Non-Interactive Zero Knowledge from Sub-exponential DDH

Abhishek Jain and Zhengzhong Jin

Johns Hopkins University, Baltimore, MD
{abhishek,zzjin}@cs.jhu.edu

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 [31] 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) [22]. Applications of NIZKs abound and include advanced encryption schemes [46,23], 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 [22], a major line of research has been dedicated towards understanding the assumptions that are sufficient for constructing NIZKs in the CRS model [9,28,5,18,35,34,30,53,19,16,14,50,21]. By now, NIZKs for NP are known from most of the standard assumptions known to imply public-key encryption – this includes factoring related assumptions [9,28], bilinear maps [18,35,34], and more recently, learning with errors (LWE) [14,50].

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 [10] and attribute-based encryption [54,33] (and more broadly, functional encryption [54,11,48]), and NIZKs. For the former, the gap has recently started to narrow in some important cases; see, e.g., [24]. We seek to understand whether such gap is inherent for NIZKs based on standard assumptions.[1]

A recent beautiful work of Brakerski et al. [13] 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 [26] 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) [29]. 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, [26] 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 [2,42,32].

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 [2,32], curiously, even bilinear maps have so far been insufficient.[2] In contrast, statistical NIZKs based on bilinear maps are known [35,34].

1.1 Our 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 (Main Result – Informal). *Assuming sub-exponential hardness of DDH against polynomial-time attackers, there exist:*

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

[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 [16,21].

[2] A variant of statistical Zap arguments where the verifier is private-coin but the proofs are publicly verifiable is known from standard assumptions on bilinear maps [43].

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.[3]

Our results rely on the assumption that *polynomial-time* adversaries cannot distinguish Diffie-Hellman tuples from random tuples with better than sub-exponentially small advantage. To the best of our knowledge, this assumption is unaffected by known attacks on the discrete logarithm problem.[4]

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:

- 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 [35,34].
- 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 [35,34].

In particular, statistical NIZKs in the common random string model were previously only known from LWE (or circular-secure FHE) [14,50], and statistical Zap arguments were previously only known from (quasi-polynomial) LWE [2,32].

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.

ITDH generalizes and extends the recent notion of trapdoor hash functions (TDH) [25] 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.

[3] Following [43], 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, [49] proved the impossibility of statistical NIZKs where adaptive soundness is proven via a black-box reduction to falsifiable assumptions [44].

[4] There are well-known attacks for discrete logarithm over $\mathbb{Z}_q^*$ that require sub-exponential time and achieve constant success probability [1,20]. However, as observed in [16], a 2^t time algorithm with constant successful probability does not necessarily imply a polynomial time attack with 2^{-t} successful probability.

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) [25,13]. 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 2 (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.

Our Approach: Round Collapsing, *Twice*. We follow the correlation intractability framework for NIZKs implemented in a recent remarkable sequence of works [16,36,14,50,13,21]. The central idea of this framework is to instantiate the random oracle in the Fiat-Shamir paradigm [29] by so-called *correlation intractable hash functions* (CIH) [17]. In particular, given a CIH for all efficiently searchable relations, this approach can be used to collapse the rounds of so-called trapdoor sigma protocols [14] to obtain NIZKs in the CRS model.

The works of [14,50] used (leveled) fully homomorphic encryption to construct CIH for all efficiently searchable relations and therefore required LWE-related assumptions. Recently, Brakerski et al. [13] demonstrated a new approach for constructing CIH via (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. 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. 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.

Theorem 3 (Informal). *Assuming sub-exponential hardness of DDH against polynomial-time attackers, there exists a CIH for* TC^0*.*

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 the above theorem can be built from DDH. This allows us to construct NIZK and Zap arguments in Theorem 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. [32].

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.

Due to page limits, we defer our constructions of NIZKs and Zap arguments to the full version.

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. A family of hash functions defined by a tuple of algorithms $(\mathsf{Gen}, \mathsf{Hash})$ is said to be *correlation intractable* (CI) 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 $y = C(x)$.

The CI framework instantiates the random oracle in the Fiat-Shamir paradigm for NIZKs via a family of CIH $(\mathsf{Gen}, \mathsf{Hash})$. 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.

Prior Work. A sequence of works [15,38,16,36,21] 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. [14] constructed CIH for all efficiently searchable relations from circular-secure fully homomorphic encryption. Subsequently, Peikert and Shiehian [50] obtained a similar result based on standard LWE.

Very recently, Brakerski et al. [13] leveraged the compactness properties of (rate-1) trapdoor hash functions to build CIH from standard assumptions. Specifically, assuming DDH (or other standard assumptions such as Quadratic Residuosity or LWE), they construct CIH for functions that can be approximated by a distribution on constant-degree polynomials. While this is a small class, [13] show that it nevertheless suffices for constructing NIZKs for NP. Specifically, they 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.

Main Challenges. We now briefly discuss the main conceptual challenges in buildings NIZKs based only on DDH (in light of the work of [13]).

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).[5] 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 [27] (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 polynomials. Specifically, [55,56,47,41] 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.

[5] The breakthrough work of [12] 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.

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 [13]. 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 6.2 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 purposes to consider circuits that consist of only not-threshold gates.

At a high-level, we implement the following two-step blueprint for constructing ITDH:

- **Step 1 (Depth-1 Circuits):** First, we build an ITDH for a simple circuit family $\mathcal{T}$ 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.

Overall, our construction uses only one cryptographic tool, namely, TDH for linear functions. As we will see later, we will use additional ideas to introduce non-linearity in the computation.

In the following, we elaborate on each of these steps. We first focus on step 2, namely, the sequential composition step, and discuss the main challenges therein. We will later describe how we implement step 1.

Controlling the Error. Recall that 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.

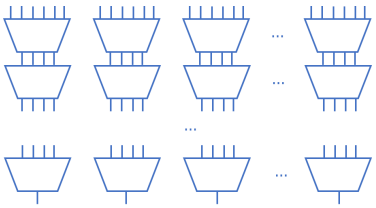

Fig. 1. Parallel structure. The top (resp., bottom) layer corresponds to input (resp., output) wires.

In such circuits, each output bit only depends on the input to *one* parallel repetition. Hence, the spreading of one Hamming error is controlled in one parallel execution. We leverage this observation to prove approximate correctness of the sequential composition.

Input Passing. 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, a plausible way to implement Step 2 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 resolve this issue, we keep the secret-sharing structure of the output in every ITDH intact. Instead, we extend the functionality of the ITDH in Step 1 so that the output of the i^{th} ITDH can be computed *within* the $(i+1)^{\text{th}}$ ITDH. Specifically, in Step 1, we construct an ITDH for a circuit family $\mathcal{T}^{\oplus}$ where every circuit consists of a single layer of Xor-*then*-Not-Threshold gates, namely, gates that first XOR the input with a pre-hardwired string and then compute the not-threshold operation on the resulting 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.

Putting together these ideas in a careful manner, we are able to implement Step 2. We refer the reader to the technical section for more details on this step.

ITDH for $\mathcal{T}^{\oplus}$. We now discuss how we implement *revised* step 1, namely constructing an ITDH for $\mathcal{T}^{\oplus}$, where every circuit consists of a single layer of Xor-*then*-Not-Threshold gates. At a high-level, we proceed in the following three steps:

- We first "decompose" a circuit in $\mathcal{T}^{\oplus}$ as the composition of two linear functions.
- Next, we use a 1-level ITDH, which is implied by TDH, to compute each of these linear functions.
- Finally, we "compose" the two ITDH executions sequentially to obtain a 2-level ITDH for $\mathcal{T}^{\oplus}$.

An observant reader may wonder how we decompose the computation of threshold gates into linear functions. Indeed, composition of linear functions is still a linear function, while a threshold gate involves non-linear computation. As we will soon see, our decomposition strategy crucially relies on some "offline" processing by the parties on the intermediate encoding and decoding values between different TDH executions. This introduces the desired non-linearity in the computation.

For simplicity, let us focus on the simpler goal of computing a *single* Xor-*then*-Not-Threshold gate. Our ideas easily extend to the more general setting. To compute such a gate, we proceed in three simple steps.

- First, bitwise xor the input string x with another string y, where y is hardwired in the circuit description.
- Next, sum the elements in the string $x \oplus y$.
- Finally, compare the summation with the threshold t (defined by the gate).

For the *first* step, let a and b be two bits at (say) the i^{th} coordinate of x and y, respectively. Then $a \oplus b = 1$ if and only if $a = 0 \wedge b = 1$ or $a = 1 \wedge b = 0$. Hence, $a \oplus b = (1-a) \cdot b + a \cdot (1-b)$. Since b is part of the circuit description, the right hand side is a linear function of a over $\mathbb{Z}$. For the *second* step, we simply sum over the result of step 1 on all coordinates. Combining the first step and the second step, this summation is still a linear function of x over $\mathbb{Z}$, and thus we can use a TDH for linear functions to compute such a summation. We note, however, that the known construction of TDH in [25,13] is only for linear functions over $\mathbb{Z}_2$. We therefore extend the TDH construction in [25,13] 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 *third* step as a linear function. We start with a simpler case. Suppose that the summation obtained 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 x, 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 $\mathbb{1}_{<t} = (1, 1, \ldots, 1, 0, \ldots, 0)$ is a vector with 1's on the first t coordinates, and 0's on on the remaining coordinates. We can therefore express the comparison as the inner product between $\mathbb{1}_{\mathsf{sum}}$ and $\mathbb{1}_{<t}$, which is a linear function over $\mathbb{1}_{\mathsf{sum}}$. Hence, such a comparison can be computed by a TDH for linear functions over $\mathbb{Z}_2$.

The above discussion is oversimplified, however, since the sender and the receiver do not have the value sum. Instead, at the end of the "previous" TDH execution, the sender and the receiver only obtained encoding e and a decoding d, respectively, such that $(\mathsf{e}+\mathsf{d}) \bmod R = \mathsf{sum}$. Fortunately, we can still express the comparison $(\mathsf{e}+\mathsf{d}) \bmod R < t$ as

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

where $\mathbb{1}_{\mathsf{e}}$ is the indicator vector for e and $\mathbb{1}_{j,<t} = \sum_{j=0}^{t-1} \mathbb{1}_{(j-\mathsf{d}) \bmod R}$. This expression works because comparing $(\mathsf{e}+\mathsf{d}) \bmod R < t$ is equivalent to checking if there exists a $0 \leq j < t$ such that $(\mathsf{e}+\mathsf{d}) \bmod R = j$, which is equivalent to checking whether $\mathsf{e} = (j - \mathsf{d}) \bmod R$. Note that the right hand side of this formula is a linear function of $\mathbb{1}_{\mathsf{e}}$, and can thus be computed using a TDH for linear functions over $\mathbb{Z}_2$.

In the above two executions of TDH, the sender processes e from the first TDH execution to obtain $\mathbb{1}_{\mathsf{e}}$, and uses it as the input to the second TDH. The receiver processes d from the first TDH execution to obtain $\mathbb{1}_{j,<t}$, and uses it as the function for the second TDH execution. Note that this intermediate processing is non-linear, since computing the indicator vector can be done by several equality checks, and equality check is not a linear function. Hence, it introduces the necessary non-linearity in the computation, but is done "outside" of the TDH executions.

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" [28].[6] Our NIZK satisfies either statistical ZK and

[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.

non-adaptive soundness, or computational ZK and adaptive soundness. Crucially, our transformation does *not* require "CRS switching" in the security proof and hence works for both 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.

Lossy Public Key Encryption. The lossy public key encryption we use is essentially the same as in [40,51,3]. 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).

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 precomputation 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 $\mathsf{PreComp}$. 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 [52], 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. [13] 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" [28] 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 average-hard *search* problem, where the instance is subjected to the *uniform* distribution. For our purposes, the discrete logarithm problem suffices. 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 [34,34].

We remark that the above idea can be easily generalized to other settings. For example, starting from LWE-based single-theorem statistical NIZKs [50], 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 [50].

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 [27]. 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 [2,32] 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, $2^{-\log^2 \lambda}$) [37]. 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 [39,37,2,32] 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 [45]. By exploiting the specific structure of the generic construction as well as the fact that Naor-Pinkas OT decryption 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 the full version 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

[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.

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 $\mathsf{Ham}(x,y) = |\{i \mid x_i \neq y_i\}|$.

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.

We defer more preliminaries to the full version.

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 $\mathsf{Hash\&Enc}$, 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 $\mathcal{C} = \{\mathcal{C}_{n,u}\}_{n,u}$ be a family of circuits, where each circuit $f \in \mathcal{C}_{n,u}$ is a circuit of input length n and output length u. An L-level interactive trapdoor hashing protocol for the circuit family $\mathcal{C}$ 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 $f \in \mathcal{C}_{n,u}$, a level $(\ell-1)$ hash value $\mathsf{h}_{\ell-1}$ and trapdoor $\mathsf{td}_{\ell-1}$ (for $\ell = 1$, $\mathsf{h}_{\ell-1} = \mathsf{td}_{\ell-1} = \bot$). 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 $\mathsf{e}_{\ell-1} = \bot$.
– $\mathsf{Dec}(\mathsf{td}_L, \mathsf{h}_L)$: 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 $f \in \mathcal{C}_{n,u}$ and any sequence of security parameters $(\lambda_1, \ldots, \lambda_L)$, we have

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

where e, d are obtained by the following procedure: Let $\mathsf{h}_0 = \mathsf{td}_0 = \mathsf{e}_0 = \bot$. 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 $\mathcal{D}$,

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

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. 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. Moreover, 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 5.2, 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 5.3, 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 $\mathcal{C} = \{\mathcal{C}_{n,u}\}_{n,u}$ be a circuit family, where for any n and u, $\mathcal{C}_{n,u}$ contains circuits with n-bit inputs and u-bit outputs. For any $\mathcal{C}$, we define an Xor-*then*-Compute circuit family $\mathcal{C}^{\oplus} = \{\mathcal{C}^{\oplus}_{n,u}\}_{n,u}$ consisting of circuits that *first* compute a bit-wise xor operation on the input with a fixed string and *then* compute a circuit in $\mathcal{C}$ on the resulting value.

Specifically, $\mathcal{C}^{\oplus}_{n,u}$ 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 $C \in \mathcal{C}_{n,u}$ 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 $\mathcal{F} = \{\mathcal{F}_{n,R}\}_{n,R}$ 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 $\mathsf{k}_1 = (1, \mathsf{hk}_1, \{\mathsf{ek}_{1,i}\}_{i\in[u]})$.
- 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 $\mathbb{1}_{\mathsf{e}}$ is the indicator vector for any e.

$$\mathsf{h}_2 \leftarrow \mathsf{TDH.Hash}(\mathsf{hk}_2, \{\mathbb{1}_{\mathsf{e}_{1,i}}\}_{i\in[u]})$$

- 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 defer the proof of approximate correctness and leveled function privacy to the full version.

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 $\sum_{i \in I} x_i \cdot (1 - y_i) + (1 - x_i) \cdot y_i$.

Fig. 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 $x = (x_1, \ldots, x_n)$.

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 $0 \leq j < t$.

Fig. 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 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, an w-parallel composition of a circuit f' is a new circuit f that computes w copies of f' in parallel. More formally, for any circuit family $\mathcal{C}$, we define a corresponding parallel-composition circuit family as follows:

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

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 $\mathcal{C}$ is a new circuit family $\mathcal{C}[\overrightarrow{w}_{\downarrow L}] = \{\mathcal{C}[\overrightarrow{w}_{\downarrow L}]_{n,s}\}_{n,s}$, where each circuit $f \in \mathcal{C}[\overrightarrow{w}_{\downarrow L}]_{n,s}$ 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 $f'_{\ell,1}, f'_{\ell,2}, \ldots, f'_{\ell,w} \in \mathcal{C}$. 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 2 (Parallel-and-Sequential-Composition). *Let $\mathcal{C} = \{\mathcal{C}_{n,u}\}_{n,u}$ be a circuit family, where each circuit in $\mathcal{C}_{n,u}$ has input length n and output length u. For any polynomials $w = w(n), L = L(n)$, and integer s, we say that $\mathcal{C}[\overrightarrow{w}_{\downarrow L}] = \{\mathcal{C}[\overrightarrow{w}_{\downarrow L}]_{n,s}\}_{n,s}$ is a family of* w-parallel-and-L-sequential-composition circuits *if every circuit $f \in \mathcal{C}[\overrightarrow{w}_{\downarrow L}]_{n,s}$ 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 $f'_{\ell,j} \in \mathcal{C}_{n'_\ell,n'_{\ell+1}}$, and $n_\ell = n'_\ell \cdot w$,

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

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 $\mathcal{C}[\overrightarrow{w}_{\downarrow L}]$. Let $\mathcal{C} = \{\mathcal{C}_{n,u}\}_{n,u}$ be any circuit family, and let $\mathcal{C}[\overrightarrow{w}]$ be the corresponding w-parallel composition circuit family. Let $\mathcal{C}[\overrightarrow{w}]^{\oplus} = \{\mathcal{C}[\overrightarrow{w}]^{\oplus}_{n,u}\}_{n,u}$ be the "Xor-*then*-Compute" circuit family defined w.r.t. $\mathcal{C}[\overrightarrow{w}]$. Let $\mathsf{ITDH} = (\mathsf{ITDH.KGen}, \mathsf{ITDH.Hash\&Enc}, \mathsf{ITDH.Dec})$ be a 2-level interactive trapdoor hashing protocol for $\mathcal{C}[\overrightarrow{w}]^{\oplus} = \{\mathcal{C}[\overrightarrow{w}]^{\oplus}_{n,u}\}_{n,u}$ 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 $\mathcal{C}[\overrightarrow{w}_{\downarrow L}]$ 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 $f_\ell^{\oplus \mathsf{d}_{\ell-1}} \in \mathcal{C}[\overrightarrow{w}]^{\oplus}_{n_\ell,n_{\ell+1}}$

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

- 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

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

- 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 $\mathsf{td}_{\ell'-1} = \mathsf{td}_{\ell,1}$.

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

- 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'. We defer the proof of approximate correctness and leveled function privacy to the full version.

5.3 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 2. We first show that any circuit in TC^0_L can be converted to a circuit in $\mathcal{T}[\vec{w}_{\downarrow L}]$.

Lemma 1. *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 $x \in \{0,1\}^n$, $f(x) = f'(x, x, \ldots, x)$.*

We defer the proof to the full version.

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 5.2 to obtain an ITDH for the circuit family $\mathcal{T}[\vec{w}_{\downarrow L}]$, and therefore an ITDH for TC^0_L.

Theorem 4. *If for any inverse polynomial τ in the security parameter, there exists a trapdoor hash function TDH for linear function family $\mathcal{F}$ 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 $\lambda_1 < \lambda_2 < \ldots < \lambda_{2L} < w/2L$, $\epsilon(w, \lambda_1, \ldots, \lambda_L) = 2^{-2w+O(1)}$.*

We defer the proof to the full version.

ITDH for P/poly. Since any circuit in P/poly can be converted to a layered circuit as in Lemma 1, 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 3 (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.

Correlation Intractable Hash CIH

– $\mathsf{Gen}(1^\lambda)$:
  - For each $\ell \in [L]$, set $\lambda_\ell = \lambda^{\frac{1}{2}(\frac{c}{2})^{L-\ell}}$.
  - Compute simulated receiver's messages for ITDH:

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

  - Sample a mask $\mathsf{mask} \leftarrow \{0,1\}^w$ uniformly at random.
  - Output $\mathsf{k} = \big(\{\mathsf{k}_\ell\}_{\ell\in[L]}, \mathsf{mask}\big)$.

– $\mathsf{Hash}(\mathsf{k}, x)$:
  - Parse $\mathsf{k} = (\{\mathsf{k}_\ell\}_{\ell\in[L]}, \mathsf{mask})$.
  - Let $\mathsf{e}_0 = \bot$. Compute hash values and encodings for ITDH:

$$\forall \ell \in [L], (\mathsf{h}_\ell, \mathsf{e}_\ell) \leftarrow \mathsf{ITDH.Hash\&Enc}(\mathsf{k}_\ell, x, \mathsf{e}_{\ell-1}).$$

  - Output $\mathsf{e} \oplus \mathsf{mask}$, where $\mathsf{e} = \mathsf{e}_L$.

Fig. 4. Description of CIH.

We construct a correlation intractable hash function $\mathsf{CIH} = (\mathsf{CIH.Gen}, \mathsf{CIH.Hash})$ for TC^0_L in Figure 4.

Theorem 5 (Correlation Intractability). *If $w = \Omega(\lambda)$, the construction in Figure 4 is sub-exponential correlation intractable for the circuit class TC^0_L.*

Acknowledgements. 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., 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
10. 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
11. 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
12. 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
13. 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
14. 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
15. 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
16. 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

17. 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
18. 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
19. 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
20. 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
21. 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
22. 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
23. 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
24. 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
25. 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
26. 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
27. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
28. 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
29. 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
30. 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

31. 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
32. 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
33. 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
34. 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
35. 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
36. 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
37. 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
38. 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
39. 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
40. 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
41. 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
42. 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
43. 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

44. 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
45. 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)
46. 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
47. 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
48. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptol. ePrint Arch. **2010**, 556 (2010), http://eprint.iacr.org/2010/556
49. 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
50. 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
51. 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
52. Reif, J.H., Tate, S.R.: On threshold circuits and polynomial computation. SIAM Journal on Computing **21**(5), 896–908 (1992)
53. 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
54. 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
55. 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
56. 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