Non-Interactive Zero Knowledge from Sub-exponential DDH

Abhishek Jain Zhengzhong Jin

Johns Hopkins University

Abstract

We provide the first constructions of non-interactive zero-knowledge and Zap arguments for NP based on the sub-exponential hardness of Decisional Diffie-Hellman against polynomial time adversaries (without use of groups with pairings).

Central to our results, and of independent interest, is a new notion of *interactive trapdoor hashing protocols*.

1 Introduction

Zero-knowledge (ZK) proofs [36] are a central object in the theory and practice of cryptography. A ZK proof allows a prover to convince a verifier about the validity of a statement without revealing any other information. ZK proofs have found wide applications in cryptography in all of their (interactive) avatars, but especially so in the non-interactive form where a proof consists of a single message from the prover to the verifier. This notion is referred to as *non-interactive zero knowledge* (NIZK) [26]. Applications of NIZKs abound and include advanced encryption schemes [54, 27], signature schemes [4, 7], blockchains [6], and more.

Since NIZKs for non-trivial languages are impossible in the plain model, the traditional (and de facto) model for NIZKs allows for a trusted setup that samples a common reference string (CRS) and provides it to the prover and the verifier algorithms. Starting from the work of [26], a major line of research has been dedicated towards understanding the assumptions that are sufficient for constructing NIZKs in the CRS model [10, 32, 5, 21, 40, 39, 35, 61, 22, 19, 17, 58, 25]. By now, NIZKs for NP are known from most of the standard assumptions known to imply public-key encryption – this includes factoring related assumptions [10, 32], bilinear maps [21, 40, 39], and more recently, learning with errors (LWE) [17, 58].

Notable exceptions to this list are standard assumptions related to the discrete-logarithm problem such as the Decisional Diffie-Hellman (DDH) assumption. In particular, the following question has remained open for three decades:

Do there exist NIZKs for NP based on DDH?

From a conceptual viewpoint, an answer to the above question would shed further light on the cryptographic complexity of NIZKs relative to public-key encryption. It would also improve our understanding of the power of groups with bilinear maps relative to non-pairing groups in cryptography. There are (at least) two prominent examples where bilinear maps have traditionally had an edge – advanced encryption schemes such as identity-based [12] and attribute-based encryption [62, 38] (and more broadly, functional encryption [62, 13, 56]), and NIZKs. For the former, the gap has recently started to narrow in some important cases; see, e.g., [28]. We seek to understand whether such gap is inherent for NIZKs based on standard assumptions.[1]

[1] If we allow for non-standard assumptions (albeit those not known to imply public-key encryption), then this gap is not inherent, as demonstrated by [19, 25].

A recent beautiful work of Brakerski et al. [15] demonstrates that this gap disappears if we additionally rely on the hardness of the learning parity with noise (LPN) problem. Namely, they construct NIZKs assuming that DDH and LPN are *both* hard. NIZKs based on the *sole* hardness of DDH, however, still remain elusive.

Zaps. Dwork and Naor [30] introduced the notion of *Zaps*, aka two-round public-coin proof systems in the plain model (i.e., without a trusted setup) that achieve a weaker form of privacy known as witness-indistinguishability (WI) [33]. Roughly speaking, WI guarantees that a proof for a statement with multiple witnesses does not reveal which of the witnesses was used in the computation of the proof.

Despite this seeming weakness, [30] proved that (assuming one-way functions) Zaps are equivalent to statistically-sound NIZKs in the common *random* string model. This allows for porting some of the known results for NIZKs to Zaps; specifically, those based on factoring assumptions and bilinear maps. Subsequently, alternative constructions of Zaps were proposed based on indistinguishability obfuscation [8]. Very recently, computationally-sound Zaps, aka Zap *arguments* were constructed based on quasi-polynomial LWE [2, 49, 37].

As in the case of NIZKs, constructing Zaps (or Zap arguments) for NP based on standard assumptions related to discrete-logarithm remains an open problem. Moreover, if we require *statistical* privacy, i.e., statistical Zap arguments , curiously, even bilinear maps have so far been insufficient. This is in contrast to statistical NIZKs, where constructions based on bilinear maps are known [40, 39]. Presently, statistical Zap arguments are only known based on quasi-polynomial LWE [2, 37]. A recent work of [50] constructs a variant of statistical Zap arguments based on bilinear maps that achieves public verifiability, but falls short of achieving public coin property.

1.1 Our Results

I. Main Results. In this work, we construct (statistical) NIZK and Zap arguments for NP based on the sub-exponential hardness of DDH against polynomial-time adversaries in standard groups.

Theorem 1.1 (Main Result – Informal)**.** *Assuming sub-exponential hardness of DDH, there exist:*

- *(Statistical) NIZK arguments for NP in the common random string model.*
- *Statistical Zap arguments for NP.*

Our NIZK achieves adaptive, multi-theorem statistical zero knowledge and non-adaptive soundness. By relaxing the zero-knowledge guarantee to be computational, we can achieve adaptive soundness. Our Zap argument achieves adaptive statistical witness indistinguishability and non-adaptive soundness.[2]

Our results in Theorem 1.1 rely on the assumption that *polynomial-time* adversaries cannot distinguish Diffie-Hellman tuples from random tuples in standard $\mathbb{Z}_q^*$ group with better than sub-exponentially small advantage. Alternatively, if we also assume hardness against sub-exponential time adversaries, then we can instantiate Theorem 1.1 using Elliptic curves over $\mathbb{F}_p$ with a prime $p > 3$ (see Appendix B). To the best of our knowledge, our assumption is unaffected by known attacks on the discrete logarithm problem.[3]

Discussion. While our primary focus is on constructing NIZKs and Zap arguments from DDH, we note that our constructions enjoy certain properties that have previously not been achieved even using bilinear maps:

[2]Following [50], by standard complexity leveraging, our statistical NIZK and Zap arguments can be upgraded (without changing our assumption) to achieve adaptive soundness for all instances of a priori (polynomially) bounded size. For the "unbounded-size" case, [57] proved the impossibility of statistical NIZKs where adaptive soundness is proven via a black-box reduction to falsifiable assumptions [52].

[3]There are well-known attacks for discrete logarithm over $\mathbb{Z}_q^*$ that require sub-exponential time and achieve constant success probability [1, 24]. However, as observed in [19], a 2^t time algorithm with constant successful probability does not necessarily imply a polynomial time attack with 2^{-t} successful probability.

- Our NIZK constructions rely on a common *random* string setup unlike prior schemes based on bilinear maps that require a common *reference* string for achieving statistical ZK [40, 39].
- Our statistical Zap argument is the first group-based construction (irrespective of whether one uses bilinear maps or not). Known constructions of Zaps from bilinear maps only achieve computational WI [40, 39].

In particular, statistical NIZKs in the common random string model were previously only known from LWE (or circular-secure FHE) [17, 58], and statistical Zap arguments were previously only known from (quasi-polynomial) LWE [2, 37].

II. Correlation-Intractatable Hash Functions. In order to obtain our main results, we follow the correlation intractability (CI) framework for Fiat-Shamir [33] implemented in a recent remarkable sequence of works [19, 41, 17, 58, 15, 25]. The central idea of this framework is to instantiate the random oracle in the Fiat-Shamir transformation by *correlation intractable hash functions* (CIH) [20]. Roughly speaking, a family of hash functions $(\mathsf{Gen}, \mathsf{Hash})$ is said to be correlation intractable for a relation class $\mathcal{R}$ if for any $R \in \mathcal{R}$, given a hash key k sampled by Gen, an adversary cannot find an input x such that $(x, \mathsf{Hash}(\mathsf{k}, x)) \in R$. In the sequel, we focus on searchable relations where R is associated with a circuit C and $(x, y) \in R$ if and only if $y = C(x)$.

A sequence of works [18, 43, 19, 41, 25] have constructed CIH for various classes of (not necessarily efficiently searchable) relations from well-defined, albeit strong assumptions that are not well understood. Recently, Canetti et al. [17] constructed CIH for all efficiently searchable relations from circular-secure fully homomorphic encryption. Subsequently, Peikert and Shiehian [58] obtained a similar result based on standard LWE. More recently, Brakerski et al. [15] constructed CIH for relations that can be approximated by constant-degreee polynomials (over $\mathbb{Z}_2$) based on various standard assumptions.

In this work, we expand the class of searchable relations that can be supported by CIH without relying on LWE. Specifically, we construct CIH for constant-depth threshold circuits from sub-exponential DDH.

Theorem 1.2 (Informal)**.** *Assuming sub-exponential hardness of DDH against polynomial-time attackers, there exists a CIH for* TC^0.

In fact, we can trade-off between the hardness assumption on DDH and the depth of the circuits that CIH can support. Assuming sub-exponential hardness of DDH against *sub-exponential time* adversaries, we can obtain CIH for threshold circuits of depth $O(\log \log n)$. Moreover, assuming *exponential* hardness of DDH against polynomial time adversaries (which can be conjectured to hold in elliptic curve groups), we can obtain CIH for log-depth threshold circuits, i.e., TC^1. We refer the reader to Section 6.4 for details.

While our primary interest in this work is using CIH for constructing NIZKs (and Zap arguments), we note that recent works (e.g., [17, 48]) have also explored applications of CIH to succinct arguments [45, 51], verifiable delay functions [11] and establishing hardness of complexity classes such as PPAD [23]. Our constructions of CIH may therefore be of independent interest for applications beyond NIZKs.

1.1.1 Main Tool: Interactive Trapdoor Hashing Protocols

Towards obtaining our results, we introduce the notion of *interactive trapdoor hashing protocols* (ITDH). An ITDH for a function family F is an interactive protocol between two parties – a sender and a receiver – where the sender holds an input x and the receiver holds a function $f \in F$. At the end of the protocol, the parties obtain an additive secret-sharing of $f(x)$. An ITDH must satisfy the following key properties:

- The sender must be *laconic* in that the length of each of its messages (consisting of a hash value) is independent of the input length.

- The receiver's messages must *hide* the function f (the exact formulation of this property is nuanced).

ITDH generalizes and extends the recent notion of trapdoor hash functions (TDH) [29] to *multi-round interactive protocols*. Indeed, ignoring some syntactic differences, a TDH can be viewed as an ITDH where both the receiver and the sender send a *single* message to each other.

Our primary motivation for the study of ITDH is to explore the feasibility of a richer class of computations than what can be supported by known constructions of TDH. Presently, TDH constructions are known for a small class of computations such as linear functions and constant-degree polynomials (based on various assumptions such as DDH, Quadratic Residuosity, and LWE) [29, 15]. We demonstrate that ITDH can support a much broader class of computations.

Assuming DDH, we construct a constant-round ITDH protocol for TC^0 circuits. While ITDH for TC^0 suffices for our main application, our approach can be generalized to obtain a polynomial-round ITDH for P/poly.

Theorem 1.3 (Informal)**.** *Assuming DDH, there exists a constant-round ITDH for* TC^0*.*

We view ITDH as a natural generalization of TDH that might allow for a broader pool of applications. While our present focus is on the class of computations, it is conceivable that the use of interaction might enable additional properties in the future that are not possible (or harder to achieve) in the non-interactive setting.

From ITDH to NIZKs: Round Collapsing, *Twice*. The work of [17] shows that given a CIH for all efficiently searchable relations, the Fiat-Shamir transformation can be used to collapse the rounds of so-called *trapdoor sigma protocols* to obtain NIZKs in the CRS model. Presently, however, CIH for all efficiently searchable relations are only known from LWE-related assumptions [58, 17].

Recently, Brakerski et al. [15] demonstrated a new approach for constructing CIH from (rate-1) TDH by crucially exploiting the laconic sender property of the latter. This raises hope for potential instantiations of CIH – ideally for all efficiently searchable relations – from other standard assumptions (such as DDH). So far, however, this approach has yielded CIH only for relations that can be approximated by constant-degree polynomials over $\mathbb{Z}_2$ due to limitations of known results for TDH. This severely restricts the class of compatible trapdoor sigma protocols that can be used for constructing NIZKs via the CIH framework. Indeed, Brakerski et al. rely crucially on LPN to construct such sigma protocols.

Somewhat counter-intuitively, we use *interaction* to address the challenge of constructing NIZKs solely from DDH. Specifically, we show that by using interaction – via the abstraction of ITDH – we can expand the class of functions that can be computed with a laconic sender (as per Theorem 1.3). Furthermore, if an ITDH is sufficiently function-private (where the amount of security required depends on the round complexity), then we can *collapse its rounds* to construct CIH. Using this approach, we construct a CIH for TC^0 based on sub-exponential DDH (as per Theorem 1.2).

Expanding the class of relations for CIH in turn expands the class of compatible trapdoor sigma protocols. In particular, we show that trapdoor sigma protocols for NP compatible with CIH from Theorem 1.2 can be built from DDH. This allows us to construct NIZK and Zap arguments in Theorem 1.1.

Overall, our approach for constructing NIZKs involves **two stages of round collapsing** – we first collapse rounds of ITDH to construct CIH, and then use CIH to collapse rounds of trapdoor sigma protocols to obtain NIZKs. Our construction of Zaps follows a similar blueprint, where the first step is the same as in the case of NIZKs and the second round-collapsing step is similar to the recent works of Badrinarayanan et al. [2] and Goyal et al. [37].

1.2 Guide to the paper

We present the technical overview in Section 2 and the necessary preliminaries in Section 3. We define and construct ITDH in Sections 4 and Section 5 respectively, and construct CIH for TC^0 in Section 6. Then we construct NIZKs in Section 7, and Zaps in Section 8.

2 Technical Overview

Our constructions rely on the correlation-intractability framework for instantiating the Fiat-Shamir paradigm. We start by recalling this framework.

Fiat-Shamir via Correlation Intractability. The correlation intractability (CI) framework instantiates the random oracle in the Fiat-Shamir paradigm for NIZKs via a family of correlation intractable hash functions $(\mathsf{Gen}, \mathsf{Hash})$. Let Σ be a sigma protocol for a language $\mathcal{L}$ where the messages are denoted as α, β and γ. To obtain a NIZK in the CRS model, we collapse the rounds of Σ by computing β as the output of $\mathsf{Hash}(k, \alpha)$ for a key k sampled by Gen and fixed as part of CRS.

We now recall the argument for soundness of the resulting scheme. From the special soundness of Σ, for any $x \notin \mathcal{L}$ and any α, there exists a *bad challenge function* BadC such that the only possible accepting transcript (α, β, γ) must satisfy $\beta = \mathsf{BadC}(\alpha)$. In other words, any cheating prover must find an α such that $\beta = \mathsf{Hash}(\mathsf{k}, \alpha) = \mathsf{BadC}(\alpha)$. However, if $(\mathsf{Gen}, \mathsf{Hash})$ is CI for the relation searchable by BadC, then such an adversary must not exist.

Note that in general, BadC may not be efficiently computable. However, for *trapdoor sigma protocols*, BadC is efficiently computable given a "trapdoor" associated with the protocol. In this case, we only require CI for efficiently searchable relations.

Main Challenges. As mentioned earlier, the recent work of Brakerski et al. [15] leverages the compactness properties of (rate-1) trapdoor hash functions to build CIH for functions that can be approximated by a distribution on constant-degree polynomials. While this is a small class, [15] show that by relying on the LPN assumption, it is possible to construct trapdoor sigma protocols where the bad challenge function has probabilistic constant-degree representation. By collapsing the rounds of this protocol, they obtain NIZKs for NP from LPN and DDH (or other standard assumptions that suffice for constructing TDH).

We now briefly discuss the main conceptual challenges in buildings NIZKs based solely on DDH. On the one hand, (non-pairing) group-based assumptions seem to have less structure than lattice assumptions; for example, we can only exploit linear homomorphisms. Hence it is not immediately clear how to construct rate-1 trapdoor hash functions from DDH beyond (probabilistic) linear functions or constant-degree polynomials (a constant-degree polynomial is also a linear function of its monomials).[4] On the other hand, it seems that we need CIH for more complicated functions in order to build NIZKs from (only) DDH via the CIH framework.

Indeed, the bad challenge function in trapdoor sigma protocols involves (at least) *extraction* from the commitment scheme used in the protocol, and it is unclear whether such extraction can be represented by probablistic constant-degree polynomials when the commitment scheme is constructed from standard group-based assumptions. For example, the decryption circuit for the ElGamal encryption scheme [31] (based on DDH) is in a higher complexity class, and is not known to have representation by probabilistic constant-degree polynomials. Indeed, there are known lower-bounds for functions that can be approximated by probabilistic

[4] The breakthrough work of [14] shows that in the case of homomorphic secret-sharing, it is in fact possible to go beyond linear homomorphisms in traditional groups. The communication complexity of the sender in their scenario, however, grows with the input length and is *not* compact as in the case of TDH.

polynomials. Specifically, [63, 64, 55, 47] proved that approximating a n fan-in majority gate by probabilistic polynomials over binary field with a small constant error requires degree at least $\Omega(\sqrt{n})$.

Roadmap. We overcome the above dilemma by exploiting the power of interaction.

- In Section 2.1, we introduce the notion of interactive trapdoor hashing protocols (ITDH) – a generalization of TDH to multi-round interactive protocols. We show that despite increased interaction, ITDH can be used to build CIH. Namely, we devise a round-collapsing approach to construct CIH from ITDH.
- We next show that ITDH can capture a larger class of computations than what can be supported by known constructions of TDH. Namely, we construct a constant-round ITDH protocol for TC^0 where the sender is laconic (Section 2.2).
- Finally, we demonstrate that using DDH, it is possible to construct trapdoor sigma protocols where the bad challenge function can be computed in low depth. Using such sigma protocols, we build multi-theorem (statistical) NIZK and statistical Zap arguments for NP (Sections 2.3 and 2.4, respectively).

2.1 Interactive Trapdoor Hashing Protocols

We start by providing an informal definition of ITDH and then describe our strategy for constructing CIH from ITDH.

Defining ITDH. An L-level ITDH is an interactive protocol between a "sender" and a "receiver", where the receiver's input is a circuit f and the sender's input is a string x. The two parties jointly compute $f(x)$ by multiple rounds of communication that are divided into L *levels*. Each level $\ell \in [L]$ consists of two consecutive protocol messages – a receiver's message, followed by the sender's response:

- First, the receiver uses f (and prior protocol information) to compute a key k_ℓ and trapdoor td_ℓ. It sends the key k_ℓ to the sender.
- Upon receiving this message, the sender computes a hash value h_ℓ together with an encoding e_ℓ. The sender sends h_ℓ to the receiver but keeps e_ℓ to herself. (The encoding e_ℓ can be viewed as sender's "private state" used for computing the next level message.)

Upon receiving the level L (i.e., final) message h_L from the sender, the receiver computes a decoding value d using the trapdoor. The function output $f(x)$ can be recovered by computing $\mathsf{e} \oplus \mathsf{d}$, where e is the *final* level encoding computed by the sender. We require the following properties from ITDH:

- **Compactness**: The sender's message in every level must be *compact*. Specifically, for every level $\ell \in [L]$, the size of the hash value h_ℓ is bounded by the security parameter, and is independent of the length of the sender's input x and the size of the circuit f.
- **Approximate Correctness**: For an overwhelming fraction of the random tapes for the receiver, for any input x, the Hamming distance between $\mathsf{e} \oplus \mathsf{d}$ and $f(x)$ must be small. Note that this is an *adaptive* definition in that the input x is chosen after the randomness for the receiver is fixed.
- **Leveled Function Privacy**: The receiver's messages computationally hide the circuit f. Specifically, we require that the receiver's message in every level can be simulated without knowledge of the circuit f. Moreover, we allow the privacy guarantee to be *different* for each level by use of different security parameters for different levels.

As we discuss in Section 4.1, barring some differences in syntax, trapdoor hash functions can be viewed as 1-level ITDH. We refer the reader to the technical sections for a formal definition of ITDH.

CIH from ITDH. We now describe our round-collapsing strategy for constructing CIH from ITDH. Given an L-level ITDH for a circuit family $\mathcal{C}$, we construct a family of CIH for relations searchable by $\mathcal{C}$ as follows:

- **Key Generation:** The key generation algorithm uses the function-privacy simulator for ITDH to compute a simulated receiver message for *every* level. It outputs a key k consisting of L simulated receiver messages (one for each level) as well as a random mask mask.
- **Hash Function:** Given a key k and an input x, the hash function uses the ITDH sender algorithm on input x to perform an ITDH protocol execution "in its head." Specifically, for every level $\ell \in [L]$, it reads the corresponding receiver message in the key k and uses it to computes the hash value and the encoding for that level. By proceeding in a level-by-level fashion, it obtains the final level encoding e. It outputs $\mathsf{e} \oplus \mathsf{mask}$.

We now sketch the proof for correlation intractability. For simplicity, we first consider the case when $L = 1$. We then extend the proof strategy to the multi-level case.

For $L = 1$, the proof of correlation intractability resembles the proof in [15]. We first switch the simulated receiver message in the CIH key to a "real" message honestly computed using a circuit $C \in \mathcal{C}$. Now, suppose that the adversary finds an x such that $\mathsf{Hash}(\mathsf{k}, x) = C(x)$. Then by approximate correctness of ITDH, $C(x) \approx \mathsf{e} \oplus \mathsf{d}$, where the " $\approx$ " notation denotes closeness in Hamming distance. This implies that $\mathsf{e} \oplus \mathsf{d} \approx \mathsf{e} \oplus \mathsf{mask}$, and thus $\mathsf{d} \approx \mathsf{mask}$. However, once we fix the randomness used by the receiver, d *only depends on* h. Since h is compact, the value d is exponentially "sparse" in its range. Therefore, the probability that $\mathsf{d} \approx \mathsf{mask}$ is exponentially small, and thus such an input x exists with only negligible probability.

Let us now consider the multi-level case. Our starting idea is to switch the simulated receiver messages in the CIH key to "real" messages in a level-by-level manner. However, note that the honest receiver message at each level depends on the hash value sent by the sender in the previous level, and at the time of the key generation of the CIH, the sender's input has not been determined. Hence, it is not immediately clear how to compute the honest receiver message at each level without knowing the sender's input.

To get around this issue, at each level ℓ, we first simply *guess* the sender's hash value $\mathsf{h}_{\ell-1}$ in the previous level $(\ell - 1)$, and then switch the simulated receiver message in level ℓ to one computed honestly using the ITDH receiver algorithm on input $\mathsf{h}_{\ell-1}$. To ensure this guessing succeeds with high probability, we rely on the *compactness* of the hash values. Specifically, let λ_ℓ denote the security parameter for the ℓ^{th} level in ITDH (as mentioned earlier, we allow the security parameters for each level to be different). Then the guessing of the level $(\ell - 1)$ hash value succeeds with probability $2^{-\lambda_{\ell-1}}$. We set $\lambda_{\ell-1}$ to be sublinear in λ, where λ is the security parameter for CIH. Then, when we reach the final level, all our guesses are successful with probability $2^{-(\lambda_1+\lambda_2+\ldots+\lambda_L)}$, which is sub-exponential in λ. Since the probability of $\mathsf{d} \approx \mathsf{mask}$ can be exponentially small in λ, we can still get a contradiction.

However, the above argument assumes the function privacy is perfect, which is not the case. Indeed, at every level, we must also account for the adversary's distinguishing advantage when we switch a simulated message to a real message. In order to make the above argument go through, we need the distinguishing advantage to be a magnitude smaller than $2^{-\lambda_{\ell-1}}$ (for every ℓ). That is, we require ITDH to satisfy sub-exponential leveled functional privacy. Now, the distinguishing advantage can be bounded by $2^{-\lambda_\ell^c}$, where $0 < c < 1$ is a constant. Once we choose λ_ℓ large enough, then $2^{-\lambda_\ell^c}$ can be much smaller than $2^{-\lambda_{\ell-1}}$, and thus the above argument goes through as long as L is not too large.

In particular, there is room for trade-off between the number of levels in ITDH that we can collapse and the amount of leveled function privacy required. If we wish to rely on *polynomial time* and sub-exponential advantage assumptions, then the above transformation requires the number of levels to be *constant*. If we allow for *sub-exponential time* (and sub-exponential advantage) assumptions, then the above transformation can work for up to $O(\log\log\lambda)$ levels. We refer the reader to Section 6.4 for more details.

2.2 Constructing ITDH

We now provide an overview of our construction of constant-round ITDH for TC^0. Let *not-threshold* gate be a gate that computes a threshold gate and then outputs its negation. Since not-threshold gates are universal for threshold circuits, it suffices for our purpose to consider circuits that consist of only not-threshold gates.

The starting template for our construction consists of the following natural two-step approach reminiscent of classical secure computation protocols [34]:

- **Step 1 (Depth-1 Circuits):** First, we build an ITDH for a simple circuit family where each circuit is simply a *single* layer of layer of not-threshold gates.
- **Step 2 (Sequential Composition):** Next, to compute circuits with larger depth, we *sequentially compose* multiple instances of ITDH from the first step, where the output of the i^{th} ITDH is used as an input in the $(i+1)^{\text{th}}$ ITDH.

Input Passing. While natural, the above template doesn't work straight out of the box. Recall that the protocol output in any ITDH execution is "secret shared" between the sender and the receiver, where the sender holds the final level encoding e, and the receiver holds the decoding d. Then the first challenge in the sequential composition is how to continue the circuit computation when the result of the previous ITDH $\mathsf{e} \oplus \mathsf{d}$ is not known to the sender and the receiver.

A plausible way to resolve this challenge is for the receiver to simply send the decoding in the i^{th} ITDH to the sender so that the latter can compute the output, and then use it as input in the $(i+1)^{\text{th}}$ ITDH. However, this leaks intermediate wire values (of the TC^0 circuit that we wish to compute) to the sender, thereby compromising function privacy. Note that the reverse strategy of requiring the sender to send the encoding to the receiver (to allow output computation) also does not work since it violates the compactness requirement on the sender's messages to the receiver.

To overcome this challenge, we keep the secret-sharing structure of the output in every ITDH intact. Instead, we extend the functionality of ITDH for depth-1 threshold circuits so that the output of the i^{th} ITDH can be computed *within* the $(i+1)^{\text{th}}$ ITDH. Specifically, we first construct an ITDH for a circuit family $\mathcal{T}^{\oplus}$ where every circuit consists of a single layer of Xor-*then*-Not-Threshold gates. Such a gate first computes xor of its input with a vector pre-hardwired in the gate description, and then computes a not-threshold gate over the xor-ed value.

This allows for resolving the above problem as follows: the final-level encoding from the i^{th} ITDH constitutes the sender's input in the $(i+1)^{\text{th}}$ ITDH. On the other hand, the decoding in the i^{th} ITDH is used as the pre-hardwired string in the circuit computed by the $(i+1)^{\text{th}}$ ITDH.

ITDH for a single Xor-*then*-Not-Threshold Gate. We now describe the main ideas for computing a *single* Xor-*then*-Not-Threshold gate. Our ideas readily extend to the case where we want to compute a single layer of such gates.

To construct an ITDH for a single Xor-*then*-Not-Threshold gate, we only rely on trapdoor hash functions (TDH) for linear functions. Crucially, however, we use *interaction* to go beyond computing linear functions. At a high-level, we first "decompose" an Xor-*then*-Not-Threshold gate as the composition of two linear functions. We then use TDH for computing each of these linear functions separately. Finally, we "compose" the two TDH executions sequentially to obtain a 2-level ITDH for an Xor-*then*-Not-Threshold gate.

An observant reader may wonder how we decompose a Xor-*then*-Not-Threshold gate into computation of linear functions. Indeed, the composition of linear functions is still a linear function, while such a threshold

gate involves non-linear computation. As we will soon see, our decomposition strategy crucially relies on "offline" processing by the parties on the intermediate values between the two TDH executions. This introduces the desired non-linearity in the computation.

Let the vector $x \in \{0,1\}^n$ be the input to the Xor-*then*-Not-Threshold gate, $y \in \{0,1\}^n$ be the binary vector pre-hardwired in the gate description, and let t be the threshold. The Xor-*then*-Not-Threshold gate computes whether the number of 1's in $x \oplus y$ is smaller than t. To compute such a gate, we proceed in the following three simple steps:

- *Xor*: First, xor the input vector x with y, where y is part of the gate description.
- *Summation*: Second, sum the elements in the vector $x \oplus y$ over $\mathbb{Z}$.
- *Comparison*: Finally, compare the summation with the threshold t.

We now describe how to express each step as a linear function. For the first step, let x_i and y_i be two bits at (say) the i^{th} coordinate of x and y, respectively. Then $x_i \oplus y_i = 1$ if and only if $x_i = 0 \wedge y_i = 1$ or $x_i = 1 \wedge y_i = 0$. Hence, $x_i \oplus y_i = (1 - x_i) \cdot y_i + x_i \cdot (1 - y_i)$. Since y_i is part of the circuit description, the right hand side is a linear function of x_i over $\mathbb{Z}$.

In the second step, we simply sum over all the coordinates of $x \oplus y$. Since the summation is a linear function, and the first step is also linear, composing these two linear functions, we obtain a linear function of x over $\mathbb{Z}$. Then we can use a TDH for linear functions for this task. We note, however, that the construction of TDH in [29, 15] only works for linear functions over $\mathbb{Z}_2$. We therefore extend their construction to arbitrary polynomial modulus. In our case, since the summation cannot be more than n, it suffices to choose the modulo $(n+1)$.

We now proceed to express the comparison in the final step as a linear function. Suppose the summation value from the second step is $\mathsf{sum} \in \{0, 1, 2, \ldots, n\}$ and we want to compare it with a threshold t. Let $\mathbb{1}_{\mathsf{sum}}$ denote the indicator vector of sum, i.e., $\mathbb{1}_{\mathsf{sum}} = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$, where the $(\mathsf{sum}+1)^{\text{th}}$ coordinate is 1, and all other coordinates are 0. Then, we have that

$$\mathsf{sum} < t \iff \langle \mathbb{1}_{\mathsf{sum}}, \mathbb{1}_{<t} \rangle = 1,$$

where $\mathbb{1}_{<t} = (1, 1, \ldots, 1, 0, 0, \ldots, 0)$ is a vector with 1's on the first t-coordinates, and 0's on the remaining coordinates. We can therefore express the comparison in the final step as an inner product of $\mathbb{1}_{\mathsf{sum}}$ and $\mathbb{1}_{<t}$, which is a linear function of $\mathbb{1}_{\mathsf{sum}}$. This means that we can again use a TDH for linear functions for performing this computation.

Note, however, that the sender and the receiver do not directly obtain the summation value sum after the first TDH execution. Indeed, after the first TDH execution, the sender obtains an encoding e and the receiver obtains a decoding d such that $(\mathsf{e} + \mathsf{d}) \bmod R = \mathsf{sum}$. Thus, we need a mechanism to perform the final step even though neither party holds sum.[5]

Fortunately, we can still express the comparison $(\mathsf{e} + \mathsf{d}) \bmod R < t$ as

$$(\mathsf{e} + \mathsf{d}) \bmod R < t \Leftrightarrow \langle \mathbb{1}_{\mathsf{e}}, \mathbb{1}_{\mathsf{d},<t} \rangle = 1,$$

where $\mathbb{1}_{\mathsf{d},<t} = \sum_{j=0}^{t-1} \mathbb{1}_{(j-\mathsf{d}) \bmod R}$. The above equation follows from the fact that checking $(\mathsf{e} + \mathsf{d}) \bmod R < t$ is equivalent to check whether there exists a $j \in \{0, 1, \ldots, t-1\}$ such that $(\mathsf{e} + \mathsf{d}) \bmod R = j$, which is equivalent to checking $\mathsf{e} = (j - \mathsf{d}) \bmod R$. Note that by this equation, we express the comparison as a linear function of $\mathbb{1}_{\mathsf{e}}$ over $\mathbb{Z}_2$. Hence, the comparison in the final step can be computed by another TDH.

[5] For reasons as discussed earlier, the straightforward idea of simply requiring one of the two parties to send their secret share to the other party (for computing sum) does not work.

Between the two executions of TDH, the sender processes e from the first TDH to obtain $\mathbb{1}_{\mathsf{e}}$, and use it as the input to the second TDH. Similarly, the receiver processes d from the first TDH to obtain $\mathbb{1}_{\mathsf{d},<t} = \sum_{j=0}^{t-1} \mathbb{1}_{(j-\mathsf{d}) \bmod R}$, and use the linear function $\langle \cdot, \mathbb{1}_{\mathsf{d},<t} \rangle$ as the input to the second TDH. Note that this intermediate processing is non-linear, since computing the indicator vector can be done by several equality checks, and the equality check is not a linear function. Hence, it introduces the necessary non-linearity in the computation, but is done "outside" of the TDH execution.

Controlling the Error. We now discuss another issue that arises in the implementation of our template. Recall that an ITDH guarantees only approximate correctness, i.e., the xor of the final-level encoding e and decoding d is "close" (in terms of Hamming distance) to the true function output. Then, in a sequential composition of an ITDH protocol, *each* execution only guarantees approximate correctness. This means that the errors could *spread* across the executions, ultimately causing *every output bit of the final execution to be incorrect*. For example, suppose a coordinate of the output for an intermediate execution is flipped and later, the computation of every output bit depends on this flipped output bit. In this case, every output bit could be incorrect.

To overcome this issue, we observe that any circuit can be converted to a new circuit that satisfies a "parallel" structure demonstrated in Figure 1.

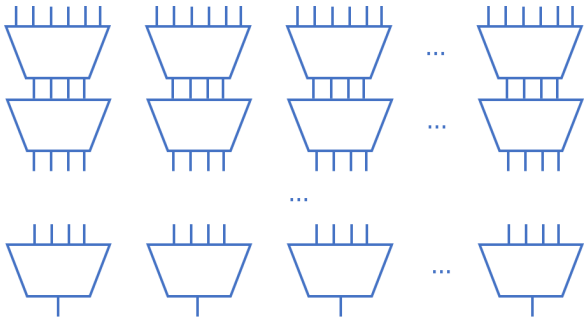


Figure 1: Parallel structure. The top (resp., bottom) layer corresponds to input (resp., output) wires.

In such circuits, each output bit only depends on the input to *one* parallel execution. Hence, the spreading of one Hamming error is controlled in one parallel execution. This allows us to prove approximate correctness of the sequential composition.

2.3 Constructing NIZKs

Armed with our construction of CIH, we now sketch the main ideas underlying our construction of (statistical) multi-theorem NIZK for NP. We proceed in the following two steps:

1. First, using CIH for TC^0, we construct a non-interactive witness indistinguishable (NIWI) argument for NP in the common random string model. Our construction satisfies either statistical WI and non-adaptive soundness, or computational WI and adaptive soundness.

2. We then transform the above NIWI into an adaptive, *multi-theorem* NIZK for NP in the common random string model via a variant of the Feige-Lapidot-Shamir (FLS) "OR-trick" [32].[6] Our NIZK satisfies either statistical ZK and non-adaptive soundness, or computational ZK and adaptive soundness. Crucially, unlike the classical FLS transformation, our transformation does *not* require "CRS switching" in

[6] By using "programmable" CIH, one could directly obtain NIZKs in the first step. However, the resulting NIZK only achieves *single-theorem* ZK; hence an additional step is still required to obtain multi-theorem NIZKs.

the security proof and hence works for both statistical and computational ZK cases seamlessly while preserving the distribution of the CRS in the underlying NIWI.

Statistical NIZKs. In the remainder of this section, we focus on the construction of *statistical* NIZKs. We briefly discuss the steps necessary for obtaining the computational variant (with adaptive soundness) at the end of the section.

Towards implementing the first of the above two steps, we first build the following two ingredients:

- A lossy public key encryption scheme with an additional property that we refer to as *low-depth decryption*, from DDH. Roughly speaking, this property requires that there exists a TC^0 circuit Dec that takes as input any ciphertext ct and a secret key sk, and outputs the correct plaintext.
- A trapdoor sigma protocol for NP with bad challenge function in TC^0 from the above lossy public key encrytion scheme. We also require the trapdoor sigma protocol to satisfy an additional "knowledge extraction" property, which can be viewed as an analogue of special soundness for trapdoor sigma protocols. Looking ahead, we use this property to construct NIWIs with argument of knowledge property, which in turn is required for our FLS variant for constructing NIZKs.

Lossy Public Key Encryption. The lossy public key encryption we use is essentially the same as in [46, 59, 3]. We start by briefly describing the scheme.

A public key $\mathsf{pk} = \begin{bmatrix} g^1 & g^b \\ g^a & g^c \end{bmatrix}$ is a matrix of elements in a group $\mathbb{G}$. When the matrix $\begin{bmatrix} 1 & b \\ a & c \end{bmatrix}$ is singular (i.e., $c = ab$), then the public key is in the "injective mode" and the secret key is $\mathsf{sk} = a$; when the matrix is non-singular (i.e., $c \neq ab$), then the public key is in the "lossy mode." The encryption algorithm is described as follows:

$$\mathsf{Enc}\left(\mathsf{pk}, m \in \{0,1\}; r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}\right) = \begin{bmatrix} (g^1)^{r_1} \cdot (g^b)^{r_2} \\ (g^a)^{r_1} \cdot (g^c)^{r_2} \cdot g^m \end{bmatrix} = g^{\begin{bmatrix} 1 & b \\ a & c \end{bmatrix}\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + \begin{bmatrix} 0 \\ m \end{bmatrix}}.$$

Let us now argue the low-depth decryption property. Let $[c_1, c_2]^T$ denote the ciphertext obtained by encrypting a message m using an injective mode public key pk with secret key $\mathsf{sk} = a$. To decrypt the ciphertext, we can compute $c_1^{-a} \cdot c_2 = g^m$ and then comparing with $1_{\mathbb{G}}$ to recover m. However, it is not known whether c_1^{-a} can be computed in TC^0 (recall that a depends on the security parameter).

In the following, we assume the DDH group is a subgroup of $\mathbb{Z}_q^*$, for some positive integer q. For the instantiation from Elliptic curves over $\mathbb{F}_p$ with a prime $p > 3$, see Appendix B for more details.

Towards achieving the low-depth decryption property, we use the following observation. Let $a_0, a_1, \ldots a_\lambda$ be the binary representation of a. Then, we have that

$$\left(c_1^{-2^0}\right)^{a_0} \cdot \left(c_1^{-2^1}\right)^{a_1} \cdot \left(c_1^{-2^2}\right)^{a_2} \cdot \ldots \cdot \left(c_1^{-2^\lambda}\right)^{a_\lambda} \cdot c_2 = g^m.$$

Note that given $[c_1, c_2]^T$, one can "precompute" $c_1^{-2^0}, c_1^{-2^1}, \ldots, c_1^{-2^\lambda}$ without using the secret key sk. In our application to NIZKs and Zaps, such pre-computation can be performed by the prover and the verifier.

We leverage this observation to slightly modify the definition of low-depth decryption to allow for a *deterministic* polynomial-time "pre-computation" algorithm PreComp. Specifically, we require that the output of $\mathsf{Dec}(\mathsf{PreComp}(1^\lambda, \mathsf{ct}), \mathsf{sk})$ is the correct plaintext m. We set $\mathsf{PreComp}(1^\lambda, c) = (c_1^{-2^0}, c_1^{-2^1}, \ldots, c_1^{-2^\lambda}, c_2)$, and allow the circuit Dec to receive $c_1^{-2^0}, c_1^{-2^1}, \ldots, c_1^{-2^\lambda}, c_2$ and $a_0, a_1, \ldots, a_\lambda$ as input. The decryption circuit Dec proceeds in the following steps:

- For each $i = 0, 1, \ldots, \lambda$, it chooses g_i to be either $1_{\mathbb{G}}$ or $c_1^{-2^i}$, such that $g_i = (c_1^{-2^i})^{a_i}$. This computation can be done in constant depth, and is hence in TC^0.
- Multiply the values $g_0, g_2, \ldots, g_\lambda$ and c_2. From [60], this iterative multiplication can be computed in TC^0 when we instantiate $\mathbb{G}$ as a subgroup of $\mathbb{Z}_q^*$.
- Compare the resulting value with $1_{\mathbb{G}}$. If they are equal, then output 0. Otherwise output 1.

Since each of the above steps can be computed in TC^0, we have that Dec is also in TC^0.

Trapdoor Sigma Protocol for NP. Recently, Brakerski et al. [15] constructed a "commit-and-open" style trapdoor sigma protocol where the only cryptographic primitive used is a commitment scheme. Crucially, the bad challenge function for their protocol involves the following two computations: extraction from the commitment, and a post-extraction verification using 3-CNF. By exploiting the specific form of their bad challenge function, we construct a trapdoor sigma protocol for NP with our desired properties by simply instantiating the commitment scheme in their protocol with the above lossy encryption scheme.

Let us analyze the bad challenge function of the resulting trapdoor sigma protocol. Since our lossy public key encryption satisfies the low-depth decryption property, the first step of the bad challenge computation can be done in TC^0. Next, note that the second step of the bad challenge computation is also in TC^0 since it involves evaluation of 3-CNF which can be computed in AC^0. Thus, the bad challenge function is in TC^0.

We observe that our protocol also satisfies a *knowledge extraction* property which requires that one can efficiently extract a witness from a *single* accepting transcript (α, β, γ) by using a trapdoor (namely, the secret key of the lossy public key encryption), if β does not equal to the output of the bad challenge function evaluated on α. We use this property to construct NIWIs with argument of knowledge property.

NIWI from Fiat-Shamir via CIH. We construct NIWI arguments in the CRS model by using CIH to collapse the rounds of our trapdoor sigma protocol repeated λ times in parallel. The CRS of the resulting construction contains a public-key of lossy public key encryption scheme from above and a CIH key. When the public key is in lossy mode, the NIWI achieves statistical WI property and non-adaptive argument of knowledge property.

To prove the argument of knowledge property, we observe that for any accepting transcript $(\{\alpha_i\}_{i\in[\lambda]}, \{\beta_i\}_{i\in[\lambda]}, \{\gamma_i\}_{i\in[\lambda]})$, it follows from correlation intractability of the CIH that $\{\beta_i\}_{i\in[\lambda]}$ is *not* equal to the outputs of the bad challenge function evaluated on $\{\alpha_i\}_{i\in[\lambda]}$. Hence, there exists at least one index i^* such that β_{i^*} is not equal to the output of the bad challenge function on α_{i^*}. We can now extract a witness by relying on the knowledge extraction property of the i^*-th parallel execution of the trapdoor sigma protocol.

From NIWI to Multi-theorem NIZK. The FLS "OR-trick" [32] is a standard methodology to transform NIWIs (or single-theorem NIZKs) into multi-theorem NIZKs. Roughly speaking, the trick involves supplementing the CRS with an instance (say) y of a hard-on-average decision problem and requiring the prover to prove that either the "original" instance (say) x *or* y is true. This methodology involves switching the CRS either in the proof of soundness or zero-knowledge, which can potentially result in a degradation of security. E.g., in the former case, one may end up with non-adaptive (computational) soundness while in the latter case, one may end up with computational ZK even if the underlying scheme achieves statistical privacy. The instance y also needs to be chosen carefully depending on the desired security and whether one wants the resulting CRS to be a reference string or a random string.

We consider a variant of the "OR-trick" that does not require CRS switching and preserves the distribution of the CRS of the underlying scheme. We supplement the CRS with an instance of hard-on-average *search*

problem, where the instance is subjected to the *uniform* distribution, and can be sampled *together with* a witness. For our purposes, the discrete logarithm problem suffices. To sample the instance uniformly at random together with a witness, we firstly sample a secret exponent, and then set the instance as the exponent raised to a group generator. The ZK simulator simply uses the secret exponent of the discrete-log instance in the CRS to simulate the proof. On the other hand, soundness can be argued by relying on the computational hardness of the discrete-log problem. One caveat of this transformation is that the proof of soundness requires the underlying NIWI to satisfy argument of knowledge property. We, note, however, that this property is usually easy to achieve (in the CRS model). Using this approach, we obtain statistical multi-theorem NIZK arguments in the common *random* string model from sub-exponential DDH. Previously, group-based statistical NIZKs were known only in the common reference string model [39, 39].

We remark that the above idea can be easily generalized to other settings. For example, starting from LWE-based single-theorem statistical NIZKs [58], one can embed the Shortest Integer Solution (SIS) problem in the CRS to build *multi-theorem* statistical NIZKs in the common random string model. This settles an open question stated in the work of [58].

Computational NIZKs with Adaptive Soundness. Using essentially the same approach as described above, we can also construct computational NIZKs for NP with adaptive soundness. The main difference is that instead of using lossy public-key encryption scheme in the construction of trapdoor sigma protocols, we use ElGamal encryption scheme [31]. Using the same ideas as for our lossy public-key encryption scheme, we observe that the ElGamal encryption scheme also satisfies *low-depth decryption* property. This allows us to follow the same sequence of steps as described above to obtain a computational NIZK for NP with adaptive soundness in the common *random* string model.[7]

2.4 Constructing Zaps

At a high-level, we follow a similar recipe as in the recent works of [2, 37] who construct statistical Zap arguments from quasi-polynomial LWE.

The main idea in these works is to replace the (non-interactive) commitment scheme in a trapdoor sigma protocol with a *two-round* statistical-hiding commitment scheme in the plain model and then collapse the rounds of the resulting protocol using CIH, as in the case of NIZKs. Crucially, unlike the non-interactive commitment scheme that only allows for extraction in the CRS model, the two-round commitment scheme must support extraction in the plain model. The key idea for achieving such an extraction property (in conjunction with statistical-hiding property) is to allow for successful extraction with only negligible but still much larger than sub-exponential probability (for example, $2^{-\log^2 \lambda}$) [42]. By carefully using complexity leveraging, one can prove soundness of the resulting argument system.

Statistical-Hiding Commitment with Low-depth Extraction. We implement this approach by replacing the lossy public-key encryption scheme in our NIWI construction (from earlier) with a two-round statistical hiding commitment scheme. Since we need the bad challenge function of the sigma protocol to be in TC^0, we require the commitment scheme to satisfy an additional *low-depth extraction* property.

To construct such a scheme, we first observe that the construction of (public-coin) statistical-hiding extractable commitments in [44, 42, 2, 37] only makes black-box use of a two-round oblivious transfer (OT) scheme. We instantiate this generic construction via the Naor-Pinkas OT scheme based on DDH [53]. By exploiting the specific structure of the generic construction as well as the fact that Naor-Pinkas OT decryption

[7] We note that one could obtain computational NIZKs with adaptive soundness by simply "switching the CRS" in our construction of statistical NIZKs. However, the resulting scheme in this case is in the common *reference* string model.

can be computed in TC^0, we are able to show that the extraction process can also be performed in TC^0. We refer the reader to Section 8 for more details.

3 Preliminaries

For any positive integer $N \in \mathbb{Z}, N > 0$, denote $[N] = \{1, 2, \ldots, N\}$. For any integer $R > 0$, and $x \in \mathbb{Z}_R$, $0 \leq x < R$, the indicator vector $\mathbb{1}_x$ of x is a vector in $\{0,1\}^R$, where the $(x+1)^{\text{th}}$ position is 1, and all other coordinates are zero. A binary relation $\mathcal{R}$ is a subset of $\{0,1\}^* \times \{0,1\}^*$.

Statistical Distance. For any two discrete distributions P, Q, the statistical distance between P and Q is defined as $\mathsf{SD}(P,Q) = \sum_i \left|\Pr[P=i] - \Pr[Q=i]\right|/2$ where i takes all the values in the support of P and Q.

Hamming Distance. Let n be an integer, and S be a set, and $x = (x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ be two tuples in S^n, the Hamming distance $\mathsf{Ham}(x,y)$ is defined as $\mathsf{Ham}(x,y) = \left|\{i \mid x_i \neq y_i\}\right|$.

Threshold Gate. Let $x_1, x_2, \ldots, x_n$ be n binary variables. A threshold gate is defined as the following function:

$$\mathsf{Th}_t(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \sum_{i \in [n]} x_i \geq t \\ 0 & \text{Otherwise} \end{cases}$$

Not-Threshold Gate. A not-threshold gate $\overline{\mathsf{Th}}_t$ is the negation of a threshold gate.

Threshold Circuits and TC^0. A threshold circuit is a directed acyclic graph, where each node either computes a threshold gate of unbounded fan-in or a negation gate.

In this work, for any constant L, we use TC^0_L to denote the class of L-depth polynomial-size threshold circuits. When the depth L is not important or is clear from the context, we omit it and simply denote the circuit class TC^0_L as TC^0. The not-threshold gate is universal for TC^0, since we can convert any threshold circuit of constant depth to a constant depth circuit that only contains not-threshold gates. The conversion works as follows: for each negation gate, we convert it to a not-threshold gate with a single input and threshold $t = 1$. For each threshold gate, we convert it to a not-threshold gate with the same input and threshold and then compose it with a negation gate, where the negation gate can be implemented as a not-threshold gate.

3.1 Number-Theoretic Assumptions

Discrete Logarithm Assumption. In the following, we state the discrete logarithm (DL) assumption.

Definition 3.1 (Discrete Logarithm)**.** *A prime-order group generator is an algorithm $\mathcal{G}$ that takes the security parameter λ as input, and outputs a tuple $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group of prime order $p(\lambda)$, and g is a generator of $\mathbb{G}$. We say that the DL problem is hard in $\mathcal{G}$, if for any n.u. PPT adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that,*

$$\Pr\left[(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda), h \leftarrow \mathbb{G}, x \leftarrow \mathcal{A}(1^\lambda, \mathbb{G}, p, g, h) : g^x = h\right] \leq \nu(\lambda).$$

We say that the DL is sub-exponentially hard in $\mathcal{G}$, if there exists a constant $0 < c < 1$ such that for any n.u. PPT adversary, the success probability is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

Decisional Diffie-Hellman Assumption. In the following, we state the decisional Diffie-Hellman (DDH) assumption.

Definition 3.2 (Decisional Diffie-Hellman)**.** *Let $\mathcal{G}$ be a prime-order group generator (as in Definition 3.1). We say that $\mathcal{G}$ satisfies the DDH assumption if for any n.u. PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\lambda)$ such that*

$$\Big|\Pr\left[(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda), a, b \leftarrow \mathbb{Z}_p : \mathcal{D}(1^\lambda, \mathbb{G}, p, g, g^a, g^b, g^{ab}) = 1\right] - \Pr\left[(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda), a, b, c \leftarrow \mathbb{Z}_p : \mathcal{D}(1^\lambda, \mathbb{G}, p, g, g^a, g^b, g^c) = 1\right]\Big| \leq \nu(\lambda)$$

We say that $\mathcal{G}$ satisfies the sub-exponential DDH assumption, if there exists a constant $0 < c < 1$ such that for any n.u. PPT distinguisher, the advantage $\nu(\lambda)$ is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

3.2 Non-Interactive Argument Systems

We recall the syntax and security properties associated with non-interactive argument systems in the common random string (CRS) model.

A non-interactive argument system for an NP language $\mathcal{L}$ with associated relation $\mathcal{R}$ is a tuple of algorithms $\Pi = (\mathsf{CGen}, \mathsf{P}, \mathsf{V})$ described as follows.

- $\mathsf{CGen}(1^\lambda)$: It takes as input the security parameter λ, and outputs a common random string crs.
- $\mathsf{P}(\mathsf{crs}, x, w)$: It takes as input a common random string crs, an instance $x \in \mathcal{L}$, a witness ω, and outputs a proof π.
- $\mathsf{V}(\mathsf{crs}, x, \pi)$: It takes as input a common random string crs, an instance x, a proof π, and decides to accept (output 1) or reject (output 0) the proof.

We now define various properties of non-interactive proof systems that we consider in this work.

- **Completeness:** For any instance $x \in \mathcal{L}$, and any witness ω of x, we have

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda), \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, \omega) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] = 1.$$

- **Computational Soundness:** For any n.u. PPT cheating prover P^*, there exists a negligible function $\nu(\lambda)$ such that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda), (x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}) : x \notin \mathcal{L} \wedge \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] \leq \nu(\lambda).$$

  We say that the proof system achieves sub-exponential computational soundness, if there exists a constant $0 < c < 1$ such that for any n.u. PPT cheating prover, the success probability is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

  We refer to the above as **adaptive** soundness. If we modify the above definition s.t. the adversary chooses the statement x before obtaining the CRS, then the resulting notion is referred to as **non-adaptive** soundness.

- **Argument of Knowledge:** There exists a PPT extractor $E = (E_1, E_2)$ such that, for any n.u. PPT prover P^*, there exists a negligible function $\nu(\lambda)$ such that

$$\Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), (\widetilde{\mathsf{crs}}, \mathsf{td}) \leftarrow E_1(1^\lambda), \pi \leftarrow \mathsf{P}^*(\widetilde{\mathsf{crs}}), \omega \leftarrow E_2(\mathsf{td}, x, \pi) : \mathcal{R}(x, \omega) = 1\right] \geq \Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), \mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] - \nu(\lambda).$$

  We say that the proof system achieves sub-exponential non-adaptive argument of knowledge, if there exists a constant $0 < c < 1$ such that for any n.u. PPT cheating prover, $\nu(\lambda)$ is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

  We refer to the above as **non-adaptive** argument of knowledge. If we modify the above definition such that the adversary chooses the statement x after obtaining the CRS, and $\mathsf{SD}(\mathsf{crs}, \widetilde{\mathsf{crs}}) \leq \nu(\lambda)$, then the resulting notion is referred to as **adaptive** argument of knowledge. (Note that this definition is slightly stronger in the sense that we require the CRS output by CGen and E_1 to be statistically close.)

- **Adaptive Statistical Witness Indistinguishability (SWI):** For any unbounded adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that

$$|\Pr[\mathsf{Expr}_0 = 1] - \Pr[\mathsf{Expr}_1 = 1]| \leq \nu(\lambda),$$

  where Expr_b, for every $b \in \{0, 1\}$, is defined as the following experiment:

  Experiment Expr_b**:**

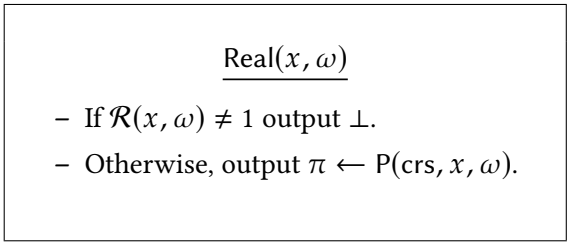
  - $\mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda)$.
  - $(x, \omega_0, \omega_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs})$.
  - If $\mathcal{R}(x, \omega_0) \neq 1$ or $\mathcal{R}(x, \omega_1) \neq 1$, output 0 and halt.
  - $\pi \leftarrow \mathsf{P}(\mathsf{crs}, x, \omega_b)$.
  - Output $\mathcal{A}(\mathsf{crs}, \pi)$.

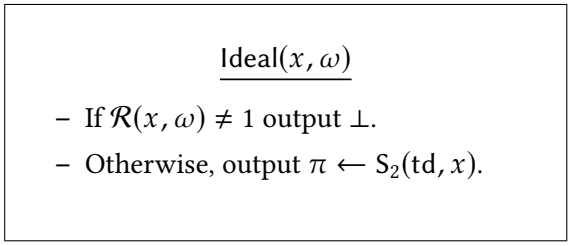
  We say that the proof system satisfies adaptive **computational** witness indistinguishability if the above condition holds for any non-uniform PPT adversary.

- **Adaptive Statistical Zero Knowledge (SZK):** There exists a simulator $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$ such that for any unbounded adversary $\mathcal{A}$ and polynomial $Q(\lambda)$, there exists a negligible function $\nu(\lambda)$ such that

$$\left|\Pr\left[\mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda) : \mathcal{A}^{\mathsf{Real}(\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1\right] - \Pr\left[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{S}_1(1^\lambda) : \mathcal{A}^{\mathsf{Ideal}(\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1\right]\right| \leq \nu(\lambda)$$

  where the oracles $\mathsf{Real}(\cdot,\cdot)$ and $\mathsf{Ideal}(\cdot,\cdot)$ are defined as follows, and $\mathcal{A}$ makes at most $Q(\lambda)$ queries to each oracle.





We say that the proof system satisfies adaptive **computational** zero knowledge if the above condition holds for any non-uniform PPT adversary.

3.3 Statistical Zap Arguments

Zaps [30] are two-round witness indistinguishable proof systems with a public-coin verifier message. Below, we define statistical Zap arguments, i.e., Zaps that achieve statistical WI property and computational soundness.

A statistical Zap argument for an NP language L is a two-round protocol (P, V) with a public-coin verifier message that satisfies the following properties:

- **Completeness:** For every $x \in L$ and witness ω for x, we have that

$$\Pr\left[\mathsf{Out}_{\mathsf{V}}\left(\mathsf{P}(1^\lambda, x, \omega) \leftrightarrow \mathsf{V}(1^\lambda, x)\right) = 1\right] = 1$$

  where $\mathsf{Out}_{\mathsf{V}}(e)$ is the output of V in a protocol execution e.

- **Computational Soundness:** For any non-uniform PPT prover P^*, there exists a negligible function $\nu(\cdot)$ such that for any $x \notin L$, we have that

$$\Pr\left[\mathsf{Out}_{\mathsf{V}}\left(\mathsf{P}^*(1^\lambda, x) \leftrightarrow \mathsf{V}(1^\lambda, x)\right) = 1\right] < \nu(\lambda)$$

  The above is referred to as **non-adaptive** soundness. If we modify the above definition s.t. the adversary chooses the statement x after receiving the verifier's message, then the resulting notion is referred to as **adaptive** soundness.

- **Statistical Witness Indistinguishability:** For any unbounded verifier V^*, there exists a negligible function $\nu(\cdot)$ such that for every $x \in L$, and witnesses ω_1, ω_2 for x, we have that

$$\mathsf{SD}\left(\mathsf{Trans}\left(\mathsf{P}(1^\lambda, x, \omega_1) \leftrightarrow \mathsf{V}^*(1^\lambda, x)\right), \mathsf{Trans}\left(\mathsf{P}(1^\lambda, x, \omega_2) \leftrightarrow \mathsf{V}^*(1^\lambda, x)\right)\right) < \nu(\lambda)$$

  where $\mathsf{Trans}(e)$ is the transcript of a protocol execution e.

3.4 Two-Round Oblivious Transfer

Definition 3.3. *A statistical sender-private oblivious transfer (OT) is a tuple of algorithms* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$*:*

$\mathsf{OT}_1(1^\lambda, b)$**:** *On input security parameter* λ*, a bit* $b \in \{0,1\}$*,* OT_1 *outputs the first round message* ot_1 *and a state* st*.*

$\mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, m_0, m_1)$**:** *On input security parameter* λ*, a first round message* ot_1*, two bits* $m_0, m_1 \in \{0,1\}$*,* OT_2 *outputs the second round message* ot_2*.*

$\mathsf{OT}_3(1^\lambda, \mathsf{ot}_2, \mathsf{st})$**:** *On input security parameter* λ*, the second round message* ot_2*, and the state generated by* OT_1*,* OT_3 *outputs a message* m*.*

We require the following properties:

Correctness *For any* $b, m_0, m_1 \in \{0,1\}$*,*

$$\Pr[(\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OT}_1(1^\lambda, b), \mathsf{ot}_2 \leftarrow \mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, m_0, m_1), m \leftarrow \mathsf{OT}_3(1^\lambda, \mathsf{ot}_2, \mathsf{st}) : m = m_b] = 1$$

Statistical Sender Privacy *There exists a negligible function $\nu(\lambda)$ and an deterministic exponential time extractor* OTExt *such that for any (potential maliciously generated)* ot_1, $\mathsf{OTExt}(1^\lambda, \mathsf{ot}_1)$ *outputs a bit* $b \in \{0,1\}$. *Then for any* $m_0, m_1 \in \{0,1\}$, *we have*

$$\mathsf{SD}\left(\mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, m_0, m_1), \mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, m_b, m_b)\right) \leq \nu(\lambda)$$

Pseudorandom Receiver's Message *For any* $b \in \{0,1\}$, *let* ot_1 *be the first round message generated by* $\mathsf{OT}_1(1^\lambda, b)$. *For any n.u. PPT adversary* $\mathcal{D}$, *there exists a negligible function* $\nu(\lambda)$ *such that, for any* $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\mathcal{D}(1^\lambda, \mathsf{ot}_1) = 1\right] - \Pr\left[u \leftarrow \{0,1\}^{|\mathsf{ot}_1|} : \mathcal{D}(1^\lambda, u) = 1\right]\right| \leq \nu(\lambda)$$

Furthermore, we say that the OT satisfies the sub-exponential pseudorandom receiver's message property, if there exists a constant $0 < c < 1$ *such that for any n.u. PPT adversary, the advantage* $\nu(\lambda)$ *is bounded by* $2^{-\lambda^c}$ *for any sufficiently large* λ.

Lemma 3.4. *Assuming DDH, there exists a two-round oblivious transfer.*

A two-round oblivious transfer from DDH was constructed by [53]. Their construction satisfies correctness and statistical sender-privacy. Further, the receiver's message in their scheme is (sub-exponentially) pseudorandom, assuming (sub-exponential) DDH.

3.5 Rate-1 Trapdoor Hash Functions

We recall the notion of (rate-1) trapdoor hash functions (TDH) introduced in [29]. For our constructions, we require TDH with an "enhanced correctness" property, as defined in [15].

Previously, [29] constructed TDH for index predicates. Their construction was later generalized by [15] to linear functions and constant degree polynomials over $\mathbb{Z}_2$. In this work, we consider a further generalized family of functions, namely, linear functions over $\mathbb{Z}_R$, where R is a polynomial in the security parameter.

Definition 3.5 (Linear Function Family)**.** $\mathcal{F} = \{\mathcal{F}_{n,R}\}_{n,R}$ *is a family of linear functions over* $\mathbb{Z}_R$ *if every* $f \in \mathcal{F}_{n,R}$ *is of the form:*

$$f(x_1, x_2, \ldots, x_n) = (a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n) \bmod R,$$

where $0 \leq a_i < R$ *for every* $i \in \{0, \ldots, n\}$.

Definition. A trapdoor hash function for $\mathcal{F}$ is a tuple of algorithms $\mathsf{TDH} = (\mathsf{HKGen}, \mathsf{EKGen}, \mathsf{Hash}, \mathsf{Enc}, \mathsf{Dec})$ described as follows:

- $\mathsf{HKGen}(1^\lambda, 1^n, 1^R)$: The hash key generation algorithm takes as input a security parameter λ, input length n, and a modulo R. It outputs a hash key hk.
- $\mathsf{EKGen}(\mathsf{hk}, f)$: The encoding key generation algorithm takes as input a hash key hk and a circuit $f \in \mathcal{F}_n$, and it outputs an encoding key ek together with a trapdoor td.
- $\mathsf{Hash}(\mathsf{hk}, x)$: The hashing algorithm takes as input a hash key hk and an input value x, and it outputs a hash value $\mathsf{h} \in \{0,1\}^\eta$.

- $\mathsf{Enc}(\mathsf{ek}, x)$: The encoding algorithm takes as input an encoding key ek and an input $x \in \{0,1\}^n$, and it outputs an encoding $\mathsf{e} \in \mathbb{Z}_R$.
- $\mathsf{Dec}(\mathsf{td}, h)$: The decoding algorithm takes as input a trapdoor td and the hash value h, and it outputs a value $\mathsf{d} \in \mathbb{Z}_R$.

We require TDH to satisfy the following properties:

- **Compactness:** The bit-length η of a hash value h is *independent* of n, and is a fixed polynomial in λ. For simplicity, in this work, we require that $\eta \leq \lambda$.
- **τ-Enhanced Correctness:** For any $\lambda, n, R \in \mathbb{N}$, any string $\mathsf{h} \in \{0,1\}^{\eta(\lambda)}$, any $f \in \mathcal{F}_{n,R}$, and any hk output by $\mathsf{HKGen}(1^\lambda, 1^n, 1^R)$, we have

$$\Pr[(\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{EKGen}(\mathsf{hk}, f) : \forall x \text{ s.t. } \mathsf{Hash}(\mathsf{hk}, x) = \mathsf{h}, f(x) = (\mathsf{e} + \mathsf{d}) \bmod R] \geq 1 - \tau(\lambda),$$

  where $\mathsf{e} = \mathsf{Enc}(\mathsf{ek}, x)$, $\mathsf{d} = \mathsf{Dec}(\mathsf{td}, \mathsf{h})$, and the probability is over the randomness of EKGen.
- **Function Privacy:** There exists a simulator Sim and a negligible function $\nu(\lambda)$ such that, for any polynomials n and R in the security parameter λ, there exists a constant $c < 1$ such that for any $\lambda \in \mathbb{N}$, any function $f \in \mathcal{F}_{n,R}$, and any n.u. PPT adversary $\mathcal{D}$,

$$\Big|\Pr\left[\mathsf{hk} \leftarrow \mathsf{HKGen}(1^\lambda, 1^n, 1^R), (\mathsf{ek}, \mathsf{td}) \leftarrow \mathsf{EKGen}(\mathsf{hk}, f) : \mathcal{D}(1^\lambda, (\mathsf{hk}, \mathsf{ek})) = 1\right] -$$
$$\Pr\left[\mathsf{hk} \leftarrow \mathsf{HKGen}(1^\lambda, 1^n, 1^R), \widetilde{\mathsf{ek}} \leftarrow \mathsf{Sim}(1^\lambda, 1^n, 1^R) : \mathcal{D}(1^\lambda, (\mathsf{hk}, \widetilde{\mathsf{ek}})) = 1\right]\Big| \leq \nu(\lambda)$$

  We say that the TDH achieves sub-exponential function privacy, if there exists a constant $0 < c < 1$ such that for any n.u. PPT adversary, the advantage $\nu(\lambda)$ is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

Theorem 3.6. *Assuming sub-exponential DDH, for any inverse polynomial τ in the security parameter λ, there exists a* TDH *construction for the linear function family $\mathcal{F} = \{\mathcal{F}_{n,R}\}_{n,R}$ with $\tau(\lambda)$-enhanced correctness and sub-exponential function privacy.*

The proof of this theorem follows via a simple modification of the TDH construction in [29]. For completeness, we prove it in Appendix A.

4 Interactive Trapdoor Hashing Protocols

In this section, we define interactive trapdoor hashing protocols (ITDH). At a high-level, ITDH is a generalization of trapdoor hash functions – which can be viewed as two-round two-party protocols with specific structural and communication efficiency properties – to multi-round protocols.

More specifically, an interactive trapdoor hashing protocol involves two parties – a sender and a receiver. The sender has an input x, while the receiver has a circuit f. The two parties jointly compute $f(x)$ over several rounds of interaction. We structure the protocols in multiple *levels*, where a level consists of the following two successive rounds:

- The receiver generates a key k and a trapdoor td using a key generation algorithm KGen, which takes as input the circuit f, the level number, and some additional internal state of the receiver. Then it sends k to the sender.

- Upon receiving a key k, the sender computes a hash value h and an encoding e using the algorithm Hash&Enc, which takes as input x, the key k, the level number, and the previous level encoding. Then it sends the hash h to the receiver, and keeps e as an internal state.

Finally, there is a decoding algorithm Dec that takes the internal state of the receiver after the last level as input, and outputs a decoding value d. Ideally, we want the output $f(x)$ to be $\mathsf{e} \oplus \mathsf{d}$.

In the following, we proceed to formally define this notion and its properties.

Per-level Security Parameter. In our formal definition of ITDH, we allow the security parameter to be *different* for every level. This formulation is guided by our main application, namely, constructing correlation-intractable hash functions (see Section 6). Nevertheless, we note that ITDH could also be meaningfully defined w.r.t. a single security parameter for the entire protocol.

4.1 Definition

Let $C = \{C_{n,u}\}_{n,u}$ be a family of circuits, where each circuit $f \in C_{n,u}$ is a circuit of input length n and output length u. An L-level interactive trapdoor hashing protocol for the circuit family C is a tuple of algorithms $\mathsf{ITDH} = (\mathsf{KGen}, \mathsf{Hash\&Enc}, \mathsf{Dec})$ that are described below.

We use $\lambda_1, \ldots, \lambda_L$ to denote the security parameters for different levels. Throughout this work, these parameters are set so that they are polynomially related. That is, there exists a λ such that $\lambda_1, \ldots, \lambda_L$ are polynomials in λ.

- $\mathsf{KGen}(1^{\lambda_\ell}, \ell, f, \mathsf{h}_{\ell-1}, \mathsf{td}_{\ell-1})$: The key generation algorithm takes as input a security parameter λ_ℓ (that varies with the level number), a level number ℓ, a circuit $f \in C_{n,u}$, a level $(\ell-1)$ hash value $\mathsf{h}_{\ell-1}$ and trapdoor $\mathsf{td}_{\ell-1}$ (for $\ell = 1$, $\mathsf{h}_{\ell-1} = \mathsf{td}_{\ell-1} = \perp$). It outputs an ℓ^{th} level key k_ℓ and a trapdoor td_ℓ.
- $\mathsf{Hash\&Enc}(\mathsf{k}_\ell, x, \mathsf{e}_{\ell-1})$: The hash-and-encode algorithm takes as input a level ℓ hash key k_ℓ, an input x, and a level $(\ell-1)$ encoding $\mathsf{e}_{\ell-1}$. It outputs an ℓ^{th} level hash value h_ℓ and an encoding $\mathsf{e}_\ell \in \{0,1\}^u$. When $\ell = 1$, we let $\mathsf{e}_{\ell-1} = \perp$.
- $\mathsf{Dec}(\mathsf{td}_L, \mathsf{h}_L)$: The decoding algorithm takes as input a level L trapdoor td_L and hash value h_L, and outputs a value $\mathsf{d} \in \{0,1\}^u$.

We require ITDH to satisfy the following properties:

- **Compactness:** For each level $\ell \in [L]$, the bit length of h_ℓ is at most λ_ℓ.
- (Δ, ϵ)-**Approximate Correctness:** For any $n, u \in \mathbb{N}$, any circuit $f \in C_{n,u}$ and any sequence of security parameters $(\lambda_1, \ldots, \lambda_L)$, we have

$$\Pr_{r_1, r_2, \ldots, r_L}\left[\forall x \in \{0,1\}^n, \mathsf{Ham}(\mathsf{e} \oplus \mathsf{d}, f(x)) < \Delta(u)\right] > 1 - \epsilon(u, \lambda_1, \ldots, \lambda_L),$$

  where e, d are obtained by the following procedure: Let $\mathsf{h}_0 = \mathsf{td}_0 = \mathsf{e}_0 = \perp$. For $\ell = 1, 2, \ldots, L$,

  - Compute $(\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{KGen}(1^{\lambda_\ell}, \ell, f, \mathsf{h}_{\ell-1}, \mathsf{td}_{\ell-1}; r_\ell)$ using random coins r_ℓ.
  - Hash and encode the input x: $(\mathsf{h}_\ell, \mathsf{e}_\ell) \leftarrow \mathsf{Hash\&Enc}(\mathsf{k}_\ell, x, \mathsf{e}_{\ell-1})$.

  Finally, let $\mathsf{e} = \mathsf{e}_L$ be the encoding at the final level, and $\mathsf{d} = \mathsf{Dec}(\mathsf{td}_L, \mathsf{h}_L)$.

- **Leveled Function Privacy:** There exist a simulator Sim and a negligible function $\nu(\cdot)$ such that for any level $\ell \in [L]$, any polynomials $n(\cdot)$ and $u(\cdot)$ in the security parameter, any circuit $f \in \mathcal{C}_{n,u}$, any trapdoor $\mathsf{td}' \in \{0,1\}^{|\mathsf{td}_{\ell-1}|}$, any hash value $\mathsf{h}' \in \{0,1\}^{|\mathsf{h}_{\ell-1}|}$, and any n.u. PPT distinguisher $\mathcal{D}$,

$$\left|\Pr\left[(\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{KGen}(1^{\lambda_\ell}, \ell, f, \mathsf{h}', \mathsf{td}') : \mathcal{D}(1^{\lambda_\ell}, \mathsf{k}_\ell) = 1\right] - \Pr\left[\widetilde{\mathsf{k}}_\ell \leftarrow \mathsf{Sim}(1^{\lambda_\ell}, 1^n, 1^u, \ell) : \mathcal{D}(1^{\lambda_\ell}, \widetilde{\mathsf{k}}_\ell) = 1\right]\right| \leq \nu(\lambda_\ell).$$

  We say that the ITDH satisfies sub-exponential leveled function privacy, if there exists a constant $0 < c < 1$ such that for any n.u. PPT distinguisher, $\nu(\lambda_\ell)$ is bounded by $2^{-\lambda_\ell^c}$ for any sufficiently large λ_ℓ.

  Note that since the security parameters for different levels are polynomially related, $n(\cdot)$ and $u(\cdot)$ are polynomials in λ_ℓ iff they are polynomials in λ.

Relationship with Trapdoor Hash Functions. A 1-level ITDH is essentially the same as TDH, except that in TDH, there are two kinds of keys: a hash key and an encoding key(see Section 3.5) . In particular, a hash value is computed using the hash key and can be reused with different encoding keys for different functions. In 1-level ITDH, however, the receiver's message only consists of one key that is used by the sender for computing both the hash value and the encoding. Therefore, the hash value is not reusable for different functions.

We choose the above formulation of ITDH for the sake of a simpler and cleaner definition that suffices for our applications. If we consider multi-bit output functions, then the above difference disappears, since we can combine multiple functions into one multi-bit output function and encode it using one key.

5 Construction of ITDH

In this section, we construct an interactive trapdoor hashing protocol (ITDH) for TC^0 circuits. We refer the reader to Section 2 for a high-level overview of our approach. The remainder of this section is organized as follows:

- **Depth-1 Circuits:** In Section 5.1, we first construct a 2-level ITDH protocol for $\mathcal{T}^{\oplus}$ – roughly speaking, a family of depth-1 Xor-*then*-Not-Threshold circuits (see below for the precise definition of $\mathcal{T}^{\oplus}$).
- **Sequential Composition:** Next, in Section 5.4, we present a sequential composition theorem for ITDH where we show how to compose L instances of a 2-level ITDH for some circuit family to obtain a $2L$-level ITDH for a related circuit family.
- **Construction for TC^0:** Finally, in Section 5.7, we put these two constructions together to obtain an ITDH for TC^0.

5.1 ITDH for $\mathcal{T}^{\oplus}$

We start by introducing some notation and definitions.

XOR-*then*-Compute Circuits. Let $C = \{C_{n,u}\}_{n,u}$ be a circuit family, where for any n and u, $C_{n,u}$ contains circuits with n-bit inputs and u-bit outputs. For any C, we define an Xor-*then*-Compute circuit family $C^{\oplus} =$

$\{C^{\oplus}_{n,u}\}_{n,u}$ consisting of circuits that *first* compute a bit-wise xor operation on the input with a fixed string and *then* compute a circuit in C on the resulting value.

Specifically, $C^{\oplus}_{n,u}$ contains all the circuit $\mathrm{C}^{\oplus y} : \{0,1\}^n \to \{0,1\}^u$, where $y \in \{0,1\}^n$ and there exists a $C \in C_{n,u}$ such that for every $x \in \{0,1\}^n$,

$$\mathrm{C}^{\oplus y}(x) = C(x \oplus y).$$

Circuit Families $\mathcal{T}$ and $\mathcal{T}^{\oplus}$. We define a circuit family $\mathcal{T} = \{\mathcal{T}_{n,u}\}_{n,u}$ consisting of depth-1 not-threshold circuits, i.e., a single layer of not-threshold gates (see Section 3). Specifically, $\mathcal{T}_{n,u}$ contains all circuits $\mathrm{T}_{\vec{t},\vec{I}} : \{0,1\}^n \to \{0,1\}^u$ where $\vec{t} = \{t_1, \ldots, t_u\}$ is a set of positive integers, and $\vec{I} = \{I_1, \ldots, I_u\}$ is a collection of sets $I_j \subseteq [n]$ s.t. for any $x \in \{0,1\}^n$,

$$\mathrm{T}_{\vec{t},\vec{I}}(x) = \left(\overline{\mathsf{Th}}_{t_1}(x[I_1]), \ldots, \overline{\mathsf{Th}}_{t_u}(x[I_u])\right),$$

where for any index set $I_j = \{i_1, i_2, \ldots, i_w\} \subseteq [n]$, we denote $x[I_j] = (x_{i_1}, x_{i_2}, \ldots, x_{i_w})$ as the projection of string x to the set I_j.

The function family $\mathcal{T}^{\oplus} = \{\mathcal{T}^{\oplus}_{n,u}\}_{n,u}$ is defined as the Xor-*then*-Compute family corresponding to $\mathcal{T}$. We denote the circuits in $\mathcal{T}^{\oplus}_{n,u}$ as $\mathrm{T}^{\oplus y}_{\vec{t},\vec{I}}$, where $\vec{t}, \vec{I}$ and y are as defined above.

For a high-level overview of our construction, see Section 2.2. We now proceed to give a formal description of our construction.

Construction of ITDH for $\mathcal{T}^{\oplus}$. We construct a 2-level interactive trapdoor hashing protocol $\mathsf{ITDH} = (\mathsf{KGen}, \mathsf{Hash\&Enc}, \mathsf{Dec})$ for the circuit family $\mathcal{T}^{\oplus}$ as defined above. Our construction relies on the following ingredient: a trapdoor hash function $\mathsf{TDH} = (\mathsf{TDH.HKGen}, \mathsf{TDH.EKGen}, \mathsf{TDH.Hash}, \mathsf{TDH.Enc}, \mathsf{TDH.Dec})$ for the linear function family $\mathcal{F} = \{\mathcal{F}_{n,R}\}_{n,R}$ (see Definition 3.5) that achieves τ-enhanced correctness and function privacy.

For ease of exposition, we describe the algorithms of ITDH *separately* for each level. The first level algorithms of ITDH internally use TDH to evaluate a circuit (defined below) with input length $n_1 = n$ and modulus $R_1 = n + 1$. The second level algorithms of ITDH internally use TDH to evaluate another circuit (defined below) with input length $n_2 = R_1 \cdot u$ and modulus $R_2 = 2$. We use λ_1 and λ_2 to denote the security parameters input to the first and second level algorithms, respectively.

- **Level 1** $\mathsf{KGen}(1^{\lambda_1}, 1, \mathrm{T}^{\oplus y}_{\vec{t},\vec{I}}, \mathsf{h}_0 = \bot, \mathsf{td}_0 = \bot)$**:**
  - Sample a hash key of TDH w.r.t. security parameter λ_1, input length $n_1 = n$ and modulus $R_1 = n + 1$
    $$\mathsf{hk}_1 \leftarrow \mathsf{TDH.HKGen}(1^{\lambda_1}, 1^{n_1 = n}, 1^{R_1 = n+1})$$
  - Parse $\vec{I} = \{I_1, \ldots, I_u\}$. For every $i \in [u]$, sample an encoding key:
    $$(\mathsf{ek}_{1,i}, \mathsf{td}_{1,i}) \leftarrow \mathsf{TDH.EKGen}(\mathsf{hk}_1, \mathsf{XorSum}_{I_i,y})$$
    where for any set $I \subseteq [n]$, $\mathsf{XorSum}_{I,y}$ is the linear function described in Figure 2.
  - Output $(\mathsf{k}_1, \mathsf{td}_1)$ where $\mathsf{k}_1 = (1, \mathsf{hk}_1, \{\mathsf{ek}_{1,i}\}_{i\in[u]})$ and $\mathsf{td}_1 = \{\mathsf{td}_{1,i}\}_{i\in[u]}$.
- **Level 1** $\mathsf{Hash\&Enc}(\mathsf{k}_1, x, \mathsf{e}_0 = \bot)$**:**
  - Parse $\mathsf{k}_1 = (1, \mathsf{hk}_1, \{\mathsf{ek}_{1,i}\}_{i\in[u]})$.

Linear Function $\mathsf{XorSum}_{I,y}(x_1, \ldots, x_n)$ over $\mathbb{Z}_{R_1}$

- Let $y = (y_1, y_2, \ldots, y_n)$.
- Compute and output $\sum_{i \in I} x_i \cdot (1 - y_i) + (1 - x_i) \cdot y_i$.

Figure 2: Description of the linear function $\mathsf{XorSum}_{I,y}$. This function computes the sum over $\mathbb{Z}_{R_1}$ of I values obtained by bit-wise XOR of $y[I]$ and $x[I]$, where $x = (x_1, \ldots, x_n)$.

  - Compute "first level" hash over x: $\mathsf{h}_1 \leftarrow \mathsf{TDH.Hash}(\mathsf{hk}_1, x)$
  - For every $i \in [u]$, compute a "first level" encoding: $\mathsf{e}_{1,i} \leftarrow \mathsf{TDH.Enc}(\mathsf{ek}_{1,i}, x)$
  - Output $(\mathsf{h}_1, \mathsf{e}_1)$, where $\mathsf{e}_1 = \{\mathsf{e}_{1,i}\}_{i \in [u]}$.

- **Level 2** $\mathsf{KGen}(1^{\lambda_2}, 2, \mathrm{T}^{\oplus y}_{\vec{t},\vec{I}}, \mathsf{h}_1, \mathsf{td}_1)$**:**
  - Parse $\mathsf{td}_1 = \{\mathsf{td}_{1,i}\}_{i \in [u]}$. For every $i \in [u]$, decode h_1: $\mathsf{d}_{1,i} \leftarrow \mathsf{TDH.Dec}(\mathsf{td}_{1,i}, \mathsf{h}_1)$
  - Sample a new hash key of TDH w.r.t. security parameter λ_2, input length $n_2 = R_1 \cdot u$ and modulus $R_2 = 2$,
$$\mathsf{hk}_2 \leftarrow \mathsf{TDH.HKGen}(1^{\lambda_2}, 1^{n_2 = R_1 \cdot u}, 1^{R_2 = 2}).$$
  - Parse $\vec{t} = \{t_1, \ldots, t_u\}$. For each $i \in [u]$, sample a new encoding key
$$(\mathsf{ek}_{2,i}, \mathsf{td}_{2,i}) \leftarrow \mathsf{TDH.EKGen}(\mathsf{hk}_2, \mathsf{AddTh}_{i,t_i,\mathsf{d}_{1,i}}),$$
    where for any index $i \in [u]$, positive integer t and value $\mathsf{d} \in \mathbb{Z}_{R_1}$, $\mathsf{AddTh}_{i,t,\mathsf{d}}$ is the linear function defined in the Figure 3.
  - Output $(\mathsf{k}_2, \mathsf{td}_2)$, where $\mathsf{k}_2 = (2, \mathsf{hk}_2, \{\mathsf{ek}_{2,i}\}_{i \in [u]})$ and $\mathsf{td}_2 = \{\mathsf{td}_{2,i}\}_{i \in [u]}$.

- **Level 2** $\mathsf{Hash\&Enc}(\mathsf{k}_2, x, \mathsf{e}_1)$**:**
  - Parse $\mathsf{k}_2 = (2, \mathsf{hk}_2, \{\mathsf{ek}_{2,i}\}_{i \in [u]})$, and $\mathsf{e}_1 = \{\mathsf{e}_{1,i}\}_{i \in [u]}$.
  - Compute "second level" hash over $\{\mathbb{1}_{\mathsf{e}_{1,i}}\}_{i \in [u]}$, where $\mathbb{1}_{\mathsf{e}_{1,i}}$ is the indicator vector for $\mathsf{e}_{1,i}$.
$$\mathsf{h}_2 \leftarrow \mathsf{TDH.Hash}(\mathsf{hk}_2, \{\mathbb{1}_{\mathsf{e}_{1,i}}\}_{i \in [u]})$$
  - For any $i \in [u]$, compute "second level" encoding: $\mathsf{e}_{2,i} \leftarrow \mathsf{TDH.Enc}(\mathsf{ek}_{2,i}, \{\mathbb{1}_{\mathsf{e}_{1,j}}\}_{j \in [u]})$.
  - Output $(\mathsf{h}_2, \mathsf{e}_2)$, where $\mathsf{e}_2 = \{\mathsf{e}_{2,i}\}_{i \in [u]}$.

- **Decoding** $\mathsf{Dec}(\mathsf{td}_2, \mathsf{h}_2)$**:**
  - Parse $\mathsf{td}_2 = \{\mathsf{td}_{2,i}\}_{i \in [u]}$. For every $i \in [u]$, decode h_2: $\mathsf{d}_{2,i} \leftarrow \mathsf{TDH.Dec}(\mathsf{td}_{2,i}, \mathsf{h}_2)$.
  - Output $\mathsf{d} = \{\mathsf{d}_{2,i}\}_{i \in [u]}$.

This completes the description of ITDH. We prove that it achieves approximate correctness and leveled function privacy in Lemmas 5.1 and 5.2, respectively.

Linear Function $\mathsf{AddTh}_{i,t,\mathsf{d}}(\vec{\mathbb{e}})$ over $\mathbb{Z}_2$

- Let $\vec{\mathbb{e}} = (\mathbb{e}_1, \ldots, \mathbb{e}_u)$, where $\mathbb{e}_j \in \{0,1\}^{R_1}$ for every $j \in [u]$.
- Compute and output the inner product: $\langle \mathbb{e}_i, \mathbb{f} \rangle \bmod 2$, where $\mathbb{f} = \sum_{j=0}^{t-1} \mathbb{1}_{(j-\mathsf{d}) \bmod R_1}$ is the sum of indicator vectors for $(j - \mathsf{d}) \bmod R_1$, for $0 \le j < t$.

Figure 3: Description of the linear function $\mathsf{AddTh}_{i,t,\mathsf{d}}$. For any $\mathsf{e}_1, \mathsf{e}_2, \ldots, \mathsf{e}_u \in \mathbb{Z}_{R_1}$, this function computes whether $(\mathsf{e}_i + \mathsf{d}) \bmod R_1$ is less than the threshold t. The actual input $\vec{\mathbb{e}}$ to the function is such that $\mathbb{e}_i$ is the indicator vector for e_i.

5.2 Proof of Approximate Correctness

Lemma 5.1 (Approximate Correctness)**.** *For any function $\Delta(u)$, the proposed protocol* ITDH *satisfies (Δ, ϵ)-approximate correctness, where*

$$\epsilon = 2\,(2e \cdot \max\{\tau(\lambda_1), \tau(\lambda_2)\} \cdot u/\Delta)^{\Delta/2} \cdot 2^{\lambda_1+\lambda_2}$$

u is the output length of the circuit, and e is the base for natural logarithms.

Proof. We first establish some notation that we shall use throughout the proof. Whenever necessary, we augment a variable with $*$ in the superscript to denote the "ideal" value of the variable, whereas the "real" – and possibly *erroneous* – value is denoted without any emphasis.

Level 1. For each $i \in [u]$, let $\mathsf{sum}_i = (\mathsf{e}_{1,i} + \mathsf{d}_{1,i}) \bmod R_1$. By the τ-enhanced correctness of TDH, for any fixed hk_1, fixed h_1 and index i, we have

$$\Pr\Big[(\mathsf{ek}_{1,i}, \mathsf{td}_{1,i}) \leftarrow \mathsf{TDH.EKGen}(\mathsf{hk}_1, \mathsf{XorSum}_{I_i,y}) : \forall x : \mathsf{h}_1 = \mathsf{TDH.Hash}(\mathsf{hk}_1, x), \mathsf{sum}_i = \mathsf{XorSum}_{I_i,y}(x)\Big] > 1 - \tau(\lambda_1)$$

Denote $\mathsf{sum}_i^* = \mathsf{XorSum}_{I_i,y}(x)$. Then, for any fixed hk_1, and fixed h_1, since the encoding keys $\{\mathsf{ek}_{1,i}\}_{i\in[u]}$ are sampled independently, for any $\Delta' \in [u]$, we have

$$\Pr_{\{\mathsf{ek}_{1,i}\}_{i\in[u]}}\left[\exists x : \mathsf{h}_1 = \mathsf{TDH.Hash}(\mathsf{hk}_1, x), \mathsf{Ham}(\{\mathsf{sum}_i\}_{i\in[u]}, \{\mathsf{sum}_i^*\}_{i\in[u]}) > \Delta'\right] < \tau(\lambda_1)^{\Delta'}\binom{u}{\Delta'} \le \left(\frac{e \cdot \tau(\lambda_1) \cdot u}{\Delta'}\right)^{\Delta'},$$

where the second inequality follows from the upper bound for the combinatorial coefficients $\binom{u}{\Delta'} < (e \cdot u/\Delta')^{\Delta'}$.

Next, by taking the union bound on the choice of h_1, we have that for any fixed hk_1,

$$\Pr_{\{\mathsf{ek}_{1,i}\}_{i\in[u]}}\left[\exists x, \mathsf{Ham}\left(\{\mathsf{sum}_i\}_{i\in[u]}, \{\mathsf{sum}_i^*\}_{i\in[u]}\right) > \Delta'\right] < \left(\frac{e \cdot \tau(\lambda_1) \cdot u}{\Delta'}\right)^{\Delta'} \cdot 2^{\lambda_1}.$$

Finally, by averaging over all possible choices of hk_1, the above bound still holds when hk_1 is sampled from HKGen.

Level 2. Similarly, in the second level, let's consider two *arbitrary* encoding $\{\mathsf{e}'_{1,i}\}_{i\in[u]}$ and decoding $\{\mathsf{d}'_{1,i}\}_{i\in[u]}$. We will wire them with $\{\mathsf{e}_{1,i}\}_{i\in[u]}$ and $\{\mathsf{d}_{1,i}\}_{i\in[u]}$ in the first level later. Then we have

$$\Pr_{\mathsf{hk}_2,\{\mathsf{ek}_{2,i}\}_{i\in[u]}}\left[\exists\{\mathsf{e}'_{1,i}\}_{i\in[u]}, \mathsf{Ham}\left(\{\mathsf{out}_i\}_{i\in[u]}, \{\mathsf{out}^*_i\}_{i\in[u]}\right) > \Delta'\right] < \left(\frac{e\cdot\tau(\lambda_2)\cdot u}{\Delta'}\right)^{\Delta'}\cdot 2^{\lambda_2}, \tag{1}$$

where $\mathsf{out}^*_i = \mathsf{AddTh}_{i,t_i,\mathsf{d}'_{1,i}}(\{\mathbb{1}_{\mathsf{e}'_{1,j}}\}_{j\in[u]})$, and out_i is obtained by the following procedure.

- $\forall i\in[u], (\mathsf{ek}_{2,i},\mathsf{td}_{2,i}) \leftarrow \mathsf{TDH.EKGen}(\mathsf{hk}_2, \mathsf{AddTh}_{i,t_i,\mathsf{d}'_{1,i}})$.
- $\mathsf{h}_2 \leftarrow \mathsf{TDH.Hash}(\mathsf{hk}_2, \{\mathbb{1}_{\mathsf{e}'_{1,i}}\}_{i\in[u]}), \forall i\in[u], \mathsf{e}_{2,i} \leftarrow \mathsf{TDH.Enc}(\mathsf{ek}_{2,i}, \{\mathbb{1}_{\mathsf{e}'_{1,j}}\}_{j\in[u]})$.
- $\forall i\in[u], \mathsf{d}_{2,i} \leftarrow \mathsf{TDH.Dec}(\mathsf{td}_{2,i}, \mathsf{h}_2)$.
- $\forall i\in[u], \mathsf{out}_i = (\mathsf{e}_{2,i}+\mathsf{d}_{2,i}) \bmod 2$.

Now, we fix the random coins r_1 used by the first level key generation algorithm. Let hk_1 and $\{\mathsf{ek}_{1,i},\mathsf{td}_{1,i}\}_{i\in[u]}$ be the values generated by the first level key generation algorithm using randomness r_1. Let $\mathsf{e}_{1,i}$ be the first level encodings computed by the sender using the keys $\mathsf{ek}_{1,i}$ and input x. Let out_i and out^*_i be as defined above, except that they are computed w.r.t. $\mathsf{e}_{1,i}$.

Then from Equation 1, for any fixed r_1 and fixed d'_1, if we let $\{\mathsf{e}'_{1,i}\}_{i\in[u]} = \{\mathsf{e}_{1,i}\}_{i\in[u]}$, we have

$$\Pr_{\mathsf{hk}_2,\{\mathsf{ek}_{2,i}\}_{i\in[u]}}\left[\exists x : \mathsf{d}'_1 = \{\mathsf{d}_{1,i}\}_{i\in[u]}, \mathsf{Ham}\left(\{\mathsf{out}_i\}_{i\in[u]}, \{\mathsf{out}^*_i\}_{i\in[u]}\right) > \Delta'\right] < \left(\frac{e\cdot\tau(\lambda_2)\cdot u}{\Delta'}\right)^{\Delta'}\cdot 2^{\lambda_2},$$

where $\mathsf{d}_{1,i} = \mathsf{TDH.Dec}(\mathsf{td}_{1,i}, \mathsf{h}_1)$, and we only consider every x such that $\{\mathsf{d}_{1,i}\}_{i\in[u]}$ derived from r_1 and x is equal to d'_1. To further remove such a constraint on x, we need to take an union bound on all possible choices of $\{\mathsf{d}_{1,i}\}_{i\in[u]}$.

Since $\mathsf{td}_{1,i}$ is fixed by r_1, the total possibilities of $\{\mathsf{d}_{1,i}\}_{i\in[u]}$ is bounded by the number of possible choices of h_1, which is at most 2^{λ_1}. Hence, applying the union bound on d'_1, we derive that for any fixed r_1,

$$\Pr_{\mathsf{hk}_2,\{\mathsf{ek}_{2,i}\}_{i\in[u]}}\left[\exists x, \mathsf{Ham}\left(\{\mathsf{out}_i\}_{i\in[u]}, \{\mathsf{out}^*_i\}_{i\in[u]}\right) > \Delta'\right] < \left(\frac{e\cdot\tau(\lambda_2)\cdot u}{\Delta'}\right)^{\Delta'}\cdot 2^{\lambda_1+\lambda_2}$$

By averaging over all possible choices of r_1, this bound still holds when r_1 is sampled uniformly at random.

Putting it all together. From the above, except $(e\max\{\tau(\lambda_1),\tau(\lambda_2)\}\cdot u/\Delta')^{\Delta'}\cdot(2^{\lambda_1}+2^{\lambda_1+\lambda_2})$ fraction of the random coins, we have

$$\forall x, \mathsf{Ham}\left(\{\mathsf{sum}_i\}_{i\in[u]}, \{\mathsf{sum}^*_i\}_{i\in[u]}\right) \le \Delta' \text{ and } \mathsf{Ham}\left(\{\mathsf{out}_i\}_{i\in[u]}, \{\mathsf{out}^*_i\}_{i\in[u]}\right) \le \Delta'.$$

From the triangle inequality of Hamming distance, we have

$$\mathsf{Ham}\left(\{\mathsf{out}_i\}_{i\in[u]}, f(x)\right) \le \mathsf{Ham}\left(\{\mathsf{out}_i\}_{i\in[u]}, \{\mathsf{out}^*_i\}_{i\in[u]}\right) + \mathsf{Ham}\left(\{\mathsf{out}^*_i\}_{i\in[u]}, f(x)\right)$$

On the right hand side, the first term is bounded by Δ'. For the second term, out^*_i in fact only depends on sum_i. This is because $\mathsf{out}^*_i = \mathsf{AddTh}_{i,t_i,\mathsf{d}_{1,i}}(\{\mathbb{1}_{\mathsf{e}_{1,j}}\}_{j\in[u]})$. Then by construction, AddTh outputs 1 if and only if $(\mathsf{e}_{1,i}+\mathsf{d}_{1,i}) \bmod R_1 < t_i$, which is equivalent to $\mathsf{sum}_i < t_i$.

Since $\mathsf{Ham}(\{\mathsf{sum}_i\}_{i\in[u]}, \{\mathsf{sum}^*_i\}_{i\in[u]}) \le \Delta'$, we know that there are at most Δ' Hamming errors in $\{\mathsf{sum}_i\}_{i\in[u]}$, and these errors lead to at most Δ' Hamming errors in $\{\mathsf{out}^*_i\}_{i\in[u]}$. Therefore, we obtain that $\mathsf{Ham}\left(\{\mathsf{out}^*_i\}_{i\in[u]}, f(x)\right) \le \Delta'$.

Hence, we have

$$\Pr_{r_1,\mathsf{hk}_2,\{\mathsf{ek}_{2,i}\}_{i\in[u]}}[\exists x, \mathsf{Ham}(\{\mathsf{out}_i\}_{i\in[u]}, f(x)) > 2\Delta'] < (e\cdot\max\{\tau(\lambda_1),\tau(\lambda_2)\}\cdot u/\Delta')^{\Delta'}\cdot(2^{\lambda_1}+2^{\lambda_1+\lambda_2})$$
$$< 2\,(e\cdot\max\{\tau(\lambda_1),\tau(\lambda_2)\}\cdot u/\Delta')^{\Delta'}\cdot 2^{\lambda_1+\lambda_2}$$

By letting $\Delta' = \Delta(u)/2$, we finish the proof. □

5.3 Proof of Leveled Function Privacy

Lemma 5.2 (Leveled Function Privacy)**.** *The proposed protocol* ITDH *satisfies leveled function privacy property.*

Proof. We build the simulator $\mathsf{Sim}(1^{\lambda_\ell}, 1^n, 1^u, \ell)$ in Figure 4.

Simulator $\mathsf{Sim}(1^{\lambda_\ell}, 1^n, 1^u, \ell)$

- If $\ell = 1$, let $n_\ell = n, R_\ell = R_1 = n$. Otherwise let $n_\ell = R_1 \cdot u = n \cdot u, R_\ell = 2$.
- Sample a hash key: $\mathsf{hk}_\ell \leftarrow \mathsf{TDH.HKGen}(1^{\lambda_\ell}, 1^{n_\ell}, 1^{R_\ell})$
- For each $i \in [u]$, compute a simulated encoding key:
$$\mathsf{ek}_{\ell,i} \leftarrow \mathsf{TDH.Sim}(1^{\lambda_\ell}, 1^{n_\ell}, 1^{R_\ell})$$
- Output $\mathsf{k}_\ell = (\ell, \mathsf{hk}_\ell, \{\mathsf{ek}_{\ell,i}\}_{i\in[u]})$.

Figure 4: Simulator $\mathsf{Sim}(1^\lambda, 1^n, 1^u, \ell)$

We construct a series of hybrids to prove that the output of Sim is indistinguishable from an honestly sampled key for any circuit $\mathrm{T}^{\oplus y}_{\vec{t},\vec{I}} \in \mathcal{T}^{\oplus}_{n,u}$. We only prove indistinguishability for the first level, or $\ell = 1$. The proof for the second level ($\ell = 2$) follows similarly.

Hyb_0: This is the "real world", where the keys are computed using the key generation algorithm $\mathsf{KGen}(1^{\lambda_1}, 1, \mathrm{T}^{\oplus y}_{\vec{t},\vec{I}}, h_0 = \bot, \mathsf{td}_0 = \bot)$.

$\mathsf{Hyb}^{i^*}_1$: In this hybrid, for every $i < i^*$, the encoding key $\mathsf{ek}_{1,i}$ is computed using the simulator TDH.Sim. For every $i \ge i^*$, the encoding key $\mathsf{ek}_{1,i}$ is computed honestly using TDH.EKGen.

- For each $i < i^*$, let $\mathsf{ek}_{1,i} \leftarrow \mathsf{TDH.Sim}(1^{\lambda_1}, 1^{n_1}, 1^{R_1})$.
- For each $i \ge i^*$, let $(\mathsf{ek}_{1,i}, \mathsf{td}_{1,i}) \leftarrow \mathsf{TDH.EKGen}(\mathsf{hk}_1, \mathsf{XorSum}_{I_i,y})$.

Hyb_2: This hybrid is the same as the simulator $\mathsf{Sim}(1^{\lambda_1}, 1^n, 1^u, 1)$.

From the description of the hybrids, it follows that Hyb_0 is identical to Hyb^1_1, and Hyb_2 is identical to Hyb^{u+1}_1. Hence, it suffices to show that $\mathsf{Hyb}^{i^*}_1$ and $\mathsf{Hyb}^{i^*+1}_1$ are indistinguishable. For $i^* \in [u]$, suppose that there exists a n.u. PPT distinguisher $\mathcal{D}$ that distinguishes between $\mathsf{Hyb}^{i^*}_1$ and $\mathsf{Hyb}^{i^*+1}_1$ with non-negligible advantage $\delta(\lambda_1)$. We build a distinguisher $\mathcal{D}'$ who breaks the function privacy of TDH.

The distinguisher $\mathcal{D}'(1^{\lambda_1}, (\mathsf{hk}, \mathsf{ek}))$ takes as input the security parameter λ_1, a hash key hk and an encoding key ek. For each $i < i^*$, it runs the simulator $\mathsf{TDH.Sim}(1^{\lambda_1}, 1^{n_1}, 1^{R_1})$ to generate an encoding key $\mathsf{ek}_{1,i}$. For $i = i^*$, it sets $\mathsf{ek}_{1,i} = \mathsf{ek}$ as the encoding key. For each $i > i^*$, it computes an encoding key $(\mathsf{ek}_{1,i}, \mathsf{td}_{1,i}) \leftarrow \mathsf{TDH.EKGen}(\mathsf{hk}, \mathsf{XorSum}_{I_i,y})$ using the key generation algorithm. Finally, $\mathcal{D}'$ runs distinguisher $\mathcal{D}$ with input $(1, \mathsf{hk}, \{\mathsf{ek}_{1,i}\}_{i\in[u]})$, and returns the output of $\mathcal{D}$.

Now, if ek in $(\mathsf{hk}, \mathsf{ek})$ is generated using $\mathsf{TDH.EKGen}$, then $\mathcal{D}'$ successfully simulates the hybrid $\mathsf{Hyb}_1^{i^*}$ for the distinguisher $\mathcal{D}'$. Hence,

$$\Pr\left[\mathcal{D}(1^{\lambda_1}, \mathsf{Hyb}_1^{i^*}) = 1\right] = \Pr\left[\begin{smallmatrix}\mathsf{hk}\leftarrow\mathsf{TDH.HKGen}(1^{\lambda_1},1^{n_1},1^{R_1}),\\ (\mathsf{ek},\mathsf{td})\leftarrow\mathsf{TDH.EKGen}(\mathsf{hk},\mathsf{XorSum}_{I_{i^*},y})\end{smallmatrix} : \mathcal{D}'(1^{\lambda_1}, (\mathsf{hk}, \mathsf{ek})) = 1\right]$$

On the other hand, if ek is generated using $\mathsf{TDH.Sim}$, then $\mathcal{D}'$ successfully simulates the hybrid $\mathsf{Hyb}_1^{i^*+1}$ for the distinguisher $\mathcal{D}'$. Hence,

$$\Pr\left[\mathcal{D}(1^{\lambda_1}, \mathsf{Hyb}_1^{i^*+1}) = 1\right] = \Pr\left[\begin{smallmatrix}\mathsf{hk}\leftarrow\mathsf{TDH.HKGen}(1^{\lambda_1},1^{n_1},1^{R_1}),\\ \mathsf{ek}\leftarrow\mathsf{TDH.Sim}(1^{\lambda_1},1^{n_1},1^{R_1})\end{smallmatrix} : \mathcal{D}'(1^{\lambda_1}, (\mathsf{hk}, \mathsf{ek})) = 1\right]$$

Since TDH achieves function privacy, the difference of the probabilities on the right hand side of the above two equations is bounded by a negligible function. Hence, there exists a negligible function $\nu(\cdot)$ such that $|\Pr[\mathcal{D}'(1^{\lambda_1}, \mathsf{Hyb}_1^{i^*}) = 1] - \Pr[\mathcal{D}'(1^{\lambda_1}, \mathsf{Hyb}_1^{i^*+1}) = 1]| \leq \nu(\lambda_1)$.

Since we have u hybrids, and u is polynomial in λ, we conclude that the construction satisfies leveled function privacy. □

Remark 5.3. *If the underlying* TDH *satisfies sub-exponential leveled function privacy, then the proposed construction of* ITDH *also satisfies the sub-exponential leveled function privacy.*

5.4 ITDH Composition

In this section, we establish a sequential composition theorem for ITDH. Roughly speaking, we show how a 2-level ITDH for an "Xor-*then*-Compute" circuit family can be executed sequentially L times to obtain an ITDH for a related circuit family (the exact transformation is more nuanced; see below). The main benefit of sequential composition is that it can be used to increase the depth of circuits that can be computed by ITDH.

We start by introducing some notation and terminology for circuit composition that we shall use in the sequel.

Parallel Composition. Let w be a positive integer. Informally, a w-parallel composition circuit f is a structured circuit that computes w circuits $f'_1, f'_2, \ldots, f'_w$ in parallel. More formally, for any circuit family C, we define a corresponding parallel-composition circuit family as follows:

Definition 5.4 (Parallel Composition)**.** *For any circuit family C and any polynomial $w = w(n)$, we say that $C[\overrightarrow{w}] = \{C[\overrightarrow{w}]_{n,u}\}_{n,u}$ is a family of w-parallel composition circuits if for every $f \in C[\overrightarrow{w}]_{n,u}$, there exists a sequence of circuits $f'_1, f'_2, \ldots, f'_w \in C_{n',u'}$ such that $n = n' \cdot w(n)$ and $u = u' \cdot w(n)$, and for any input $x = (x_1, x_2, \ldots, x_w) \in \{0,1\}^{n=n'\cdot w}$ (where every $x_i \in \{0,1\}^{n'}$), we have*

$$f(x_1, x_2, \ldots, x_w) = (f'_1(x_1), f'_2(x_2), \ldots, f'_w(x_w)).$$

Parallel-and-Sequential-Composition. For any circuit family C, we now define another circuit family obtained via parallel *and* sequential composition of circuits in C.

Informally speaking, for any polynomials $w(n)$ and $L(n)$ and an integer s, a w-parallel-and-L-sequential-composition of a circuit family C is a new circuit family $C[\overrightarrow{w}_{\downarrow L}] = \{C[\overrightarrow{w}_{\downarrow L}]_{n,s}\}_{n,s}$, where each circuit $f \in C[\overrightarrow{w}_{\downarrow L}]_{n,s}$

is computed by a sequence of circuits $f_1, f_2, \ldots, f_L$. For any input x, to compute $f(x)$, we firstly evaluate f_1 on input x, then use the output $f_1(x)$ as the input to the circuit f_2, and so on, such that the output of f_L is the output of f. Furthermore, we require that for every $\ell \in [L]$, f_ℓ is an m-parallel composition of some sequence of circuits $f'_{\ell,1}, f'_{\ell,2}, \ldots, f'_{\ell,w} \in C$. For the ease of presentation, we fix the output length of the circuit f_ℓ for every $\ell < L$ as s, and the output length of f as w.

Definition 5.5 (Parallel-and-Sequential-Composition)**.** *Let $C = \{C_{n,u}\}_{n,u}$ be a circuit family, where each circuit in $C_{n,u}$ has input length n and output length u. For any polynomials $w = w(n), L = L(n)$, and integer s, we say that $C[\overrightarrow{\downarrow_L w}] = \{C[\overrightarrow{\downarrow_L w}]_{n,s}\}_{n,s}$ is a family of w-*parallel-and-L-sequential-composition circuits *if every circuit $f \in C[\overrightarrow{\downarrow_L w}]_{n,s}$ is of the form*

$$f = f_L \circ f_{L-1} \circ \ldots \circ f_1$$

where for every $\ell \in [L]$, $f_\ell : \{0,1\}^{n_\ell} \to \{0,1\}^{n_{\ell+1}}$ satisfies $n_1 = n, n_2 = n_3 = \ldots = n_{L-1} = s, n_L = w$. Furthermore, there exists a sequence of integers $\{n'_\ell\}_\ell$ and circuits $\{f'_{\ell,j}\}_{\ell\in[L],j\in[w]}$, where $f'_{\ell,j} \in C_{n'_\ell,n'_{\ell+1}}$, and $n_\ell = n'_\ell \cdot w$,

$$f_\ell(x_1, \ldots, x_w) = \left(f'_{\ell,1}(x_1), f'_{\ell,2}(x_2), \ldots, f'_{\ell,w}(x_w)\right)$$

for every $x = (x_1, \ldots, x_w) \in \{0,1\}^{n'_\ell \cdot w}$, where $x_i \in \{0,1\}^{n'_\ell}$ for every $i \in [w]$.

Construction of ITDH for $C[\overrightarrow{\downarrow_L w}]$. Let $C = \{C_{n,u}\}_{n,u}$ be any circuit family, and let $C[\overrightarrow{w}]$ be the corresponding w-parallel composition circuit family. Let $C[\overrightarrow{w}]^\oplus = \{C[\overrightarrow{w}]^\oplus_{n,u}\}_{n,u}$ be the "Xor-*then*-Compute" circuit family defined w.r.t. $C[\overrightarrow{w}]$. Let $\mathsf{ITDH} = (\mathsf{ITDH.KGen}, \mathsf{ITDH.Hash\&Enc}, \mathsf{ITDH.Dec})$ be a 2-level interactive trapdoor hashing protocol for $C[\overrightarrow{w}]^\oplus = \{C[\overrightarrow{w}]^\oplus_{n,u}\}_{n,u}$ with (Δ, ϵ)-approximate correctness and leveled function privacy.

Given ITDH, we construct a $2L$-level interactive trapdoor hashing protocol $\mathsf{ITDH}' = (\mathsf{KGen}, \mathsf{Hash\&Enc}, \mathsf{Dec})$ for the circuit family $C[\overrightarrow{\downarrow_L w}]$ as defined above. For ease of exposition, we describe the algorithms of ITDH′ for "odd" and "even" levels separately.

- **Level $\ell' = 2\ell - 1$, $\mathsf{KGen}(1^{\lambda_{\ell'}}, \ell', f, \mathsf{h}_{\ell'-1}, \mathsf{td}_{\ell'-1})$:**
  - If $\ell = 1$, set d_0 to be an all zero string of length n.
  - If $\ell \geq 2$, decode $\mathsf{h}_{\ell'-1}$: $\mathsf{d}_{\ell-1} \leftarrow \mathsf{ITDH.Dec}(\mathsf{td}_{\ell'-1}, \mathsf{h}_{\ell'-1})$
  - Let $f_1, \ldots, f_L$ be such that $f = f_L \circ f_{L-1} \circ \ldots \circ f_1$ (as defined above), where f_ℓ has input length n_ℓ and output length $n_{\ell+1}$.
  - Compute a key w.r.t. security parameter $\lambda_{\ell'}$ and the "Xor-*then*-Compute" circuit $f_\ell^{\oplus \mathsf{d}_{\ell-1}} \in C[\overrightarrow{w}]^\oplus_{n_\ell,n_{\ell+1}}$

    $$(\mathsf{k}_{\ell,1}, \mathsf{td}_{\ell,1}) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_{\ell'}}, 1, f_\ell^{\oplus \mathsf{d}_{\ell-1}}, \bot, \bot).$$

  - Output $(\mathsf{k}_{\ell'}, \mathsf{td}_{\ell'})$ where $\mathsf{k}_{\ell'} = (\ell', \mathsf{k}_{\ell,1})$ and $\mathsf{td}_{\ell'} = \mathsf{td}_{\ell,1}$.
- **Level $\ell' = 2\ell - 1$, $\mathsf{Hash\&Enc}(\mathsf{k}_{\ell'}, x, \mathsf{e}_{\ell'-1})$:**
  - If $\ell = 1$, let $x_\ell = x$, otherwise, let $x_\ell = \mathsf{e}_{\ell'-1}$. Execute

    $$(\mathsf{h}_{\ell,1}, \mathsf{e}_{\ell,1}) \leftarrow \mathsf{ITDH.Hash\&Enc}(\mathsf{k}_{\ell,1}, x, \bot)$$

  - Output $(\mathsf{h}_\ell = \mathsf{h}_{\ell,1}, \mathsf{e}_\ell = (x_\ell, \mathsf{e}_{\ell,1}))$.

- **Level $\ell' = 2\ell$, $\mathsf{KGen}(1^{\lambda_{\ell'}}, \ell', f, \mathsf{h}_{\ell'-1}, \mathsf{td}_{\ell'-1})$:**
  - Parse $\mathsf{h}_{\ell'-1} = \mathsf{h}_{\ell,1}$, and $\mathsf{td}_{\ell'-1} = \mathsf{td}_{\ell,1}$.

$$(\mathsf{k}_{\ell,2}, \mathsf{td}_{\ell,2}) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_{\ell'}}, 2, f_\ell^{\oplus \mathsf{d}_{\ell-1}}, \mathsf{h}_{\ell,1}, \mathsf{td}_{\ell,1})$$

  - Output $(\mathsf{k}_{\ell'}, \mathsf{td}_{\ell'})$, where $\mathsf{k}_{\ell'} = (\ell', \mathsf{k}_{\ell,2})$, and $\mathsf{td}_{\ell'} = \mathsf{td}_{\ell,2}$.
- **Level $\ell' = 2\ell$, $\mathsf{Hash\&Enc}(\mathsf{k}_{\ell'}, x, \mathsf{e}_{\ell'-1})$:**
  - Parse $\mathsf{e}_{\ell'-1} = (x_\ell, \mathsf{e}_{\ell,1})$, $\mathsf{k}_{\ell'} = \mathsf{k}_{\ell,2}$.
  - Output $(\mathsf{h}_{\ell'}, \mathsf{e}_{\ell'}) \leftarrow \mathsf{Hash\&Enc}(\mathsf{k}_{\ell,2}, x_\ell, \mathsf{e}_{\ell,1})$.
- **Decoding $\mathsf{Dec}(\mathsf{td}_{2L}, \mathsf{h}_{2L})$:**
  - Output $\mathsf{d} \leftarrow \mathsf{ITDH.Dec}(\mathsf{td}_{2L}, \mathsf{h}_{2L})$.

This completes the description of ITDH′.

5.5 Proof of Approximate Correctness

Lemma 5.6 (Approximate Correctness)**.** *For any circuit $f \in C[\vec{w}]_{n,s}$, ITDH′ satisfies (Δ', ϵ')-approximate correctness, where*

$$\Delta' = \sum_{\ell \in [L]} \Delta_\ell(n_{\ell+1}), \qquad \epsilon' = \sum_{\ell \in [L]} \epsilon_\ell(n_{\ell+1}, \lambda_{2\ell-1}, \lambda_{2\ell}) \cdot 2^{\lambda_1 + \lambda_2 + \ldots + \lambda_{2\ell-2}}$$

Proof. We start by bounding the error at each level ℓ.

Bounding error at ℓ-th level. For each $\ell \in [L]$, let $r_{2\ell-1}$ and $r_{2\ell}$ be the random coins used for the KGen in the $(2\ell-1)^{\text{th}}$ level and $2\ell^{\text{th}}$ level, respectively. For each $\ell \in [L]$, let $\ell' = 2\ell - 1$ be the starting level number for f_ℓ. Since the underlying ITDH satisfies $(\Delta_\ell(u), \epsilon_\ell(u, \lambda_1, \lambda_2))$-approximate correctness, for any fixed $\ell \in [L]$, and any fixed $\mathsf{d}'_{\ell-1}$ we have

$$\Pr_{r_{\ell'}, r_{\ell'+1}} \left[\exists x'_\ell : \mathsf{Ham}(\mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell, f_\ell^{\oplus \mathsf{d}'_{\ell-1}}(x'_\ell)) > \Delta_\ell(n_{\ell+1}) \right] < \epsilon_\ell(n_{\ell+1}, \lambda_{\ell'}, \lambda_{\ell'+1}),$$

where the $\mathsf{e}_{2\ell}$ and d_ℓ are obtained by executing the ITDH protocol with sender's input x'_ℓ, and receiver's input $f_\ell^{\oplus d'_{\ell-1}}$, and the randomness is over the random coins $r_{\ell'}, r_{\ell'+1}$.

Hence, if we fix the random coins $r_1, r_2, \ldots, r_{\ell'-1}$, and also fix $\mathsf{d}'_{\ell-1}$, then we have

$$\Pr_{r_{\ell'}, r_{\ell'+1}} \left[\exists x : \mathsf{d}_{\ell-1} = \mathsf{d}'_{\ell-1}, \mathsf{Ham}(\mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell, f_\ell^{\oplus \mathsf{d}'_{\ell-1}}(x_\ell)) > \Delta_\ell(n_{\ell+1}) \right] < \epsilon_\ell(n_{\ell+1}, \lambda_{\ell'}, \lambda_{\ell'+1}),$$

where the $\mathsf{e}_{2\ell}$ and d_ℓ are obtained by executing the ITDH′ protocol to the $(\ell'+1)^{\text{th}}$ level, with sender's input x, receiver's input f, and the random coins $r_1, r_2, \ldots, r_{\ell'-1}, r_{\ell'}, r_{\ell'+1}$. Note that in this probability, we only consider all x such that d_ℓ obtained from the execution equals to the fixed $\mathsf{d}'_{\ell-1}$. To further remove this restriction on x, we need to take an union bound on all possible choice of $\mathsf{d}_{\ell-1}$.

Since we fixed $r_1, r_2, \ldots, r_{\ell'-1}$, the decoding $\mathsf{d}_{\ell-1}$ only depends on $\mathsf{h}_1, \mathsf{h}_2, \ldots, \mathsf{h}_{\ell'-1}$. Hence, the total number of possibilities of $\mathsf{d}_{\ell-1}$ is bounded by $2^{\lambda_1+\lambda_2+\ldots+\lambda_{\ell'-1}}$. By taking an union bound, for any fixed $r_1, r_2, \ldots, r_{\ell'-1}$, we have

$$\Pr_{r_{\ell'}, r_{\ell'+1}}\left[\exists x, \mathsf{Ham}(\mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell, f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell)) > \Delta_\ell(n_{\ell+1})\right] < \epsilon_\ell(n_{\ell+1}, \lambda_{\ell'}, \lambda_{\ell'+1}) \cdot 2^{\lambda_1+\lambda_2+\ldots+\lambda_{\ell'-1}},$$

By averaging over all possibilities of $r_1, r_2, \ldots, r_{\ell'-1}$, the above inequality still holds when $r_1, r_2, \ldots, r_{\ell'-1}$ are sampled uniformly at random.

Now, except with probability

$$\sum_{\ell \in [L]} \epsilon_\ell(n_{\ell+1}, \lambda_{\ell'}, \lambda_{\ell'+1}) \cdot 2^{\lambda_1+\lambda_2+\ldots+\lambda_{\ell'-1}},$$

we have that for any x, and any $\ell \in [L]$, $\mathsf{Ham}(\mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell, f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell)) \leq \Delta(n_{\ell+1})$.

Controlling the Error Spread. For each $\ell \in [L]$, denote out_ℓ^* as the ideal (intermediate) outputs, $\mathsf{out}_\ell^* = f_\ell \circ f_{\ell-1} \circ \cdots \circ f_1(x)$. Then we have $\mathsf{out}_\ell^* = f_\ell(\mathsf{out}_{\ell-1}^*)$. We denote the real (intermediate) outputs as $\mathsf{out}_\ell = \mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell$ in the honest execution. Next, instead of bounding the Hamming distance out_ℓ and out_ℓ^* directly, we bound the following Hamming distance Ham_w over a larger alphabet.

For any two strings $a, b \in \{0,1\}^n$, where $n = n' \cdot w$, we firstly "partition" a as $a = (a_1, a_2, \ldots, a_w)$ where $a_i \in \{0,1\}^{n'}, i \in [w]$, and b as $b = (b_1, b_2, \ldots, b_w)$ where $b_i \in \{0,1\}^{n'}, i \in [w]$. We define the Hamming distance Ham_w between a and b as the number of index $i \in [w]$ such that a_i and b_i differ. Then we have

$$\begin{aligned}
\mathsf{Ham}_w(\mathsf{out}_\ell, \mathsf{out}_\ell^*) &\leq \mathsf{Ham}_w(\mathsf{out}_\ell, f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell)) + \mathsf{Ham}_w(f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell), \mathsf{out}_\ell^*) && (2)\\
&= \mathsf{Ham}_w(\mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell, f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell)) + \mathsf{Ham}_w(f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell), f_\ell(\mathsf{out}_{\ell-1}^*)) && (3)\\
&\leq \Delta(n_{\ell+1}) + \mathsf{Ham}_w(f_\ell(x_\ell \oplus \mathsf{d}_{\ell-1}), f_\ell(\mathsf{out}_{\ell-1}^*)) && (4)\\
&\leq \Delta(n_{\ell+1}) + \mathsf{Ham}_w(\mathsf{out}_{\ell-1}, \mathsf{out}_{\ell-1}^*) && (5)
\end{aligned}$$

The first inequality comes from the triangular inequality of the Hamming distance. The second line follows from the definition of out_ℓ and out_ℓ^*. The third line follows from the bound between $\mathsf{e}_{2\ell} \oplus \mathsf{d}_\ell$ and $f_\ell^{\oplus \mathsf{d}_{\ell-1}}(x_\ell)$, and the definition of the circuit $f_\ell^{\oplus \mathsf{d}_{\ell-1}}$. The fourth line follows from the fact that f_ℓ is a w-parallel composition circuit, hence, the "partitioned" Hamming errors between the output of f_ℓ is bounded by the "partitioned" Hamming errors between the input of f_ℓ. Recursively applying the Equation 5, we have

$$\mathsf{Ham}(\mathsf{e}_{2L} \oplus \mathsf{d}, f(x)) = \mathsf{Ham}_w(\mathsf{out}_L, \mathsf{out}_L^*) \leq \sum_{\ell \in [L]} \Delta(n_{\ell+1}).$$

By the definition of approximate correctness, we finish the proof. □

5.6 Proof of Leveled Function Privacy

Lemma 5.7 (Leveled Function Privacy)**.** *The construction above satisfies leveled function privacy.*

Proof. Since each key in construction ITDH′ is also a key of ITDH, the leveled function-privacy follows directly from the leveled function privacy of the underlying protocol ITDH. □

Remark 5.8. *If the underlying* ITDH *satisfies sub-exponential leveled function privacy, then the* ITDH′ *also satisfies sub-exponential leveled function privacy.*

5.7 ITDH for TC^0

We now describe how we can put the above constructions together to obtain an ITDH for TC^0. Recall that, we use the notation TC^0_L to denote the class of L-depth TC^0 circuits.

Let $\mathcal{T}[\vec{w}_{\downarrow L}]$ be the circuit family obtained by w-parallel-and-L-sequential composition of the circuit family $\mathcal{T}$, as per Definition 5.5. We first show that any circuit in TC^0_L can be converted to a circuit in $\mathcal{T}[\vec{w}_{\downarrow L}]$.

Lemma 5.9. *TC^0_L can be computed in $\mathcal{T}[\vec{w}_{\downarrow L}]$. Specifically, for any circuit $f \in \mathsf{TC}^0_L$ with n bit input and w output bits, we convert it in polynomial time to a circuit $f' \in \mathcal{T}[\vec{w}_{\downarrow L}]$ such that, for any $x \in \{0,1\}^n$, $f(x) = f'(x, x, \ldots, x)$.*

Proof. For any circuit $f \in \mathsf{TC}^0_L$ with w output bits, we can always convert it to a layered circuit (of the same depth L). Hence, we obtain a series of depth-1 circuits $f'_1, f'_2, \ldots, f'_L$ such that $f' = f'_L \circ f'_{L-1} \circ \ldots f'_1$. To make it a w-parallel-and-L-sequential-composition circuit, we repeat the input for w times, and for every $j \in [w]$, we compute the j^{th} output bit of $f'_L \circ f'_{L-1} \circ \ldots \circ f'_1$ on the j^{th} repetition of the input. □

Next, we combine the construction of ITDH for the circuit family $\mathcal{T}^{\oplus}$ from Section 5.1 together with the sequential composition theorem in section 5.4 to obtain an ITDH for the circuit family $\mathcal{T}[\vec{w}_{\downarrow L}]$, and therefore an ITDH for TC^0_L.

Theorem 5.10. *If for any inverse polynomial τ in the security parameter, there exists a trapdoor hash function TDH for linear function family $\mathcal{F}$ (as defined in Definition 3.5) with τ-enhanced correctness and sub-exponential function privacy, then for any constants $L = O(1)$, $\alpha = O(1)$, and any polynomial w in the security parameter, there exists a $2L$-level interactive trapdoor hashing protocol for TC^0_L that achieves (Δ, ϵ)-approximate correctness and sub-exponential function privacy, where $\Delta(w) = \alpha \cdot w$ and for any $\lambda_1 < \lambda_2 < \ldots < \lambda_{2L} < w/2L$, $\epsilon(w, \lambda_1, \ldots, \lambda_L) = 2^{-2w+O(1)}$.*

Proof. By Lemma 5.9, since each circuit in TC^0_L can be converted to a circuit in $\mathcal{T}[\vec{w}_{\downarrow L}]$, it suffices to construct ITDH for $\mathcal{T}[\vec{w}_{\downarrow L}]$. Since the circuit family $\mathcal{T}[\vec{w}]^{\oplus}$ is a subset of $\mathcal{T}^{\oplus}$, we combine the ITDH for $\mathcal{T}^{\oplus}$ with the generic composition in section 5.4. From Lemma 5.1, we have that the interactive trapdoor hashing protocol ITDH for circuit family $\mathcal{T}^{\oplus}$ satisfies $(\Delta, \epsilon = 2(2e \cdot \max\{\tau(\lambda_1), \tau(\lambda_2)\} \cdot u/\Delta)^{\Delta/2} \cdot 2^{\lambda_1+\lambda_2})$-approximate correctness for any Δ. Setting $\Delta_\ell(n_{\ell+1}) = \alpha w/L$, we have

$$\epsilon_\ell(u, \lambda_{2\ell-1}, \lambda_{2\ell}) = 2\left(\frac{2eL \cdot \tau_\ell \cdot u}{\alpha w}\right)^{\frac{\alpha w}{2L}} \cdot 2^{\lambda_{2\ell-1}+\lambda_{2\ell}},$$

where $\tau_\ell = \max\{\tau(\lambda_{2\ell-1}), \tau(\lambda_{2\ell})\}$.

From Lemma 5.6, for any security parameters $\lambda_1 < \lambda_2 < \ldots < \lambda_{2L}$, we have that ITDH′ satisfies (Δ', ϵ')-approximate correctness, where

$$\Delta'(m) = \sum_{\ell \in [L]} \Delta_\ell(n_{\ell+1}) \leq L \cdot \alpha w/L \leq \alpha w$$

$$\epsilon'(m, \lambda_1, \lambda_2, \ldots, \lambda_{2L}) = \sum_{\ell \in [L]} \epsilon_\ell(n_{\ell+1}, \lambda_{2\ell-1}, \lambda_{2\ell}) \cdot 2^{\lambda_1+\lambda_2+\ldots+\lambda_{2\ell-2}} \leq 2L \cdot \left(\frac{2eL \cdot \tau' \cdot s}{\alpha w}\right)^{\frac{\alpha w}{2L}} \cdot 2^{2L \cdot \lambda_{2L}},$$

where $\tau' = \max\{\tau(\lambda_1), \tau(\lambda_2), \ldots, \tau(\lambda_{2L})\}$, and s is the upper bound for n_ℓ. We set τ such that

$$\tau' < \frac{2^{-6L/\alpha} \alpha w}{2eL \cdot s},$$

which is an inverse polynomial. Hence, there exists a TDH construction with τ-enhanced correctness. Since $2L \cdot \lambda_{2L} < w$, we bound ϵ' by $\epsilon' < 2L \cdot 2^{-3w} \cdot 2^{w} < 2^{-2w+O(1)}$.

For the function privacy, since we assume sub-exponential leveled function privacy for the TDH, by Remark 5.3, the ITDH satisfies sub-exponential function privacy. Then by Remark 5.8, the ITDH′ construction satisfies sub-exponential function privacy. Since s is a polynomial in λ, we prove the theorem. □

ITDH for P/poly. Since any circuit in P/poly can be converted to a layered circuit as in Lemma 5.9, the above construction of ITDH for TC^0 can be naturally extended to obtain a polynomial-level ITDH for P/poly.

6 Correlation Intractable Hash Functions for TC^0

In this section, we build correlation intractable hash functions for the circuit family TC^0.

6.1 Definition

Correlation intractable hash (CIH) function is a tuple of algorithms $\mathsf{CIH} = (\mathsf{Gen}, \mathsf{Hash})$ described as follows:

- $\mathsf{Gen}(1^\lambda)$: It takes as input a security parameter λ and outputs a key k.
- $\mathsf{Hash}(\mathsf{k}, x)$: It takes as input a hash key k and a string x, and outputs a binary string y of length $w = w(\lambda)$.

We require CIH to satisfy the following property:

- **Correlation Intractability:** Recall that, a binary relation R is a subset of $\{0,1\}^* \times \{0,1\}^*$. We say that CIH is correlation intractable for a class of binary relations $\{\mathcal{R}_\lambda\}_\lambda$ if there exists a negligible function $\nu(\lambda)$ such that, for any $\lambda \in \mathbb{N}$, any n.u. PPT adversary $\mathcal{A}$, and any $R \in \mathcal{R}_\lambda$,

$$\Pr\left[\mathsf{k} \leftarrow \mathsf{Gen}(1^\lambda), x \leftarrow \mathcal{A}(1^\lambda, \mathsf{k}) : (x, \mathsf{Hash}(\mathsf{k}, x)) \in R\right] \leq \nu(\lambda)$$

We say that the CIH is sub-exponential correlation intractable, if there exists a constants c such that for any n.u. PPT adversary, its successful probability is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

Definition 6.1 (CIH for TC^0)**.** *Let $n(\lambda), w(\lambda)$ be polynomials. Let $L = O(1)$ be a constant. Recall that, we use TC^0_L to denote the class of L-depth threshold circuits. We say that* CIH *is a CIH for TC^0_L, if* CIH *is correlation intractable for the class of relations $\{\mathcal{R}_\lambda\}_\lambda$, where $\mathcal{R}_\lambda = \{R_{f,\lambda} \mid f \in \mathsf{TC}^0_L\}$, and*

$$R_{f,\lambda} = \{(x, y) \in \{0,1\}^{n(\lambda)} \times \{0,1\}^{w(\lambda)} \mid y = f(x)\}$$

6.2 Our Construction

For any $L = O(1)$, we show a generic transformation from an L-level ITDH for TC^0_L to a CIH for the same circuit family.

CIH for TC^0. Let $\mathsf{ITDH} = (\mathsf{ITDH.KGen}, \mathsf{ITDH.Hash\&Enc}, \mathsf{ITDH.Dec})$ be an L-level interactive trapdoor hashing protocol for the circuit class TC^0_L that satisfies the following properties:

- $(0.01w, 2^{-2w+O(1)})$-approximate correctness.

- Sub-exponential leveled function privacy. Let Sim be the leveled function privacy simulator. Let c be the constant in the sub-exponential security definition.

We construct a correlation intractable hash function $\mathsf{CIH} = (\mathsf{CIH.Gen}, \mathsf{CIH.Hash})$ for TC^0_L in Figure 5.

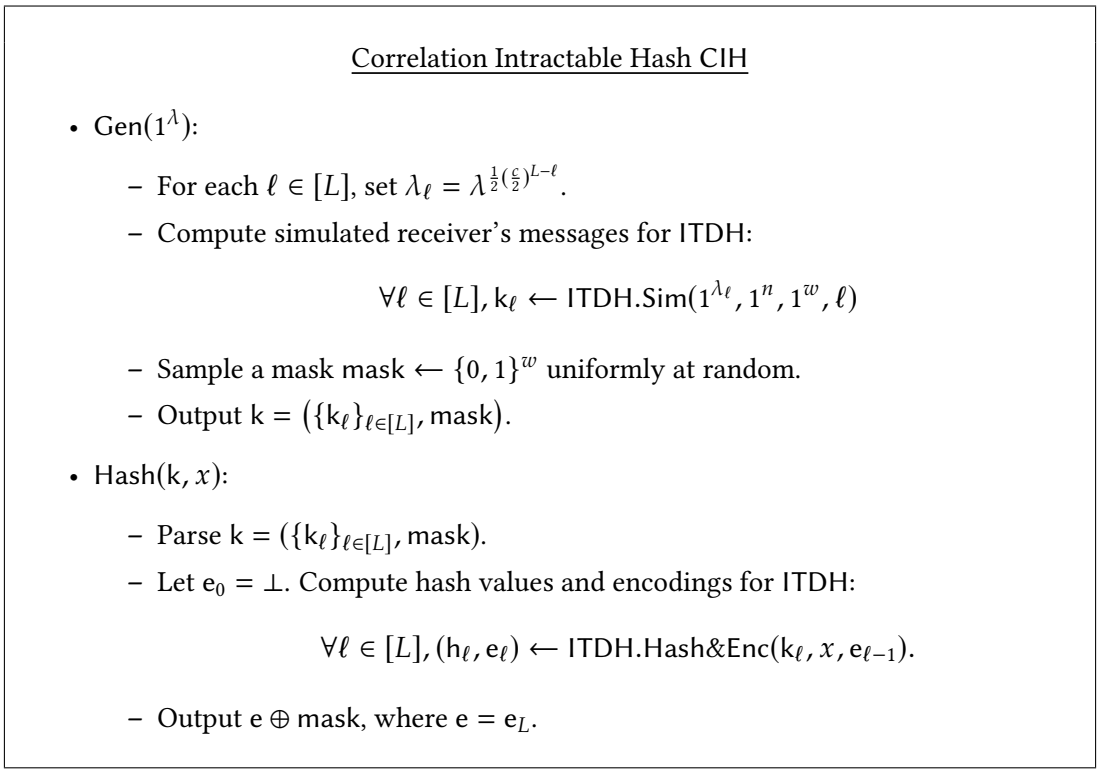


Figure 5: Description of CIH.

Theorem 6.2 (Correlation Intractability). *If $w = \Omega(\lambda)$, the construction in Figure 5 is sub-exponential correlation intractable for the circuit class TC^0_L.*

6.3 Proof of Correlation Intractability

We prove Theorem 6.2 by contradiction. Let $\{f_\lambda\}_\lambda$ be a sequence of circuits in TC^0_L, and let $\mathcal{A}$ be a n.u. PPT adversary breaking correlation intractability with probability $\epsilon(\lambda)$.

We find the contradiction by constructing a series of hybrids.

- Hyb_0: In this hybrid, if the adversary's attack successes, then output 1, otherwise output 0.
  - Sample the CIH key $\mathsf{k} \leftarrow \mathsf{Gen}(1^\lambda)$, and run the adversary $x \leftarrow \mathcal{A}(1^\lambda, \mathsf{k})$.
  - If $\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x)$, then output 1, otherwise output 0.
- $\mathsf{Hyb}_1^{\ell^*}$: This hybrid is the same as Hyb_0, except that we additionally guess the hash value h_{ℓ^*-1} by sampling h'_{ℓ^*-1} from uniform distribution.
  - Let $\mathsf{h}_0 = \mathsf{td}_0 = \perp$. For $\ell = 1, 2, \ldots, \ell^* - 1$, if $\ell > 1$, let $\mathsf{h}'_{\ell-1} \leftarrow \{0,1\}^{\lambda_{\ell-1}}$. Let
  $$(\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_\ell}, \ell, f_\lambda, \mathsf{h}'_{\ell-1}, \mathsf{td}_{\ell-1})$$
  - If $\ell^* > 1$, sample $\mathsf{h}'_{\ell^*-1} \leftarrow \{0,1\}^{\lambda_{\ell^*-1}}$ uniformly at random. Let $\mathsf{k}_{\ell^*} \leftarrow \mathsf{ITDH.Sim}(1^{\lambda_{\ell^*}}, 1^n, 1^w, \ell^*)$.

- For $\ell = \ell^* + 1, \ldots, L$, let $\mathsf{k}_\ell \leftarrow \mathsf{ITDH.Sim}(1^{\lambda_\ell}, 1^n, 1^w, \ell)$.
- Sample $\mathsf{mask} \leftarrow \{0,1\}^w$ uniformly at random. Let $\mathsf{k} = (\{\mathsf{k}_\ell\}_{\ell \in [L]}, \mathsf{mask})$.
- Run the adversary $x \leftarrow \mathcal{A}(1^\lambda, \mathsf{k})$.
- If $\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x)$ and $\forall i \in [\ell^* - 1], \mathsf{h}'_\ell = \mathsf{h}_\ell$, then output 1. Otherwise, output 0.

- $\mathsf{Hyb}_{1.5}^{\ell^*}$: This hybrid is the same as $\mathsf{Hyb}_1^{\ell^*}$, except that we replace the ℓ^{th} level key with a "real key" generated by ITDH.KGen.
  - Let $\mathsf{h}_0 = \mathsf{td}_0 = \perp$. For $\ell = 1, 2, \ldots, \ell^*$, if $\ell > 1$, let $\mathsf{h}'_{\ell-1} \leftarrow \{0,1\}^{\lambda_{\ell-1}}$. Let
  $$(\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_\ell}, \ell, f_\lambda, \mathsf{h}'_{\ell-1}, \mathsf{td}_{\ell-1})$$
  - For $\ell = \ell^* + 1, \ldots, L$, let $\mathsf{k}_\ell \leftarrow \mathsf{ITDH.Sim}(1^{\lambda_\ell}, 1^n, 1^w, \ell)$.
  - Sample $\mathsf{mask} \leftarrow \{0,1\}^w$ uniformly at random. Let $\mathsf{k} = (\{\mathsf{k}_\ell\}_{\ell \in [L]}, \mathsf{mask})$.
  - Run the adversary $x \leftarrow \mathcal{A}(1^\lambda, \mathsf{k})$.
  - If $\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x)$ and $\forall i \in [\ell^* - 1], \mathsf{h}'_\ell = \mathsf{h}_\ell$, then output 1. Otherwise, output 0.

- Hyb_2: This hybrid is the same as Hyb_1^{L+1}.
  - Let $\mathsf{h}_0 = \mathsf{td}_0 = \perp$. For $\ell = 1, 2, \ldots, L$, if $\ell > 1$, let $\mathsf{h}'_{\ell-1} \leftarrow \{0,1\}^{\lambda_{\ell-1}}$. Let
  $$(\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_\ell}, \ell, f_\lambda, \mathsf{h}'_{\ell-1}, \mathsf{td}_{\ell-1})$$
  - Sample $\mathsf{h}'_L \leftarrow \{0,1\}^{\lambda_L}$ uniformly at random.
  - Sample $\mathsf{mask} \leftarrow \{0,1\}^w$ uniformly at random. Let $\mathsf{k} = (\{\mathsf{k}_\ell\}_{\ell \in [L]}, \mathsf{mask})$.
  - Run the adversary $x \leftarrow \mathcal{A}(1^\lambda, \mathsf{k})$.
  - If $\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x)$ and $\forall i \in [L], \mathsf{h}'_\ell = \mathsf{h}_\ell$, then output 1. Otherwise, output 0.

Lemma 6.3. *For any sufficiently large λ, $\Pr[\mathsf{Hyb}_{1.5}^{\ell^*} = 1] \geq \Pr[\mathsf{Hyb}_1^{\ell^*} = 1] - 2^{-\lambda_{\ell^*}^c}$.*

Proof. For n.u. PPT adversary $\mathcal{A}$, we build the following distinguisher $\mathcal{D}$ for the sub-exponential function privacy in Figure 6. When the challenger computes k_{ℓ^*} from ITDH.KGen, the distinguisher $\mathcal{D}$ simulates the environments for $\mathcal{A}$, and hence

$$\Pr\left[\mathsf{k}_{\ell^*} \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_{\ell^*}}, \ell^*, f_\lambda, \mathsf{h}'_{\ell^*-1}, \mathsf{td}_{\ell^*-1}) : \mathcal{D}(1^{\lambda_{\ell^*}}) = 1\right] = \Pr[\mathsf{Hyb}_{1.5}^{\ell^*} = 1].$$

Similarly, we also have

$$\Pr\left[\mathsf{k}_{\ell^*} \leftarrow \mathsf{ITDH.Sim}(1^{\lambda_{\ell^*}}, 1^n, 1^w, \ell^*) : \mathcal{D}(1^{\lambda_{\ell^*}}) = 1\right] = \Pr[\mathsf{Hyb}_1^{\ell^*} = 1].$$

Note that, the distinguisher $\mathcal{D}(1^{\lambda_{\ell^*}})$ runs in $\mathrm{poly}(\lambda)$ time, since $\lambda = \mathrm{poly}(\lambda_{\ell^*})$, the distinguisher also runs in $\mathrm{poly}(\lambda_{\ell^*})$ time. If the ITDH satisfies the sub-exponential function privacy property, the probabilities on the left hand sides differ by at most $2^{-\lambda_{\ell^*}^c}$, where c is a constant. Hence, we finish proving the lemma. □

Lemma 6.4. $\Pr[\mathsf{Hyb}_1^{\ell^*+1} = 1] \geq \Pr[\mathsf{Hyb}_{1.5}^{\ell^*} = 1]/2^{\lambda_{\ell^*}}$.

Distinguisher $\mathcal{D}(1^{\lambda_{\ell^*}})$

- For each $\ell < \ell^*$, generate k_ℓ using ITDH.KGen,

$$\mathsf{h}'_{\ell-1} \leftarrow \{0,1\}^{\lambda_{\ell-1}}, (\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_\ell}, \ell, f_\lambda, \mathsf{h}'_{\ell-1}, \mathsf{td}_{\ell-1}).$$

- If $\ell^* > 1$, sample $\mathsf{h}'_{\ell^*} \leftarrow \{0,1\}^{\ell^*}$ uniformly at random, query the challenger with h'_{ℓ^*-1} and td_{ℓ^*-1}, and get k_{ℓ^*} from the challenger.
- For each $\ell > \ell^*$, generate k_ℓ using ITDH.Sim,

$$\mathsf{k}_\ell \leftarrow \mathsf{ITDH.Sim}(1^{\lambda_\ell}, 1^n, 1^w, \ell).$$

- Sample $\mathsf{mask} \leftarrow \{0,1\}^w$ uniformly at random, and let $\mathsf{k} = (\{\mathsf{k}_\ell\}_{\ell\in[L]}, \mathsf{mask})$.
- Run the adversary $\mathcal{A}(1^\lambda, \mathsf{k})$.
- If $\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x)$ and $\forall i \in [\ell^* - 1], \mathsf{h}'_\ell = \mathsf{h}_\ell$, then output 1. Otherwise, output 0.

Figure 6: Description of the distinguisher $\mathcal{D}$.

Proof. The difference between $\mathsf{Hyb}_{1.5}^{\ell^*}$ and $\mathsf{Hyb}_1^{\ell^*+1}$ is that, in $\mathsf{Hyb}_1^{\ell^*+1}$, we guess the hash value h'_{ℓ^*}. Hence,

$$\begin{aligned}
\Pr\left[\mathsf{Hyb}_1^{\ell^*+1} = 1\right] &= \Pr_{\mathsf{Hyb}_1^{\ell^*+1}}\left[\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x) \wedge (\forall i \in [\ell^*], \mathsf{h}'_\ell = \mathsf{h}_\ell)\right] \\
&= \Pr_{\mathsf{Hyb}_1^{\ell^*+1}}\left[\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x) \wedge (\forall i \in [\ell^* - 1], \mathsf{h}'_\ell = \mathsf{h}_\ell) \wedge \mathsf{h}'_{\ell^*} = \mathsf{h}_{\ell^*}\right] \\
&= \Pr_{\mathsf{Hyb}_1^{\ell^*+1}}\left[\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x) \wedge (\forall i \in [\ell^* - 1], \mathsf{h}'_\ell = \mathsf{h}_\ell)\right] \Pr\left[\mathsf{h}'_{\ell^*} = \mathsf{h}_{\ell^*}\right] \\
&= \Pr_{\mathsf{Hyb}_{1.5}^{\ell^*}}[\mathsf{Hash}(\mathsf{k}, x) = f_\lambda(x) \wedge (\forall i \in [\ell^* - 1], \mathsf{h}'_\ell = \mathsf{h}_\ell)]/2^{\lambda_{\ell^*}} \\
&= \Pr\left[\mathsf{Hyb}_{1.5}^{\ell^*} = 1\right]/2^{\lambda_{\ell^*}}.
\end{aligned}$$

The first line follows from the construction of the hybrid $\mathsf{Hyb}_1^{\ell^*+1}$. The second line is obtained by considering the cases $i \in [\ell^* - 1]$ and $i = \ell^*$ separately. The third line follows from the independence of h'_{ℓ^*} and all other random variables. The fourth line follows from the fact that the bit length of h'_{ℓ^*} is λ_{ℓ^*}. The fifth line follows from the definition of $\mathsf{Hyb}_{1.5}^{\ell^*}$. Hence, we finish proving the lemma. □

Lemma 6.5. $\Pr[\mathsf{Hyb}_2 = 1] < 2^{-\Omega(\lambda)}$.

Proof. In Hyb_2, we check if $\forall i \in [L], \mathsf{h}'_\ell = \mathsf{h}_\ell$. Note that if such check passes, then $\mathsf{Hash}(\mathsf{k}, x)$ equals to $\mathsf{e} \oplus \mathsf{mask}$, where e is the encoding in the final level in an honest execution. Hence,

$$\Pr[\mathsf{Hyb}_2 = 1] \le \Pr_{\mathsf{mask}\leftarrow\{0,1\}^w, r_1, r_2, \ldots, r_L}\left[\exists x : \mathsf{e} \oplus \mathsf{mask} = f_\lambda(x)\right],$$

where $r_1, r_2, \ldots, r_L$ are the random coins for the ITDH, and the encoding e is obtained from the following procedure.

Let $\mathsf{h}_0 = \mathsf{td}_0 = \mathsf{e}_0 = \perp$. For $\ell = 1, 2, \ldots, L$,

- Compute $(\mathsf{k}_\ell, \mathsf{td}_\ell) \leftarrow \mathsf{ITDH.KGen}(1^{\lambda_\ell}, \ell, f_\lambda \mathsf{h}_{\ell-1}, \mathsf{td}_{\ell-1}; r_\ell)$ with random coins r_ℓ.
- Hash the input x using the hash key $(\mathsf{h}_\ell, \mathsf{e}_\ell) \leftarrow \mathsf{ITDH.Hash\&Enc}(\mathsf{k}_\ell, x, \mathsf{e}_{\ell-1})$

Finally, let $\mathsf{e} = \mathsf{e}_L$ be the encoding at the final level, and also let $\mathsf{d} = \mathsf{ITDH.Dec}(\mathsf{td}_L, \mathsf{h}_L)$.

Since the ITDH satisfies $(0.01w, 2^{-2w+O(1)})$-approximate correctness, we have

$$\Pr_{r_1, r_2, \ldots, r_L}[\exists x, \mathsf{Ham}(\mathsf{e} \oplus \mathsf{d}, f_\lambda(x)) > 0.01w] < 2^{-2w+O(1)}.$$

Hence, except with probability $2^{-2w+O(1)}$, we have that $\forall x, \mathsf{Ham}(\mathsf{e} \oplus \mathsf{d}, f_\lambda(x)) \leq 0.01w$. Denote the Hamming error between $\mathsf{e} \oplus \mathsf{d}$ and $f_\lambda(x)$ as ε. Then we have $f_\lambda(x) = \mathsf{e} \oplus \mathsf{d} \oplus \varepsilon$, and the weight of ε is at most $0.01w$. Now, we have,

$$\begin{aligned}\Pr_{\mathsf{mask} \leftarrow \{0,1\}^w, r_1, r_2, \ldots, r_L}[\exists x : \mathsf{e} \oplus \mathsf{mask} = f_\lambda(x)] &\leq 2^{-2m+O(1)} + \Pr_{\mathsf{mask} \leftarrow \{0,1\}^w, r_1, r_2, \ldots, r_L}[\exists x, \varepsilon : \mathsf{e} \oplus \mathsf{mask} = \mathsf{e} \oplus \mathsf{d} \oplus \varepsilon] \\ &\leq 2^{-2m+O(1)} + \Pr_{\mathsf{mask} \leftarrow \{0,1\}^w, r_1, r_2, \ldots, r_L}[\exists x, \varepsilon : \mathsf{mask} = \mathsf{d} \oplus \varepsilon]\end{aligned}$$

Note that, for any fixed random coins $r_1, r_2, \ldots, r_L$, the decoding value d only depends on $\mathsf{h}_1, \mathsf{h}_2, \ldots, \mathsf{h}_L$. The number of possible choice of d is $2^{\lambda_1+\lambda_2+\ldots+\lambda_L} \leq 2^{2\lambda_L} \leq 2^{2\lambda^{1/2}}$. The number of possible values of ε is at most $\binom{w}{0.01w} \leq (100e)^{0.01w} \leq 2^{w/2}$. Hence, we have

$$\Pr_{\mathsf{mask} \leftarrow \{0,1\}^w, r_1, r_2, \ldots, r_L}[\exists x, \varepsilon : \mathsf{mask} = \mathsf{d} \oplus \varepsilon] < 2^{2\lambda^{1/2}} \cdot 2^{w/2} \cdot 2^{-w} = 2^{-(w/2-2\lambda^{1/2})} = 2^{-\Omega(\lambda)}$$

□

Completing the Proof. Let $\epsilon(\lambda) = \Pr[\mathsf{Hyb}_0 = 1]$. We first claim that, for each $\ell^* \in [L+1]$, we have

$$\Pr\left[\mathsf{Hyb}_1^{\ell^*} = 1\right] \geq (\epsilon(\lambda) - \ell^* 2^{-\lambda_1^2+2\lambda_1})/2^{2\lambda_{\ell^*-1}},$$

where $\lambda_0 = 0$.

We now prove this claim by induction on ℓ^*. For $\ell^* = 1$, since Hyb_1^1 and Hyb_0 are identical, we have $\Pr\left[\mathsf{Hyb}_1^1 = 1\right] = \Pr\left[\mathsf{Hyb}_0 = 1\right] \geq \epsilon(\lambda)$. Hence, then the claim holds for $\ell^* = 1$. Now, we assume the claim holds for ℓ^*, we prove that the claim holds for $\ell^* + 1$ as follows.

From Lemma 6.3, we have

$$\Pr\left[\mathsf{Hyb}_{1.5}^{\ell^*} = 1\right] \geq \Pr\left[\mathsf{Hyb}_1^{\ell^*} = 1\right] - 2^{\lambda_{\ell^*}^c} \geq \epsilon(\lambda) - \ell^* 2^{-\lambda_1^2+2\lambda_1}/2^{2\lambda_{\ell^*-1}} - 2^{-\lambda_{\ell^*}^c}.$$

By the choice of the parameters, we have $\lambda_{\ell^*} = (\lambda_{\ell^*-1})^{\frac{2}{c}}$. Hence, the right hand side is bounded by

$$\begin{aligned}\epsilon(\lambda) - \ell^* 2^{-\lambda_1^2+2\lambda_1}/2^{2\lambda_{\ell^*-1}} - 2^{-\lambda_{\ell^*-1}^2} &= (\epsilon(\lambda) - \ell^* 2^{-\lambda_1^2+2\lambda_1} - 2^{-\lambda_{\ell^*-1}^2+\lambda_{\ell^*-1}})/2^{2\lambda_{\ell^*-1}} \\ &\geq (\epsilon(\lambda) - (\ell^*+1)2^{-\lambda_1^2+2\lambda_1})/2^{2\lambda_{\ell^*-1}}\end{aligned}$$

Then, by Lemma 6.4, we have

$$\begin{aligned}\Pr\left[\mathsf{Hyb}_1^{\ell^*+1} = 1\right] \geq \Pr[\mathsf{Hyb}_{1.5}^{\ell^*} = 1]/2^{\lambda_{\ell^*}} &\geq (\epsilon(\lambda) - (\ell^*+1)2^{-\lambda_1^2+2\lambda_1})/2^{2\lambda_{\ell^*-1}+\lambda_{\ell^*}} \\ &> (\epsilon(\lambda) - (\ell^*+1)2^{-\lambda_1^2+2\lambda_1})/2^{2\lambda_{\ell^*}}.\end{aligned}$$

Hence, we finish prove the claim.

By this claim, and the fact that Hyb_2 is identical to Hyb_1^{L+1} we know that

$$\Pr[\mathsf{Hyb}_2 = 1] = \Pr[\mathsf{Hyb}_1^{L+1} = 1] \geq (\epsilon(\lambda) - (L+1)2^{-\lambda_1^2+2\lambda_1})/2^{2\lambda^{1/2}}.$$

From Lemma 6.5, we have $\Pr[\mathsf{Hyb}_2 = 1] < 2^{-\Omega(\lambda)}$. Hence, we have

$$\epsilon(\lambda) < 2^{-\Omega(\lambda)+2\lambda^{1/2}} + (L+1)2^{-\lambda_1^2+2\lambda_1}.$$

Since $L = O(1)$ and $\lambda_1 = \lambda^{\Theta(1)}$, we finish the proof.

6.4 On the Trade-off between DDH-hardness and the Circuit Class for CIH

In the previous subsections, we constructed CIH for TC^0 based on sub-exponential hardness of DDH against polynomial time adversaries. In this section, we show that we can in fact trade-off between the hardness assumption on DDH and the depth of the circuit class for CIH.

CIH for $O(\log\log\lambda)$-depth Threshold Circuits. If we assume sub-exponential hardness of DDH against sub-exponential time adversaries, then we can obtain CIH for $O(\log\log\lambda)$-depth threshold circuits.

Theorem 6.6 (CIH for $O(\log\log\lambda)$-depth Threshold Circuits.). *If we assume there exists a constant $0 < c < 1$ such that for any non-uniform adversary running in time $2^{O(\lambda^c)}$, the advantage for DDH is bounded by $2^{-\Omega(\lambda^c)}$, then there exists a construction of CIH for any polynomial size circuits with depth $\frac{1-o(1)}{4\log(2/c)}\log\log\lambda$, and output length $\Omega(\lambda)$.*

We can set the security parameters for i^{th} level as $\lambda_i = (\log^2\lambda)^{(\frac{2}{c})^i}$, for $i = 1, 2, \ldots, \frac{1}{4\log(2/c)}\log\log\lambda \cdot (1-o(1))$. Note that at the last level, $\lambda_L \leq \lambda^{1/2}$. Then the proofs in Section 6.3 can be extended to CIH for $\frac{1-o(1)}{4\log(2/c)}\log\log\lambda$-depth threshold circuits. Note that here we crucially rely on the sub-exponential time assumption, since an adversary that runs in time polynomial in λ is an adversary that runs in time sub-exponential in λ_i.

CIH for TC^1. If we assume exponential hardness of DDH against polynomial time adversaries, then we can obtain CIH for TC^1, i.e., log-depth threshold circuits.

Theorem 6.7 (CIH for TC^1). *If we assume there exists a constant $A > 1$ such that for any non-uniform adversary running in polynomial time, the advantage for DDH is bounded by $2^{-\Omega(\lambda/A)}$. Then there exists a construction of CIH for any polynomial size threshold circuits with depth $\lfloor\frac{1}{3}\cdot\log_{2A}\lambda\rfloor$ and output length $\Omega(\lambda)$.*

We can set the security parameter for i^{th} level as $\lambda_i = \lambda^{1/3}\cdot(2A)^i$, where $i = 1, 2, \ldots, \lfloor\frac{1}{3}\cdot\log_{2A}\lambda\rfloor$. Note that at the last level, $\lambda_L \leq \lambda^{2/3}$. Then the proofs in Section 6.3 also work for TC^1 circuits.

7 Non-Interactive (Statistical) Zero-Knowledge Arguments for NP

In this section, we present our constructions of NIZK arguments for NP in the common random string model. We present two variants: one that achieves statistical zero knowledge and non-adaptive soundness, and another that achieves computational zero knowledge and adaptive soundness. For most of this section, we focus on the first variant, namely, statistical NIZK arguments for NP. We obtain the computational variant via simple modifications to our first construction.

The rest of this section is organized as follows:

- In Section 7.1, we construct a lossy public key encryption scheme LPKE with *low-depth decryption property*, which essentially requires that the decryption circuit is in TC^0.
- Using LPKE, we construct a trapdoor sigma protocol for NP that achieves statistical honest-verifier zero knowledge. This protocol achieves two additional key properties – *low-depth bad challenge function* and *knowledge extraction*. We describe this construction in Section 7.2.
- Next, in Section 7.3, we construct non-interactive statistical witness indistinguishable (NISWI) arguments for NP in the common random string model via the correlation-intractability framework. Namely, we use CIH for TC^0 to collapse the rounds of λ parallel-repetitions of the trapdoor sigma protocol from the previous step to obtain NISWIs. We further prove that the resulting scheme achieves (sub-exponential) non-adaptive argument of knowledge property.
- In Section 7.4, we describe a variant of the FLS "OR-trick" [32] to transform NISWI for NP from the previous step to multi-theorem *statistical* NIZK for NP, while preserving the distribution of the CRS.
- Finally, in Section 7.5, we sketch our construction of multi-theorem *computational* NIZK arguments for NP with *adaptive* soundness in the common random string model. This variant is obtained via the same steps as above, except that we replace the lossy public-key encryption scheme (used for constructing trapdoor sigma protocol) with Elgamal encryption [31].

7.1 Lossy Public Key Encryption with Low-Depth Decryption

A lossy public key encryption scheme is a tuple of algorithms $(\mathsf{LossyGen}, \mathsf{Gen}, \mathsf{Enc})$, which proceeds as follows.

- $\mathsf{Gen}(1^\lambda)$: The key generation algorithm takes as input a security parameter λ, and it outputs a public key pk and a secret key sk.
- $\mathsf{LossyGen}(1^\lambda)$: The lossy public key generation algorithm takes as input the security parameter, and it outputs a lossy public key $\widetilde{\mathsf{pk}}$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm takes as input a public key pk, and a message $m \in \mathbb{Z}_2$, it outputs a ciphertext ct.

We require the algorithms to satisfy the following properties.

- **Key Indistinguishability:** For any n.u. PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\lambda)$ such that for any $\lambda \in \mathbb{N}$,

$$\left| \Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{D}(1^\lambda, \mathsf{pk}) = 1\right] - \Pr\left[\widetilde{\mathsf{pk}} \leftarrow \mathsf{LossyGen}(1^\lambda) : \mathcal{D}(1^\lambda, \widetilde{\mathsf{pk}}) = 1\right] \right| \leq \nu(\lambda).$$

  Furthermore, we say the scheme satisfies sub-exponential key indistinguishability, if there exists a constants c and λ_0 such that for any n.u. PPT distinguisher, the advantage $\nu(\lambda)$ is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

- **Statistical Semantic Security in Lossy Mode:** There exists a negligible function $\nu(\lambda)$ s.t. for any two messages $m_1, m_2 \in \mathbb{Z}_2$,

$$\mathsf{SD}\left(\left(\widetilde{\mathsf{pk}}, \mathsf{Enc}(\widetilde{\mathsf{pk}}, m_1)\right), \left(\widetilde{\mathsf{pk}}, \mathsf{Enc}(\widetilde{\mathsf{pk}}, m_2)\right)\right) \leq \nu(\lambda),$$

where $\widetilde{\mathsf{pk}} \leftarrow \mathsf{LossyGen}(1^\lambda)$, and the randomness of the distribution is over the randomness of LossyGen and Enc.

- **Low-Depth Decryption:** There exists a sequence of circuits $\{\mathsf{Dec}_\lambda\}_\lambda$ in TC^0 and a deterministic polynomial-time algorithm PreComp such that, for any $\lambda \in \mathbb{N}$ and any message $m \in \mathbb{Z}_2$,

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

7.1.1 Construction

We describe our lossy public key encryption scheme LPKE in Figure 7. Our construction is essentially the same as the dual-mode encryption scheme in [59]. We show that this construction achieves low-depth decryption property.

We instantiate the group $\mathbb{G}$ as the standard prime order subgroup of $\mathbb{Z}_q^*$, with efficient group membership testing algorithm. For instantiation from Elliptic curves, see Appendix B.

Lossy Public Key Encryption LPKE

- $\mathsf{Gen}(1^\lambda)$:
  - Generate a group using $\mathcal{G}$: $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$.
  - Sample $a, b \leftarrow \mathbb{Z}_p$.
  - Output $\left(\mathsf{pk} = \left(\mathbb{G}, p, \begin{bmatrix} g & g^b \\ g^a & g^{ab} \end{bmatrix}\right), \mathsf{sk} = a\right)$.
- $\mathsf{LossyGen}(1^\lambda)$:
  - Generate a group using $\mathcal{G}$: $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$.
  - Sample the elements uniformly at random from $\mathbb{G}$, $g_{12}, g_{21}, g_{22} \leftarrow \mathbb{G}$.
  - Output $\widetilde{\mathsf{pk}} = \left(\mathbb{G}, p, \begin{bmatrix} g & g_{12} \\ g_{21} & g_{22} \end{bmatrix}\right)$.
- $\mathsf{Enc}(\mathsf{pk}, m; r)$:
  - Parse $\mathsf{pk} = \left(\mathbb{G}, p, \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}\right)$, $m \in \{0, 1\}$ and $r = (r_1, r_2) \in \mathbb{Z}_p^2$.
  - Output the ciphertext $\mathsf{ct} = \begin{bmatrix} g_{11}^{r_1} \cdot g_{12}^{r_2} \\ g_{21}^{r_1} \cdot g_{22}^{r_2} \cdot g^m \end{bmatrix}$

Figure 7: Construction of the lossy public key encryption.

We now prove that LPKE achieves the required properties.

Lemma 7.1 (Statistical Semantic Security in Lossy Mode)**.** *The proposed scheme* LPKE *satisfies statistical semantic security in lossy mode.*

Proof. For any lossy public key

$$\widetilde{\mathsf{pk}} = \left(\mathbb{G}, p, \begin{bmatrix} g^1 & g^a \\ g^b & g^c \end{bmatrix}\right),$$

where a, b, c are sampled from uniform distribution over $\mathbb{Z}_p$, with probability $1 - 1/|\mathbb{G}|$, $c \neq ab$. When it happens, the matrix $\begin{bmatrix} 1 & a \\ b & c \end{bmatrix}$ is non-singular and hence the ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\widetilde{\mathsf{pk}}, m)$ is uniformly distributed over $\mathbb{G}^2$ and is independent of m. Hence,

$$\mathsf{SD}((\widetilde{\mathsf{pk}}, \mathsf{Enc}(\widetilde{\mathsf{pk}}, m)), (\widetilde{\mathsf{pk}}, U)) \leq 1/|\mathbb{G}|,$$

where U is the uniform distribution over $\mathbb{G}^2$. By the triangular inequality of statistical distance, $(\widetilde{\mathsf{pk}}, \mathsf{Enc}(\widetilde{\mathsf{pk}}, m_1))$ and $(\widetilde{\mathsf{pk}}, \mathsf{Enc}(\widetilde{\mathsf{pk}}, m_2))$ are statistically indistinguishable, for any messages $m_1, m_2 \in \mathbb{Z}_2$. We finish the proof. □

Lemma 7.2 (Key Indistinguishability)**.** *Assuming (sub-exponential) DDH, the proposed scheme* LPKE *satisfies (sub-exponential) key indistinguishability.*

Proof. Since an injective mode public key is a DDH tuple and a lossy public key is subjected to uniformly distribution, the (sub-exponential) key indistinguishability follows from the (sub-exponential) DDH assumption directly. □

Lemma 7.3 (Low-Depth Decryption)**.** *The proposed scheme* LPKE *satisfies low-depth decryption property.*

Proof. We construct the following algorithm PreComp and the circuit family $\{\mathsf{Dec}_\lambda\}_\lambda$. For any $\mathsf{sk} = a$, we denote $(a_0, a_1, \ldots, a_\lambda)$ to be binary representation of a, i.e., $a = a_0 2^0 + a_1 2^1 + \ldots + a_\lambda 2^\lambda$ where $a_i \in \{0, 1\}$. Since Dec_λ is a circuit over boolean value, it takes as the inputs in the binary representation form. Hence, we can assume Dec_λ takes $a_0, a_1, \ldots, a_\lambda$ as input.

- $\mathsf{PreComp}(1^\lambda, c = (c_1, c_2) \in \mathbb{G}^2)$: Output $(c_1^{-2^0}, c_1^{-2^1}, c_1^{-2^2}, \ldots, c_1^{-2^\lambda}, c_2)$.
- $\mathsf{Dec}_\lambda\left((c'_0, c'_1, c'_2, \ldots, c'_\lambda, c_2), \mathsf{sk} = (a_0, a_1, \ldots, a_\lambda)\right)$:
  - For each $i \in 0, 1, \ldots, \lambda$, if $a_i = 0$, let $g_i = 1_\mathbb{G}$. Otherwise, let $g_i = c'_i$.
  - Compute $u = g_0 \cdot g_1 \cdot \ldots \cdot g_\lambda \cdot c_2$, where the iterative multiplication is over $\mathbb{Z}_q^*$.
  - Compare u with $1_\mathbb{G}$. If $u = 1_\mathbb{G}$, output 0. Otherwise output 1.

We first argue the correctness of Dec_λ. It is easy to see that for any ciphertext $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; r)$ of the message $m \in \{0, 1\}$ and randomness r using public key pk, we have that $\mathsf{Dec}_\lambda(\mathsf{PreComp}(1^\lambda, c), \mathsf{td}) = c_1^{-a} \cdot c_2 = m$.

Next, we argue the low-depth property. Note that the first step of Dec_λ can be easily computed by a constant depth threshold circuit. From [60], we have that the second step, which involves multiplication of λ inputs mod a prime q, can also be computed in TC^0. For the third step, the comparison between g_0 and $1_\mathbb{G}$ can be computed in TC^0. Hence, by composing these circuits, we obtain a circuit for Dec_λ in TC^0. □

7.2 Trapdoor Sigma Protocol

In this section, we construct trapdoor sigma protocols for NP with low-depth bad challenge functions, and with an additional knowledge extraction property. We start by providing a formal definition.

Definition. A trapdoor sigma protocol for a language $\mathcal{L}$ is a tuple of algorithms $\Sigma = (\mathsf{Gen}_{\mathsf{zk}}, \mathsf{Gen}_{\mathsf{sound}}, \mathsf{P}, \mathsf{V})$ described as follows:

- $\mathsf{crs}_{\mathsf{zk}} \leftarrow \mathsf{Gen}_{\mathsf{zk}}(1^\lambda)$: It takes as input the security parameter and outputs a uniformly distributed "ZK mode" CRS $\mathsf{crs}_{\mathsf{zk}}$.
- $(\mathsf{crs}_{\mathsf{sound}}, \mathsf{td}) \leftarrow \mathsf{Gen}_{\mathsf{sound}}(1^\lambda)$: It takes as input the security parameter and outputs a "soundness mode" CRS $\mathsf{crs}_{\mathsf{sound}}$ and a trapdoor td.
- **Round 1:** At the start of the protocol, the prover P and the verifier V receive a CRS crs and an instance $x \in \mathcal{L}$. The prover P additionally receives a witness ω. It computes a first round message α and sends it to V.
- **Round 2:** The verifier V sends a random challenge $\beta \in \{0, 1\}$ to P.
- **Round 3:** Upon receiving β, the prover P computes a third round message γ and sends it to the verifier V.
- **Verification:** Upon receiving γ, V either accepts or rejects the transcript (α, β, γ).

We require Σ to satisfy the following properties:

- **Completeness:** For any crs generated by $\mathsf{Gen}_{\mathsf{zk}}$ or $\mathsf{Gen}_{\mathsf{sound}}$, $x \in \mathcal{L}$ and any witness ω for x,

$$\Pr\left[\mathsf{Out}_{\mathsf{V}}\left(\mathsf{P}\left(\mathsf{crs}, x, \omega\right) \leftrightarrow \mathsf{V}\left(\mathsf{crs}, x\right)\right) = \mathsf{accept}\right] = 1,$$

  where $\mathsf{Out}_{\mathsf{V}}(e)$ is the output of V in a protocol execution e.

- **CRS Indistinguishability:** For any n.u. PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\lambda)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\mathsf{crs}_{\mathsf{zk}} \leftarrow \mathsf{Gen}_{\mathsf{zk}}(1^\lambda) : \mathcal{D}(1^\lambda, \mathsf{crs}_{\mathsf{zk}}) = 1\right] - \Pr\left[\mathsf{crs}_{\mathsf{sound}} \leftarrow \mathsf{LossyGen}(1^\lambda) : \mathcal{D}(1^\lambda, \mathsf{crs}_{\mathsf{sound}}) = 1\right]\right| \leq \nu(\lambda).$$

  Furthermore, we say the scheme satisfies sub-exponential CRS indistinguishability, if there exists a constant $0 < c < 1$ such that for any n.u. PPT distinguisher, the advantage $\nu(\lambda)$ is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

- **Adaptive Statistical Honest-Verifier Zero Knowledge:** There exists a PPT simulator S and a negligible function $\nu(\lambda)$ such that, for any unbounded adversary $\mathcal{A}$,

$$\left|\Pr\left[\mathsf{Real}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Ideal}(1^\lambda) = 1\right]\right| \leq \nu(\lambda),$$

  where the experiments Real and Ideal are described as follows:

$\underline{\mathsf{Real}(1^\lambda):}$	$\underline{\mathsf{Ideal}(1^\lambda):}$
$\mathsf{crs}_{\mathsf{zk}} \leftarrow \mathsf{Gen}_{\mathsf{zk}}(1^\lambda)$.	$\mathsf{crs}_{\mathsf{zk}} \leftarrow \mathsf{Gen}_{\mathsf{zk}}(1^\lambda)$.
$(x, \omega) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{zk}})$.	$(x, \omega) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{zk}})$.
If $\mathcal{R}(x, \omega) \neq 1$, output 0.	If $\mathcal{R}(x, \omega) \neq 1$ output 0.
$\alpha \leftarrow \mathsf{P}(1^\lambda, \mathsf{crs}_{\mathsf{zk}}, x, \omega)$.	$\beta \leftarrow \{0,1\}$.
$\beta \leftarrow \{0,1\}$, and $\gamma \leftarrow \mathsf{P}(\beta)$.	$(\alpha, \gamma) \leftarrow \mathsf{S}(1^\lambda, \mathsf{crs}_{\mathsf{zk}}, x, \beta)$.
Output $\mathcal{A}(\alpha, \beta, \gamma)$.	Output $\mathcal{A}(\alpha, \beta, \gamma)$.

We say that the trapdoor sigma protocol satisfies adaptive **computational** honest-verifier zero knowledge, if the above condition holds for any non-uniform PPT adversary.

- **Bad Challenge Function in TC^0:** There exists a sequence of circuits $\{\mathsf{BadC}_\lambda\}_\lambda$ in TC^0, and a *deterministic* polynomial time algorithm $\mathsf{PreComp}(\cdot,\cdot)$, such that for any $(\mathsf{crs}_{\mathsf{sound}}, \mathsf{td}) \leftarrow \mathsf{Gen}_{\mathsf{sound}}(1^\lambda)$, instance $x \notin \mathcal{L}$ and any accepting transcript (α, β, γ) for x,

$$\beta = \mathsf{BadC}_\lambda(\mathsf{PreComp}(1^\lambda, \alpha), \mathsf{td}).^{8}$$

- **Knowledge Extraction:** There exists a polynomial time extractor Ext such that, for any soundness mode CRS $(\mathsf{crs}_{\mathsf{sound}}, \mathsf{td}) \leftarrow \mathsf{Gen}_{\mathsf{sound}}(1^\lambda)$, any instance $x \in \{0,1\}^*$, and any accepting transcript (α, β, γ), if $\beta \neq \mathsf{BadC}_\lambda(\mathsf{PreComp}(1^\lambda, \alpha), \mathsf{td})$, then

$$\Pr\left[\omega \leftarrow \mathsf{Ext}(\alpha, \beta, \gamma, \mathsf{td}) : \mathcal{R}(x, \omega) = 1\right] = 1.$$

Remark 7.4. *The bad challenge function we define above satisfies the "instance-universality" property [15]. Namely, the bad challenge function does not depend on the instance x. As in [15], we rely on this property to achieve adaptive soundness for our computational NIZK construction.*

7.2.1 Construction

We construct our desired trapdoor sigma protocol for NP by instantiating the commitment scheme in the trapdoor sigma protocol of [15] (Construction 3.1) with the lossy public key encryption from Section 7.1.

Protocol from [15], Slightly Modified. Brakerski et al. [15] constructed a so-called "commit-and-open" trapdoor sigma protocol where in the first round, the prover commits to a string bit-by-bit, and then in the third round, the prover opens some positions of the commitments and provides some other information to complete the proof. The crucial property of their construction, that we shall use, is that the bad challenge function only consists of the following two computations: extraction from the commitments sent by the prover, and verification of a 3-CNF. While the first step is common to trapdoor sigma protocols, the second step is due to the use of Cook-Levin theorem in [15] (for reducing the depth of the bad challenge function). Recall that for any polynomial size circuit $C(\cdot)$, from the Cook-Levin theorem, we can convert it efficiently to a 3-CNF $\Phi_C(\cdot,\cdot)$ such that, for any input x, $C(x) = 1$ if and only if $\Phi_C(x,\cdot)$ is satisfiable. Furthermore, for any $C(x) = 1$, we can compute efficiently a witness ω' such that $\Phi_C(x, \omega') = 1$.

We now briefly describe the trapdoor sigma protocol, which is slightly modified from [15].

[8] Note that this implies that for any false instance $x \notin \mathcal{L}$ and any α, there is a unique β for which there exists an accepting γ.

- **CRS Generation:** The CRS crs contains a commitment key. The commitment key is generated with a trapdoor td that allows one to extract the message from the commitment.
- **First Round:** The prover prepares the first round message α in Blum's protocol [9]. In addition, the prover prepares the third round response γ_1 for the challenge bit $\beta = 1$. Let $C(\alpha, \gamma_1) = \mathsf{V}(\mathsf{crs}, \alpha, 1, \gamma_1)$ be the verification circuit for the challenge bit $\beta = 1$ in Blum's protocol.[9] The prover applies a Cook-Levin reduction to C and the assigment (α, γ_1), and obtains a 3-CNF $\Phi_C(\cdot, \cdot)$ and a witness ω'. Then the prover sends the bit-by-bit commitments $c_{\gamma_1} \leftarrow \mathsf{Com}(\gamma_1), c_{\omega'} \leftarrow \mathsf{Com}(\omega')$ and α to the verifier.
- **Second Round:** The verifier sends a random challenge $\beta \leftarrow \{0, 1\}$ to the prover.
- **Third Round:** Upon receiving the random challenge $\beta \in \{0, 1\}$, if $\beta = 0$, then the prover responds with the third round message in the Blum's protocol. If $\beta = 1$, the prover opens the commitments to γ_1 and ω', and sends γ_1, ω' and their openings to the verifier.
- **Verification:** Given the transcript, if $\beta = 0$, the verifier performs the same verification as in Blum's protocol. If $\beta = 1$, the verifier checks if the openings of γ_1, ω' are correct, and checks if $\Phi_C((\alpha, \gamma_1), \omega') = 1$.
- **Bad Challenge Function:** It proceeds in the following two steps:
  - **Extraction:** It uses td to extract the (γ_1, ω') from the commitments.
  - **Verfication:** If $\Phi_C((\alpha, \gamma_1), \omega') = 1$, then output 1, otherwise output 0.

Our Construction. We construct our desired trapdoor sigma protocol Σ for NP by instantiating the commitment scheme Com in the above protocol with the lossy public key encryption LPKE in Section 7.1. A CRS of the resulting protocol contains a public key of LPKE; when the public key is lossy (resp., injective), the CRS is in ZK (resp., soundness) mode.

We now prove that the resulting protocol satisfies our required properties. The CRS indistinguishability follows directly from the key indistinguishability of LPKE. The completeness property follows from the completeness of the underlying Blum's protocol and the Cook-Levin theorem. The construction in [15] is proven to achieve adaptive computational zero-knowledge property. Here, we observe that when we instantiate the commitment scheme with LPKE, the resulting construction achieves adaptive *statistical* zero knowledge property since LPKE satisfies statistical semantic security in lossy mode.

Next, we argue that the construction satisfies the bad challenge function in TC^0 and the knowledge extraction properties.

Bad Challenge Function in TC^0. As described above, the bad challenge function for the trapdoor sigma protocol of [15] consists of two steps: extraction from the commitments, and an evaluation of Φ_C. For our instantiation, the first step can be computed as $\mathsf{LPKE.Dec}_\lambda(\mathsf{LPKE.PreComp}(1^\lambda, \cdot), \mathsf{td})$, where $\{\mathsf{LPKE.Dec}_\lambda\}_\lambda$ is the sequence of decryption circuits in TC^0 for LPKE and LPKE.PreComp is the associated deterministic pre-computation algorithm. Furthermore, the second step that involves evaluation of Φ_C can be computed in AC^0.

We therefore define BadC_λ and PreComp for our trapdoor sigma protocol as follows:

$$\mathsf{PreComp}(1^\lambda, (c_{\gamma_1}, c_{\omega'}, \alpha)) = (\mathsf{LPKE.PreComp}(1^\lambda, c_{\gamma_1}), \mathsf{LPKE.PreComp}(1^\lambda, c_{\omega'}), \alpha),$$
$$\mathsf{BadC}_\lambda((c'_{\gamma_1}, c'_{\omega'}, \alpha), \mathsf{td}) = \Phi_C((\alpha, \mathsf{LPKE.Dec}_\lambda(c'_{\gamma_1}, \mathsf{td})), \mathsf{LPKE.Dec}_\lambda(c'_{\omega'}, \mathsf{td})).$$

From the above, it follows that $\mathsf{BadC}_\lambda \in \mathsf{TC}^0$.

[9] Here we assume that the verification circuit does not take the instance x as input for the challenge $\beta = 1$. Recall that Blum's protocol only checks whether the committed graph is a cycle graph for one of the challenge. Hence such a property can be achieved.

Knowledge Extraction. To argue knowledge extraction, we leverage the special soundness of Blum's protocol. Recall that special soundness guarantees that given two accepting transcripts with different challenges where the first round messages are the same, one can efficiently extract a witness.

For any accepting transcript (α', β, γ) with $\beta \neq \mathsf{BadC}_\lambda(\mathsf{PreComp}(1^\lambda, \alpha'), \mathsf{td})$, where $\alpha' = (c_{\gamma_1}, c_{\omega'}, \alpha)$, we consider two cases.

- **Case 1** $\beta = 0$**:** Then the bad challenge function outputs 1. We first execute the extraction step in the bad challenge function, and obtain (γ_1, ω'). As the bad challenge function outputs 1, γ_1 is an accepting witness for the Blum's protocol. Since γ is the other accepting response for the challenge $\beta = 0$ for Blum's protocol, we obtain two accepting transcripts, and hence can extract a witness via the special soundness property of Blum's protocol.

- **Case 2** $\beta = 1$**:** Then the bad challenge function outputs 0. We will argue that this is impossible. Recall that, γ contains (γ_1, ω') and the openings with respect to their commitments. Since the transcript is accepting, $\Phi_C((\alpha, \gamma_1), \omega') = 1$ and the openings are accepted. Recall that, once we instantiate the commitment scheme with our lossy public key encryption, the opening contains the randomness used in the encryption. Hence, since the openings are accepted, the bad challenge function also extracts the same (γ_1, ω'). However, the output value of the bad challenge function implies that $\Phi_C((\alpha, \gamma_1), \omega') = 0$, which contradicts $\Phi_C((\alpha, \gamma_1), \omega') = 1$. Hence, this case is impossible.

From the above, we have that only the first case is possible. This concludes the proof of the knowledge extraction property.

7.3 Non-Interactive Statistical Witness Indistinguishable Argument for NP

We construct a non-interactive statistical witness indistinguishable (NISWI) argument system $\Pi = (\mathsf{CGen}, \mathsf{P}, \mathsf{V})$ for NP in the common random string model with adaptive statistical witness indistinguishability and (sub-exponential) non-adaptive argument of knowledge properties.

Our construction relies on the following two ingredients:

- A trapdoor sigma protocol $\Sigma = (\Sigma.\mathsf{Gen}_{\mathsf{zk}}, \Sigma.\mathsf{Gen}_{\mathsf{sound}}, \Sigma.\mathsf{P}, \Sigma.\mathsf{V})$ for $\mathcal{L}$. Let $\{\Sigma.\mathsf{BadC}_\lambda\}_\lambda$ be the family of bad challenge functions in TC^0 and $\Sigma.\mathsf{PreComp}$ be the deterministic "pre-computation" algorithm associated with Σ, and let $\Sigma.\mathsf{Ext}$ be the knowledge extractor associated with Σ.

- A correlation intractable hash function $\mathsf{CIH} = (\mathsf{CIH.Gen}, \mathsf{CIH.Hash})$ for TC^0. Furthermore, we require that the CIH satisfies sub-exponential correlation intractability.

Construction of Π. Our scheme Π is described in Figure 8.

Lemma 7.5 (Completeness)**.** *The proposed scheme Π satisfies completeness.*

Proof. Let $\pi = (\{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$ be any proof generated by an honest prover for an instance $x \in \mathcal{L}$. From the completeness of the sigma protocol Σ, we have that

$$\Pr\left[\mathsf{out}_i \leftarrow \Sigma^{(i)}.\mathsf{V}(\mathsf{crs}_{\mathsf{zk}}, x, (\alpha_i, \beta_i, \gamma_i)) : \mathsf{out}_i = 1\right] = 1.$$

Hence, the verifier accepts the proof with probability 1. □

Lemma 7.6 (Adaptive SWI)**.** *The proposed scheme Π is adaptive statistical witness indistinguishable.*

Proof. We prove the adaptive statistical WI property by constructing a series of hybrids.

NISWI Argument System Π for NP

CRS Generation $\mathsf{CGen}(1^\lambda)$**:** Sample a CIH key $\mathsf{k} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$, and a CRS $\mathsf{crs}_{\mathsf{zk}} \leftarrow \Sigma.\mathsf{Gen}_{\mathsf{zk}}(1^\lambda)$. Output $\mathsf{crs} = (\mathsf{k}, \mathsf{crs}_{\mathsf{zk}})$.

Prover $\mathsf{P}(\mathsf{crs}, x, \omega)$**:** The prover receives as input a CRS $\mathsf{crs} = (\mathsf{k}, \mathsf{crs}_{\mathsf{zk}})$, an instance x and a witness ω where $\mathcal{R}(x, \omega) = 1$.

The prover runs $w = \lambda$ copies of the trapdoor sigma protocol Σ, denoted as $\Sigma^{(1)}, \ldots, \Sigma^{(w)}$ in parallel:

- Compute first round prover messages: $\alpha_i \leftarrow \Sigma^{(i)}.\mathsf{P}(1^\lambda, \mathsf{crs}_{\mathsf{zk}}, x, \omega)$.
- Compute verifier challenges:

$$\{\beta_i\}_{i\in[w]} \leftarrow \mathsf{CIH.Hash}(\mathsf{k}, \{\Sigma^{(i)}.\mathsf{PreComp}(1^\lambda, \alpha_i)\}_{i\in[w]}).$$

- Compute third round prover messages: $\gamma_i \leftarrow \Sigma^{(i)}.\mathsf{P}(\beta_i)$.

It outputs the proof $\pi = \left(\{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]}\right)$.

Verifier $\mathsf{V}(\mathsf{crs}, x, \pi)$**:** The verifier takes as input a CRS $\mathsf{crs} = (\mathsf{k}, \mathsf{crs}_{\mathsf{zk}})$, an instance x and a proof π. It performs the following steps:

- Parse the proof $\pi = \left(\{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]}\right)$.
- Compute verifier challenges:

$$\{\beta_i\}_{i\in[w]} \leftarrow \mathsf{CIH.Hash}\left(\mathsf{k}, \left\{\Sigma^{(i)}.\mathsf{PreComp}(1^\lambda, \alpha_i)\right\}_{i\in[w]}\right).$$

- For every $i \in [w]$, verify the transcript $(\alpha_i, \beta_i, \gamma_i)$ of $\Sigma^{(i)}$: $\mathsf{out}_i \leftarrow \Sigma^{(i)}.\mathsf{V}\left(\mathsf{crs}_{\mathsf{zk}}, x, (\alpha_i, \beta_i, \gamma_i)\right)$. If any $\mathsf{out}_i = 0$, then output reject. Otherwise output accept.

Figure 8: NISWI Argument System Π for NP

- Hyb_0**:** This hybrid is simply the experiment Expr_0 in the definition of adaptive SWI. Let (ω_0, ω_1) be the two witnesses chosen by the adversary.
- $\mathsf{Hyb}_1^{i^*}$**:** This hybrid is the same as Hyb_0, except that for each $i < i^*$, we compute (α_i, γ_i) using ω_1. For each $i \geq i^*$, we compute (α_i, γ_i) using ω_0.
- $\mathsf{Hyb}_2^{i^*}$**:** This hybrid is the same as $\mathsf{Hyb}_1^{i^*}$, except that, before the output, we guess a random $\beta \leftarrow \{0, 1\}$. Let $\{\beta_i\}_{i\in[w]} = \mathsf{CIH.Hash}(\mathsf{k}, \{\Sigma^{(i)}.\mathsf{PreComp}(1^\lambda, \alpha_i)\}_{i\in[w]})$. If $\beta = \beta_{i^*}$, then proceed to output. Otherwise, start running the hybrid from the beginning again. If the guessing process fails for λ times, then abort.
- $\mathsf{Hyb}_3^{i^*}$**:** This hybrid is the same as $\mathsf{Hyb}_2^{i^*}$, except that for $i = i^*$, we run the simulator on the guessed challenge β to compute $(\alpha_{i^*}, \gamma_{i^*}) \leftarrow \Sigma.\mathsf{S}(1^\lambda, \mathsf{crs}_{\mathsf{zk}}, x, \beta)$.

- Hyb_4 This hybrid is identical to the experiment Expr_1, where all (α_i, γ_i) tuples are computed using ω_1.

We now show that Hyb_0 and Hyb_4 are statistically indistinguishable.

$\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1^1$: This follows from the definition of these two hybrids.

$\mathsf{Hyb}_1^{i^*} \approx_s \mathsf{Hyb}_2^{i^*}$: Since β is sampled uniformly at random, it is independent of β_{i^*}. Hence each guess in $\mathsf{Hyb}_2^{i^*}$ is correct with probability $1/2$, and thus the hybrid aborts with probability $1/2^\lambda$. Conditioned on the event that $\mathsf{Hyb}_2^{i^*}$ doesn't abort, its output distribution is identical to $\mathsf{Hyb}_1^{i^*}$. Hence, the statistical distance between $\mathsf{Hyb}_1^{i^*}$ and $\mathsf{Hyb}_2^{i^*}$ is at most $1/2^\lambda$.

$\mathsf{Hyb}_2^{i^*} \approx_s \mathsf{Hyb}_3^{i^*}$: This follows from the adaptive statistical honest verifier zero-knowledge property of Σ.

$\mathsf{Hyb}_3^{i^*} \approx_s \mathsf{Hyb}_1^{i^*+1}$: This also follows from the adaptive statistical honest verifier zero-knowledge property of Σ.

$\mathsf{Hyb}_1^{w+1} \equiv \mathsf{Hyb}_4$: This follows from the definition of the hybrids above.

□

Lemma 7.7 (Non-adaptive Argument of Knowledge)**.** *The proposed scheme* Π *satisfies sub-exponential non-adaptive argument of knowledge.*

Proof. Let Ext be the extractor in the knowledge extraction property of the trapdoor sigma protocol. We build the extractor in Figure 9.

Extractor $E_1(1^\lambda)$

- Sample $\mathsf{k} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$, $(\mathsf{crs}_{\mathsf{sound}}, \mathsf{td}) \leftarrow \Sigma.\mathsf{Gen}_{\mathsf{sound}}(1^\lambda)$.
- Output $(\mathsf{crs} = (\mathsf{k}, \mathsf{crs}_{\mathsf{sound}}), \mathsf{td})$.

Extractor $E_2(\mathsf{td}, x, \pi)$

- Parse $\pi = (\{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$.
- Let $\{\beta_i\}_{i\in[w]} = \mathsf{CIH.Hash}\left(\mathsf{k}, \left\{\Sigma.\mathsf{PreComp}(1^\lambda, \alpha_i)\right\}_{i\in[w]}\right)$.
- If $\beta_i = \Sigma.\mathsf{BadC}_\lambda(\Sigma.\mathsf{PreComp}(1^\lambda, \alpha_i), \mathsf{td})$ for all $i \in [w]$, then abort.
- Let i^* be the smallest i such that $\beta_i \neq \Sigma.\mathsf{BadC}_\lambda(\Sigma.\mathsf{PreComp}(1^\lambda, \alpha_i), \mathsf{td})$.
- Output $\Sigma.\mathsf{Ext}(\alpha_{i^*}, \beta_{i^*}, \gamma_{i^*}, \mathsf{td})$.

Figure 9: Extractor $E = (E_1, E_2)$.

Since the only difference between the CRS generated by CGen and the CRS obtained from E_1 is that, CGen invokes $\Sigma.\mathsf{Gen}_{\mathsf{zk}}$ while E_1 invokes $\Sigma.\mathsf{Gen}_{\mathsf{sound}}$, by the sub-exponential CRS indistinguishability, for any n.u. PPT cheating prover P^* we have

$$\Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow E_1(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] \geq$$

$$\Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), \mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] - \nu(\lambda),$$

where $\nu(\lambda)$ is a sub-exponential function.

By the correlation intractability of CIH, there exists a sub-exponential function ν' such that

$$\Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow E_1(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : E_2(\mathsf{td}, x, \pi) \text{ abort}\right] \leq \nu'(\lambda).$$

Hence, we have

$$\begin{aligned}\Pr[x \leftarrow \mathsf{P}^*(1^\lambda), (\mathsf{crs}, \mathsf{td}) \leftarrow E_1(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1 \wedge E_2(\mathsf{td}, x, \pi) \text{ does not abort}] \geq \\ \Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), \mathsf{crs} \leftarrow \mathsf{CGen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] - \nu(\lambda) - \nu'(\lambda),\end{aligned}$$

When $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, we have that $(\alpha_i, \beta_i, \gamma_i)$ is accepted, for any $i \in [w]$. By the knowledge extraction property of the trapdoor sigma protocol, when E_2 does not abort, we have $\mathcal{R}(x, \omega) = 1$. Hence, we finish the proof. □

7.4 From NISWI to Multi-Theorem NIZK

We now provide a general transformation from a statistical NISWI argument system for NP with non-adaptive argument of knowledge property to an adaptive, multi-theorem statistical NIZK argument system for NP with non-adaptive soundness. Our transformation relies on the hardness of discrete logarithm and uses a slight variant of the "OR-trick" from [32].

Construction. Let $\Pi' = (\Pi'.\mathsf{Gen}, \Pi'.\mathsf{P}, \Pi'.\mathsf{V})$ be a NISWI argument system for an NP-Complete language $\mathcal{L}'$ with (sub-exponential) non-adaptive argument of knowledge property. Let $\mathcal{G}$ be a prime-order group generator for which discrete logarithm is hard.

We build a NISZK argument system $\Pi = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ for any NP language $\mathcal{L}$, with adaptive statistical zero-knowledge and (sub-exponential) non-adaptive computational soundness property. The construction is described in Figure 10.

Lemma 7.8 (Completeness)**.** *The proposed scheme* Π *satisfies completeness.*

Proof. For any instance $x \in \mathcal{L}$, and any witness ω of x, by the definition of $\mathcal{L}_{\mathsf{or}}$, we have that x_{or} constructed in Figure 10 is in $\mathcal{L}_{\mathsf{or}}$, and $\omega_{\mathsf{or}} = (\omega, 0)$ is a witness for x_{or}. Hence, the completeness follows from the completeness of the underlying protocol Π'. □

Lemma 7.9 (Multi-Theorem Adaptive Statistical Zero-Knowledge)**.** *The proposed scheme* Π *is multi-theorem adaptive statistical zero knowledge.*

Proof. We build the following simulator $\mathsf{S} = (\mathsf{S}_1, \mathsf{S}_2)$.

- $\underline{\mathsf{S}_1(1^\lambda)}$:
  - Sample a CRS of Π': $\Pi'.\mathsf{crs} \leftarrow \Pi'.\mathsf{Gen}(1^\lambda)$.
  - Generate a prime order group $(\mathbb{G}, p, g) \leftarrow \mathsf{G}(1^\lambda)$.
  - Sample $t \leftarrow \mathbb{Z}_p$, and let $h = g^t$.
  - Output $\mathsf{crs} = (\Pi'.\mathsf{crs}, (\mathbb{G}, p, g, h))$, and $\mathsf{td} = (t, \mathsf{crs})$.
- $\underline{\mathsf{S}_2(\mathsf{td}, x)}$:

NISZK Argument System Π for Language $\mathcal{L}$.

- **CRS Generation** $\mathsf{Gen}(1^\lambda)$**:**
  - Sample a CRS for Π': $\Pi'.\mathsf{crs} \leftarrow \Pi'.\mathsf{Gen}(1^\lambda)$.
  - Generate a prime order group $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$.
  - Sample $h \leftarrow \mathbb{G}$ uniformly at random.
  - Output $\mathsf{crs} = (\Pi'.\mathsf{crs}, (\mathbb{G}, p, g, h))$
- **Prover** $\mathsf{P}(\mathsf{crs}, x, \omega)$**:**
  - Let $\mathcal{L}_{\mathsf{or}}$ be an NP language, where an instance $(x, (\mathbb{G}, p, g, h)) \in \mathcal{L}_{\mathsf{or}}$ if and only if there exists a witness (ω, a) such that either ω is a witness for $x \in \mathcal{L}$ or $g^a = h$.
  - Let $x_{\mathsf{or}} = (x, (\mathbb{G}, p, g, h))$ and $\omega_{\mathsf{or}} = (\omega, 0)$. Use NP reduction on $(x_{\mathsf{or}}, \omega_{\mathsf{or}})$ to obtain an instance $x' \in \mathcal{L}'$ and witness ω' for x'.
  - Compute a proof for Π': $\pi \leftarrow \Pi'.\mathsf{P}(\Pi'.\mathsf{crs}, x', \omega')$.
  - Output π.
- **Verifier** $\mathsf{V}(\mathsf{crs}, x, \pi)$**:**
  - Parse $\mathsf{crs} = (\Pi'.\mathsf{crs}, (\mathbb{G}, p, g, h))$.
  - Let $x_{\mathsf{or}} = (x, (\mathbb{G}, p, g, h))$. Use NP reduction on x_{or} to obtain x'.
  - Output $\mathsf{out} \leftarrow \Pi'.\mathsf{V}(\Pi'.\mathsf{crs}, x', \pi)$.

Figure 10: NISZK argument system Π for language $\mathcal{L}$.

- Let $x_{\mathsf{or}} = (x, (\mathbb{G}, p, g, h))$, and $\widetilde{\omega}_{\mathsf{or}} = (0, t)$. Use NP reduction on $(x_{\mathsf{or}}, \widetilde{\omega}_{\mathsf{or}})$ to obtain an instance $x' \in \mathcal{L}'$ and witness $\widetilde{\omega}'$ for x'. Run the prover algorithm of Π' to compute:

$$\pi \leftarrow \Pi'.\mathsf{P}(\Pi'.\mathsf{crs}, x', \widetilde{\omega}').$$

- Output π.

Since t is sampled uniformly at random from $\mathbb{Z}_p$ and g is a generator of $\mathbb{G}$, g^t is uniformly distributed over $\mathbb{G}$. Hence, the distribution of crs generated by the simulator is identical to that in the real execution. Then, the adaptive statistical zero knowledge property of Π follows from the adaptive statistical witness indistinguishability of Π'. □

Lemma 7.10 (Non-adaptive Computational Soundness)**.** *Assuming (sub-exponential) hardness of discrete logarithm, the proposed scheme* Π *satisfies (sub-exponential) non-adaptive computational soundness.*

Proof. Suppose there exists a n.u. PPT adversary P^*, and a non-negligible function $\epsilon(\lambda)$ such that

$$\Pr\left[x \leftarrow \mathsf{P}^*(1^\lambda), \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : x \notin \mathcal{L} \wedge \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] > \epsilon(\lambda),$$

for infinite many $\lambda \in \mathbb{N}$. Then there exists infinite many λ such that, there exists a random coin r_λ and

$x_\lambda = \mathsf{P}^*(1^\lambda; r_\lambda)$ such that

$$\Pr\left[x_\lambda = \mathsf{P}^*(1^\lambda; r_\lambda), \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}) : x_\lambda \notin \mathcal{L} \wedge \mathsf{V}(\mathsf{crs}, x_\lambda, \pi) = 1\right] > \epsilon(\lambda),$$

where the probability is only over the randomness of $\mathsf{Gen}(1^\lambda)$ and $\mathsf{P}^*(\mathsf{crs})$. Since the probability is greater than 0, $x_\lambda \notin \mathcal{L}$.

We build the following non-uniform cheating prover P' for Π':

- $\mathsf{P}'(1^\lambda)$ computes $x_\lambda = \mathsf{P}^*(1^\lambda; r_\lambda)$, and generates a prime order group $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$. Then it samples $h \leftarrow \mathbb{G}$ and outputs $x_{\mathsf{or}} = (x_\lambda, (\mathbb{G}, p, g, h))$.
- Upon receiving input $\Pi'.\mathsf{crs}$, P' sets $\mathsf{crs} = (\Pi'.\mathsf{crs}, (\mathbb{G}, p, g, h))$. It computes $\pi \leftarrow \mathsf{P}^*(\mathsf{crs})$ and outputs π.

From the non-adaptive argument of knowledge property of Π', there exists an extractor $E = (E_1, E_2)$ and a negligible function $\nu(\lambda)$ such that

$$\Pr[x_{\mathsf{or}} \leftarrow \mathsf{P}'(1^\lambda), (\Pi'.\mathsf{crs}, \mathsf{td}) \leftarrow E_1(1^\lambda), \pi \leftarrow \mathsf{P}'(\Pi'.\mathsf{crs}), \omega_{\mathsf{or}} \leftarrow E_2(\mathsf{td}, x_{\mathsf{or}}, \pi) : \mathcal{R}_{\mathsf{or}}(x_{\mathsf{or}}, \omega_{\mathsf{or}}) = 1] \tag{6}$$

$$\geq \Pr[x_{\mathsf{or}} \leftarrow \mathsf{P}'(1^\lambda), \Pi'.\mathsf{crs} \leftarrow \Pi'.\mathsf{Gen}(1^\lambda), \pi \leftarrow \mathsf{P}'(\Pi'.\mathsf{crs}) : \Pi'.\mathsf{V}(\Pi'.\mathsf{crs}, x_{\mathsf{or}}, \pi) = 1] - \nu(\lambda) \geq \epsilon(\lambda) - \nu(\lambda), \tag{7}$$

where the relation $\mathcal{R}_{\mathsf{or}}(x_{\mathsf{or}}, \omega_{\mathsf{or}}) = 1$, if and only if ω_{or} is a witness for the instance $x_{\mathsf{or}} \in \mathcal{L}_{\mathsf{or}}$.

Next, we build the following adversary $\mathcal{A}$ for the discrete logarithm problem:

- $\mathcal{A}$ receives as input a security parameter λ, a group $\mathbb{G}$ and its order p, a generator g, and $h \in \mathbb{G}$.
- It computes $x_\lambda = \mathsf{P}^*(1^\lambda; r_\lambda)$, and $x_{\mathsf{or}} = (x_\lambda, (\mathbb{G}, p, g, h))$.
- Next, it computes $(\Pi'.\mathsf{crs}, \mathsf{td}) \leftarrow E_1(1^\lambda)$, and sets $\mathsf{crs} = (\Pi'.\mathsf{crs}, (\mathbb{G}, p, g, h))$.
- Finally it computes $\pi \leftarrow \mathsf{P}^*(\mathsf{crs})$ and $\omega_{\mathsf{or}} \leftarrow E_2(\mathsf{td}, x_{\mathsf{or}}, \pi)$. It parses $\omega_{\mathsf{or}} = (\omega, t)$, and outputs t.

When the input to $\mathcal{A}$ is computed $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$ and $h \leftarrow \mathbb{G}$, the distribution of $\mathsf{crs} = (\Pi'.\mathsf{crs}, (\mathbb{G}, p, g, h))$ is identical to the distribution of the $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ in the real execution of the protocol Π. Hence, the adversary $\mathcal{A}$ correctly simulates the cheating prover P'. Since $x_\lambda \notin \mathcal{L}$, if $\mathcal{R}_{\mathsf{or}}(x_{\mathsf{or}}, \omega_{\mathsf{or}} = (\omega, t)) = 1$, then $g^t = h$. Therefore,

$$\begin{aligned}\Pr\left[(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda), h \leftarrow \mathbb{G}, t \leftarrow \mathcal{A}(1^\lambda, \mathbb{G}, p, g, h) : g^t = h\right] \\ \geq \Pr[x_{\mathsf{or}} \leftarrow \mathsf{P}'(1^\lambda), (\Pi'.\mathsf{crs}, \mathsf{td}) \leftarrow E_1(1^\lambda), \pi \leftarrow \mathsf{P}'(\Pi'.\mathsf{crs}), \\ \omega_{\mathsf{or}} \leftarrow E_2(\mathsf{td}, x_{\mathsf{or}}, \pi) : \mathcal{R}_{\mathsf{or}}(x_{\mathsf{or}}, \omega_{\mathsf{or}}) = 1] \geq \epsilon(\lambda) - \nu(\lambda),\end{aligned}$$

for infinite many λ, which is a contradiction.

□

7.5 Computational NIZKs for NP with Adaptive Soundness

In this section, we describe our construction of computational NIZK for NP with adaptive soundness in the common random string model. We proceed via the same steps as in the construction of statistical NIZK for NP, except that we replace the lossy public-key encryption scheme in the construction of trapdoor sigma protocol with Elgamal encryption.

We outline each of the steps below.

Lemma 7.11. *If we replace the lossy public key encryption in the trapdoor sigma protocol in Section 7.2 with ElGamal encryption [31], then the resulting trapdoor sigma protocol $\overline{\Sigma}$ satisfies adaptive computational honest-verifier zero-knowledge and the bad challenge in TC^0 property. Furthermore, the resulting protocol is in the common random string model.*

Proof sketch. The proof of adaptive computational honest-verifier zero-knowledge follows from the computational semantic security of ElGammal encryption scheme.

For the bad challenge in TC^0 property, recall that a ciphertext in ElGamal encryption scheme is of the form $(c_1, c_2) \in \mathbb{G}^2$ and a secret key $\mathsf{sk} = s$ is an element in $\mathbb{Z}_p$. Furthermore, the decryption process simply involves computing $c_1^s \cdot c_2$, which is the same as in the case of lossy public-key encryption scheme in Section 7.1. Hence, using the same argument as in Lemma 7.3, it follows that the ElGamal encryption scheme also satisfies low-depth decryption property. Then, following the same argument as in Section 7.2, we have that $\overline{\Sigma}$ satisfies the bad challenge in TC^0 property.

Finally, we note that since a public key of ElGamal encryption scheme is of the form (g^s, g) where $g \leftarrow \mathbb{G}$, the public key is uniformly distributed over $\mathbb{G}^2$. Hence, $\overline{\Sigma}$ is in the common random string model. □

Theorem 7.12. *If we replace the trapdoor sigma protocol in Π with $\overline{\Sigma}$ obtained from Lemma 7.11 then the resulting protocol $\overline{\Pi}$ satisfies adaptive computational witness indistinguishability and (sub-exponential) adaptive argument of knowledge in the common random string model.*

Proof sketch. The proof of computational witness indistinguishability relies on the computational honest-verifier zero-knowledge property, and its proof follows the same idea as Lemma 7.6. The adaptive argument of knowledge property follows the same argument as in Lemma 7.7. □

Theorem 7.13. *In Figure 10, if we replace the NISWI Π' with the computational NIWI $\overline{\Pi}$ obtained from Theorem 7.12 then the resulting protocol satisfies adaptive multi-theorem adaptive computational zero-knowledge and (sub-exponential) adaptive computational soundness in the common random string model.*

The proof follows in the same manner as in Lemma 7.9 and Lemma 7.10.

8 Statistical Zap Arguments for NP

In this section, we describe our construction of statistical Zap arguments for NP. Our construction closely follows the recent works of [2, 37] who constructed statistical Zap arguments from quasi-polyomial hardness of LWE.

We proceed in the following two steps:

- In Section 8.1, we construct a two-round (public-coin) statistical-hiding commitment scheme with *low-depth extraction property*, which essentially requires that the extraction circuit is in TC^0.
- Next, in Section 8.2, we construct statistical Zap arguments by replacing the lossy public-key encryption scheme used in our construction of NISWI in Section 7.3 with the two-round commitment scheme from the previous step.

8.1 Statistical Hiding Commitments with Low-Depth Extraction

A (public-coin) statistical hiding commitment scheme with low-depth extraction (ECOM) is an interactive protocol between a receiver and a sender: in the first round, the receiver sends a commitment key to the sender. Using the key, the sender computes a commitment to some value and sends it to the receiver. We require such schemes with a statistical hiding property as well as a low-depth extraction property.

Here we are interested in two-round protocols – in the plain model – that achieve security against malicious receivers. More specifically, in such protocols, statistical hiding property is required to hold with respect to even adversarially chosen commitment keys. At the same time, we require the commitments to be extractable, i.e., it should be possibly to efficiently extract the committed value from the sender's message. The extraction property is only required to hold in a special "extraction mode" which is determined by choice of the commitment key.

The key to achieving these two properties simultaneously is to allow the "extraction mode" to happen with only negligible probability [42]. Following the syntax in [2], we provide an additional input – a random string $\mathbf{b}$ – to the commitment algorithm. The "extraction mode" key generation algorithm also takes a random string $\widetilde{\mathbf{b}}$ as an additional input. Consequently, extraction is only required when the strings $\mathbf{b}$ and $\widetilde{\mathbf{b}}$ are *equal*. As in prior works [42, 2, 37], this weak extraction guarantee suffices for our target application by careful use of complexity leveraging.

For our purposes, we further require that the extraction process can be represented via *low-depth* computation, namely, TC^0 circuits.

Definition. For any security parameter λ, let $\mathbb{G}$ denote a cyclic group of order $p = p(\lambda)$. Let $\mu = \mu(\lambda)$ be a polynomial in λ.

A group-based statistical hiding commitment scheme with low-depth extraction, and with message space $\mathbb{Z}_2$ and key space $\mathcal{K}$, is a tuple of algorithms $\mathsf{ECOM} = (\mathsf{Gen}, \mathsf{ExtGen}, \mathsf{Com})$ described as follows:

- $\mathsf{Gen}(1^\lambda)$: It takes as input a security parameter λ, and outputs a uniformly distributed "normal mode" commitment key K.
- $\mathsf{ExtGen}(1^\lambda, \widetilde{\mathbf{b}})$: It takes as input the security parameter λ, and a string $\widetilde{\mathbf{b}}$ of length μ, and outputs an "extraction mode" commitment key $\widetilde{K}$ and a trapdoor td.
- $\mathsf{Com}(\mathbf{b}, K, m; r)$: It takes as input the security parameter λ, an integer μ, a binary string $\mathbf{b} \in \{0,1\}^\mu$, a message m, and the random coins r, and outputs a commitment c.

We require ECOM to satisfy the following properties.

- **Key Verifiability:** There exists a PPT algorithm KeyVer such that, for any string K,

$$\Pr\left[\mathsf{KeyVer}(1^\lambda, K) = 1 \iff K \in \mathcal{K}\right] = 1.$$

- **Key Indistinguishability:** For any n.u. PPT distinguisher $\mathcal{D}$, there exists a negligible function $\nu(\cdot)$ such that, for any $\lambda \in \mathbb{N}$, any $\widetilde{\mathbf{b}} \in \{0,1\}^\mu$,

$$\left|\Pr\left[K \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{D}(1^\lambda, K) = 1\right] - \Pr\left[(\widetilde{K}, \mathsf{td}) \leftarrow \mathsf{ExtGen}(1^\lambda, \widetilde{\mathbf{b}}) : \mathcal{D}(1^\lambda, \widetilde{K}) = 1\right]\right| \leq \nu(\lambda).$$

  We say that the ECOM achieves sub-exponential key indistinguishability, if there exists a constants c such that for any n.u. PPT distinguisher, the advantage $\nu(\lambda)$ is bounded by $2^{-\lambda^c}$ for any sufficiently large λ.

- **Statistical Hiding:** For any key $K \in \mathcal{K}$, any $m_1, m_2 \in \mathbb{Z}_2$, there exists a negligible function $\nu(\cdot)$ such that, for any $\lambda \in \mathbb{N}$,

$$\mathsf{SD}\left((\mathbf{b}, \mathsf{Com}(\mathbf{b}, K, m_1)), (\mathbf{b}, \mathsf{Com}(\mathbf{b}, K, m_2))\right) \leq \nu(\lambda).$$

- **Low-Depth Extraction:** We say that ECOM supports extraction in TC^0 if there exists a sequence of circuits $\{\mathsf{Dec}_\lambda(\cdot,\cdot)\}_\lambda$ in TC^0 and a deterministic polynomial-time algorithm $\mathsf{PreComp}(1^\lambda,\cdot)$ such that, for any $\lambda \in \mathbb{N}$, any binary string $\widetilde{\mathbf{b}} \in \{0,1\}^\mu$, and extraction mode key $(\widetilde{K}, \mathsf{td}) \leftarrow \mathsf{ExtGen}(1^\lambda, \widetilde{\mathbf{b}})$, and any message $m \in \mathbb{Z}_2$,

$$\Pr\left[c \leftarrow \mathsf{Com}(\widetilde{\mathbf{b}}, \widetilde{K}, m) : m' \leftarrow \mathsf{Dec}_\lambda(\mathsf{PreComp}(1^\lambda, c), \mathsf{td}) : m = m'\right] = 1.$$

8.1.1 Construction

Our construction follows the work of [42] who constructed statistical hiding extractable commitments generically from two-round oblivious transfer protocols. To achieve the low-depth extraction property, we instantiate their construction with the Naor-Pinkas oblivious transfer scheme based on DDH [53].

- $\mathsf{Gen}(1^\lambda)$: Sample and output μ strings uniformly at random, where each string is of the same length as an OT receiver first round message. Let K denote the output.
- $\mathsf{ExtGen}(1^\lambda, \widetilde{\mathbf{b}})$: Parse $\widetilde{\mathbf{b}} = (\widetilde{b}_1, \widetilde{b}_2, \ldots, \widetilde{b}_\mu)$. Generate μ first round OT messages

  $$\mathsf{OT}_1(1^\lambda, \widetilde{b}_1), \mathsf{OT}_1(1^\lambda, \widetilde{b}_2), \ldots, \mathsf{OT}_1(1^\lambda, \widetilde{b}_\mu),$$

  and output them as commitment key $\widetilde{K}$. Let td denote the set of random strings used for computing the OT messages.
- $\mathsf{Com}(\mathbf{b}, K, m)$: Parse $\mathbf{b} = (b_1, b_2, \ldots, b_\mu)$ and $K = \{\mathsf{ot}_{1,i}\}_{i\in[\mu]}$.
  - Randomly sample $m_1, m_2, \ldots, m_\mu \in \mathbb{Z}_2$ such that $\sum m_i \bmod 2 = m$.
  - For each $i \in [\mu]$, compute the second round OT message $c_i = \mathsf{OT}_2(\mathsf{ot}_{1,i}, m_{i,0}, m_{i,1})$ with $m_{i,b_i} = m_i$, and sample $m_{i,1-b_i} \leftarrow \mathbb{Z}_2$ uniformly at random.
  - Output the second round messages $\{c_i\}_{i\in[\mu]}$ as the commitment.

The key verifiability follows directly from the underlying oblivious transfer protocol and the group instantiation. The key indistinguishability property follows from the pseudorandomness of receiver's message in Naor-Pinkas OT. The statistical hiding property follows the same argument as in [42, 2, 37]. Intuitively, for any key in the key space, one can use an inefficient extractor to extract $\widetilde{\mathbf{b}}$. Since $\mathbf{b}$ is sampled uniformly at random, with probability $1 - 2^{-\mu}$, there exists an index i such that $b_i \neq \widetilde{b}_i$. Hence, from the statistical sender-privacy of the OT, m_i is statistically hidden. Since $m_1, m_2, \ldots, m_\mu$ constitute an n-out-of-n secret sharing of m, m is also statistically hidden.

Low-Depth Extraction. When $\mathbf{b} = \widetilde{\mathbf{b}}$, to extract m, we need to proceed in the following two steps.

1. Use OT_3 and td to decrypt $m_1, m_2, \ldots, m_\mu$ from the commitment.
2. Compute $m = m_1 + m_2 + \ldots + m_\mu$ in $\mathbb{Z}_2$.

When we instantiate the OT with Naor-Pinkas OT, the output computation algorithm OT_3 is of the same form as the decryption algorithm of the lossy public key encryption in Section 7.1. Hence, it also satisfies the low-depth decryption property, and there exists a TC^0 circuit sequence $\{\mathsf{Dec}_\lambda\}_\lambda$ and a deterministic pre-computation algorithm $\mathsf{PreComp}$ such that $m_i = \mathsf{Dec}_\lambda(\mathsf{PreComp}(1^\lambda, c_i), \mathsf{td}_i)$, where td_i is the receiver's randomness for the i^{th} OT. For the second step, the summation of $m_1, m_2, \ldots, m_\mu$ in $\mathbb{Z}_2$ can be computed in TC^0 [60]. Composing these two steps, we prove the low-depth extraction property.

8.2 Construction of Statistical Zap Arguments

We now describe our construction of statistical Zap arguments for NP with non-adaptive soundness. We rely on the following ingredients:

- A statistical-hiding commitment scheme $\mathsf{ECOM} = (\mathsf{Gen}, \mathsf{ExtGen}, \mathsf{Com})$ with low-depth extraction and sub-exponential key indistinguishability.

  Let $\{\mathsf{ECOM.Dec}_\lambda\}$ be the sequence of low-depth decryption circuits and let ECOM.PreComp be the deterministic pre-computation algorithm associated with ECOM.

- A correlation intractable hash function CIH for TC^0, with sub-exponential correlation intractability.

- A "commit-and-open" trapdoor sigma protocol Σ for NP with low-depth bad challenge function, from Section 7.2.

Each of these ingredients can be based on sub-exponential DDH. We thus obtain our construction based on the same assumption.

Construction. Our construction and its security proof resembles the recent works of [2, 37]. Below, we sketch the construction and the proof of security. In the following, we set $\mu = \log^2 \lambda$.

- **Verifier:** It generates $K \leftarrow \mathsf{ECOM.Gen}(1^\lambda)$ and a CIH key $\mathsf{k} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$, and sends (K, k) to the prover.

- **Prover:** Upon receiving a message (K, k), it first checks if $K \in \mathcal{K}$ by checking $\mathsf{KeyVer}(1^\lambda, K) = 1$. If the check fails, then abort.

  Next, it randomly samples $\mathbf{b} \leftarrow \{0,1\}^\mu$, and emulates $w = \lambda$ executions of the trapdoor sigma protocol Σ in parallel, as follows.

  - For each $i \in [w]$, run the prover's first round algorithm for Σ to generate the first round message α_i, where the commitment scheme is instantiated as $\mathsf{ECOM.Com}(\mathbf{b}, K, \cdot; \cdot)$.
  - Compute the verfier's challenges by using CIH: $\{\beta_i\}_{i\in[w]} \leftarrow \mathsf{CIH.Hash}(\mathsf{k}, \{\mathsf{ECOM.PreComp}(1^\lambda, \alpha_i)\}_{i\in[w]})$.
  - For each $i \in [w]$, run the prover's third round algorithm for Σ, with challenge bit β, where the commitment is instantiated as $\mathsf{ECOM.Com}(\mathbf{b}, K, \cdot; \cdot)$. Let γ_i be the computed message.

  Finally, the prover sends $\pi = (\mathbf{b}, \{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$ to the verifier.

- **Verification:** Given the transcript $((K, \mathsf{k}), \pi)$, the verifier parses $\pi = (\mathbf{b}, \{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$, and computes

  $$\{\beta_i\}_{i\in[w]} = \mathsf{CIH.Hash}(\mathsf{k}, \{\mathsf{ECOM.PreComp}(1^\lambda, \alpha_i)\}_{i\in[w]}).$$

  For each $i \in [w]$, it verifies if $(\alpha_i, \beta_i, \gamma_i)$ is a accepting transcript for Σ, where the commitment is instantiated as $\mathsf{ECOM.Com}(\mathbf{b}, K, \cdot; \cdot)$.

This completes the description of the protocol. The completeness of the protocol directly follows from the completeness of the underlying sigma protocol Σ. The proof of statistical witness indistinguishability resembles the proof of Lemma 7.6. The only differences is that, here, for *any* $K \in \mathcal{K}$, when $\mathbf{b} \leftarrow \{0,1\}^\mu$ is sampled from uniform distribution, the commitment $\mathsf{ECOM.Com}(\mathbf{b}, K, \cdot; \cdot)$ is statistical hiding. Below, we argue that the protocol achieves non-adaptive computational soundness.

Non-adaptive Computational Soundness. For any instance $x \notin \mathcal{L}$, and any cheating prover P^* with success probability $\epsilon(\lambda)$, we build the following hybrids.

- Hyb_0: In this hybrid, the cheating prover tries to break the soundness of Zaps.
  - $K \leftarrow \mathsf{ECOM.Gen}(1^\lambda), \mathsf{k} \leftarrow \mathsf{CIH.Gen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(1^\lambda, (K, \mathsf{k}))$
  - If $\pi = (\mathbf{b}, \{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$ is accepted, then output 1. Otherwise output 0.

  Since the cheating prover succeeds with probability $\epsilon(\lambda)$, we have that $\Pr[\mathsf{Hyb}_0 = 1] = \epsilon(\lambda)$.

- Hyb_1: This hybrid is the same as Hyb_0, except that we additionally guess $\mathbf{b}$ by sampling $\widetilde{\mathbf{b}}$ uniformly at random.
  - $\widetilde{\mathbf{b}} \leftarrow \{0,1\}^\mu, K \leftarrow \mathsf{ECOM.Gen}(1^\lambda), \mathsf{k} \leftarrow \mathsf{CIH.Gen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(1^\lambda, (K, \mathsf{k}))$.
  - If $\pi = (\mathbf{b}, \{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$ is accepted and $\mathbf{b} = \widetilde{\mathbf{b}}$, then output 1. Otherwise output 0.

  Since the only difference between this hybrid and Hyb_0 is that we additionally guess $\widetilde{\mathbf{b}} \leftarrow \{0,1\}^\mu$, we have
  $$\Pr[\mathsf{Hyb}_1 = 1] \geq \epsilon(\lambda)/2^\mu.$$

- Hyb_2: This hybrid is the same as Hyb_1, except that we switch the K to a extraction key $\widetilde{K}$.
  - $\widetilde{\mathbf{b}} \leftarrow \{0,1\}^\mu, \widetilde{K} \leftarrow \mathsf{ECOM.ExtGen}(1^\lambda, \widetilde{\mathbf{b}}), \mathsf{k} \leftarrow \mathsf{CIH.Gen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(1^\lambda, (\widetilde{K}, \mathsf{k}))$.
  - If $\pi = (\mathbf{b}, \{\alpha_i\}_{i\in[w]}, \{\gamma_i\}_{i\in[w]})$ is accepted and $\mathbf{b} = \widetilde{\mathbf{b}}$, then output 1. Otherwise output 0.

  Since the only difference between this hybrid and Hyb_1 is the ECOM key generation, let $0 < c < 1$ be the constant in the sub-exponential key indistinguishability definition, we have
  $$\Pr[\mathsf{Hyb}_2 = 1] \geq \Pr[\mathsf{Hyb}_1 = 1] - 2^{-\lambda^c} \geq \epsilon(\lambda)/2^\mu - 2^{-\lambda^c}.$$

Now we argue that there exists a constant $0 < c' < 1$ such that $\Pr[\mathsf{Hyb}_2 = 1] \leq 2^{-\lambda^{c'}}$ for any sufficiently λ. When the proof π is accepted and $\mathbf{b} = \widetilde{\mathbf{b}}$, the commitment scheme $\mathsf{ECOM.Com}(\mathbf{b}, K, \cdot;\cdot)$ is in "extraction" mode. By the same argument as in Section 7.2, we can prove the trapdoor sigma protocol instantiated with $\mathsf{ECOM.Com}(\mathbf{b}, K, \cdot;\cdot)$ as the commitment scheme satisfies the bad challenge function in TC^0 property. Hence, one can compute $\{\mathsf{PreComp}(1^\lambda, \alpha_i)\}_{i\in[w]}$ from any accepting transcript, which constitutes an attack for the correlation intractability of CIH. Since we assume the CIH is sub-exponential correlation intractable, let c' be the constant in the definition, we have $\Pr[\mathsf{Hyb}_2 = 1] \leq 2^{-\lambda^{c'}}$, for any sufficiently large λ.

Combining the two inequalities, we obtain $\epsilon(\lambda)/2^\mu - 2^{-\lambda^c} \leq \Pr[\mathsf{Hyb}_2 = 1] \leq 2^{-\lambda^{c'}}$, and thus $\epsilon(\lambda) \leq 2^\mu \cdot (2^{-\lambda^c} + 2^{-\lambda^{c'}})$. When we set $\mu = \log^2 \lambda$, $\epsilon(\lambda)$ is sub-exponential.

9 Acknowledgements

We would like to thank Yuval Ishai and Prabhanjan Ananth for helpful discussions. The authors were supported in part by an NSF CNS grant 1814919, NSF CAREER award 1942789 and Johns Hopkins University Catalyst award. The first author was additionally supported in part by Office of Naval Research grant N00014-19-1-2294.

References

[1] Adleman, L.: A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In: 20th Annual Symposium on Foundations of Computer Sciences. pp. 55–60 (1979)

[2] Badrinarayanan, S., Fernando, R., Jain, A., Khurana, D., Sahai, A.: Statistical ZAP arguments. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 642–667. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45727-3_22

[3] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg, Germany, Cologne, Germany (Apr 26–30, 2009). https://doi.org/10.1007/978-3-642-01001-9_1

[4] Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg, Germany, Warsaw, Poland (May 4–8, 2003). https://doi.org/10.1007/3-540-39200-9_38

[5] Bellare, M., Yung, M.: Certifying cryptographic tools: The case of trapdoor permutations. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 442–460. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1993). https://doi.org/10.1007/3-540-48071-4_31

[6] Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014. pp. 459–474. IEEE Computer Society (2014)

[7] Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg, Germany, New York, NY, USA (Mar 4–7, 2006). https://doi.org/10.1007/11681878_4

[8] Bitansky, N., Paneth, O.: ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 401–427. Springer, Heidelberg, Germany, Warsaw, Poland (Mar 23–25, 2015). https://doi.org/10.1007/978-3-662-46497-7_16

[9] Blum, M.: How to prove a theorem so no one else can claim it. In: In: Proceedings of the International Congress of Mathematicians. pp. 1444–1451 (1987)

[10] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC. pp. 103–112. ACM Press, Chicago, IL, USA (May 2–4, 1988). https://doi.org/10.1145/62212.62222

[11] Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 757–788. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/978-3-319-96884-1_25

[12] Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001). https://doi.org/10.1007/3-540-44647-8_13

[13] Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg, Germany, Providence, RI, USA (Mar 28–30, 2011). https://doi.org/10.1007/978-3-642-19571-6_16

[14] Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 509–539. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53018-4_19

[15] Brakerski, Z., Koppula, V., Mour, T.: NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 738–767. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_26

[16] Brier, E., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 335–345. Springer, Heidelberg, Germany, Paris, France (Feb 12–14, 2002). https://doi.org/10.1007/3-540-45664-3_24

[17] Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1082–1090. ACM Press, Phoenix, AZ, USA (Jun 23–26, 2019). https://doi.org/10.1145/3313276.3316380

[18] Canetti, R., Chen, Y., Reyzin, L.: On the correlation intractability of obfuscated pseudorandom functions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 389–415. Springer, Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016). https://doi.org/10.1007/978-3-662-49096-9_17

[19] Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 91–122. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78381-9_4

[20] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004). https://doi.org/10.1145/1008731.1008734, https://doi.org/10.1145/1008731.1008734

[21] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg, Germany, Warsaw, Poland (May 4–8, 2003). https://doi.org/10.1007/3-540-39200-9_16

[22] Canetti, R., Lichtenberg, A.: Certifying trapdoor permutations, revisited. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 476–506. Springer, Heidelberg, Germany, Panaji, India (Nov 11–14, 2018). https://doi.org/10.1007/978-3-030-03807-6_18

[23] Choudhuri, A.R., Hubácek, P., Kamath, C., Pietrzak, K., Rosen, A., Rothblum, G.N.: Finding a nash equilibrium is no easier than breaking Fiat-Shamir. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1103–1114. ACM Press, Phoenix, AZ, USA (Jun 23–26, 2019). https://doi.org/10.1145/3313276.3316400

[24] Coppersmith, D., Odlyzko, A.M., Schroeppel, R.: Discrete logarithms in gf(p). Algorithmica **1**(1), 1–15 (Jan 1986). https://doi.org/10.1007/BF01840433, https://doi.org/10.1007/BF01840433

[25] Couteau, G., Katsumata, S., Ursu, B.: Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 442–471. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45727-3_15

[26] De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge proof systems. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 52–72. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1988). https://doi.org/10.1007/3-540-48184-2_5

[27] Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC. pp. 542–552. ACM Press, New Orleans, LA, USA (May 6–8, 1991). https://doi.org/10.1145/103418.103474

[28] Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017). https://doi.org/10.1007/978-3-319-63688-7_18

[29] Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 3–32. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_1

[30] Dwork, C., Naor, M.: Zaps and their applications. In: 41st FOCS. pp. 283–293. IEEE Computer Society Press, Redondo Beach, CA, USA (Nov 12–14, 2000). https://doi.org/10.1109/SFCS.2000.892117

[31] Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)

[32] Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st FOCS. pp. 308–317. IEEE Computer Society Press, St. Louis, MO, USA (Oct 22–24, 1990). https://doi.org/10.1109/FSCS.1990.89549

[33] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12

[34] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987). https://doi.org/10.1145/28395.28420

[35] Goldreich, O., Rothblum, R.D.: Enhancements of trapdoor permutations. Journal of Cryptology **26**(3), 484–512 (Jul 2013). https://doi.org/10.1007/s00145-012-9131-8

[36] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press, Providence, RI, USA (May 6–8, 1985). https://doi.org/10.1145/22145.22178

[37] Goyal, V., Jain, A., Jin, Z., Malavolta, G.: Statistical zaps and new oblivious transfer protocols. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 668–699. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45727-3_23

[38] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 89–98. ACM Press, Alexandria, Virginia, USA (Oct 30 – Nov 3, 2006). https://doi.org/10.1145/1180405.1180418, available as Cryptology ePrint Archive Report 2006/309

[39] Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2006). https://doi.org/10.1007/11818175_6

[40] Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006). https://doi.org/10.1007/11761679_21

[41] Holmgren, J., Lombardi, A.: Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In: Thorup, M. (ed.) 59th FOCS. pp. 850–858. IEEE Computer Society Press, Paris, France (Oct 7–9, 2018). https://doi.org/10.1109/FOCS.2018.00085

[42] Kalai, Y.T., Khurana, D., Sahai, A.: Statistical witness indistinguishability (and more) in two messages. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 34–65. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78372-7_2

[43] Kalai, Y.T., Rothblum, G.N., Rothblum, R.D.: From obfuscation to the security of Fiat-Shamir for proofs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 224–251. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017). https://doi.org/10.1007/978-3-319-63715-0_8

[44] Khurana, D., Sahai, A.: How to achieve non-malleability in one or two rounds. In: Umans, C. (ed.) 58th FOCS. pp. 564–575. IEEE Computer Society Press, Berkeley, CA, USA (Oct 15–17, 2017). https://doi.org/10.1109/FOCS.2017.58

[45] Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press, Victoria, BC, Canada (May 4–6, 1992). https://doi.org/10.1145/129712.129782

[46] Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 320–339. Springer, Heidelberg, Germany, San Francisco, CA, USA (Mar 19–21, 2008). https://doi.org/10.1007/978-3-540-78524-8_18

[47] Kopparty, S.: AC^0 lower bounds and pseudorandomness. Lecture notes for 'Topics in Complexity Theory and Pseudorandomness' (2013), https://sites.math.rutgers.edu/~sk1233/courses/topics-S13/lec4.pdf

[48] Lombardi, A., Vaikuntanathan, V.: Fiat-shamir for repeated squaring with applications to PPAD-hardness and VDFs. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 632–651. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_22

[49] Lombardi, A., Vaikuntanathan, V., Wichs, D.: 2-message publicly verifiable WI from (subexponential) LWE. Cryptology ePrint Archive, Report 2019/808 (2019), https://eprint.iacr.org/2019/808

[50] Lombardi, A., Vaikuntanathan, V., Wichs, D.: Statistical ZAPR arguments from bilinear maps. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 620–641. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45727-3_21

[51] Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453. IEEE Computer Society Press, Santa Fe, NM, USA (Nov 20–22, 1994). https://doi.org/10.1109/SFCS.1994.365746

[52] Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003). https://doi.org/10.1007/978-3-540-45146-4_6

[53] Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA. pp. 448–457. ACM-SIAM, Washington, DC, USA (Jan 7–9, 2001)

[54] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press, Baltimore, MD, USA (May 14–16, 1990). https://doi.org/10.1145/100216.100273

[55] Oliveira, I.C., Santhanam, R., Srinivasan, S.: Parity Helps to Compute Majority. In: Shpilka, A. (ed.) 34th Computational Complexity Conference (CCC 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 137, pp. 23:1–23:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2019). https://doi.org/10.4230/LIPIcs.CCC.2019.23, http://drops.dagstuhl.de/opus/volltexte/2019/10845

[56] O'Neill, A.: Definitional issues in functional encryption. IACR Cryptol. ePrint Arch. **2010**, 556 (2010), http://eprint.iacr.org/2010/556

[57] Pass, R.: Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 334–354. Springer, Heidelberg, Germany, Tokyo, Japan (Mar 3–6, 2013). https://doi.org/10.1007/978-3-642-36594-2_19

[58] Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 89–114. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26948-7_4

[59] Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008). https://doi.org/10.1007/978-3-540-85174-5_31

[60] Reif, J.H., Tate, S.R.: On threshold circuits and polynomial computation. SIAM Journal on Computing **21**(5), 896–908 (1992)

[61] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014). https://doi.org/10.1145/2591796.2591825

[62] Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg, Germany, Aarhus, Denmark (May 22–26, 2005). https://doi.org/10.1007/11426639_27

[63] Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Aho, A. (ed.) 19th ACM STOC. pp. 77–82. ACM Press, New York City, NY, USA (May 25–27, 1987). https://doi.org/10.1145/28395.28404

[64] Smolensky, R.: On representations by low-degree polynomials. In: 34th FOCS. pp. 130–138. IEEE Computer Society Press, Palo Alto, CA, USA (Nov 3–5, 1993). https://doi.org/10.1109/SFCS.1993.366874

A Extension of Rate-1 Trapdoor Hash Functions

In this section, we present an extension (and a slight simplification) of trapdoor hash functions based on DDH in [15] to any polynomial modulo setting.

Construction of Trapdoor Hash for Linear Functions over $\mathbb{Z}_R$. In the following construction, we use a hash function $\phi : \mathbb{G} \to \{0,1\}^*$, which is sampled from a hash function family Φ. Here we can use a pseudo-random function as in the previous work [15], or k-wise independent hash functions.

- $\mathsf{HKGen}(1^\lambda, 1^n, 1^R)$:
  - Generate a group $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$.
  - For each $i \in [n]$, sample an element uniformly at random from the group $\mathbb{G}$, $h_i \leftarrow \mathbb{G}$.
  - Output $\mathsf{hk} = (\mathbb{G}, p, g, \{h_i\}_{i\in[n]}, R)$.
- $\mathsf{EKGen}(\mathsf{hk}, f)$:
  - Let the linear function
$$f(x_1, x_2, \ldots, x_n) = (a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n) \bmod R, \text{where } 0 \le a_i < R.$$
  - Sample a secret $s \leftarrow \mathbb{Z}_p$ uniformly at random. For each $i \in [n]$, let $f_i = h_i^s \cdot g^{a_i}$.
  - Set the parameters $\tau' = \lceil \log_2(2R^2 \cdot (n+1)/\tau) \rceil + 1$, and $T = 2^{\tau'} \lceil \ln(2/\tau) \rceil + 1$.
  - Let Φ be a family of pseudo-random functions from $\mathbb{G}$ to $\{0,1\}^{\tau'}$. Sample $\phi \leftarrow \Phi$.
  - Output $(\mathsf{ek} = (\phi, \{f_i\}_{i\in[n]}), \mathsf{td} = (s, a_0))$.
- $\mathsf{Hash}(\mathsf{hk}, x)$:
  - Let $x = (x_1, x_2, \ldots, x_n)$, where $0 \le x_i < R$, for each $i \in [n]$.
  - Output $\mathsf{h} = \prod_{i\in[n]} h_i^{x_i}$
- $\mathsf{Enc}(\mathsf{ek}, x)$:
  - Let $h_e \leftarrow \prod_{i\in[n]} f_i^{x_i}$.
  - Output $\mathsf{DLog}(g, \phi, R, h_e)$.
- $\mathsf{Dec}(\mathsf{td}, \mathsf{h})$:
  - Let $h_d = h^s \cdot g^{-a_0}$.
  - Output $(-\mathsf{DLog}(g, \phi, R, h_d)) \bmod R$.

Lemma A.1. *Let S be an integer, and $0 < \tau < 1$ be a real number. Let $\Phi = \{\phi : \mathbb{G} \to \{0,1\}^{\tau'}\}$ be a pseudo-random hash function family with output length $\tau' = \lceil \log_2(2S/\tau) \rceil + 1$, and set parameter $T = 2^{\tau'} \lceil \ln(2/\tau) \rceil + 1$. Then, for any cyclic group $\mathbb{G}$ of order p with generator g, and any $h \in \mathbb{G}$, we have*

$$\Pr_{\phi\leftarrow\Phi}\left[\forall 0 \le x < S, (\mathsf{DLog}(g, \phi, R, h \cdot g^x) - \mathsf{DLog}(g, \phi, R, h)) \bmod R = x\right] > 1 - \tau - \mathsf{negl}(\lambda).$$

Furthermore, if S is a polynomial in n, and τ is an inverse polynomial in n, then running time of the algorithm DLog *is also a polynomial in n.*

Proof. The proof follows the same strategy as Proposition B.2 in [15]. Since the input to ϕ is fixed before ϕ is sampled, we can switch ϕ to a random function. For ease of presentation, let the event E denote $\forall 0 \le x < S, (\mathsf{DLog}(g, \phi, R, h \cdot g^x) - \mathsf{DLog}(g, \phi, R, h)) \bmod R = x$. We consider three cases:

Distributed Discrete Logarithm $\mathsf{DLog}(g, \phi, R, h)$

- Let $i = 0$
- Loop while $i \le T$
  - If $\phi(h \cdot g^i) = 0$, output $(i \bmod R)$.
  - Let $i = i + 1$.

  Output 0.

Figure 11: Description of the distributed discrete logarithm algorithm DLog.

- **Case 1:** There exists a $0 \le i < S$ such that $\phi(h \cdot g^i) = 0$. In this case, E may not hold, hence we bound the probability of this case. By the union bound, we bound it by $S/2^{\tau'}$.
- **Case 2:** For any $0 \le i < S$, $\phi(h \cdot g^i) \ne 0$, and there exists a $S \le i \le T$ such that $\phi(h \cdot g^i) = 0$. In this case, E always holds.
- **Case 3:** For any $i \le T$, $\phi(h \cdot g^i) \ne 0$. In this case, E may not hold. Hence, we bound the probability of this case. Since the hash function ϕ is random, we bound the probability by $(1 - 2^{-\tau'})^T$.

In total, we have

$$\Pr_{\phi}[E] \ge 1 - \left(\frac{S}{2^{\tau'}} + \left(1 - 2^{-\tau'}\right)^T\right) > 1 - (\tau/2 + \tau/2) \ge 1 - \tau.$$

□

Lemma A.2 (τ-Enhanced Correctness for TDH)**.** *The above construction of* TDH *satisfies τ-enhanced correctness.*

Proof. For any polynomial $R = R(n)$, any hash value $\mathsf{h} \in \mathbb{G}$, and any linear function $f \in \mathcal{F}_{n,R}$, and any hash key hk output by $\mathsf{HKGen}(1^\lambda, 1^n, 1^R)$,

$$\begin{aligned}
&\Pr_{\phi, \{f_i\}_{i\in[n]}}[\forall x : \mathsf{Hash}(\mathsf{hk}, x) = \mathsf{h}, f(x) = (\mathsf{e} + \mathsf{d}) \bmod R] \\
=&\Pr_{\phi \leftarrow \Phi}\left[\forall x : \mathsf{Hash}(\mathsf{hk}, x) = \mathsf{h}, f(x) = (\mathsf{DLog}(g, \phi, R, h_e) - \mathsf{DLog}(g, \phi, R, h_d)) \bmod R\right] \\
\ge&\Pr_{h \leftarrow \Phi}\left[\forall 0 \le y \le R^2 \cdot (n+1), (\mathsf{DLog}(g, \phi, R, h_d \cdot g^y) - \mathsf{DLog}(g, \phi, R, h_d)) \bmod R = y\right] \\
>&1 - \tau
\end{aligned}$$

The second line follows from the definition of e and d. The third line follows from

$$\begin{aligned}
h_e &= \prod_{i\in[n]} f_i^{x_i} = \prod_{i\in[n]} (h_i^s \cdot g^{a_i})^{x_i} = \prod_{i\in[n]} (h_i^{x_i})^s \cdot g^{\sum_i a_i x_i} = (h^s \cdot g^{-a_0}) \cdot g^{a_0 + \sum_i a_i x_i} \\
&= h_d \cdot g^{a_0 + \sum_i a_i x_i}.
\end{aligned}$$

Let $y = a_0 + \sum_{i\in[n]} a_i x_i$, then we have $0 \le y < R^2 \cdot (n+1)$. The forth line follows from Lemma A.1. □

Lemma A.3 (Sub-exponential Function Privacy)**.** *Assuming sub-exponential hardness of DDH, the construction of* TDH *satisfies sub-exponential function privacy.*

The proof of this lemma follows in the same manner as the proof of Theorem 4.2 in [29] since the ek is the same as in the construction of [29].

B Instantiation from Elliptic Curves

In this section, we instantiate the result in Theorem 1.1 from any Elliptic curves in the short Weierstrass form over $\mathbb{F}_p$ ($p \neq 2, 3$). In Lemma B.1, we show that to compute an iterative multiplication of λ group elements, we only need a $o(\log \lambda)$-depth threshold circuit. Next, when assuming DDH hardness against sub-exponential time adversaries, by complexity leveraging, we shrink the security parameter of the group to $\log^{O(1)} \lambda$, while the commitments from such groups remain hiding for polynomial time adversaries. Then the depth of the threshold circuit computing the iterative multiplication becomes $o(\log \log \lambda)$, and hence can be handled by the CIH for $O(\log \log \lambda)$-depth threshold circuit in Theorem 6.6 (See Theorem B.3).

Lemma B.1. *Let a, b be two integers, $y^2 = x^3 + ax + b$ be a short Weierstrass equation, and p be a prime with $4a^3 + 27b^2 \neq 0 \pmod p$. Then the short Weierstrass equation defines a Elliptic curve $\mathbb{G} = \{(X : Y : Z) \mid X, Y, Z \in \mathbb{F}_p, Y^2Z = X^3 + aXZ^2 + bZ^3\}$ in the finite projective space, where the identity is defined as $(0 : 1 : 0)$. Then*

- **Iterative Multiplication:** *For any integer n and security parameter λ, let the iterative multiplication function be $\mathsf{Mul}_{\lambda,n}(g_1, g_2, \ldots, g_n) = g_1 \cdot g_2 \cdot g_3 \ldots g_n$, where $g_1, g_2, \ldots, g_n \in \mathbb{G}$. Then $\mathsf{Mul}_{\lambda,n}$ can be computed by a series of polynomial-size threshold circuits $\{C_{\lambda,n}\}_{\lambda,n}$ of depth $o(\log n)$.*
- **Identity Testing:** *For any security parameter λ, define the $\mathsf{Identity}_\lambda(g)$ be the function which outputs whether g is the identity element $\mathbb{1}_\mathbb{G}$. Then $\mathsf{Identity}_\lambda$ can be computed in TC^0.*

Proof. We prove the lemma by computing $\mathsf{Mul}_{\lambda,n}$ and $\mathsf{Identity}_\lambda$, as follows.

- **Iterative Multiplication:** Our construction of the circuit computing $\mathsf{Mul}_{\lambda,n}$ is a complete B-ary tree of depth $\log_B n$, where $B = \log^* n$. Each tree node is a gate computing the iterative multiplication of its B children. The leaf nodes correspond to $g_1, g_2, \ldots, g_n$. Since the depth of the tree is $\log_B n = o(\log n)$, it suffices to show the product of B group elements can be computed in TC^0.

  Note that, for any group elements $h = (X_h : Y_h : Z_h), f = (X_f : Y_f : Z_f) \in \mathbb{G}$, if we denote $h \cdot f = u$, where $u = (X_u : Y_u : Z_u)$, then X_u, Y_u and Z_u can be computed by constant-degree polynomials in $\mathbb{Z}_p[X_h, Y_h, Z_h, X_f, Y_f, Z_f]$ using the unified point addition formula [16]. Hence, if we let $v = g_1 \cdot g_2 \cdot \ldots g_B$, where $v = (X_v : Y_v : Z_v)$, then X_v, Y_v, Z_v are $2^{O(B)}$-degree multivariate polynomials about the coordinates of $g_1, g_2, \ldots, g_B$. Since we choose $B = \log^* n$, we have at most $(O(B))^{2^{O(B)}} = \mathsf{poly}(n)$ monomials in each polynomial. Since iterative multiplication and addition in $\mathbb{Z}_p$ can be computed in TC^0 by [60], we can evaluate each monomial in TC^0 by interative multiplication, and add them in TC^0 by iterative addition. Hence, v can be computed in TC^0.
- **Identity Testing:** Given a group element $g = (X : Y : Z)$, we can test whether $g = 1_\mathbb{G}$ by checking if $X = Z = 0$. Since this checking can be done in TC^0, we can compute $\mathsf{Identity}_\lambda$ in TC^0.

□

Lemma B.2. *From any Elliptic curve in Lemma B.1, we can construct a lossy public key encryption scheme whose decryption can be decomposed to a deterministic polynomial time algorithm* PreComp *and $o(\log \lambda)$-depth threshold circuit* Dec_λ.

The proof follows the same idea as Lemma 7.3, the only difference is that we apply Lemma B.1 for the iterative multiplication and the identity testing.

Theorem B.3. *Assuming DDH over the Elliptic curve in Lemma B.1 is hard for any sub-exponential time adversary, we can obtain NIZKs and Zaps with the same properties in Theorem 1.1.*

If we assume any $2^{O(\lambda^c)}$-time adversary for DDH can obtain at most $2^{-\Omega(\lambda^c)}$ advantage, then we can set the security parameter of the lossy public key encryption scheme in the construction of NIZKs with the lossy public key encryption scheme to be $\log^{2/c} \lambda$. By Lemma B.2, the decryption circuit of such lossy public key encryption can be decomposed to a deterministic algorithm $\mathsf{PreComp}$ and $o(\log \log \lambda)$-depth threshold circuit Dec_λ. Applying the CIH in Theorem 6.6, we obtain the result.