digest d from the receiver's OT messages in cds_1.
  - Compute $(w^*, \mathsf{aux} = D^*) \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{\mathsf{LOT}}(crs_{\mathsf{LOT}}, \mathsf{st}, d, \mathsf{r}^*, \delta)$
  - $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}_3(\cdot)}(\mathsf{st})$, where $\mathcal{O}_3(\cdot)$ is identical to $\mathcal{O}_2$, except that calls to $\mathsf{Send}(crs_{\mathsf{LOT}}, d, \cdot, \cdot, \cdot)$ are replaced by calls to $\mathsf{Sim}^{\mathsf{LOT}}(crs_{\mathsf{LOT}}, d, \mathsf{aux}, \cdot)$
- Hybrid $\mathcal{H}_4$: The same as hybrid $\mathcal{H}_4$, except that $\mathcal{Z}_2$ gets access to an oracle $\mathcal{O}_4$, which is identical to $\mathcal{O}_3$, except that instead of computing
$$(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \{\ell_i^{(0)}, \ell_i^{(1)}\}_{i=1}^{q\cdot\lambda}) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}_{\mathsf{PCP}}[m, s_1, \ldots, s_\lambda]),$$
it computes
$$(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \{\ell_i\}_{i=1}^{q\cdot\lambda}) \leftarrow \mathsf{GCSim}(1^\lambda, |\mathcal{C}_{\mathsf{PCP}}|, m^*).$$

We can now state the extractor $\mathsf{Ext}_{\mathcal{Z}}$ and the simulator Sim for CDS.

$\mathsf{Ext}_{\mathcal{Z}}(crs, \mathsf{st}, \mathsf{cds}_1, \mathsf{r}^*, \delta)$: Parse $\mathsf{cds}_1 = (\mathsf{ot}_1^{(1)}, \ldots, \mathsf{ot}_1^{(|d|)})$ and run the UC-simulator OTSim (on input the first OT messages) to extract the digest d. Parse $crs = (crs_{\mathsf{OT}}, crs_{\mathsf{LOT}})$ and locally execute $D^* \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{\mathsf{LOT}}(crs_{\mathsf{LOT}}, \mathsf{st}, d, \mathsf{r}^*, \delta)$. Compute $w^* \leftarrow \mathsf{PCPExt}(D^*)$ and return w^* and $\mathsf{aux} = D^*$.

We then define our simulator Sim in the following.

$\mathsf{Sim}(crs, \mathsf{cds}_1, D^*, m^* = \{m, \perp\})$: Sample λ-many uniform random tapes $(s_1, \ldots, s_\lambda)$ and define $(L_1, \ldots, L_{q\cdot\lambda})$ as the positions queried by the PCP verifier run on such random coins in parallel. Call the garbling simulator to compute $(\tilde{\mathcal{C}}_{\mathsf{PCP}}, \{\ell_i\}_{i=1}^{q\cdot\lambda}) \leftarrow \mathsf{GCSim}(1^\lambda, |\mathcal{C}_{\mathsf{PCP}}|, m^*)$. For all $i \in \{1, \ldots, q \cdot \lambda\}$ compute $c_i \leftarrow \mathsf{Sim}^{\mathsf{LOT}}(crs_{\mathsf{LOT}}, d, D^*, (L_i, \ell_i))$, then invoke once more the garbling simulator $(\tilde{\mathcal{C}}_{\mathsf{LOT}}, \{k_i\}_{i=1}^{|d|}) \leftarrow \mathsf{GCSim}(1^\lambda, |\mathcal{C}_{\mathsf{LOT}}|, (c_1, \ldots, c_{q\cdot\lambda}))$. Finally compute $(\mathsf{ot}_2^{(1)}, \ldots, \mathsf{ot}_2^{(|d|)})$ via the UC-simulator OTSim on input $(k_1, \ldots, k_{|d|})$. Return $\mathsf{cds}_2 = (\tilde{\mathcal{C}}_{\mathsf{PCP}}, \tilde{\mathcal{C}}_{\mathsf{LOT}}, \mathsf{ot}_2^{(1)}, \ldots, \mathsf{ot}_2^{(|d|)}, L_1, \ldots, L_{q\cdot\lambda})$.

Both algorithms are PPT and the runtime of $\mathsf{Ext}_{\mathcal{Z}}$ depends only on OTSim and $\mathsf{Ext}_{\mathcal{Z}}^{\mathsf{LOT}}$ and in particular is independent of $\mathcal{Z}$. We will choose $\delta(\varepsilon) = \varepsilon/4$. Assume towards contradiction that there exists a inverse-polynomial ε such that

$$|\Pr[\mathcal{Z}(1^\lambda) = 1] - \Pr[\mathcal{EZ}(1^\lambda, \varepsilon) = 1]| > \varepsilon.$$

As $\mathcal{H}_0$ is identically distributed to $\mathcal{Z}(1^\lambda)$ and $\mathcal{H}_4$ is identically distributed to $\mathcal{EZ}(1^\lambda, \varepsilon)$, by the averaging principle there must exist an $k \in \{1, \ldots, 4\}$ such that

$$|\Pr[\mathcal{H}_k = 1] - \Pr[\mathcal{H}_{k-1} = 1]| > \varepsilon/4.$$

We will now go through all possible cases of k.

(1) $k = 1$: In this case we obtain a contradiction against the sender-security of OT.

(2) $k = 2$: In this case we obtain a contradiction against the simulation security of the garbling scheme, since the garbled and the simulated circuits are functionally equivalent.

(3) $k = 3$: In this case we will obtain a contradiction against the context security of LOT. To see this, note that by construction of $\mathcal{Z}'$ it holds that $\mathcal{H}_2$ is identically distributed to $\mathcal{Z}'$ and by construction of $\mathcal{H}_3$ it holds that $\mathcal{H}_3$ is identically distributed to the experiment $\mathcal{EZ}'$. Thus we get a contradiction against the context security of LOT.

(4) $k = 4$: To analyze this case we split the distribution conditioned on the event that the extracted witness w^* is valid or not.

(a) $\mathcal{R}(w^*, x) = 1$: By definition of the ideal functionality, $m^* = m$ and therefore the two circuits are functionally equivalent by the perfect correctness of the PCP scheme. The indistinguishability follows from the security of the garbling scheme.

(b) $\mathcal{R}(w^*, x) \neq 1$: In this case the ideal functionality sets $m^* = \perp$. Thus the two circuits are functionally equivalent except with probability

$$\begin{aligned} p &= \Pr\left[\mathsf{PCPVerify}^{\mathcal{O}[D^*[L_1]\|\ldots\|D^*[L_q]]}(x; s_1) = 1 \wedge \cdots \wedge \mathsf{PCPVerify}^{\mathcal{O}[D^*[L_{q\cdot(\lambda-1)}]\|\ldots\|D^*[L_{q\cdot\lambda}]]}(x; s_\lambda) = 1\right] \\ &= \prod_{i=1}^{\lambda} \Pr\left[\mathsf{PCPVerify}^{\mathcal{O}[D^*[L_{q\cdot(i-1)}]\|\ldots\|D^*[L_{q\cdot(i-1)+q}]]}(x; s_i) = 1\right] \\ &= \prod_{i=1}^{\lambda} \Pr\left[\mathsf{PCPVerify}^{D^*}(x; s_i) = 1\right] \end{aligned}$$

where the probability is taken over the random choice of $(s_1, \ldots, s_\lambda)$. Since w^* is not a valid witness, by the witness-extractability of the PCP we have that $p \leq 1/3^\lambda$. Therefore the two circuits are functionally equivalent (they both return $\perp$) with all but negligible probability and indistinguishability is implied by the security of the garbling scheme.

□

Then we obtain the following corollary.

Corollary 1 (Message-Indistinguishability)**.** *Let* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ *be a context-secure string-LOT, let* $(\mathsf{OTSetup}, \mathsf{OTReceive}, \mathsf{OTSend}, \mathsf{OTDecode})$ *be a UC-secure string-OT, and let* $(\mathsf{Garble}, \mathsf{Eval})$ *be a secure garbling scheme. Then the construction* $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSetup}, \mathsf{CDSDecode})$ *as defined in Figure 13 is message-indistinguishable.*

Finally we show that the scheme satisfies the notion of receiver simulation. We consider without loss of generality a notion of security that is one-time secure, in the sense that security is guaranteed only as long as the sender's message is well-formed. This can be generically upgraded to the stronger notion of reusable receiver simulation by attaching (MDV-)NIZKs to the sender's message.

Theorem 11 (Receiver Simulation)**.** *Let* $(\mathsf{OTSetup}, \mathsf{OTReceive}, \mathsf{OTSend}, \mathsf{OTDecode})$ *be a UC-secure string-OT. Then the construction* $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSetup}, \mathsf{CDSDecode})$ *as defined in Figure 13 is receiver simulatable.*

Proof of Theorem 11. We define the following series of hybrid distributions.

- Hybrid $\mathcal{H}_0$: Is identical to the scheme described in Figure 13.
- Hybrid $\mathcal{H}_1$: The common reference string crs_{OT} is replaced with a simulated common reference string.
- Hybrid $\mathcal{H}_{1+i}$ (for $i \in \{1, \ldots, |d|\}$): The message $\mathsf{ot}_1^{(i)}$ is replaced with the output of the simulator OTSim.

The last hybrid is defined to be the output of the simulator CDSSim. Observe that the distribution induced by CDSSim hides the witness w in an information-theoretic sense. Therefore all is left to be shown is that the distribution induced by $\mathcal{H}_{1+|d|}$ is computationally close to the that induced by $\mathcal{H}_0$. Assume towards contradiction that there exists some PPT algorithm $\mathcal{A}$ such that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{H}_0}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{H}_{1+|d|}}\right]\right| \geq \frac{1}{poly(\lambda)}.$$

By a trivial reduction against the UC-security of the common reference string functionality we have that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{H}_0}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{H}_1}\right]\right| = negl(\lambda).$$

It follows that there must exist an $i^* \in \{1, \ldots, d\}$ such that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{H}_{i^*}}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{H}_{i^*+1}}\right]\right| \geq \frac{1}{|d| \cdot poly(\lambda)}.$$

We show that this is not the case with a reduction to the UC-security of the string-OT scheme: The reduction is defined to be identical to $\mathcal{H}_{i^*}$ up to the point where $\mathsf{ot}_1^{(i^*)}$ is computed. Then the reduction sends the bit d_{i^*} to the challenger and receives some ot_1^*, which the reduction sets

to $\mathsf{ot}_1^{(i^*)} = \mathsf{ot}_1^*$. The remainder of the execution is identical to that of $\mathcal{H}_{i^*+1}$. The reduction is clearly PPT since it runs only efficient subroutines. Observe that when ot_1^* is the output of $\mathsf{OTReceive}$, then the distribution induced by the reduction is identical to that of $\mathcal{H}_{i^*}$. On the other hand, when ot_1^* is simulated then the reduction perfectly reproduces $\mathcal{H}_{i^*+1}$. It follows that any difference in the output of $\mathcal{A}$ can be used to break the UC-security of the string-OT scheme. This is a contradiction and concludes our proof. □

8 UC-Secure Bob-Optimized Two-Party Computation

In the following we present a UC-secure Bob-optimized two-round 2PC for P/poly. Our construction $\mathsf{2PC} = (\mathsf{2PCSetup}, \mathsf{2PCReceive}, \mathsf{2PCSend}, \mathsf{2PCDecode})$ assumes the existence of the following cryptographic primitives:

- A pseudorandom generator PRG.
- A semi-malicious LFE $(\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$.
- A perfectly correct public-key encryption scheme $(\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec})$.

Then let $\mathcal{L}_1$ be the following language

$$\mathcal{L}_1 = \left\{ (crs_{\mathsf{LFE}}, pk, d_{\mathsf{LFE}}, c_{\mathsf{PKE}}) \;\middle|\; \exists\ (x,s): \begin{array}{ll} & d_{\mathsf{LFE}} = \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_x; \mathsf{PRG}(s)) \\ \wedge & c_{\mathsf{PKE}} = \mathsf{PKEEnc}(pk, x \| s) \end{array} \right\}$$

where $\mathcal{C}_x$ is defined as the circuit with x hardwired that computes $f(x,\cdot)$. Our protocol also assumes the existence of:

- A laconic CDS scheme $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSend}, \mathsf{CDSDecode})$ for $\mathcal{L}_1$.

Furthermore, let $\mathcal{L}_2$ be defined as follows

$$\mathcal{L}_2 = \left\{ \left(\begin{array}{l} crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, \tilde{pk}, \\ d_{\mathsf{LFE}}, \mathsf{stmt}, \mathsf{cds}_1, \tilde{c}_{\mathsf{PKE}} \end{array} \right) \;\middle|\; \exists\ (y, c_{\mathsf{LFE}}): \begin{array}{ll} & c_{\mathsf{LFE}} = \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y) \\ \wedge & \mathsf{cds}_2 = \mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, c_{\mathsf{LFE}}, \mathsf{cds}_1) \\ \wedge & \tilde{c}_{\mathsf{PKE}} = \mathsf{PKEEnc}(\tilde{pk}, y) \end{array} \right\}$$

then the last building block for our protocol consists of:

- A NIZK[1] scheme $(\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ for $\mathcal{L}_2$.

The protocol is described in Figure 14. It is instructive to observe that if we were to settle for semi-honest security, then a standard LFE scheme would be sufficient. The high level idea of the construction is to compile a semi-honest LFE into a UC-secure one, without affecting the communication or the computation complexity of the protocol. Our newly developed laconic CDS serves exactly this purpose: The receiver message consists of a compressed circuit d_{LFE} plus the first message of a laconic CDS that certifies that the hash of circuit is well-formed. The sender then encrypts its input y via $c_{\mathsf{LFE}} \leftarrow \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y)$ and conditions c_{LFE} on the fact that hash is correctly computed. That is, the receiver can decode c_{LFE} from the laconic CDS if and only if it behaved honestly in the first round. Otherwise c_{LFE} (and therefore the output of the computation) is computationally hidden. The important point here is that the laconic CDS does not inflate the communicate between the parties nor the computation of P_2, although the corresponding witness is very large (proportional to the circuit size).

[1] We use NIZK instead of MDV-NIZK in favor of a simpler presentation, however the result easily generalizes.

In addition to this we also have to ensure that the UC simulator can extract without rewinding, which is done by including the inputs in standard extractable commitment (i.e., a perfectly correct public-key encryption scheme). In terms of communication we pay the price of being proportional in the size of the inputs (but not of the circuit). This seems unavoidable if one insists with UC-security. Finally a NIZK proof is attached to the message of the sender to prevent leakage from selective failure attacks.

$\mathsf{2PCSetup}(1^\lambda)$:

- Sample
  - $crs_{\mathsf{LFE}} \leftarrow \mathsf{LFESetup}(1^\lambda)$,
  - $crs_{\mathsf{CDS}} \leftarrow \mathsf{CDSSetup}(1^\lambda)$, and
  - $crs_{\mathsf{NIZK}} \leftarrow \mathsf{NIZKSetup}(1^\lambda)$.
- Sample two key pairs $(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda)$ and $(\tilde{sk}, \tilde{pk}) \leftarrow \mathsf{PKEKGen}(1^\lambda)$.
- Return $crs = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk, \tilde{pk})$.

$\mathsf{2PCReceive}(crs, x)$:

- Parse crs as $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk, \tilde{pk})$.
- Sample a uniform $s \leftarrow\!\!\$\ \{0,1\}^\lambda$.
- Compute
  - $(d_{\mathsf{LFE}}, r_{\mathsf{LFE}}) \leftarrow \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_x; \mathsf{PRG}(s))$
  - $c_{\mathsf{PKE}} \leftarrow \mathsf{PKEEnc}(pk, x\|s)$, and
  - $(\mathsf{cds}_1, r_{\mathsf{CDS}}) \leftarrow \mathsf{CDSReceive}(crs_{\mathsf{CDS}}, \mathsf{stmt}, (x, s))$.
    // where $\mathsf{stmt} = (crs_{\mathsf{LFE}}, pk, d_{\mathsf{LFE}}, c_{\mathsf{PKE}})$
- Return $m_1 = (d_{\mathsf{LFE}}, c_{\mathsf{PKE}}, \mathsf{cds}_1)$ and $r = (r_{\mathsf{LFE}}, r_{\mathsf{CDS}})$.

$\mathsf{2PCSend}(crs, m_1, y)$:

- Parse crs as $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk, \tilde{pk})$ and m_1 as $(d_{\mathsf{LFE}}, c_{\mathsf{PKE}}, \mathsf{cds}_1)$.
- Compute
  - $c_{\mathsf{LFE}} \leftarrow \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y)$,
  - $\tilde{c}_{\mathsf{PKE}} \leftarrow \mathsf{PKEEnc}(\tilde{pk}, y)$,
  - $\mathsf{cds}_2 \leftarrow \mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, c_{\mathsf{LFE}}, \mathsf{cds}_1)$, and
  - $\pi \leftarrow \mathsf{NIZKProve}(crs_{\mathsf{NIZK}}, \tilde{\mathsf{stmt}}, (y, c_{\mathsf{LFE}}))$.
    // where $\tilde{\mathsf{stmt}} = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, pk, d_{\mathsf{LFE}}, \mathsf{stmt}, \mathsf{cds}_1, \tilde{c}_{\mathsf{PKE}})$
- Return $m_2 = (\mathsf{cds}_2, \tilde{c}_{\mathsf{PKE}}, \pi)$.

$\mathsf{2PCDecode}(crs, m_2, r)$:

- Parse crs as $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk, \tilde{pk})$, m_2 as $(\mathsf{cds}_2, \tilde{c}_{\mathsf{PKE}}, \pi)$, r as $(r_{\mathsf{LFE}}, r_{\mathsf{CDS}})$.
- If $1 \neq \mathsf{NIZKVerify}(crs_{\mathsf{NIZK}}, \tilde{\mathsf{stmt}}, \pi)$, then return $\perp$.
- Else return $z = \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{CDSDecode}(crs_{\mathsf{CDS}}, \mathsf{cds}_2, r_{\mathsf{CDS}}), r_{\mathsf{LFE}})$.

Figure 14: UC-Secure Bob-Optimized 2PC protocol.

8.1 Efficiency

It is easy to see that all algorithms are PPT. We want to show a sharper bound on the running time of the algorithm $\mathsf{2PCSend}$, also known as Bob. First observe that the runtime of the subroutines $\mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}})$ and $\mathsf{PKEEnc}(\tilde{pk}, y)$ is independent of the size of the circuit $\mathcal{C}_x$, except for its depth (by definition of LFE). Note that the size of the statement for the laconic CDS protocol is that of a fixed polynomial in λ ad therefore, as discussed in 7.3, the runtime of $\mathsf{CDSSend}$ is also bounded by a fixed $poly(\lambda)$. It follows that our protocol is Bob-optimized in the following sense: The runtime of P_2 depends only on λ and on the depth of f. If we ignore the former term (which is inherited by the underlying LFE), then this is asymptotically optimal. We now analyze the communication complexity of the protocol. Observe that the first message m_1 consists of the following variables.

- d_{LFE} : The compressed version of the circuit $\mathcal{C}_x$ with the input x hardwired.
- c_{PKE} : The encryption of the input x and of the uniform seed s.
- cds_1 : The first message of a laconic CDS scheme.

Since the LFE and the CDS scheme are laconic, the size of cds_1 and d_{LFE} is a fixed polynomial in λ. Thus the size of m_1 depends only on the size of the input x and on the security parameter. We now analyze the content of the second message m_2.

- cds_2 : The second message of the laconic CDS scheme.
- $\tilde{c}_{\mathsf{PKE}}$: The encryption of the input y.
- π : The NIZK proof that certifies the honest generation of the other variables.

The first two objects are clearly independent of the circuit size, whereas the size of π depends on the circuit depth since the circuit that computes the corresponding relation contains LFEEnc as a subroutine. We can conclude that the size of m_2 is bounded by a polynomial in the size of the inputs (x, y) and of the output z, in the depth of the circuit $\mathcal{C}_x$, and in the security parameter.

8.2 Analysis

We begin by analyzing the correctness of the scheme.

Theorem 12 (Correctness)**.** *The construction* $(\mathsf{2PCSetup}, \mathsf{2PCReceive}, \mathsf{2PCSend}, \mathsf{2PCDecode})$ *as defined in Figure 14 is correct.*

Proof of Theorem 12. Observe that, since all variables are honestly computed it holds that $\mathsf{stmt} \in \mathcal{L}_1$ and $\tilde{\mathsf{stmt}} \in \mathcal{L}_2$ and therefore

$$\begin{aligned} z &= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{CDSDecode}(crs_{\mathsf{CDS}}, \mathsf{cds}_2, r_{\mathsf{CDS}}), r_{\mathsf{LFE}}) \\ &= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, c_{\mathsf{LFE}}, r_{\mathsf{LFE}}) \\ &= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y), r_{\mathsf{LFE}}) \\ &= \mathcal{C}_x(y) \\ &= f(x, y) \end{aligned}$$

since $(d_{\mathsf{LFE}}, r_{\mathsf{LFE}}) = \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_x)$. □

We now show that the protocol is secure.

Theorem 13 (Security)**.** *Let* PRG *be a pseudorandom generator, let* $(\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *be a semi-malicious secure and function-hiding LFE, let* $(\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec})$ *be a semantically secure and perfectly correct public-key encryption scheme, let* $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSend}, \mathsf{CDSDecode})$ *by a message-indistinguishable and receiver simulatable laconic CDS, and let* $(\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ *be a sound NIZK. Then the construction* $(\mathsf{2PCSetup}, \mathsf{2PCReceive}, \mathsf{2PCSend}, \mathsf{2PCDecode})$ *as defined in Figure 14 UC-realizes* $\mathcal{F}_{\mathsf{2PC}}$*.*

Proof of Theorem 13. We show that the protocol is UC-secure by analyzing the case of each corrupted party separately.

(1) Corrupted P_1 : Fix a real-world adversary $\mathcal{A}$. The simulator $\mathsf{2PCSim}$ works as follows.

- $\mathsf{2PCSim}$: The simulator samples two honestly generated strings $crs_{\mathsf{LFE}} \leftarrow \mathsf{LFESetup}(1^\lambda)$ and $crs_{\mathsf{CDS}} \leftarrow \mathsf{CDSSetup}(1^\lambda)$, then it samples a simulated string $crs_{\mathsf{NIZK}} \leftarrow \mathsf{NIZKSetup}(1^\lambda)$ and two fresh key pairs $(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda)$ and $(\tilde{sk}, \tilde{pk}) \leftarrow \mathsf{PKEKGen}(1^\lambda)$. The common reference string crs is set to $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk, \tilde{pk})$. When $\mathcal{A}$ replies with a message $(d_{\mathsf{LFE}}, c_{\mathsf{PKE}}, \mathsf{cds}_1)$ the simulator decrypts $\tilde{x} \| \tilde{s} \leftarrow \mathsf{PKEDec}(sk, c_{\mathsf{PKE}})$ and checks whether

$$d_{\mathsf{LFE}} = \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_{\tilde{x}}; \mathsf{PRG}(\tilde{s})). \tag{5}$$

  If this is not the case, the simulator computes $\mathsf{cds}_2 \leftarrow \mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, 0^{|c_{\mathsf{LFE}}|}, \mathsf{cds}_1)$ and returns to $\mathcal{A}$ the tuple $(\mathsf{cds}_2, \mathsf{PKEEnc}(\tilde{pk}, 0^{|y|}), \pi)$, where π is a simulated proof. On the other hand, if the equality holds, then the simulator sends $\tilde{x}$ to the ideal functionality and receives z in response. Then it computes $c_{\mathsf{LFE}} \leftarrow \mathsf{LFESim}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, \mathcal{C}_{\tilde{x}}, z)$ are returns to $\mathcal{A}$ the tuple $(\mathsf{cds}_2, \mathsf{PKEEnc}(\tilde{pk}, 0^{|y|}), \pi)$, where cds_2 is honestly computed and π is a simulated proof.

The simulator is clearly PPT. We show that it is also indistinguishable from the real world execution by defining a series of hybrid experiments where we gradually change the simulation until we recover the original protocol. We also show that each modification is indistinguishable to the eyes of the adversary.

- Hybrid $\mathcal{H}_0$: Identical to the simulator $\mathsf{2PCSim}$.
- Hybrid $\mathcal{H}_1$: If Equation (1) holds, then the simulator computes the ciphertext $c_{\mathsf{LFE}} \leftarrow \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y)$, where y is the input of P_2.

Note that $z = f(\tilde{x}, y) = \mathcal{C}_{\tilde{x}}(y)$, thus the indistinguishability follows from the semi-malicious security of the LFE scheme. Recall that here the adversary is allowed to choose the random coins for the compression function, which the reduction sets to $\mathsf{PRG}(\tilde{s})$.

- Hybrid $\mathcal{H}_2$: If Equation (1) does not hold, then the simulator computes cds_2 as $\mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, c_{\mathsf{LFE}}, \mathsf{cds}_1)$, where c_{LFE} is computed as in $\mathcal{H}_1$.

Since Equation (1) does not hold and the public-key encryption scheme is perfectly correct, then it must be the case that $\mathsf{stmt} \notin \mathcal{L}_1$. Thus, this modification is indistinguishable by the message-indistinguishability of the CDS scheme.

- Hybrid $\mathcal{H}_3$: The simulator no longer decrypts c_{PKE}, does not check the validity of Equation (1) and it behaves as if Equation (1) holds.

This change is only syntactical since the behavior of the simulator in $\mathcal{H}_2$ is identical regardless on whether Equation (1) holds or not.

- Hybrid $\mathcal{H}_4$: The simulator computes $\tilde{c}_{\mathsf{PKE}}$ as $\mathsf{PKEEnc}(\tilde{pk}, y)$.

This change is indistinguishable by the semantic security of the public-key encryption scheme. Note that the decryption key $\tilde{sk}$ is never used in $\mathcal{H}_3$.

- Hybrid $\mathcal{H}_5$: The common reference string crs_{NIZK} is sampled from the honest domain and the proof π is honestly computed using the correct witness (y, c_{LFE}).

This change is indistinguishable by the zero-knowledge property of the NIZK scheme. Finally, observe that the distribution induced by $\mathcal{H}_5$ is identical to that of the real-world protocol. This concludes our proof for the case of a corrupted P_1.

(2) Corrupted P_2 : Fix a real-world adversary $\mathcal{A}$. We define a simulator in the following.

- $\mathsf{2PCSim}$: The simulator samples a simulated common reference string $(crs_{\mathsf{CDS}}, \mathsf{td}) \leftarrow \mathsf{CDSSim}_1(1^\lambda)$ and sets $crs = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk, \tilde{pk})$ where all other variables are honestly generated. It then gives crs as an input to $\mathcal{A}$. When P_1 engages in a protocol run in the ideal world, the simulator computes $d_{\mathsf{LFE}} \leftarrow \mathsf{LFESim}_{\mathsf{FH}}(crs, \mathfrak{C})$ and $\mathsf{cds}_1 \leftarrow \mathsf{CDSSim}_2(crs_{\mathsf{CDS}}, \mathsf{td}, \mathsf{stmt})$. Then it sends $(d_{\mathsf{LFE}}, \mathsf{cds}_1, \mathsf{PKEEnc}(pk, 0^{|x|+|s|}))$ to $\mathcal{A}$. When $\mathcal{A}$ returns a tuple $(\mathsf{cds}_2, \tilde{c}_{\mathsf{PKE}}, \pi)$, the simulator checks that π correctly verifies. If this is the case, the simulator computes $\tilde{y} \leftarrow \mathsf{PKEDec}(\tilde{sk}, \tilde{c}_{\mathsf{PKE}})$ and returns $\tilde{y}$ to the ideal functionality. Otherwise it returns nothing.

It is easy to see that the algorithm is PPT. We now argue that the inputs of $\mathcal{A}$ generated by the simulator are computationally indistinguishable from that of the real world protocol.

- Hybrid $\mathcal{H}_0$: Identical to the simulator $\mathsf{2PCSim}$.
- Hybrid $\mathcal{H}_1$: The simulator computes $d_{\mathsf{LFE}} \leftarrow \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_x; \tilde{s})$, where x is the input of P_1 and $\tilde{s}$ is a uniform random tape of the appropriate length.

The two hybrids are computationally close by the function-hiding of the LFE scheme.

- Hybrid $\mathcal{H}_2$: The simulator computes $(d_{\mathsf{LFE}}, r_{\mathsf{LFE}}) \leftarrow \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_x; \mathsf{PRG}(s))$, where s is a uniform λ-bit string.

Indistinguishability follows from a simple invocation of the pseudorandomness of PRG.

- Hybrid $\mathcal{H}_3$: Here c_{PKE} is computed as $\mathsf{PKEEnc}(pk, x\|s)$.

Indistinguishability follows from a simple reduction to the semantic security of the public-key encryption scheme.

- Hybrid $\mathcal{H}_4$: The variable crs_{CDS} is sampled honestly and the variable cds_1 is computed via $\mathsf{CDSReceive}(crs_{\mathsf{CDS}}, \mathsf{stmt}, (x, s))$

Note that at this point $\mathsf{stmt} \in \mathcal{L}_1$ and the simulator has a valid witness for stmt. Thus indistinguishability follows from a reduction to the receiver simulatability of the CDS.

- Hybrid $\mathcal{H}_5$: The simulator no longer decrypts $\tilde{c}_{\mathsf{PKE}}$ to extract $\tilde{y}$ but sets the output of P_1 to z computed as in the real world protocol.

By the soundness of the NIZK scheme we have that

$$\begin{aligned}
z &= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{CDSDecode}(crs_{\mathsf{CDS}}, \mathsf{cds}_2, r_{\mathsf{CDS}}), r_{\mathsf{LFE}}) \\
&= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{CDSDecode}(crs_{\mathsf{CDS}}, \mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, c_{\mathsf{LFE}}, \mathsf{cds}_1), r_{\mathsf{CDS}}), r_{\mathsf{LFE}}) \\
&= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, c_{\mathsf{LFE}}, r_{\mathsf{LFE}}) \\
&= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, \mathsf{PKEDec}(sk, \tilde{c}_{\mathsf{PKE}})), r_{\mathsf{LFE}}) \\
&= \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}_x), \mathsf{PKEDec}(sk, \tilde{c}_{\mathsf{PKE}})), r_{\mathsf{LFE}}) \\
&= \mathcal{C}_x(\mathsf{PKEDec}(sk, \tilde{c}_{\mathsf{PKE}})) \\
&= \mathcal{C}_x(\tilde{y}) = f(x, \tilde{y})
\end{aligned}$$

except with negligible probability, since the encryption scheme is perfectly correct. Thus with the same probability, the output of P_1 is identical in both hybrids. Note that the distribution induced by $\mathcal{H}_5$ is identical to that of the real world protocol.

This concludes our proof. □

9 Malicious Laconic Function Evaluation

Here we lift LFE in the malicious settings without any additional assumption. We construct a protocol where the communication complexity is independent of the size of the circuit (suppressing a factor that depends on the depth of the circuit representation of $\mathcal{C}$) and is Bob-optimized. Since straight-line extraction (and therefore UC-security) is information theoretically impossible in these settings, we settle for the weaker security notion of context security. We stress that such a definition implies the standard indistinguishability-based security. Our scheme is constructed from the following ingredients:

- A semi-malicious LFE ($\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec}$).
- A perfectly correct public-key encryption scheme ($\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec}$).
- A laconic CDS scheme ($\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSend}, \mathsf{CDSDecode}$) for $\mathcal{L}_1$.
- A NIZK scheme ($\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify}$) for $\mathcal{L}_2$.

Where $\mathcal{L}_1$ is defined as

$$\mathcal{L}_1 = \left\{ (crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}) \;\middle|\; \exists\ (\mathcal{C}, s) : \quad d_{\mathsf{LFE}} = \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}; s) \right\}$$

and $\mathcal{L}_2$ is defined as follows

$$\mathcal{L}_2 = \left\{ \left(\begin{array}{l} crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, pk, \\ d_{\mathsf{LFE}}, \mathsf{stmt}, \mathsf{cds}_1, c_{\mathsf{PKE}} \end{array} \right) \;\middle|\; \exists\ (y, c_{\mathsf{LFE}}) : \begin{array}{ll} & c_{\mathsf{LFE}} = \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y) \\ \wedge & \mathsf{cds}_2 = \mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, c_{\mathsf{LFE}}, \mathsf{cds}_1) \\ \wedge & c_{\mathsf{PKE}} = \mathsf{PKEEnc}(pk, y) \end{array} \right\}.$$

The construction is given in Figure 15. The scheme is a minor modification of what already shown in Section 8: The only difference is that the receiver no longer commits to its input (which in this cases consists of the whole circuit $\mathcal{C}$) via the public-key encryption scheme. This improves the communication complexity but results in a weaker notion of security. Looking ahead, our proof will crucially rely on the context-security of the laconic CDS to extract $\mathcal{C}$.

$\bar{\mathsf{LFESetup}}(1^\lambda)$:

- Sample
  - $crs_{\mathsf{LFE}} \leftarrow \mathsf{LFESetup}(1^\lambda)$,
  - $crs_{\mathsf{CDS}} \leftarrow \mathsf{CDSSetup}(1^\lambda)$, and
  - $crs_{\mathsf{NIZK}} \leftarrow \mathsf{NIZKSetup}(1^\lambda)$.
- Sample a key pair $(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda)$.
- Return $crs = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk)$.

$\bar{\mathsf{LFECompress}}(crs, \mathcal{C})$:

- Parse crs as $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk)$.
- Sample a uniform random tape s.
- Compute
  - $(d_{\mathsf{LFE}}, r_{\mathsf{LFE}}) \leftarrow \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}; s)$ and
  - $(\mathsf{cds}_1, r_{\mathsf{CDS}}) \leftarrow \mathsf{CDSReceive}(crs_{\mathsf{CDS}}, \mathsf{stmt}, (x, s))$.
    // where $\mathsf{stmt} = (crs_{\mathsf{LFE}}, d_{\mathsf{LFE}})$
- Return $d = (d_{\mathsf{LFE}}, \mathsf{cds}_1)$ and $r = (r_{\mathsf{LFE}}, r_{\mathsf{CDS}})$.

$\bar{\mathsf{LFEEnc}}(crs, d, y)$:

- Parse crs as $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk)$ and d as $(d_{\mathsf{LFE}}, \mathsf{cds}_1)$.
- Compute
  - $c_{\mathsf{LFE}} \leftarrow \mathsf{LFEEnc}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, y)$,
  - $c_{\mathsf{PKE}} \leftarrow \mathsf{PKEEnc}(pk, y)$,
  - $\mathsf{cds}_2 \leftarrow \mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, c_{\mathsf{LFE}}, \mathsf{cds}_1)$, and
  - $\pi \leftarrow \mathsf{NIZKProve}(crs_{\mathsf{NIZK}}, \tilde{\mathsf{stmt}}, (y, c_{\mathsf{LFE}}))$.
    // where $\tilde{\mathsf{stmt}} = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, pk, d_{\mathsf{LFE}}, \mathsf{stmt}, \mathsf{cds}_1, c_{\mathsf{PKE}})$
- Return $c = (\mathsf{cds}_2, c_{\mathsf{PKE}}, \pi)$.

$\bar{\mathsf{LFEDec}}(crs, c, r)$:

- Parse crs as $(crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk)$, c as $(\mathsf{cds}_2, c_{\mathsf{PKE}}, \pi)$, and r as $(r_{\mathsf{LFE}}, r_{\mathsf{CDS}})$.
- If $1 \neq \mathsf{NIZKVerify}(crs_{\mathsf{NIZK}}, \tilde{\mathsf{stmt}}, \pi)$, then return $\perp$.
- Else return $z = \mathsf{LFEDec}(crs_{\mathsf{LFE}}, \mathsf{CDSDecode}(crs_{\mathsf{CDS}}, \mathsf{cds}_2, r_{\mathsf{CDS}}), r_{\mathsf{LFE}})$.

Figure 15: Malicious LFE Protocol.

9.1 Efficiency

The efficiency analysis follows along the lines of that of Section 8.1, except that in this case the secret input of the receiver (the circuit $\mathcal{C}$) is not given in a committed form. This means that both in terms of computation and communication, our protocol is (asymptotically) as efficient as the underlying LFE. In particular:

- The communication complexity depends only on λ, the depth of $\mathcal{C}$, and Bob's input.

- The computation is Bob-optimized (in the same sense as the underlying LFE).

We stress that the dependency with respect to the circuit depth is only present in the underlying LFE and not in the other building blocks of our protocol.

9.2 Analysis

The argument for correctness is also identical to that of Theorem 12. The security of our scheme is analyzed in the following.

Theorem 14 (Security)**.** *Let* $(\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *be a semi-malicious LFE, let* $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSend}, \mathsf{CDSDecode})$ *be a context-secure laconic CDS, and let* $(\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ *be a NIZK. Then the construction* $(\mathsf{LFE}\bar{\mathsf{S}}\mathsf{etup}, \mathsf{LFE}\bar{\mathsf{C}}\mathsf{ompress}, \mathsf{LF}\bar{\mathsf{E}}\mathsf{Enc}, \mathsf{LF}\bar{\mathsf{E}}\mathsf{Dec})$ *as defined in Figure 15 is context secure.*

Proof of Theorem 14. Let $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$ be a context for our LFE protocol. We first define a sequence of hybrids.

- Hybrid $\mathcal{H}_0$: This is the real experiment. In particular $\mathcal{H}_0$ computes
  - $crs_{\mathsf{LFE}} \leftarrow \mathsf{LFESetup}(1^\lambda)$
  - $crs_{\mathsf{CDS}} \leftarrow \mathsf{CDSSetup}(1^\lambda)$
  - $crs_{\mathsf{NIZK}} \leftarrow \mathsf{NIZKSetup}(1^\lambda)$
  - $(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda)$
  - Set $crs = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk)$
  - Compute $(\mathsf{st}, d) \leftarrow \mathcal{Z}_1(crs)$
  - Compute and return $b^* \leftarrow \mathcal{Z}_2^{\mathsf{LFEEnc}\bar{}(crs,d,\cdot)}(\mathsf{st})$.
- Hybrid $\mathcal{H}_1$: This hybrid is identical to $\mathcal{H}_0$, except that we compute $(crs_{\mathsf{NIZK}}, \mathsf{td})$ by $\mathsf{NIZKSim}_1(1^\lambda)$ and we give $\mathcal{Z}_2$ access to an oracle $\mathcal{O}_1$, which behaves identical to $\mathsf{LFEEnc}\bar{}(crs, d, \cdot)$ except that it computes π as $\mathsf{NIZKSim}_2(crs_{\mathsf{NIZK}}, \tilde{\mathsf{stmt}}, \mathsf{td})$.
- Hybrid $\mathcal{H}_2$: Identical to $\mathcal{H}_1$ except that we give $\mathcal{Z}_2$ access to an oracle $\mathcal{O}_2$ which is identical to $\mathcal{O}_1$ except that it computes $c_{\mathsf{PKE}} \leftarrow \mathsf{PKEEnc}(pk, 0^{|y|})$.
- Hybrid $\mathcal{H}_3$: We first express $\mathcal{H}_3$ as a context $\mathcal{Z}' = (\mathcal{Z}'_1, \mathcal{Z}'_2)$ for CDS. Specifically, $\mathcal{Z}'_1$ takes as input crs_{CDS}, generates crs_{LFE}, crs_{NIZK} and pk itself, runs $\mathcal{Z}_1(crs)$ and outputs a state st' and a digest cds_1 for CDS. $\mathcal{Z}'_2$ has oracle access to $\mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, \cdot, \mathsf{cds}_1)$ and computes $\mathcal{Z}_2^{\mathcal{O}_2(\cdot)}$, where it simulates $\mathcal{O}_2$ but replaces calls of $\mathcal{O}_2$ to $\mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, \cdot, \mathsf{cds}_1)$ by calls to its own oracle. Context security yields an extractor $\mathsf{Ext}_{\mathcal{Z}}^{\mathsf{CDS}}$ and a simulator $\mathsf{Sim}^{\mathsf{CDS}}$. We now define $\mathcal{H}_3$ as follows.
  - $crs_{\mathsf{LFE}} \leftarrow \mathsf{LFESetup}(1^\lambda)$
  - $crs_{\mathsf{CDS}} \leftarrow \mathsf{CDSSetup}(1^\lambda)$
  - $(crs_{\mathsf{NIZK}}, \mathsf{td}) \leftarrow \mathsf{NIZKSim}_1(1^\lambda)$
  - $(sk, pk) \leftarrow \mathsf{PKEKGen}(1^\lambda)$
  - Set $crs = (crs_{\mathsf{LFE}}, crs_{\mathsf{CDS}}, crs_{\mathsf{NIZK}}, pk)$
  - Compute $(\mathsf{st}, d) \leftarrow \mathcal{Z}_1(crs)$
  - Parse d as $(d_{\mathsf{LFE}}, \mathsf{cds}_1)$ and include all other values in a $\mathcal{Z}'$ state st'

– Compute $\mathsf{aux} = w^* = (\mathcal{C}^*, s^*) \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{\mathsf{CDS}}(crs, \mathsf{st}', \mathsf{cds}_1, \delta)$

– Compute and return $b^* \leftarrow \mathcal{Z}_2^{\mathcal{O}_3}(\mathsf{st})$, where $\mathcal{O}_3$ is identical to $\mathcal{O}_2$ but replaces calls to $\mathsf{CDSSend}(crs_{\mathsf{CDS}}, \mathsf{stmt}, \cdot, \mathsf{cds}_1)$ by $\mathsf{Sim}^{\mathsf{CDS}}(crs_{\mathsf{CDS}}, \mathsf{cds}_1, (\mathcal{C}^*, s^*), c_{\mathsf{LFE}})$.

- Hybrid $\mathcal{H}_4$: Identical to $\mathcal{H}_3$ except that we give $\mathcal{Z}_2$ access to an oracle $\mathcal{O}_4$ which is identical to $\mathcal{O}_3$ except that it checks whether

$$d_{\mathsf{LFE}} = \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}^*; s^*)$$

and computes $c_{\mathsf{LFE}} \leftarrow \mathsf{LFESim}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, \mathcal{C}^*, \mathcal{C}^*(y))$ if this is the case and sets $c_{\mathsf{LFE}} = \bot$ otherwise.

We can now define the extractor $\mathsf{Ext}_{\mathcal{Z}}$ and the simulator Sim.

$\mathsf{Ext}_{\mathcal{Z}}(crs, \mathsf{st}, d, \mathsf{r}^*, \delta)$: Parse $d = (d_{\mathsf{LFE}}, \mathsf{cds}_1)$ and run $w^* \leftarrow \mathsf{Ext}_{\mathcal{Z}}^{\mathsf{CDS}}(crs_{\mathsf{CDS}}, \mathsf{st}, \mathsf{cds}_1, \mathsf{r}^*, \delta)$. Return $\mathsf{aux} = w^* = (\mathcal{C}^*, s^*)$.

Let $\mathsf{Sim}^{\mathsf{CDS}}$ be the simulator given by the context security of the laconic CDS scheme and let td be the trapdoor generated by the simulated crs_{NIZK}. We then define our simulator Sim in the following.

$\mathsf{Sim}(crs, d, (\mathcal{C}^*, s^*), \mathcal{C}^*(y), \mathsf{td})$: Parse $d = (d_{\mathsf{LFE}}, \mathsf{cds}_1)$, then check whether

$$d_{\mathsf{LFE}} = \mathsf{LFECompress}(crs_{\mathsf{LFE}}, \mathcal{C}^*; s^*). \tag{6}$$

If this is the case compute $c_{\mathsf{LFE}} \leftarrow \mathsf{LFESim}(crs_{\mathsf{LFE}}, d_{\mathsf{LFE}}, \mathcal{C}^*, \mathcal{C}^*(y))$, else set $c_{\mathsf{LFE}} = \bot$. Compute $\mathsf{cds}_2 \leftarrow \mathsf{Sim}^{\mathsf{CDS}}(crs_{\mathsf{CDS}}, \mathsf{cds}_1, (\mathcal{C}^*, s^*), c_{\mathsf{LFE}})$. Then set $c_{\mathsf{PKE}} \leftarrow \mathsf{PKEEnc}(pk, 0^{|y|})$ and simulate $\pi \leftarrow \mathsf{NIZKSim}_2(crs_{\mathsf{NIZK}}, \mathsf{td}, \tilde{\mathsf{stmt}})$. Return $c = (\mathsf{cds}_2, c_{\mathsf{PKE}}, \pi)$.

Both algorithms are clearly PPT and in particular the runtime of $\mathsf{Ext}_{\mathcal{Z}}$ is identical to that of $\mathsf{Ext}_{\mathcal{Z}}^{\mathsf{CDS}}$. We will choose $\delta(\varepsilon) = \varepsilon/4$. Assume towards contradiction that there exists a inverse-polynomial ε such that

$$|\Pr[\mathcal{Z}(1^\lambda) = 1] - \Pr[\mathcal{EZ}(1^\lambda, \varepsilon) = 1]| > \varepsilon.$$

As $\mathcal{H}_0$ is identically distributed to $\mathcal{Z}(1^\lambda)$ and $\mathcal{H}_4$ is identically distributed to $\mathcal{EZ}(1^\lambda, \varepsilon)$, by the averaging principle there must exist an $k \in \{1, \ldots, 4\}$ such that

$$|\Pr[\mathcal{H}_k = 1] - \Pr[\mathcal{H}_{k-1} = 1]| > \varepsilon/4.$$

We will now go through all possible cases of k.

(1) $k = 1$: The two hybrids are indistinguishable by an invocation of the zero-knowledge property of NIZK.

(2) $k = 2$: The two hybrids are computationally indistinguishable by the semantic security of the public key encryption scheme. Note that the simulator never uses the secret key sk.

(3) $k = 3$: In this case we will obtain a contradiction against the context security of CDS. To see this, note that by construction of $\mathcal{Z}'$ it holds that $\mathcal{H}_2$ is identically distributed to $\mathcal{Z}'$ and by construction of $\mathcal{H}_3$ it holds that $\mathcal{H}_3$ is identically distributed to the experiment $\mathcal{EZ}'$. Thus we get a contradiction against the context security of CDS.

(4) $k = 4$: Observe that if Equation (6) is true, then it holds that the digest d_{LFE} is honestly computed with randomness s^*, which is known by the simulator. This change is undetectable by a reduction against the semi-malicious security of the underlying LFE scheme where the reduction sets the random coins to s^*.

□

The next theorem shows that the scheme is UC-secure for a corrupted sender (Bob). This implies the notion of (reusable) function hiding.

Theorem 15 (Bob UC-Security)**.** *Let* $(\mathsf{LFESetup}, \mathsf{LFECompress}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *be a function-hiding LFE, let* $(\mathsf{PKEKGen}, \mathsf{PKEEnc}, \mathsf{PKEDec})$ *be a semantically secure and perfectly correct public-key encryption scheme, let* $(\mathsf{CDSSetup}, \mathsf{CDSReceive}, \mathsf{CDSSend}, \mathsf{CDSDecode})$ *be a receiver simulatable laconic CDS, and let* $(\mathsf{NIZKSetup}, \mathsf{NIZKProve}, \mathsf{NIZKVerify})$ *be a sound NIZK. Then the construction* $(\mathsf{LFE\bar{S}etup}, \mathsf{LFE\bar{C}ompress}, \mathsf{LF\bar{E}Enc}, \mathsf{LF\bar{E}Dec})$ *as defined in Figure 15 UC-realizes* $\mathcal{F}_{\mathsf{2PC}}$ *for a corrupted Bob.*

Proof of Theorem 15. The analysis is identical to the case of corrupted P_2 in the proof of Theorem 13. □

References

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 111–120, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BDH+17] Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. Concurrently composable security with shielded super-polynomial simulators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 351–381, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

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

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

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 17–33, Cambridge, MA, USA, February 10–12, 2005. Springer, Heidelberg, Germany.

[CLP13] Ran Canetti, Huijia Lin, and Rafael Pass. From unprovability to environmentally friendly protocols. In *54th Annual Symposium on Foundations of Computer Science*, pages 70–79, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 265–281, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[DGH+19] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. Cryptology ePrint Archive, Report 2019/414, 2019. https://eprint.iacr.org/2019/414.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[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 Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan

Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 158–189, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 723–732, Victoria, BC, Canada, May 4–6, 1992. ACM Press.

[Lin03] Yehuda Lindell. General composition and universal composability in secure multiparty computation. In *44th Annual Symposium on Foundations of Computer Science*, pages 394–403, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

[LQR+19] Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New Constructions of Reusable Designated-Verifier NIZKs. Cryptology ePrint Archive, Report 2019/242, 2019. https://eprint.iacr.org/2019/242.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science*, pages 436–453, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[Nao90] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 128–136, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive Zero Knowledge for NP from (Plain) Learning With Errors. Cryptology ePrint Archive, Report 2019/158, 2019. https://eprint.iacr.org/2019/158.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable Designated-Verifier NIZKs for all NP from CDH. Cryptology ePrint Archive, Report 2019/235, 2019. https://eprint.iacr.org/2019/235.

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

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A Proof of Theorem 6

A LOT should satisfy the notions of correctness and sender security and we analyze the two property separately.

Correctness. If the underlying weak LOT is perfectly correct, then so is the resulting LOT. On the other hand, if the weak LOT is only $(1/p)$-correct, for some polynomial p, then the resulting scheme is (λ/p)-correct by a union bound. When considering string-LOT, if $p = \lambda^2$ we can pre-process the message with an error-correcting code to reduce the error probability to $1/2^\lambda$. This can be further lifted to a perfectly correct scheme using Naor's trick [Nao90]. Note that these transformations are information-theoretic and do not add any additional assumption.

Sender Security. We show that $\bar{\mathsf{LOT}}$ is sender secure, under the assumption that LOT is weakly sender secure. For the analysis it is useful to recall the following lemma from [CHS05].

Lemma 5 (Hardness Amplification [CHS05])**.** *Let $\mathcal{Y}$ be some distribution over pairs of puzzles and solutions* $(\mathsf{puzzle}, \mathsf{solution})$ *and let* Solve *be an algorithm such that*

$$\Pr\left[\mathsf{Solve}(\mathsf{puzzle}^{(1)}, \ldots, \mathsf{puzzle}^{(\lambda)}) = (\mathsf{solution}^{(1)}, \ldots, \mathsf{solution}^{(\lambda)})\right] \geq \frac{1}{poly(\lambda)}$$

where each pair $(\mathsf{puzzle}^{(i)}, \mathsf{solution}^{(i)})$ *is uniformly sampled from $\mathcal{Y}$ and poly is some polynomial function. Then for all constants $c > 0$ there exists a PPT algorithm* Amp *such that*

$$\Pr\left[\mathsf{Amp}^{\mathsf{Solve},\mathcal{Y}}(1^\lambda, \mathsf{puzzle}^*) = \mathsf{solution}^*\right] \geq c$$

where $(\mathsf{puzzle}^*, \mathsf{solution}^*)$ *is uniformly sampled from $\mathcal{Y}$.*

Before proving the main statement of the theorem, we put forward the following claim. Intuitively, this states that the adversary cannot predict the value of both K_0 and K_1 for a fixed digest d.

Claim 15.1. *Let* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{KGen}, \mathsf{Receive})$ *be a weak LOT. Then for all* $\lambda \in \mathbb{N}$*, for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, and for all polynomial functions* $poly(\lambda)$ *there exists a negligible function* $negl$ *such that*

$$\Pr\left[\varepsilon_0 > \frac{1}{poly(\lambda)} \text{ and } \varepsilon_1 > \frac{1}{poly(\lambda)}\right] = negl(\lambda)$$

where, for $\beta \in \{0,1\}$*,* ε_β *is defined as*

$$\varepsilon_\beta = \Pr\left[\mathcal{A}_2(c^{(1)}, \ldots, c^{(\lambda)}, \mathsf{st}, \beta) = (k_\beta^{(1)}, \ldots, k_\beta^{(\lambda)}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (c^{(1)}, \ldots, c^{(\lambda)}) \leftarrow\!\!\$\ \mathsf{KGen}(crs, d, L) \end{array}\right]$$

where the probabilities are taken over the random coins of Setup*,* KGen*, and* $\mathcal{A}_1$*.*

Proof of Claim 15.1. Assume the contrary, i.e., that there is an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that

$$\Pr\left[\varepsilon_0 > \frac{1}{poly(\lambda)} \text{ and } \varepsilon_1 > \frac{1}{poly(\lambda)}\right] = \frac{1}{poly(\lambda)}$$

for some polynomial function $poly$. Let S be the set of random coins of the algorithms Setup and $\mathcal{A}_1$ for which

$$\Pr\left[\mathcal{A}_2(c^{(1)}, \ldots, c^{(\lambda)}, \mathsf{st}, \beta) = (k_\beta^{(1)}, \ldots, k_\beta^{(\lambda)}) \;\middle|\; \begin{array}{l} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (d, L, \mathsf{st}) \leftarrow \mathcal{A}_1(crs) \\ (c^{(1)}, \ldots, c^{(\lambda)}) \leftarrow\!\!\$\ \mathsf{KGen}(crs, d, L) \end{array}\right] > \frac{1}{poly(\lambda)}$$

for all $\beta \in \{0,1\}$ simultaneously, where the probabilities are taken over the random coins of Send. Fix an element $s \in S$ and note that such choice fixes also the tuple (d, L, st) output by $\mathcal{A}_1$. By construction we have that

$$\Pr\left[\mathcal{A}_2(c^{(1)}, \ldots, c^{(\lambda)}, \mathsf{st}, \beta) = (k_\beta^{(1)}, \ldots, k_\beta^{(\lambda)})\right] > \frac{1}{poly(\lambda)}$$

for all $\beta \in \{0,1\}$, where $(c^{(1)}, \ldots, c^{(\lambda)}) \leftarrow\!\!\$\ \mathsf{KGen}(crs, d, L)$. By Lemma 5, we have that

$$\Pr\left[\mathsf{Amp}^{\mathcal{A}_2(\cdot, \mathsf{st}, \beta), \mathsf{KGen}}(c) = k_\beta\right] \geq 3/4$$

for all $\beta \in \{0,1\}$, where $c \leftarrow\!\!\$\ \mathsf{KGen}(crs, d, L)$. Define $\mathcal{A}^*$ to run $\mathsf{Amp}^{\mathcal{A}_2(\cdot, \mathsf{st}, \beta), \mathsf{KGen}}$, for all $\beta \in \{0,1\}$, and output whatever the algorithms output. Then

$$\Pr\left[\mathcal{A}^*(c, \mathsf{st}) = (k_0, k_1)\right] \geq 1 - \sum_\beta \Pr\left[\mathsf{Amp}^{\mathcal{A}_2(\cdot, \mathsf{st}, \beta), \mathsf{KGen}}(c) \neq k_\beta\right] \geq 1/2$$

where $c \leftarrow\!\!\$\ \mathsf{KGen}(crs, d, L)$ and the last inequality is by a union bound. It follows that $\mathcal{A}^*$ breaks the weak sender security of LOT with probability at least $\frac{1}{2 \cdot poly(\lambda)}$. This proves our claim. □

Before completing the proof of the theorem, we recall the central lemma of the result of Goldreich and Levin [GL89] that shows an efficient decoder for Hadamard codes.

Lemma 6 (Goldreich-Levin Decoding [GL89])**.** *There exists a PPT algorithm* GLDec *and a polynomial* $poly'$ *such that for any* $(n, \ell) \in \mathbb{N}$ *and for any* $t \in \{0,1\}^n$*, and any function* $f : \{0,1\}^n \to \{0,1\}$ *such that*

$$\Pr\left[f(t) = \sum_{i=1}^{n} x_i \cdot t_i\right] \geq 1/2 + 1/\ell$$

(over the random choice of t) then it holds that

$$\Pr\left[\mathsf{GLDec}^f(1^n, 1^\ell) = x\right] \geq \frac{1}{poly'(n,\ell)}.$$

We are now in the position of proving our main theorem. Assume towards contradiction that there eixsts an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the sender security of $\bar{\mathsf{LOT}}$, i.e.,

$$\Pr\left[\varepsilon_0 > \frac{1}{poly(\lambda)} \text{ and } \varepsilon_1 > \frac{1}{poly(\lambda)}\right] = \frac{1}{poly(\lambda)}$$

where ε_β is defined as in Definition 27. Let S be the set of random coins of the algorithms $\bar{\mathsf{Setup}}$ and $\mathcal{A}_1$ for which the above event happens. By assumption, sampling random tapes yields an element of S with probability at least $\frac{1}{poly(\lambda)}$. Fix an $s \in S$ and observe that this also fixes $(m_0, m_1, d, L, \mathsf{st})$. For such an s we have that

$$|\Pr\left[\mathcal{A}_2(\tilde{c}, C_0, C_1, t_0, t_1, \mathsf{st}, \beta = 0) = 1\right] - \Pr\left[\mathcal{A}_2(\tilde{c}, R_0, C_1, t_0, t_1, \mathsf{st}, \beta = 0) = 1\right]| \geq \frac{1}{poly(\lambda)}$$
$$|\Pr\left[\mathcal{A}_2(\tilde{c}, C_0, C_1, t_0, t_1, \mathsf{st}, \beta = 1) = 1\right] - \Pr\left[\mathcal{A}_2(\tilde{c}, C_0, R_1, t_0, t_1, \mathsf{st}, \beta = 1) = 1\right]| \geq \frac{1}{poly(\lambda)}$$

where the probability is taken over the random choice of $\bar{\mathsf{Send}}(crs, d, L, m_0, m_1)$ and $R_b \leftarrow\!\!\$\, \{0,1\}$, for all $b \in \{0,1\}$. Recall that $C_b = \mathsf{GLEnc}(K_b; t_b) \oplus m_b$, for all $b \in \{0,1\}$. Since (for a single bit) distinguishing implies predicting, it follows that there exists a PPT algorithm $\mathcal{A}^f$ such that

$$\Pr\left[\mathcal{A}^f(\tilde{c}, t_0, t_1, C_1, \mathsf{st}, \beta = 0) = \mathsf{GLEnc}(K_0; t_0)\right] \geq 1/2 + \frac{1}{2 \cdot poly(\lambda)}$$
$$\Pr\left[\mathcal{A}^f(\tilde{c}, t_0, t_1, C_0, \mathsf{st}, \beta = 1) = \mathsf{GLEnc}(K_1; t_1)\right] \geq 1/2 + \frac{1}{2 \cdot poly(\lambda)}$$

where the probabilities are taken over the random coins of the $\bar{\mathsf{Send}}$ algorithm, as above. Let Z_0 be the set of random variables $z_0 = (crs, \tilde{c}, K_0, K_1, t_1)$ such that the LHS of the first equation is $\geq 1/2 + \frac{1}{4 \cdot poly(\lambda)}$ and let Z_1 be defined analogously. By Markov inequality, for any $s \in S$, the probability that a random $z_b \in Z_b$ is $\geq \frac{1}{4 \cdot poly(\lambda)}$, for all $b \in \{0,1\}$. By Lemma 6, there exists a PPT decoder GLDec and a polynomial $poly'$ such that, for any fixed $z_0 \in Z_0$ and $z_1 \in Z_1$, we have

$$\Pr\left[\mathsf{GLDec}(\tilde{c}, t_1, C_1, \mathsf{st}, \beta = 0) = K_0\right] \geq \frac{1}{poly(\lambda)'}$$
$$\Pr\left[\mathsf{GLDec}(\tilde{c}, t_0, C_0, \mathsf{st}, \beta = 1) = K_1\right] \geq \frac{1}{poly(\lambda)'}$$

over the internal coins of GLDec. Let $\mathcal{A}^*(\tilde{c}, \mathsf{st}, \beta)$ be an adversary that outputs $\mathsf{GLDec}(\tilde{c}, t_{1-\beta}, C_{1-\beta}, \mathsf{st}, \beta)$ for randomly sampled $(t_{1-\beta}, C_{1-\beta})$. Then for all values in $z_0 \in Z_0$ and $z_1 \in Z_1$ we have that

$$\Pr\left[\mathcal{A}^*(\tilde{c}, \mathsf{st}, \beta = 0; t_1) = K_0\right] \geq \frac{1}{2 \cdot poly(\lambda)'}$$
$$\Pr\left[\mathcal{A}^*(\tilde{c}, \mathsf{st}, \beta = 1; t_0) = K_1\right] \geq \frac{1}{2 \cdot poly(\lambda)'}$$

fixing $t_{1-\beta}$ and over the random coins of $\mathcal{A}^*$. Here the factor $1/2$ in the loss of probability comes from guessing the correct value of $C_{1-\beta}$. It follows that

$$\Pr\left[\mathcal{A}^*(\tilde{c}, \mathsf{st}, \beta = 0) = K_0\right] \geq \frac{1}{4 \cdot poly(\lambda) \cdot 2 \cdot poly(\lambda)'}$$
$$\Pr\left[\mathcal{A}^*(\tilde{c}, \mathsf{st}, \beta = 1) = K_1\right] \geq \frac{1}{4 \cdot poly(\lambda) \cdot 2 \cdot poly(\lambda)'}$$

over the coins of $\mathcal{A}^*$ and the random choices of $(\tilde{c}, K_0, K_1)$. Expanding we have that both events

$$\Pr\left[\mathcal{A}^*((c^{(1)}, \ldots, c^{(\lambda)}), \mathsf{st}, \beta = 0) = (k_0^{(1)}, \ldots, k_0^{(\lambda)})\right] \geq \frac{1}{4 \cdot poly(\lambda) \cdot 2 \cdot poly(\lambda)'}$$
$$\Pr\left[\mathcal{A}^*((c^{(1)}, \ldots, c^{(\lambda)}), \mathsf{st}, \beta = 1) = (k_1^{(1)}, \ldots, k_1^{(\lambda)})\right] \geq \frac{1}{4 \cdot poly(\lambda) \cdot 2 \cdot poly(\lambda)'}$$

happen with probability at least $\frac{1}{poly(\lambda)}$, over the random choices of Setup, KGen, and $\mathcal{A}_1$. This contradicts Claim 15.1 and concludes our proof. □