Rate-1 Statistical Non-Interactive Zero-Knowledge

Pedro Branco[1], Nico Döttling[2], and Akshayaram Srinivasan[3]

[1]Bocconi University
[2]Helmholtz Center for Information Security (CISPA)
[3]University of Toronto

Abstract

We give the first construction of a rate-1 statistical non-interactive zero-knowledge argument of knowledge. For the $\mathsf{circuitSAT}$ language, our construction achieves a proof length of $|w| + |w|^{\epsilon} \cdot \mathsf{poly}(\lambda)$ where w denotes the witness, λ is the security parameter, ϵ is a small constant less than 1, and $\mathsf{poly}(\cdot)$ is a fixed polynomial that is independent of the instance or the witness size. The soundness of our construction follows from either the LWE assumption, or the $O(1)$-LIN assumption on prime-order groups with efficiently computable bilinear maps, or the sub-exponential DDH assumption. Previously, Gentry et al. (Journal of Cryptology, 2015) achieved NIZKs with statistical soundness and computational zero-knowledge with the aforementioned proof length by relying on the Learning with Errors (LWE) assumption.

1 Introduction

In a zero-knowledge (ZK) [GMR85] proof system for an NP language $\mathcal{L}$, a prover convinces a verifier that a statement $x \in \mathcal{L}$ without revealing any other information. In a bit more detail, a ZK proof system should satisfy three properties, namely, completeness, soundness, and zero-knowledge. The *completeness* property requires that the honest prover convinces the verifier with a proof if $x \in \mathcal{L}$. The *soundness* guarantees that an (unbounded) malicious prover cannot make the honest verifier accept the proof if $x \notin \mathcal{L}$. The *zero-knowledge* property ensures that a malicious polynomial-time verifier cannot learn any other information except that $x \in \mathcal{L}$ from its interaction with an honest prover. If the soundness of the proof system is required to hold only against computationally-bounded provers, then it is called an *argument system* [BCC88] and if the zero-knowledge property is required to hold against unbounded verifiers, then we have *statistical zero-knowledge*.

In the non-interactive setting (NIZK) [BFM88], the prover sends a single message (or proof) π to the verifier and the verifier then checks the correctness of this proof. This setting has attracted significant attention, not only due to the ubiquity of NIZKs as fundamental building blocks in cryptography [NY90, DDN91, DN00], but also because it is notoriously hard to construct such schemes. It is well-known that NIZKs are impossible to achieve without any trusted setup [BFM88, GO94]. Hence, to build NIZKs, it is usually assumed that there exists a common reference string (CRS) generated by a trusted party. Throughout this work, we will refer to NIZK for all NP in the CRS model as simply NIZK.

By now, NIZKs are known from a plethora of hardness assumptions including factoring-based assumptions [BFM88, FLS90, BDSMP91], pairing-based assumptions [CHK03, GOS12], LWE [CCH$^+$19, PS19, Wat24], sub-exponential DDH [JJ21], DDH and LPN [BKM20] and LPN and MQ [DJJ24].

Statistical Zero-Knowledge. In this work, we focus on achieving statistical zero-knowledge in the non-interactive setting. That is, we require the zero-knowledge property to hold against unbounded verifiers but the soundness is required to hold only against computationally bounded provers. This setting is of particular interest as it achieves everlasting security: even if one of the hardness assumptions used to build the scheme gets broken in the future, the secret witness of the prover remains hidden.

Minimizing Proof Length. An important measure of efficiency for a NIZK scheme is its proof length. Starting from the seminal works of Kilian [Kil92] and Micali [Mic94], the problem of minimizing the proof length has been extensively studied for nearly three decades. For the case of statistically-sound systems (a.k.a. proofs), under standard complexity-theoretic assumptions, one cannot achieve proof size that is lesser than the witness size [GH98, GVW02]. Moreover, the work of Campanelli *et al.* [CGKS23] showed that under the exponential-time hypothesis, for the case of arguments of knowledge[1], it is impossible to achieve proof size $o(|w|)$ with black-box extraction.[2] Thus, there is little difference between the lower bounds on the communication complexity for proofs and arguments of knowledge with respect to black-box extraction. Hence, we define the rate of a NIZK proof or argument of knowledge to be $|w|/|\pi|$ and call constructions with a proof size that asymptotically approaches $|w|$ to be rate-1.

Prior Work. Gentry et al. [GGI+15] gave an elegant construction of rate-1 NIZKs based on fully-homomorphic encryption (FHE) [Gen09, BV11]. Specifically, for the $\mathsf{circuitSAT}$ language, their proof size is $|w| + \mathsf{poly}(\lambda)$ where w is the witness, λ is the security parameter, and $\mathsf{poly}(\cdot)$ is a fixed polynomial that is independent of the instance or the witness size. Their construction achieves rate-1 as the ratio between the size of the witness and the size of the proof approaches 1 asymptotically. This construction achieves statistical soundness but unfortunately, does not achieve statistical zero-knowledge property.

Another interesting line of work by Katsumata et al. [KNYY19, KNYY20] constructed NIZKs for the $\mathsf{circuitSAT}$ language with a proof size of $|C| + \mathsf{poly}(\lambda)$ where C is the verification circuit in the $\mathsf{circuitSAT}$ instance. The security (both soundness and zero-knowledge) of their construction is computational and is based on standard assumptions on bilinear maps. This construction does not achieve rate-1 as the size of the verification circuit C could be polynomial in the size of the witness w.

As none of these works achieve statistical zero-knowledge, we ask:

Can we build rate-1 NIZKs with statistical zero-knowledge?

This was stated as an open problem in [PS19].

1.1 Our Result

In this work, we give a new approach to build a statistical NIZK for NP satisfying argument of knowledge with rate-1. The soundness of our construction follows from either the $O(1)$-LIN assumption on prime-order groups with efficiently computable bilinear maps, or the sub-exponential DDH assumption, or the LWE assumption. For the $\mathsf{circuitSAT}$ language, our construction achieves a proof length of $|w| + |w|^{\epsilon} \cdot \mathsf{poly}(\lambda)$ where w is the witness, λ is the security parameter, a constant $0 < \epsilon < 1$, and $\mathsf{poly}(\cdot)$ is a fixed polynomial that is independent of the instance or the witness size.

Theorem 1 (Informal)**.** *Assuming the hardness of either LWE, or $O(1)$-LIN on prime order groups with efficiently computable bilinear maps, or the sub-exponential DDH, there exists a statistical rate-1 NIZK argument of knowledge.*

2 Technical Overview

In this section, we describe the main techniques involved in our construction. Throughout this section, we will be interested in constructing a non-interactive zero-knowledge (NIZK) argument of knowledge for the $\mathsf{circuitSAT}$ language. Recall that an instance of this language comprises of a Boolean circuit C and an input x. A valid witness is a string w is such that $C(x, w) = 1$.

[1] Argument of knowledge guarantees that if a polynomial-time prover creates a valid proof for a statement x, then there exists an extractor that can extract a valid witness from this proof.

[2] In a nutshell, if that was the case, then we could use the extractor for the argument of knowledge to solve NP-complete problems better than brute-forcing over all witnesses. This is not possible under the exponential-time hypothesis.

In Section 2.1, we give a construction of a scheme that only satisfies computational zero-knowledge and has a proof size of $|C| + \mathsf{poly}(\lambda)$. Later, in Section 2.2, we show how to decrease the proof size to $|w| + \mathsf{poly}(\lambda)$ while still satisfying only computational ZK. Finally, in Section 2.3, we show how to upgrade the construction to satisfy statistical zero-knowledge property with rate-1.

2.1 Starting Point

We explain a construction of NIZK with proof size $|C| + \mathsf{poly}(\lambda)$ below.

Setup: The CRS consists of a CRS for a non-interactive extractable commitment scheme and the CRS for a batch argument for NP (BARG) scheme [BHK17, CJJ21]. At a high-level, BARGs allow the prover to convince a verifier that a set of k NP statements all belong to an NP language $\mathcal{L}$ with a proof. The size of this proof only grows poly logarithmically in the batch size. Recent constructions have shown how to achieve this primitive under LWE [CJJ22], bilinear maps [WW22], and sub-exponential DDH [CGJ+23].

Prover: The prover on witness w evaluates the circuit C on (x, w). Let the computation tableau T be the value carried by each wire in C in the above evaluation. The size of this tableau is equal to $|C|$.

- The prover commits to T using a "special" rate-1 extractable commitment. We will later describe the properties that this extractable commitment needs to satisfy. Let $\mathsf{com}(T)$ denote this rate-1 commitment.
- The prover then considers a batch NP language comprising of $|C|$ NP instances. The witness to the i-th instance is an opening for the commitment $\mathsf{com}(T)$ to the input and the output wire values of the i-th gate in the circuit. The underlying NP relation verifies that this opening is correct w.r.t. $\mathsf{com}(T)$ and then checks if the input and output values are consistent with the gate type. If i is the output gate, then it additionally verifies whether the output wire value is 1.
- The prover generates a batch argument proof π for the above batch language and sends this proof along with $\mathsf{com}(T)$ to the verifier.

Verifier: The verifier then checks if the batch proof π verifies.

Proof Size. Let us first analyze the size of the proof in the above construction. The proof consists of two parts: a rate-1 commitment $\mathsf{com}(T)$ and the batch proof π. Since the commitment is rate-1, the size of the first part is $|T| + \mathsf{poly}(\lambda)$. To achieve the required proof size, we need the size of $|\pi|$ to be at most $\mathsf{poly}(\lambda)$. The proof size in existing batch NP arguments is at least the size of one of the witnesses (in fact, it grows with the size of the verification circuit for the underlying NP relation). In the above construction, the BARG witnesses consist of the openings to the input and the output values of a particular gate. In a normal extractable commitment, the size of each opening grows at least with the length of message. Hence, the size of the batch proof is at least the size of the computation tableau and this does not lead to a construction with the required proof size. To fix this issue, we rely on a rate-1 extractable commitment scheme with fully local opening. The fully local opening property allows the size of the opening to any position (and the size of the verification circuit for this opening) to be independent of the length of the message to be committed. Such a commitment scheme can be constructed using a pseudorandom function in conjunction with an extractable commitment scheme. To commit to a long message μ, we generate a PRF key k and generate a commitment to this key using the extractable commitment scheme. We send this commitment along with $\mu_i \oplus \mathsf{PRF}(k, i)$ for each $i \in [|\mu|]$. To open to a specific position, we just need to provide an opening to the commitment to the underlying key. Thus, even if the batch argument has a polynomial blow-up in the size of one of the witnesses, the size of the batch NP proof is a fixed polynomial in the security parameter independent of the length of the computation tableau. Thus, the total proof length is $|C| + \mathsf{poly}(\lambda)$.

Zero-Knowledge. Note that in the above construction the BARG proof uses the openings to each wire value in the computation tableau as the underlying witnesses. Since a BARG proof could leak information about some of the witnesses, the above proof system may not be zero-knowledge. However, we observe that if the underlying BARG is zero-knowledge, then the above construction can be proven to satisfy zero-knowledge. Specifically, we can consider an intermediate hybrid where we generate the BARG proof using the simulator for the zero-knowledge BARG and then rely on the hiding property of the commitment scheme to switch the computation tableau T to some junk value. Finally, we observe that there is a very simple transformation from BARG to zero-knowledge BARG by additionally using NIZKs. Specifically, we consider a modified batch NP language where the instances are the same as before but the witnesses correspond to NIZK proofs that attest that each instance belongs to the NP language. Since the new witnesses computationally hide all information about the old witnesses, this transformation can be easily shown to satisfy zero-knowledge property. Additionally, the size of the new BARG proof is still a fixed polynomial in the size of the verification circuit of the underlying NP language.

Recent works of Bitansky et al. [BKP+23] and Bradley et al. [BWW23] have shown that somewhere extractable BARGs along with one-way functions imply a non-interactive zero-knowledge argument.[3] These can be easily updated to arguments of knowledge by committing to the witness via an extractable commitment and proving via a NIZK argument that this witness is valid.

Argument of Knowledge. The argument of knowledge property is a bit more subtle to argue. We start with the description of the knowledge extractor. On receiving a proof from a malicious prover, this extractor checks if the proof is valid. If it is the case, it uses the extractor for the underlying extractable commitment to extract the computation tableau. From this computation tableau, it reads off the witness and outputs the witness string.

Assume for the sake of contradiction that there exists a malicious prover that with non-negligible probability outputs a valid proof but the output of the above extractor is an incorrect witness. It follows that the extracted computation tableau has an inconsistency. That is, either the final output wire does not carry 1 or there exists a gate such that the input and output wire values are inconsistent. In either case, we show how to contradict the somewhere extractability property of the BARG. Recall that the somewhere extractability property states that the CRS for the BARG scheme can be generated w.r.t. to a secret index i^* and there is an extractor that given a valid BARG proof can extract the witness for the i^*-th instance.

Let E be the event that prover is able to produce a convincing proof and the output of the extractor is not a valid witness. To contradict the somewhere extractability property of the underlying BARG scheme, we first make a guess i^* where the inconsistency occurs. Let F be the event that our guess is correct and the probability that F happens at least $1/|C|$. Since E and F are independent, $\Pr[E \wedge F] = \frac{\Pr[E]}{|C|}$. We now make the crs for BARG to be binding at i^*. It follows from the crs indistinguishability property that $\Pr[E \wedge F] \geq \frac{\Pr[E]}{|C|} - \mathsf{negl}(\lambda)$. Now, the somewhere extractability property ensures that the witness extracted from the BARG proof is correct except with negligible probability. This means that the openings to the wire values are correct and the input and output wire values are consistent. This means that there is an inconsistency between the values output by the extractor for the extractable commitment and the witness extracted from the BARG. If E happens with non-negligible probability, then this contradicts the extraction property of the extractable commitment scheme.

2.2 Decreasing the Proof Size

The above approach achieves a proof size of $|C| + \mathsf{poly}(\lambda)$ and hence, falls short of achieving rate-1. Our main technical contribution is modifying the above approach to achieve this proof size.

Why a natural idea fails? A natural solution to decrease the proof length would be to use a rate-1 somewhere extractable commitment with fully local opening [DGKV22, PP22, BDSZ24] to commit to the

[3]The work of Bradley et al. [BWW23] additionally assumes a public-key encryption scheme (which is implied by a non-interactive extractable commitment).

computation tableau. In contrast to a fully extractable commitment, a somewhere extractable commitment is extractable at a few secret positions. The main advantage of this somewhere extractability is that the size of the commitment only grows with the number of such positions rather than growing with the length of the entire committed message. Such a commitment is said to be rate-1 if the size of the commitment is $|B| + \mathsf{poly}(\lambda)$ where B is the set of positions that are extractable. Further, the size of the local opening (and the size of the verification circuit to check the openings) are fixed polynomials that are independent of $|B|$.[4]

A natural fix to decrease the proof length in the above construction is to have the CRS for the commitment scheme to be extractable at the positions corresponding to the witness. Thus, the size of the commitment is now $|w| + \mathsf{poly}(\lambda)$. The size of the batch proof is $\mathsf{poly}(\lambda)$ and hence, the proof size is $|w| + \mathsf{poly}(\lambda)$. The zero-knowledge part can be argued using a near identical argument as before.

Unfortunately, the proof of soundness (specifically, argument of knowledge) breaks down completely. Recall that in the previous proof, we first make a guess for the position where there is an inconsistency in the computational tableau. Once we receive the proof, we extract the entire computation tableau and check whether our guess was correct or not. However, since the size of the commitment in the new construction is much less than the size of the computation tableau, it is infeasible to extract the computation tableau that was committed to by the prover. Therefore, we cannot reliably check if our guess was correct or not and the above proof fails to go through. Other natural fixes such as making the extractable commitment binding at the input and output wire values of a guessed gate also fail as the prover can start cheating at a different gate once the distribution of the CRS changes. In hindsight, the barriers we face in proving the argument of knowledge property is similar to those faced while trying to construct SNARGs for NP using somewhere extractable BARGs.

Our approach in a nutshell. The main problem with the above approach is that it seems infeasible to catch a malicious prover who cheats in generating the computation tableau while having the total communication cost to be independent of the tableau size. While batch arguments allow one to argue that the prover cannot cheat at *any* particular step of the tableau generation, what we really need is a method to ensure that the prover cannot cheat at *every* step.

The first observation that allows us to design such an approach is that the computation tableau is completely determined by the witness. This means that if we extract the witness, then we can compute the honest computation tableau. However, this observation alone is not sufficient to prove soundness as we still need to surmount significant challenges. Specifically, we still need a mechanism to ensure that the computed tableau is consistent with the the prover's tableau used in generating the proof. However, note that the size of the proof is much less than the size of the computation tableau and hence, this task seems highly non-trivial.

To resolve this, we build the computation tableau "one-step-at-a-time" inside the batch argument. Specifically, we view the computation tableau as part of an external memory that can be updated by the batch argument. At each step, the batch argument takes a digest of the current state of the computation tableau (called a snapshot), computes the output of a single gate, and then updates the digest using the output of this gate. If the digest is computed using a Merkle tree, then these updates can be done in $\mathsf{poly}(\log |C|, \lambda)$ time.

We now explain how generating the computation tableau in a step-by-step manner allows us to overcome the aforementioned challenge. Observe that extracting the "candidate" witness allows us to precisely compute the digest of each snapshot of the tableau generation. The key idea behind our proof is to ensure that at any step i of the tableau generation that the digest used by the prover is consistent with the digest computed using the extracted witness. This implies that all the steps prior to step-i have been computed correctly due to the "binding" property of the digest. If we can ensure that this holds in the final step, then we are guaranteed that all the steps have been computed correctly and hence, the extracted witness is correct.

[4]We note that we can also allow the size of the openings to be $|B|^\epsilon$ for a small enough constant $0 < \epsilon < 1$. But we ignore this to keep the exposition simple.

Our Construction. Our constructions builds on the construction of flexible SNARGs for RAM programs given in [KLVW23].

- The CRS comprises of a hash key for a collision-resistant hash function, a hash key for a somewhere extractable hash function, CRS for a rate-1 extractable commitment and a CRS for a BARG scheme.
- The prover uses the witness w to evaluate the circuit C on the input (x, w) to generate the computation tableau T. T comprises of the input x along with values carried by each internal wire in the circuit (we will deal with the witness separately).
- Let n be the number of gates in the circuit C. Let $T_1, \ldots, T_n$ be the snapshots of the computation tableau at the end of evaluating gates $1, 2, \ldots, n$ respectively. In a bit more detail, to compute T_i, we evaluate the first i gates in the circuit using the inputs (x, w). Let W_i be the sequence of $|C|$ wire values where the wires corresponding to the outputs of the first i gates are correctly set and the rest of the values set to 0. T_i is computed as $x \| W_i$. Note that $T_n = T$ and let T_0 be the initial computation tableau that is populated with input x and with 0s for each internal wire.
- The prover hashes each T_i for $i \in [n] \cup \{0\}$ using a Merkle tree to generate the hash digests $h_0, h_1, \ldots, h_n$. The prover then hashes these values using a somewhere extractable hash function to generate a hash digest h.
- The prover also generates a rate-1 extractable commitment with fully local openings to the witness w. Let us denote the commitment by $\mathsf{com}(w)$.
- The prover then considers a batch NP language consisting of n statements. The witness to the i-th instance comprises of openings to h_i, h_{i+1} w.r.t. to the hash digest h along with certain other openings ρ. If the i-th gate in the circuit reads particular bit(s) of the witness, then ρ includes those openings from $\mathsf{com}(w)$. Else, if it reads certain input wires or other internal wires, then the openings to those values w.r.t. to h_i are included in ρ. Finally, ρ also includes the opening to output wire of the i-th gate in h_i.

  The relation function checks if all the openings are correct and computes the output of the i-th gate using the values of the input wires. If it is the final output gate, it additionally checks if the computed output is 1. It then checks if the opening to the output wire value in h_i is the default value (which is 0). If it is the case, it checks if h_{i+1} is correctly obtained from h_i with the output wire value of the i-th gate correctly set to the above computed output. Note that this can be done for Merkle tree using the openings to the output wire of the i-th gate in h_i.
- It generates a batch proof π for the above batch language and sends $(\pi, \mathsf{com}(w), h)$ to the verifier along with local opening to h_0 w.r.t. to h. The verifier checks if the proof verifies, the local opening to h_0 is correct and if h_0 is the output of the Merkle tree on T_0 (which can be computed from the statement).

Proof Size. The size of $\mathsf{com}(w)$ is $|w| + \mathsf{poly}(\lambda)$ and the size of h is $\mathsf{poly}(\lambda)$. Note that size of each openings used in the witness of the BARG language is $\mathsf{poly}(\lambda)$ and therefore, the size of π is $\mathsf{poly}(\lambda)$. The size of the opening to h_0 is $\mathsf{poly}(\lambda)$ and hence, the overall proof size in our construction is $|w| + \mathsf{poly}(\lambda)$.

Argument of Knowledge. The knowledge extractor is similar to the previous construction. It first verifies if the proof is valid and if it the case, it uses the extractor for the underlying extractable commitment to compute the candidate witness and outputs this. To prove that this extractor succeeds, we rely on techniques developed in the work of Kalai et al. [KLVW23] for constructing flexible SNARGs for RAM programs. Let us give more details.

Let $T_1, \ldots, T_n$ be the snapshots of the tableau generation procedure using the candidate witness extracted by the extractor. We maintain an invariant that if the hash key for the somewhere extractable hash function is extractable at location i and the prover produces an accepting proof, then h_i that is extracted from the

hash digest h is consistent with the tableau T_i (except with negligible probability). We prove this invariant using induction with the base case being $i = 0$.

The base case holds trivially (as the proof includes an opening to h_0) and suppose the invariant holds for the i-th step. Let us generate the hash key of the somewhere extractable hash function to be extractable at location $i + 1$ and let h_{i+1} be value that is extracted from h. We need to show that h_{i+1} is consistent with T_{i+1}. Below we provide a brief sketch of this argument.

- Since the invariant holds in the i-th step, it follows that if the hash key of the somewhere extractable hash function is set to be extractable at location i, then h_i extracted from h is consistent with T_i.
- We first change the CRS for the somewhere extractable BARG scheme to be binding at the i-th instance. It follows from the CRS indistinguishability property for the BARG scheme that h_i computed as above is still consistent with T_i.
- We now extract the witness for i-th BARG instance using the corresponding extractor. By somewhere extractability, this witness is a valid witness for the i-th instance except with negligible probability. Note that this witness corresponds to openings to h_i, h_{i+1} w.r.t. h along with certain other openings denoted by ρ.
- Since the somewhere extractable hash function is binding at location i and the witness is valid, it follows that h_i extracted from the digest h is identical to the h_i extracted from the witness of the BARG scheme except with negligible probability.
- We now rely on the collision-resistance of Merkle tree and the binding of the commitment to the witness w to show that values used in the computation of the i-th gate is consistent with T_i except with negligible probability. Since the extracted witness from the BARG is correct, it follows that the output of the i-th gate is computed correctly. Once again relying on the collision-resistance of Merkle tree, it follows that h_{i+1} that is extracted from the BARG scheme is consistent with T_{i+1}.
- Now, if we switch the hash key of the somewhere extractable hash function to be binding at location $i + 1$, we still maintain the invariant that h_{i+1} that is extracted from the BARG witness is consistent with T_{i+1} except with negligible probability. This property directly follows from the index hiding of the hash key.
- Once again relying on the binding property of somewhere extractable hash function at location $i + 1$, we infer that the invariant is maintained at step $i + 1$ as the BARG witness is valid.

This argument is reminiscent of the "two-key" trick [NY90] where we alternate between extracting the hash digests using somewhere extractable BARG and somewhere extractable hash. Extending this argument to the final step shows that h_n is consistent with T_n. We can now rely on the somewhere extractability of the BARG at the final instance along with the the collision-resistance of the Merkle tree to prove that the value carried by the output wire in T_n is 1. Thus, the witness extracted by us is valid.

Zero-Knowledge. An astute reader would have noticed that the above construction does not directly satisfy the zero-knowledge property. Specifically, the digest h could reveal information about some of the bits of some h_i which in turn leaks information about the tableau T_i. Since T_i consists of the internal wire values, we cannot afford to leak this information. To fix this, we use techniques developed in constructing adaptive garbling schemes [GS18a, GOS18, GS18b, AL18] to protect the memory content. Specifically, we mask each internal wire value using the output of a PRF and we commit to the key using an extractable commitment. The batch argument additionally takes an opening to this commitment and ensures that the masks are correctly computed. This does not affect the argument of knowledge proof as we can extract the PRF key from the commitment and ensure that at each step, the digest is consistent with the masked tableau rather than the plaintext tableau. This masking ensures that even if h_i is leaked, no information about the underlying wire values is revealed to the verifier.

2.3 Statistical Zero-Knowledge

Zero-knowledge for the construction described above relies on i) Zero-knowledge of the BARG scheme, ii) rate-1 extractable commitment scheme which hides the witness, and iii) the pseudorandomness of the PRF which masks the intermediate snapshots $T_1, \ldots, T_n$ of the computation. We now show how to modify the construction in order to get a NIZK with statistical zero-knowledge, while preserving the optimal rate.

Our first (and easiest) modification is to use a statistical zero-knowledge BARG. This can be easily built from a standard BARG and a statistical NIZK (with poor rate) using the same construction described earlier.

To solve the second issue, we show how to build a rate-1 extractable commitment scheme which can be set in two different modes: a statistically hiding mode and an extractable mode. In the extractable mode, an extractor can use the trapdoor (sampled along with the CRS) to extract the committed message. In the statistically hiding mode, the committed message is statistically hidden. We additionally need this commitment to have a local opening property. In particular, we need the size of the opening to any location to be $|m|^\epsilon \cdot \mathsf{poly}(\lambda)$ (where m is the committed message) for a small enough constant $0 < \epsilon < 1$. To construct such a rate-1 commitment, we rely on a technique developed by Paneth and Pass [PP22] which uses on a rate-1 OT scheme. Specifically, we chop the message m into k blocks $m_1, \ldots, m_k$ for $k = |m|^{1-\epsilon}$. The commitment key in the extraction mode comprises of k receiver OT messages with choice bit being 0 in each of them. To commit to m, the committer computes k sender OT messages using $(m_1, \perp)$, $(m_2, \perp)$, $\ldots$, $(m_k, \perp)$ as the sender messages. If the underlying OT has rate-1, then the size of the commitment is $|m| + |m|^{1-\epsilon} \cdot \mathsf{poly}(\lambda)$. Hence, the asymptotic rate achieved by this construction is 1. The local opening to some location i comprises $(m_{i^*}, \perp)$ along with the sender randomness used to generate the i^*-th OT message where i^* is the block that includes i. Thus, the size of the local opening is $|m|^\epsilon \cdot \mathsf{poly}(\lambda)$. Additionally, if the underlying OT scheme achieves statistical sender privacy, then we can set the commitment keys to be receiver OT messages with the choice bits set to 1 and we can show that the commitment statistically hides the message.

Finally, we need a new approach to hide the intermediate snapshots of the computation as PRFs only provide computational pseudorandomness. In our new approach, instead of hiding the snapshots $T_0, \ldots, T_n$ using a PRF, we will compute them in the plain. From here, we proceed exactly as in the previous construction: first, hash each T_i using a Merkle tree to obtain h_i, and then hash the values $h_0, \ldots, h_n$ using a SEH to obtain h. The main difference is that now we use a special SEH that we call *dual-mode SEH* to hash the digests $h_0, \ldots, h_n$. This SEH can be set in two different modes: a statistically hiding mode and a somewhere extractable mode. In the somewhere extractable mode, the hash key binds to a secret location i and an extractor can use the trapdoor to extract the input bit at location i from the digest. In the statistically hiding mode, the digest provides no information about the input. To construct such a SEH, we rely on the construction of SEH from the work of Kalai et al. [KLVW23] based on rate-1 OT. The hash key in this construction comprises of $\log k$ receiver OT messages where k is the length of the input. To hash an input, we first construct a complete binary tree where the leaves are assigned the values corresponding to the bits of the input and the value assigned to any node in level-ℓ is the sender OT message using the ℓ-th receiver message in the hash key and the sender strings being equal to the values of its children. The digest is the value assigned to the root. In the somewhere extractable mode, the choice bits in the hash key consists of the bits of the secret location i. To modify this construction to have a statistically hiding mode, we add an additional leaf with a default symbol and have the hash key point to this location. If the underlying OT has statistical sender privacy, then it follows that the input is statistically hidden in this mode.

In the real scheme, the commitment and the dual-mode SEH are set up in the statistically hiding mode and this allows us to prove statistical zero-knowledge. On the other hand, in the argument of knowledge proof, we switch them to extractable mode and we can use a similar argument as in the previous scheme.

2.4 Relation to RAM SNARGs

We note that the above idea of hashing the external memory and passing the digest as input to a batch argument has already appeared in the context of constructing SNARGs for RAM computations [KPY19, CJJ22, KLVW23, BD24]. As we explain below, these results cannot be directly used to construct a rate-1

NIZK.

SNARGs for RAM computation constructed in Kalai et al. [KPY19] and Choudhuri et al. [CJJ22] satisfied a form of soundness where the adversarial prover is forced to send the external memory in the clear along with a proof for a wrong statement about the RAM program. To use this to construct a NIZK scheme, we can consider a RAM program to access the witness as the database and perform the NP verification check. To satisfy statistical zero-knowledge property, we need to commit to the database using a statistically hiding commitment. The RAM program can use this commitment as external memory and takes the decommitment as part of its private state. It then runs the NP verification circuit on the decommitted witness and checks its validity. However, we cannot give the RAM SNARG proof and the decommitment in the clear as they may leak information about the underlying witness. Instead, we can give a NIZK proof (with poor rate) that the RAM SNARG verifier will accept the underlying proof using the decommitment as the private state. This can be shown to be sound and satisfy statistical zero-knowledge. However, this construction fails to achieve rate-1. Note that the size of the decommitment to any statistically hiding commitment is at least as large as the message (from standard incompressibility argument). Thus, NIZK proof size grows as $\mathsf{poly}(|w|, \lambda)$ as it takes the decommitment as a witness. Hence, this approach doesn't seem to give a rate-1 statistical NIZK argument.

Ben-David [BD24] considered a stronger soundness notion where the adversarial prover only sends a hash of the database but produces two proofs for the output of the computation being 0 and 1. This notion of soundness is again insufficient for our purposes as there doesn't seem a way to produce another proof (different from the one given by the prover) from the hash of the database. In a recent work, Kalai et al. [KLVW23] considered a stronger soundness definition called partial input soundness. Intuitively speaking, partial input soundness guarantees that if a RAM program "touches" only a few positions of the external memory, then if we hash the external memory using a somewhere extractable hash family that is extractable at those positions, a prover cannot generate a proof for a wrong statement about such a RAM program. In our setting, the program that checks the validity of the witness touches $O(|C|)$ positions in the external memory and hence, the size of a flexible RAM SNARG proof is $\Omega(|C|)$. In contrast, our techniques ensure that the size of this proof is $|w| + \mathsf{poly}(\lambda)$ though our RAM program touches $|C|$ locations of the external memory.

3 Preliminaries

Let λ denote the cryptographic security parameter. We assume that all cryptographic algorithms implicitly take 1^λ as input. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$ there exists λ_0 such that for all $\lambda > \lambda_0$ we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function. For any $i \in [n]$, let x_i denote the symbol at the i-th co-ordinate of x, and for any $T \subseteq [n]$, let $x_T \in \{0,1\}^{|T|}$ denote the projection of x to the co-ordinates indexed by T. We use $\mathsf{supp}(X)$ to denote the support of a random variable X.

For a probabilistic algorithm A, we denote $A(x; r)$ to be the output of A on input x with the content of the random tape being r. When r is omitted, $A(x)$ denotes a distribution. For a finite set S, we denote $x \leftarrow S$ as the process of sampling x uniformly from the set S. We will use PPT to denote Probabilistic Polynomial Time algorithm. We assume w.l.o.g. that the length of the randomness for all cryptographic algorithms is λ.

Definition 1. *We say that two distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable if for every non-uniform PPT distinguisher D, we have $|\Pr[D(1^\lambda, X_\lambda) = 1]| - \Pr[D(1^\lambda, Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.*

3.1 Non-Interactive Zero-Knowledge

Non-interactive zero-knowledge (NIZKs) schemes allows a prover to non-interactively prove that a given statement is in an NP-language. At the moment, NIZKs are known from a variety of hardness assumptions

such as factoring-based assumptions [BFM88, FLS90, BDSMP91], pairing-based assumptions [GOS06], LWE [CCH+19, PS19], or subexponential DDH [JJ21].

Definition 2 (Non-interactive zero-knowledge)**.** *A non-interactive zero-knowledge (NIZK) argument of knowledge (AoK) for a* NP *language* $\mathcal{L}$*, with corresponding instance-witness relation* $\mathcal{R} : \mathcal{X} \times \mathcal{W} \to \{0,1\}$*, is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *such that:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: *It is a PPT algorithm that takes as input the unary encoding of the security parameter* 1^λ *and outputs a common reference string* crs*.*
- $\mathsf{P}(\mathsf{crs}, x, w) \to \pi$: *It is a PPT algorithm that takes as input a common reference string* crs*, a statement* $x \in \mathcal{X}$ *and a witness* $w \in \mathcal{W}$*. It outputs a proof* π*.*
- $\mathsf{V}(\mathsf{crs}, x, \pi) \to b$: *It is a deterministic algorithm that takes as input a common reference string* crs*, a statement* $x \in \mathcal{X}$ *and a proof* π*. It outputs a bit* $b \in \{0,1\}$*.*

A NIZK-AoK satisfies the following properties:

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

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \quad : \quad \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w) \end{array} \right] = 1$$

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

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

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

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

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

NIZK argument of knowledge can be constructed from a NIZK and an extractable commitment. Specifically, we commit to the witness using an extractable commitment and prove that the committed value is a valid witness. This construction satisfies both zero-knowledge and argument of knowledge properties. NIZK arguments are known from factoring [FLS90], or from bilinear maps [GOS12], or from LWE [CCH+19, PS19], or sub-exponential DDH [JJ21].

Theorem 2. *Assuming either factoring, or* $O(1)$-LIN *assumption on prime-order groups with efficiently computable bilinear maps, or LWE, or sub-exponential DDH assumption, there exists a NIZK argument of knowledge.*

3.2 Somewhere Extractable Hash Families

Definition 3 (Somewhere Extractable Hash)**.** *A somewhere extractable hash family consists of the following polynomial time algorithms:*

- $\mathsf{Gen}(1^\lambda, N, i^*) \to (\mathsf{hk}, \mathsf{td})$: *It is a probabilistic setup algorithm that takes as input the security parameter* 1^λ*, the message length* N*, and an index* $i^* \in [N]$*. It outputs a hashing key* hk *and a trapdoor* td*.*

- $\mathsf{Hash}(\mathsf{hk}, x) \to v$: *It is a deterministic algorithm that takes as input a hashing key* hk *and a message* $x \in \{0,1\}^N$*, and outputs a hash value* $v \in \{0,1\}^{\ell_{hash}}$*.*
- $\mathsf{Open}(\mathsf{hk}, x, j) \to (b, \rho)$: *It is a deterministic algorithm that takes as input a hashing key* hk*, a message* $x \in \{0,1\}^N$ *and an index* $j \in [N]$*. It outputs a bit* $b \in \{0,1\}$ *and an opening* $\rho \in \{0,1\}^{\ell_{open}}$*.*
- $\mathsf{Verify}(\mathsf{hk}, v, j, b, \rho) \to \{0,1\}$: *It is a deterministic algorithm that takes as input a hashing key* hk*, a hash value* v*, an index* $j \in [N]$*, a bit* b *and an opening* ρ*, and it outputs 1 (accept) or 0 (reject).*
- $\mathsf{Extract}(\mathsf{td}, v) \to u$: *It is a deterministic algorithm that takes as input the trapdoor* td *and a hash value* v*, and it outputs a bit* $u \in \{0,1\}$*.*

We want a somewhere extractable hash family to satisfy the following properties:

- ***Efficiency:*** *The size of the hashing key* $|\mathsf{hk}|$*, the size of the hash* ℓ_{hash}*, the size of the opening* ℓ_{open} *and the running time of* Verify *are all bounded by* $\mathsf{poly}(\lambda, \log N)$*.*
- ***Opening completeness:*** *For every* $x \in \{0,1\}^N$*,* $j \in [N]$*, we have*

$$\Pr\left[\begin{array}{ll} b = x_j & (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i^*), \\ \wedge\ \mathsf{Verify}(\mathsf{hk}, v, j, b, \rho) = 1 \quad : & v = \mathsf{Hash}(\mathsf{hk}, x), \\ & (b, \rho) = \mathsf{Open}(\mathsf{hk}, x, j) \end{array}\right] = 1$$

- ***Index hiding:*** *For any* $N = \mathsf{poly}(\lambda)$*,* $(i_0, i_1) \in [N]$*, we have:*

$$\left\{\mathsf{hk} : (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i_0)\right\} \approx_c \left\{\mathsf{hk} : (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i_1)\right\}$$

- ***Collision-Resistance:*** *For any non-uniform PPT adversary* $\mathcal{A}$*, we have:*

$$\Pr\left[\begin{array}{ll} x \neq x' & (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, N, i^*), \\ \wedge\ \mathsf{Hash}(\mathsf{hk}, x) = \mathsf{Hash}(\mathsf{hk}, x') \quad : & (x, x') \leftarrow \mathcal{A}(\mathsf{hk}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- ***Somewhere statistically (resp. computational) binding w.r.t. opening:*** *For any all-powerful (resp. non-uniform PPT) adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we have:*

$$\Pr\left[\begin{array}{ll} u \neq b \wedge \mathsf{Verify}(\mathsf{hk}, v, i^*, b, \rho) = 1 \quad : & (1^N, i^*, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda), \\ & (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^n, N, i^*), \\ & (v, i^*, b, \rho) \leftarrow \mathcal{A}_2(\mathsf{st}, \mathsf{hk}), \\ & u \leftarrow \mathsf{Extract}(\mathsf{td}, v) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Remark 1 ([DGKV22, KLVW23])**.** *Notice that we can easily convert any such somewhere extractable hash family into one that is extractable on* m *indices* $I = \{i_1, \ldots, i_m\}$ *by running each algorithm* m *times and concatenating the outputs. Under this transformation, the sizes of* ℓ_{hash}, ℓ_{open} *and the efficiency of the* Verify *will be* $|I| \cdot \mathsf{poly}(\lambda, \log N)$*. We will use the shorthand notation* $\mathsf{Gen}(1^n, N, I)$ *to denote this construction, in which case* $\mathsf{Extract}(\mathsf{td}, v)$ *will output* m *bits* $(u_i)_{i \in I}$*.*

Remark 2. *We note that collision-resistance is implied by index hiding and somewhere statistical/computational binding w.r.t. to opening.*

[KLVW23] showed how to construct a somewhere extractable hash family satisfying the above properties assuming a rate-1 string OT. This can be constructed from DCR [DJ01], from QR/DDH/LWE [DGI+19] or from $O(1)$-LIN assumption on prime-order groups with efficiently computable bilinear maps.

Theorem 3. *Assuming either QR or DDH or LWE or* $O(1)$-LIN *assumption on prime-order groups with efficiently computable bilinear maps, there exists a construction of somewhere extractable hash family.*

3.2.1 Dual-Mode Somewhere Extractable Hash

We also define a variant of SEH where the hashing key can be set in a *statistically hiding mode*, meaning that the input $x \in \{0,1\}^N$ is statistically hidden given $v \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$.

Definition 4. *A dual-mode somewhere extractable hash family consists of the following polynomial time algorithms:*

- $\mathsf{Gen}(1^\lambda, N) \to \mathsf{hk}$: *It is a probabilistic setup algorithm that takes as input the security parameter* 1^λ *and the message length* N*. It outputs a hashing key* hk*.*
- $\mathsf{ExtGen}(1^\lambda, N, i^*) \to (\mathsf{hk}, \mathsf{td})$: *It is a probabilistic setup algorithm that takes as input the security parameter* 1^λ*, the message length* N*, and an index* $i^* \in [N]$*. It outputs a hashing key* hk *and a trapdoor* td*.*
- $\mathsf{Hash}(\mathsf{hk}, x)$*,* $\mathsf{Open}(\mathsf{hk}, x, j)$*,* $\mathsf{Verify}(\mathsf{hk}, v, j, b, \rho)$ *and* $\mathsf{Extract}(\mathsf{td}, v)$ *are defined as in Definition 3.*

A dual-mode SEH should fulfill all the properties described in Definition 3 (with respect to ExtGen*) and additionally it fulfills the following properties:*

- ***Mode indistinguishability.*** *For every* $\lambda, N \in \mathbb{N}$ *and any* $i \in [N]$ *we have:*

$$\left\{\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, N)\right\} \approx_c \left\{\mathsf{hk} : (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{ExtGen}(1^\lambda, N, i)\right\}.$$

  Note that index hiding is implied by mode indistinguishability.

- ***Statistical hiding:*** *For any inputs* $x, x' \in \{0,1\}^N$ *we have:*

$$\left\{(\mathsf{hk}, v) : \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{Gen}(1^n, N) \\ v = \mathsf{Hash}(\mathsf{hk}, x) \end{array}\right\} \approx_s \left\{(\mathsf{hk}, v) : \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{Gen}(1^n, N) \\ v = \mathsf{Hash}(\mathsf{hk}, x') \end{array}\right\}.$$

Remark 3. *We remark that the construction of SEH from rate-1 OT from [KLVW23] can be easily extended into a dual-mode SEH by using a (semi-honest) statistically sender secure rate-1 OT [DGI+19, ADD+22, BBDP22, BDS23]. At a high level, we will start by hashing* $\mathsf{Hash}(x||r)$ *where* $x \in \{0,1\}^N$ *is the input to be hashed and* r *is some uniformly chosen bit. If we use a statistically sender secure rate-1 OT in the SEH construction of [KLVW23], we will get a SEH for which all the hashed bits of the input are statistically hidden, except for the extractable one. Hence, in the statistically hiding mode, we just set the extractable index to be* $N+1$*, which corresponds to* r*. Thus, by the statistical sender security of the underlying rate-1 OT we will have that* $\mathsf{Hash}(\mathsf{hk}, x||r) \approx_s \mathsf{Hash}(\mathsf{hk}, x'||r)$ *for any* $x, x' \in \{0,1\}^N$*.*

3.3 Somewhere Extractable Batch Arguments

We recall the notion of batch arguments (BARGs). This is an argument system to succinctly prove that multiple instances $x_1, \ldots, x_k$ belong to an NP language $\mathcal{L}$ via witnesses $w_1, \ldots, w_k$, with the communication complexity less than $\sum |w_i|$. At the moment, we know BARG constructions from several standard assumptions such as LWE [CJJ22], pairings [WW22], subexponential DDH [CGJ+23], or a mixture of assumptions [CJJ21, HJKS22].

In particular, let $\mathsf{BatchCSAT}$ be the following language:

$$\mathsf{BatchCSAT} = \{(C, x_1, \ldots, x_k) \ : \ \exists w_1, \ldots, w_k \text{ s.t. } \forall i \in [k],\, C(x_i, w_i) = 1\},$$

where $C_n : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}$ is a Boolean circuit of size $s(n)$ that checks a relation with instance size n and witness size $m(n)$. For simplicity, we use s and m instead of $s(n)$ and $m(n)$.

Definition 5 (Somewhere extractable BARGs)**.** *A somewhere extractable batch argument for* $\mathsf{BatchCSAT}$ *with L-succinctness consists of the following polynomial time algorithms:*

- $\mathsf{Gen}(1^\lambda, 1^k, 1^s, i^*) \to (\mathsf{crs}, \mathsf{td})$: *It is a PPT algorithm that takes as input the unary encoding of the security parameter 1^λ, the number of instances 1^k, the circuit size 1^s and an index $i^* \in [k]$. It outputs a common reference string* crs *and a trapdoor* td.

- $\mathsf{P}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \{w_i\}_{i\in[k]}) \to \pi$*: It is a PPT algorithm that takes* crs*, a circuit C, k statements $x_1, \ldots, x_k \in \{0,1\}^n$ and k witnesses $w_1, \ldots, w_k \in \{0,1\}^m$. It outputs a proof π.*

- $\mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) \to b$*: It is a deterministic algorithm that takes* crs*, a circuit C, k statements $\{x_i\}_{i\in[k]}$ and a proof π. It outputs a bit $b \in \{0,1\}$.*

- $\mathsf{Extract}(\mathsf{td}, C, \{x_i\}_{i\in[k]}, \pi) \to w^*$*: It is a deterministic algorithm that takes a trapdoor* td*, a circuit C, k statements $\{x_i\}_{i\in[k]}$ and a proof π. It outputs a witness w^* for instance x_{i^*}.*

We want a somewhere extractable batch argument to satisfy the following properties:

- ***L-succinctness:*** *The* crs *and the proof π have length at most $L(k,\lambda) \cdot \mathsf{poly}(s)$, and the verifier runs in time $L(k,\lambda) \cdot \mathsf{poly}(s) + k \cdot \mathsf{poly}(n, \lambda)$.*

- ***Completeness:*** *For all $\lambda \in \mathbb{N}$, all $k, n \in \mathsf{poly}(\lambda)$, all circuits $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ at size most s and all $(x_1, \ldots, x_k)$ and $(w_1, \ldots, w_k)$ such that $C(x_i, w_i) = 1$ and any $i^* \in [k]$, we have:*

$$\Pr\left[1 \leftarrow \mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^s, i^*) \\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \{w_i\}_{i\in[k]})\end{array}\right] = 1.$$

- ***Index hiding:*** *For all $k, s = \mathsf{poly}(\lambda)$, $i_0, i_1 \in [k]$*

$$\left\{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^s, i_0)\right\} \approx_c \left\{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^s, i_1)\right\}$$

- ***Somewhere argument of knowledge:*** *There exists a PPT extractor* Ext *such that for any PPT adversary $\mathcal{A}$ and for any $k, s = \mathsf{poly}(\lambda)$, and any index $i^* \in [k]$ we have that*

$$\Pr\left[\begin{array}{c}1 \leftarrow \mathsf{V}(\mathsf{crs}, C, \{x_i\}_{i\in[k]}, \pi) \\ \wedge \\ C(x_{i^*}, w^*) \neq 1\end{array} : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda, 1^k, 1^s, i^*) \\ (C, \{x_i\}_{i\in[k]}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w^* \leftarrow \mathsf{Ext}(\mathsf{td}, C, \{x_i\}_{i\in[k]}, \pi)\end{array}\right] \leq \mathsf{negl}.$$

Index BARGs. We say that a somewhere extractable BARG scheme is an index seBARG if the instances $x_1, \ldots, x_k$ are all of the form $x_i = i$; however, in the L-succinctness property we require that the verification algorithm runs in time $L(k,\lambda) \cdot \mathsf{poly}(s)$, since it doesn't have to read all the instances anymore.

Theorem 4 ([CJJ22, DGKV22, PP22, WW22, CGJ+23, KLVW23])**.** *Assuming either LWE or sub-exponential DDH or $O(1)$-*LIN *assumption on prime-order groups with efficiently computable bilinear maps, there exists a construction of somewhere extractable index BARGs with* $\mathsf{poly}(\lambda, \log k)$*-succinctness.*

4 Rate-1 Dual-Mode Fully Local Extractable Commitments

In this section we present a new commitment scheme with some special properties: i) It is dual-mode, meaning that we can set the CRS to be in either extractable or statistically hiding mode, ii) It is local openable, meaning that opening a bit of the message should only require a opening of sublinear size, and iii) It is rate 1, meaning that the size of the commitment is identical to the size of the message (up to sublinear additive factors).

Definition 6 (Dual-Mode Fully-local extractable commitment)**.** *A dual-mode fully-local extractable commitment (FLC) scheme parametrized by $k, \ell \in \mathbb{N}$ consists of the following polynomial time algorithms:*

- $\mathsf{Setup}(1^\lambda, k, \ell) \to \mathsf{crs}$: *It is a PPT algorithm that takes in the unary encoding of the security parameter* 1^λ *and output a common reference string* crs.
- $\mathsf{ExtSetup}(1^\lambda, k, \ell) \to (\mathsf{crs}, \mathsf{td})$: *It is a PPT algorithm that takes in the unary encoding of the security parameter* 1^λ *and output a common reference string* crs *and a trapdoor* td.
- $\mathsf{Com}(\mathsf{crs}, \mathbf{m}) \to (\mathsf{com}, \mathsf{st})$: *It is a PPT algorithm that takes in* crs *and the message* $\mathbf{m} \in \{0,1\}^k$ *and outputs a commitment string* com *(which can be decomposed as* $\mathsf{com} = (\mathbf{d}_1, \ldots, \mathbf{d}_\ell)$*) and a state* st.
- $\mathsf{LocOpen}(\mathsf{crs}, \mathsf{com}, \mathsf{st}, m, i)$: *It is a deterministic algorithm that takes in* crs*, a commitment* com*, a state* st*, a message* $m = (m_1, \ldots, m_k)$ *and an index* $i \in [k]$*, and outputs a opening* ρ.
- $\mathsf{LocVer}(\mathsf{crs}, \mathbf{d}_j, m_i, \rho) \to b$: *It is a deterministic algorithm that takes in a* crs*, a commitment* $(c, \mathbf{d}_j)$ *where* $j \in [\ell]$*, a message* m_i *and the opening* ρ *and outputs a bit* $b \in \{0,1\}$.
- $\mathsf{Extract}(\mathsf{td}, \mathsf{com}) \to m$: *It is a deterministic algorithm that takes in the trapdoor* td *and a commitment string* com *and outputs a message* m.

We require a dual-mode fully-local extractable commitment to satisfy the following properties:

- ***Mode indistinguishability:*** *For every* $\lambda, k, \ell \in \mathbb{N}$ *we have:*

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

- ***Local Opening Completeness:*** *For every message* $m = (m_1, \ldots, m_k)$ *and any indices* $j \in [\ell], i \in [k]$ *such that* $j \in \{(i-1)k/\ell, \ldots, ik/\ell\}$*, we have:*

$$\Pr\left[\begin{array}{l|ll} \mathsf{LocVer}(\mathsf{crs}, \mathbf{d}_j, m_i, \rho) = 1 - \mathsf{negl}(\lambda) & : & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{ExtSetup}(1^\lambda), \\ & & (\mathsf{com}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{crs}, m) \\ & & \rho \leftarrow \mathsf{LocOpen}(\mathsf{crs}, \mathsf{com}, \mathsf{st}, m, i) \end{array}\right] = 1$$

  where $\mathsf{com} = (\mathbf{d}_1, \ldots, \mathbf{d}_\ell)$.

- ***Statistical Hiding:*** *For any two messages* m_0, m_1 *of equal length that is* $\mathsf{poly}(\lambda)$*, we have:*

$$\begin{aligned} &\left\{(\mathsf{crs}, \mathsf{com}) : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{com} \leftarrow \mathsf{Com}(\mathsf{crs}, m_0)\right\} \approx_s \\ &\quad \left\{(\mathsf{crs}, \mathsf{com}) : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{com} \leftarrow \mathsf{Com}(\mathsf{crs}, m_1)\right\} \end{aligned}$$

- ***Extraction Correctness:*** *For any PPT adversary* $\mathcal{A}$*, we have:*

$$\Pr\left[\begin{array}{l|ll} \wedge_i \mathsf{LocVer}(\mathsf{crs}, \mathbf{d}_j, m_i, \rho_i) = 1 \wedge \mathbf{m} \neq \mathbf{m}' & : & (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{ExtSetup}(1^\lambda), \\ & & (\mathsf{com}, \mathbf{m}, \{\rho_i\}_i) \leftarrow \mathcal{A}(\mathsf{crs}), \\ & & \mathbf{m}' \leftarrow \mathsf{Extract}(\mathsf{td}, \mathsf{com}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

  where $\mathsf{com} = (\mathbf{d}_1, \ldots, \mathbf{d}_\ell)$ *and* $\mathbf{m} = (m_1, \ldots, m_k)$.

- ***Efficiency:*** *The rate-1 property states that* $|\mathsf{com}| = k + o(k) \cdot \mathsf{poly}(\lambda)$*. Additionally, we require that* LocVer *runs in time* $o(k) \cdot \mathsf{poly}(\lambda)$*, which also implies that* $|\rho| = o(k) \cdot \mathsf{poly}(\lambda)$.

Remark 4 (Simple construction achieving computational hiding)**.** *If we drop the statistically hiding property, then this notion can be easily achieved with an extractable commitment and a PRF. To commit to the message, simply compute a commitment to the PRF key* K *and compute* $m' = m \oplus (\mathsf{PRF}(K, 1), \ldots, \mathsf{PRF}(K, k))$*. It is easy to see that this construction is extractable and local openable. This construction only achieves computational hiding, as the PRF only provides computational pseudorandomness.*

4.1 Rate-1 Dual-Mode Fully Local Extractable Commitment from Rate-1 OT

We present a generic construction from rate-1 oblivious transfer with (semi-honest) statistical sender security [DGI+19, ADD+22, BBDP22, BDS23]. At the moment, we know how to build this primitive from hardness assumptions such as DCR [DJ01], LWE [BDGM19], DDH, QR [DGI+19], k-LIN [KLVW23] or a combination of DDH and LPN [BBDP22, BDS23].

Definition 7 (Rate-1 oblivious transfer)**.** *A rate-1 oblivious transfer (OT) consists of the following algorithms:*

- $\mathsf{OTR}(1^\lambda, b) \to (\mathsf{st}, \mathsf{ot}_1)$: *It is a PPT algorithm that takes in the unary encoding of the security parameter 1^λ and a bit $b \in \{0,1\}$, and outputs a private state st and an OT receiver message ot_1.*
- $\mathsf{OTS}(\mathsf{ot}_1, (m_0, m_1); r) \to \mathsf{ot}_2$: *It is a PPT algorithm that takes in an OT message ot_1 and a pair of messages (m_0, m_1) (using random coins $r \in \lambda$), and outputs an OT sender message ot_2*
- $\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2) \to m$: *It is a deterministic algorithm that takes in a private state st and an OT sender message ot_2, and outputs a message m.*
- ***Correctness:*** *For every $\lambda \in \mathbb{N}$ $b \in \{0,1\}$ and every pair of messages (m_0, m_1) we have:*

$$\Pr\left[\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2) = m_b : \begin{array}{c}(\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, b) \\ \mathsf{ot}_2 \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (m_0, m_1))\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

- ***Receiver security:*** *For every $\lambda \in \mathbb{N}$ we have that:*

$$\left\{\mathsf{ot}_1 : (\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, 0)\right\} \approx_c \left\{\mathsf{ot}_1 : (\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, 1)\right\}$$

- ***Semi-honest statistical sender security:*** *For every $\lambda \in \mathbb{N}$, any $b \in \{0,1\}$ and any pair of messages (m_0, m_1) we have:*

$$\left\{(\mathsf{ot}_2, \mathsf{st}) : \begin{array}{c}(\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, b) \\ \mathsf{ot}_2 \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (m_0, m_1))\end{array}\right\} \approx_s \left\{(\mathsf{ot}_2, \mathsf{st}) : \begin{array}{c}(\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, b) \\ \mathsf{ot}_2 \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (m_b, m_b))\end{array}\right\}.$$

- ***Efficiency:*** *We require that $|\mathsf{ot}_2| = |m_b| + \mathsf{poly}(\lambda)$.*

We need an additional property for the rate-1 OT called verifiable correctness, which was defined in [KLVW23].

Definition 8. *An OT scheme $(\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ is said to be verifiable correct if there is polynomial-time deterministic algorithm $\mathsf{Valid}(\mathsf{ot}_1, m_0, m_1, r) \to \{0,1\}$ such that:*

- *If $1 = \mathsf{Valid}(\mathsf{ot}_1, m_0, m_1, r)$ then*

$$\Pr\left[\mathsf{OTD}(\mathsf{st}, \mathsf{ot}_2) = m_b : \begin{array}{c}(\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, b) \\ \mathsf{ot}_2 \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (m_0, m_1); r)\end{array}\right] = 1.$$

- *For all b, m_0, m_1, r,*

$$\Pr\left[1 = \mathsf{Valid}(\mathsf{ot}_1, m_0, m_1, r) : (\mathsf{st}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, b)\right] = 1 - \mathsf{negl}(\lambda).$$

Rate-1 OT schemes with (semi-honest) statistical sender security can be built from DDH, QR or LWE [DGI+19, BBDP22]. We now sketch a construction of a (semi-honest) statistical sender secure rate-1 OT from $\mathcal{O}(1)$−LIN assumption. This follows from a simple modification of the construction given in [KLVW23]. Like in [KLVW23], we give a construction of a rate-1 batch OT scheme which implies a rate-1 string OT.

Rate-1 OT with semi-honest statistical sender security from $\mathcal{O}(1)-$LIN. We show how to modify the rate-1 OT from $\mathcal{O}(1)-$LIN assumption given in [KLVW23] to provide semi-honest statistical sender security.

- **Receiver's message**: Let $\mathbf{b} \in \{0,1\}^{n-k}$ be the vector of receiver's choice bits. Pad these bits with default bits to obtain a vector of length n. We will slightly abuse the notation and use $\mathbf{b}$ to denote this padded vector. Let $\mathbf{X} \in \{0,1\}^{n \times n}$ be a matrix whose diagonal elements are given by $\mathbf{b}$. Let $\mathbf{Y} \in \{0,1\}^{(k+n)\times n}$ be another matrix that as $\mathbf{X}$ in the bottom block and zeroes elsewhere. The receiver chooses a random matrix $\mathbf{B} \in \mathbb{Z}_p^{(k+n)\times n}$ with rank k. This is done by choosing the first k rows of $\mathbf{B}$ uniformly at random and then choosing the last n rows to be random linear combinations of the first k rows. Let the $(k+i)$-th row be given by $\mathbf{v}_i \cdot \mathbf{B}[1,\ldots,k]$ where $\mathbf{B}[1,\ldots,k]$ denotes the first k rows of $\mathbf{B}$. The receiver's message is given by $g^{(\mathbf{B}+\mathbf{Y})}$. Note that the k-LIN assumption implies that $g^{\mathbf{B}}$ chosen as above is indistinguishable from $g^{\mathbf{U}}$ where $\mathbf{U}$ is uniformly chosen from $\mathbb{Z}_p^{(k+n)\times n}$. This ensures that the receiver's choice bits are computationally hidden.

- **Sender's message.** Let $\mathbf{s}_0, \mathbf{s}_1 \in \{0,1\}^{n-k}$ be the vector of sender's inputs in the batch OT. The sender samples $2k$ random elements from $\mathbb{Z}_p$ and extends $\mathbf{s}_0, \mathbf{s}_1$ with k elements each to obtain new vectors in $\mathbb{Z}_p^n$. We slightly abuse the notation and denote the new vectors as $\mathbf{s}_0, \mathbf{s}_1$ respectively. The sender computes
$$\mathbf{h} = g^{(\mathbf{B}+\mathbf{Y})\cdot(\mathbf{s}_1-\mathbf{s}_0)} \odot g^{0^k \| \mathbf{s}_0}.$$
where $\odot$ denotes point-wise product. It then parses $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3) \in \mathbb{G}^k \times \mathbb{G}^{n-k} \times \mathbb{G}^k$. For each $i \in [n-k]$, it computes $z_i = \mathsf{DDLog}(h_{2,i})$ where DDLog is the Distributed Discrete Log introduced in [BGI16]. It sends $(\mathbf{h}_1, (z_1, \ldots, z_{n-k}))$.

- **Output Computation.** Let $\mathbf{u}_i = (-\mathbf{v}_i, \mathbf{e}_i)$ be the vector that is in the left kernel of $\mathbf{B}$ for each $i \in [n]$ (where $\mathbf{e}_i$ is the i-th unit vector of length n). This implies that for each $i \in [n-k]$,
$$((-\mathbf{v}_i) \diamond \mathbf{h}_1) \cdot h_{2,i} = g^{\mathbf{u}_i\left((\mathbf{B}+\mathbf{Y})\cdot(\mathbf{s}_1-\mathbf{s}_0)+0^k\|\mathbf{s}_0\right)} = g^{s_{b_i,i}}$$
where $(-\mathbf{v}_i) \diamond \mathbf{h}_1$ denotes applying the linear map $-\mathbf{v}_i$ to the exponent of the group elements $\mathbf{h}_1$. This means that with probability at least $1 - 1/n\lambda$, for each $i \in [n-k]$, $\mathsf{DDLog}((-\mathbf{v}_i) \diamond \mathbf{h}_1)) \oplus z_i = s_{b_i,i}$. Therefore, by union bound, the probability of correctness of the complete construction is $1 - 1/\lambda$. This can be bootstrapped to $1 - \mathsf{negl}(\lambda)$ correctness via the same techniques in [DGI+19].

- **Statistical Sender Privacy.** Note that except with negligible probability, $z_i = \mathsf{DDLog}((-\mathbf{v}_i) \diamond \mathbf{h}_1)) \oplus s_{b_i,i}$. This can be simulated given the knowledge of the output and $\mathbf{h}_1$. Furthermore, since the last k elements of $\mathbf{s}_1 - \mathbf{s}_0$ are uniformly chosen from $\mathbb{Z}_p^k$, it follows that $\mathbf{B}[1,\ldots,k] \cdot (\mathbf{s}_1 - \mathbf{s}_0)$ is uniformly distributed with overwhelming probability. This means that $\mathbf{h}_1$ is uniformly distributed and this completes the statistical sender security proof.

Construction. We give a formal description of a rate-1 FLC from rate-1 OT. Let $(\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ be a rate-1 OT which is verifiable correct.

- $\mathsf{Setup}(1^\lambda)$:
  1. Run $(\mathsf{st}_{OT}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, 1)$.
  2. Sample $\mathbf{u} \leftarrow\$ \{0,1\}^t$.
  3. Output $\mathsf{crs} = (\mathsf{ot}_1, \mathbf{u})$.
- $\mathsf{ExtSetup}(1^\lambda)$:
  1. Run $(\mathsf{st}_{OT}, \mathsf{ot}_1) \leftarrow \mathsf{OTR}(1^\lambda, 0)$.

2. Sample $\mathbf{u} \leftarrow\!\!\$\ \{0,1\}^t$.
3. Output $\mathsf{crs} = (\mathsf{ot}_1, \mathbf{u})$ and $\mathsf{td} = \mathsf{st}_{OT}$.

- $\mathsf{Com}(\mathsf{crs}, \mathbf{m})$:
  1. Parse $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$ where each $\mathbf{m}_i \in \{0,1\}^t$ and $t = k/\ell$.
  2. For all $i \in [\ell]$ compute $\mathsf{ot}_{2,i} \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (\mathbf{m}_i, \mathbf{u}); r_i)$ using random coins $r_i \leftarrow\!\!\$\ \{0,1\}^\lambda$.
  3. Output $\mathsf{com} = \{\mathsf{ot}_{2,i}\}_{i\in[\ell]}$ and $\mathsf{st} = \{r_i\}_{i\in[\ell]}$.
- $\mathsf{LocOpen}(\mathsf{crs}, \mathsf{com}, \mathsf{st}, \mathbf{m}, i \in [k])$:
  1. Parse $\mathsf{com} = \{\mathsf{ot}_{2,i}\}_{i\in[\ell]}$, $\mathsf{st} = \{r_i\}_{i\in[\ell]}$, and $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_j)$
  2. Let $j \in [\ell]$ such that $i \in \{(j-1)k/\ell, jk/\ell\}$.
  3. Output $\rho = (r_j, \mathbf{m}_j)$.
- $\mathsf{LocVer}(\mathbf{d}_j, m_i, \rho)$:
  1. Parse $\mathbf{d}_j = \mathsf{ot}_{2,j}$, $\rho = (r_j, \mathbf{m}_j)$ and $\mathbf{m}_j = (m_{j,1}, \ldots, m_{j,k/\ell})$.
  2. If $\mathsf{ot}_{2,j} \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (\mathbf{m}_j, \mathbf{u}); r_j)$, if $m_{j,i \mod k/\ell} = m_i$ and if $1 = \mathsf{Valid}(\mathsf{ot}_1, \mathbf{m}_j, \mathbf{u}, r_j)$, output 1.
- $\mathsf{Extract}(\mathsf{td}, \mathsf{com})$:
  1. Parse $\mathsf{td} = \mathsf{st}_{OT}$ and $\mathsf{com} = \{\mathsf{ot}_{2,i}\}_{i\in[\ell]}$.
  2. For all $i \in [\ell]$ compute $\mathbf{m}_i \leftarrow \mathsf{OTD}(\mathsf{st}_{OT}, \mathsf{ot}_{2,i})$.
  3. Output $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$

Efficiency. Set $\ell = k^{1-\epsilon}$ for $\epsilon > 0$. The commitment com is composed by $\{\mathsf{ot}_{2,i}\}_{i\in[\ell]}$ where

- $|\mathsf{ot}_{2,i}| = k/\ell + \mathsf{poly}(\lambda) = k^\epsilon + \mathsf{poly}(\lambda)$.

Hence the rate ϱ_{com} is

$$\varrho_{\mathsf{com}} = \frac{|\mathsf{com}|}{|\mathbf{m}|} = \frac{k^{1-\epsilon} \cdot |\mathsf{ot}_{2,i}|}{k} = 1 + \frac{\mathsf{poly}(\lambda)}{k^\epsilon}.$$

On the other hand, the size of the opening ρ for a position $i \in [k]$ is

$$|\rho| = |r_j| + \mathbf{m}_j = k^\epsilon + \mathsf{poly}(\lambda)$$

which is sublinear in the size of the message k. Finally, the algorithm LocVer runs in time $P(k^\epsilon, \lambda)$ for a fixed polynomial P. For a proper choice of ϵ we have that LocVer runs in time $o(k) \cdot \mathsf{poly}(\lambda)$.

Remark 5. *Note that the choice of ϵ can be tuned at setup time. In particular, the parameter $\epsilon > 0$ (which determines the size of the opening ρ and the runtime of LocVer) can be as small as we want. This will be important to set up the parameters that allow us to achieve rate-1 NIZKs with statistical zero-knowledge (see Section 5).*

Analysis. We now proceed with the analysis of the scheme. Local opening completeness follows easily from the verifiable correctness of the underlying OT scheme. We proceed to show all other security properties of the scheme.

Lemma 1. *The construction presented above is dual-mode indistinguishable given that the underlying rate-1 OT is receiver secure.*

Proof. Assume that there is an adversary $\mathcal{A}$ that breaks the mode indistinguishability of the commitment scheme. This adversary can be directly used as an adversary against the receiver security of the underlying OT. □

Lemma 2. *The construction presented above has extraction correctness given that the underlying rate-1 OT is verifiable correct.*

Proof. Let $(\mathsf{com}, \mathbf{m}, \{\rho_i\})$ be the output of the adversary in the extraction correctness game. If $\mathsf{LocVer}(\mathsf{crs}, \mathbf{d}_j, m_i, \rho_i) = 1$ for all $i \in [k]$ then this means that $\mathsf{ot}_{2,j} \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (\mathbf{m}_j, \mathbf{u}); r_j)$, $m_{j,i \mod k/\ell} = m_i$ and $1 = \mathsf{Valid}(\mathsf{ot}_1, \mathbf{m}_j, \mathbf{u}, r_j)$. By the verifiable correctness of the underlying OT scheme, this immediately implies that for all $j \in [\ell]$ compute $\mathbf{m}_j \leftarrow \mathsf{OTD}(\mathsf{st}_{OT}, \mathsf{ot}_{2,j})$. □

Lemma 3. *The construction presented above is statistical hiding given that the underlying rate-1 OT is semi-honest statistical sender secure.*

Proof. In the statistically hiding mode, the crs is composed by ot_1 which is the output of $\mathsf{OTR}(1^\lambda, 1)$ and the commitment is of the form $\mathsf{ot}_{2,i} \leftarrow \mathsf{OTS}(\mathsf{ot}_1, (\mathbf{m}_i, \mathbf{u}); r_i)$. Thus, breaking statistically hiding of the commitment is equivalent to breaking statistical sender security of the underlying OT. □

5 Rate-1 Statistical Non-Interactive Zero-Knowledge

In this section we show how to build a rate-1 NIZK for which the zero-knowledge property holds statistically.

Building Blocks. The construction uses the following building blocks:

- A rate-1 dual-mode fully local extractable commitment scheme $(\mathsf{Setup}_C, \mathsf{ExtSetup}_C, \mathsf{Com}, \mathsf{LocOpen}, \mathsf{LocVer}, \mathsf{Extract}_C)$ (see Definition 6), for which the local openings have sublinear size m^ε, for any $\varepsilon < 0$ (see Remark 5).
- A dual-mode somewhere extractable hash family $(\mathsf{Gen}_H, \mathsf{ExtGen}_H, \mathsf{Hash}_H, \mathsf{Open}_H, \mathsf{Verify}_H, \mathsf{Extract}_H)$ with statistical hiding (Definition 6).
- A NIZK-AoK $(\mathsf{Setup}_N, \mathsf{P}_N, \mathsf{V}_N)$ with statistical zero knowledge for an NP language $\mathcal{L}_N$ described below.
- A somewhere extractable batch argument $(\mathsf{Gen}_B, \mathsf{P}_B, \mathsf{V}_B, \mathsf{Extract}_B)$ for the language defined below. Let $s(n)$ be the size of C_n. For simplicity, we will use s instead of $s(n)$.
- A $(\mathsf{MDGen}, \mathsf{MDHash}, \mathsf{MDRead}, \mathsf{MDVerify}, \mathsf{MDWrite})$ be a Merkle-Damgård hash tree using a collision resistant hash key hk. MDRead gives a local opening to any location i. $\mathsf{MDVerify}$ checks if this opening is correct. $\mathsf{MDWrite}$ to write to a location i first takes takes a local opening to location i, runs $\mathsf{MDVerify}$ and then, updates the hash using the updated value at location i.
- A NIZK-AoK $(\mathsf{Setup}_{N'}, \mathsf{P}_{N'}, \mathsf{V}_{N'})$ with statistical zero knowledge for the language

$$\mathcal{L}' = \{(\mathsf{hk}_H, v, 0, \mathsf{st}_0) : \exists \rho_0 \text{ s.t. } \mathsf{Verify}_H(\mathsf{hk}_0, v, 0, \mathsf{st}_0, \rho_0) = 1\}.$$

Construction. We give the formal description of a rate-1 NIZK-AoK for CSAT.

- $\mathsf{Setup}(1^\lambda)$:
  1. Sample $\mathsf{crs}_C \leftarrow \mathsf{Setup}_C(1^\lambda)$.
  2. Sample $\mathsf{crs}_N \leftarrow \mathsf{Setup}_N(1^\lambda)$ and $\mathsf{crs}_{N'} \leftarrow \mathsf{Setup}_{N'}(1^\lambda)$.
  3. Sample $\mathsf{hk} \leftarrow \mathsf{Gen}_H(1^\lambda, N, 1)$ where $N = T \cdot \lambda$.
  4. Sample $\mathsf{hk}' \leftarrow \mathsf{MDGen}(1^\lambda)$.

5. Sample $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, 1)$ where $k = T$.
6. Output $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$.

- $\mathsf{P}(\mathsf{crs}, x, w)$: Let $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$.
  1. Compute $(\mathsf{com}, \mathsf{st}_w) \leftarrow \mathsf{Com}(\mathsf{crs}_C, w)$.
  2. Let $\mathsf{init}_0 = x \| 0^S$ and compute $\mathsf{st}_0 = \mathsf{MDHash}(\mathsf{hk}', \mathsf{init}_0)$.
  3. Evaluate $\mathcal{C}_n(w, x)$ to compute all the values carried by each wire. Let $\mathsf{wire}_{n+m+1}, \ldots, \mathsf{wire}_{n+m+S}$ be the values carried by the internal wires.
  4. For every g from 1 to T:
     (a) Let i and j be the input wires to g-th gate and let ℓ be the output wire.
     (b) Set $(b_i^g, \rho_i^g) = \mathsf{MDRead}(\mathsf{hk}', \mathsf{st}_{g-1}, \mathsf{init}_{g-1}, i)$, $(b_j^g, \rho_j^g) = \mathsf{MDRead}(\mathsf{hk}', \mathsf{st}_{g-1}, \mathsf{init}_{g-1}, j)$ and $(b_\ell^g, \rho_\ell^g) = \mathsf{MDRead}(\mathsf{hk}', \mathsf{st}_{g-1}, \mathsf{init}_{g-1}, \ell)$.
     (c) Let init_g be same as init_{g-1} except that ℓ-th entry is set to wire_ℓ.
     (d) Compute $\mathsf{st}_g = \mathsf{MDHash}(\mathsf{hk}', \mathsf{init}_g)$.
  5. Compute $v = \mathsf{Hash}(\mathsf{hk}, \mathsf{st}_0 \| \ldots \| \mathsf{st}_T)$ and for each $i \in [0, T]$, compute $(\mathsf{st}_i, \rho_i) \leftarrow \mathsf{Open}(\mathsf{hk}, v, \mathsf{st}_0 \| \ldots \| \mathsf{st}_T, i)$.
  6. Decompose com into $(\mathbf{d}_1, \ldots, \mathbf{d}_m)$ where each block $\mathbf{d}_i$ contains w_i.[5]
  7. For each $g \in [T]$, compute the witness $\tilde{w}_g$ for the language described below with the instance being $\tilde{x}_g = (\mathsf{crs}_N, v, \mathsf{hk}, \mathsf{st}_0, g, \mathbf{d}_i^g, \mathbf{d}_j^g)$.
  8. Compute $\pi_B = \mathsf{P}_B(\mathsf{crs}_B, (\tilde{x}_1, \ldots, \tilde{x}_T), (\tilde{w}_1, \ldots, \tilde{w}_T))$.
  9. Set $x' = (\mathsf{hk}, v, 0, \mathsf{st}_0)$ and $w' = \rho_0$. Compute $\sigma \leftarrow \mathsf{P}_{N'}(\mathsf{crs}_{N'}, x', w')$.
  10. Output $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
- $\mathsf{V}(\mathsf{crs}, x, \pi)$:
  1. Parse π as $(\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$
  2. Compute $\mathsf{init}_0 = x \| 0^S$ and check if $\mathsf{st}_0 = \mathsf{MDHash}(\mathsf{hk}', \mathsf{init}_0)$.
  3. Decompose com into $(\mathbf{d}_1, \ldots, \mathbf{d}_m)$ where each block $\mathbf{d}_i$ contains the bit w_i.
  4. Set $x' = (\mathsf{hk}, v, 0, \mathsf{st}_0)$ and check if $\mathsf{V}_{N'}(\mathsf{crs}_{N'}, x', \sigma) = 1$.
  5. For each $g \in T$, set $\tilde{x}_g = (\mathsf{crs}_N, v, \mathsf{hk}, \mathsf{st}_0, g, \mathbf{d}_i^g, \mathbf{d}_j^g)$.
  6. Check if $\mathsf{V}_B(\mathsf{crs}_B, (\tilde{x}_1, \ldots, \tilde{x}_T), \pi_B) = 1$.
  7. If all the checks pass, V outputs 1.

Language for BARGs. We describe a language for our BARG scheme.

- **Input:** The CRS for a NIZK scheme crs_N, the hash digests v, hash key hk for SEH, a hash key hk' for MDHash, $\mathsf{st}_0^* \in \{0,1\}^\lambda$, an index $g \in [T]$ and two blocks $\mathbf{d}_i^g, \mathbf{d}_j^g$ of the commitment.
- **Witness:** A proof π_N.
- **Description of Relation Circuit:** The circuit checks if $V_N(\mathsf{crs}_N, (\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0^*, \mathsf{com}, g, \mathbf{d}_i^g, \mathbf{d}_j^g), \pi_N) = 1$ for the NP language $\mathcal{L}_N$ defined below. Let s be the size of this relation circuit.
- **Description of $\mathcal{L}_N$:** $\tilde{x}_g = (\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0^*, g, \mathbf{d}_i^g, \mathbf{d}_j^g) \in \mathcal{L}_N$ if there exists a witness $(\mathsf{st}_{g-1}, \rho_{g-1})$, (st_g, ρ_g), (b_i^g, ρ_i^g), $(\mathbf{d}_i^g, \delta_i^g)$, (b_j^g, ρ_j^g), $(\mathbf{d}_j^g, \delta_j^g)$, (b_ℓ^g, ρ_ℓ^g) such that:

[5] This can be done by decomposing $(\mathbf{c}_1, \ldots, \mathbf{c}_\ell)$ (as per Definition 6) and computing $(\mathbf{d}_1, \ldots, \mathbf{d}_m)$ where $\mathbf{d}_{i(j-1)+1} = \cdots = \mathbf{d}_{i(j-1)} = \mathbf{c}_j$ for all $i \in [k/\ell]$ and $j \in [\ell]$. In other words, each block $\mathbf{d}_i$ is identical to the block $\mathbf{c}_j$ for which we can open w_i.

1. If $g = 1$, check if $\mathsf{st}_0^* = \mathsf{st}_0$.
2. Check if $\mathsf{Verify}_H(\mathsf{hk}, v, g-1, \mathsf{st}_{g-1}, \rho_{g-1}) = 1$ and $\mathsf{Verify}_H(\mathsf{hk}, v, g, \mathsf{st}_g, \rho_g) = 1$.
3. Let i and j be the input wires to g-th gate in C and let ℓ be the output wire.
4. If $i \in [m]$ (that is, if i corresponds to a witness wire that is given as input to the circuit), check if $\mathsf{LocVer}(\mathsf{crs}_C, \mathbf{d}_i^g, b_i^g, \delta_i^g) = 1$. Similarly, if $j \in [m]$ (that is, if i corresponds to a witness wire that is given as input to the circuit), check if $\mathsf{LocVer}(\mathsf{crs}_C, \mathbf{d}_j^g, b_j^g, \delta_j^g) = 1$.
5. Check if $\mathsf{MDVerify}(\mathsf{hk}', \mathsf{st}_{g-1}, b_i^g, \rho_i^g) = 1$, $\mathsf{MDVerify}(\mathsf{hk}', \mathsf{st}_{g-1}, b_j^g, \rho_j^g) = 1$ and $\mathsf{MDVerify}(\mathsf{hk}', \mathsf{st}_{g-1}, b_\ell^g, \rho_\ell^g) = 1$.
6. Check if $b_\ell^g = 0$.
7. Compute $b_\ell = \mathsf{NAND}(b_i^g, b_j^g)$. If $g = T$, check if $b_\ell = 1$.
8. Check if $\mathsf{st}_g = \mathsf{MDWrite}(\mathsf{hk}', v, \ell, (b_\ell^g, \rho_\ell^g), b_\ell)$.

5.1 Proof Size

We first prove the following claim which will be used in the analysis of the proof size.

Claim 1. *The size of a proof π_B the BARG scheme presented above is $|w|^\varepsilon \cdot \mathsf{poly}(\log T, \lambda)$ for some $\varepsilon < 0$.*

Proof. In order to prove this claim, we start by showing that the NIZK verifier for language $\mathcal{L}_N$ runs in time $|w|^{\varepsilon_N} \cdot \mathsf{poly}(\lambda)$. The NIZK verifier runs in time $P_N(|\mathcal{C}_N|, \lambda)$ where $\mathcal{C}_N$ is the relation circuit for $\mathcal{L}_N$, for some fixed polynomial P_N. Notice that all operations in $\mathcal{C}_N$ run in time $\mathsf{poly}(\lambda)$, except for $\mathsf{LocVer}(\mathsf{crs}_C, \mathbf{d}_i^g, b_i^g, \delta_i^g) = 1$ and $\mathsf{LocVer}(\mathsf{crs}_C, \mathbf{d}_j^g, b_j^g, \delta_j^g) = 1$ (in case any of $i, j \in [m]$). In order to prove this, it is sufficient to prove that that the language $\mathcal{L}_N$ can be verified using a $\mathsf{poly}(\lambda)$ circuit as the NIZK verifier has a $\mathsf{poly}(\lambda)$ overhead over the size of this circuit.

- The size of st_{g-1} and st_g are $\mathsf{poly}(\lambda)$ as we set the SEH to be extractable at a single location from 1 to T.
- Since the SEH has local openings, it follows that the size of ρ_g and ρ_{g-1} are $\mathsf{poly}(\log T, \lambda) = \mathsf{poly}(\lambda)$. Therefore, the size of Verify_H is $\mathsf{poly}(\lambda)$.
- The size of $\mathsf{MDVerify}$ and the size of the openings $\rho_i^g, \rho_j^g, \rho_\ell^g$ are $\mathsf{poly}(\lambda)$.
- The computation of the PRF and the bit b_ℓ can be done using a $\mathsf{poly}(\lambda)$ sized circuit.
- $\mathsf{MDWrite}$ can be done using a circuit of size $\mathsf{poly}(\lambda)$.

Hence, it follows that the $\mathcal{L}_N$ can be verified using a circuit of size $\mathsf{poly}(\lambda)$ and this completes the proof of the claim.

By definition, LocVer runs in time $|w|^{\varepsilon_C}$ for any $\varepsilon_C < 0$. Setting ε_C to be small enough so that $P_N(|\mathcal{C}_N|, \lambda) = |w|^{\varepsilon'} \cdot \mathsf{poly}(\lambda)$ yields that Verify_N runs in time sublinear in $|w|$. Note that, using the FLC construction of Section 4.1, we have full control over the choice of ε_C (see Remark 5).

Finally, the BARG scheme adds an overhead of $Q_B(|\mathcal{C}_V|, \log T, \lambda)$ where $|\mathcal{C}_V|$ is the NIZK verifier circuit, for some fixed polynomial Q_B. Again, by setting ε' to be small enough, we obtain the desired result. □

Size of the proof π. The proof π is composed by $(\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$ where:

- $|\mathsf{com}| = |w| + o(|w|) \cdot \mathsf{poly}(\lambda)$.
- $|v| = \mathsf{poly}(\lambda)$, since we just need to extract st_i (which is of size $\mathsf{poly}(\lambda)$).
- $|\mathsf{st}_0| = \mathsf{poly}(\lambda)$ and $|\sigma| = \mathsf{poly}(\lambda)$ since the underlying NIZK for language $\mathcal{L}'$ only blows up the size of the proof by a factor of $\mathsf{poly}(\lambda)$ when compared to the size of the witness ρ, which is $|\rho_0| = \mathsf{poly}(\lambda)$.

- $|\pi_B| = o(|w|) \cdot \mathsf{poly}(\log T, \lambda)$ (from Claim 1).

Thus, we have that the rate ϱ_π of the proof is

$$\begin{aligned}\varrho_\pi = \frac{\pi}{|w|} &= \frac{|\mathsf{com}| + |v| + |\mathsf{st}_0| + |\sigma| + |\pi_B|}{|w|} = \frac{|w| + o(|w|) \cdot \mathsf{poly}(\lambda)}{|w|} \\ &= 1 + o(|w|) \cdot \mathsf{poly}(\lambda)/|w|\end{aligned}$$

which approaches 1 for large enough $|w|$.

Online/Offline Verification. We observe that for a slight modificaion of our construction, it is possible to split the verification into an offline/online steps where a majority of the verifier work is done in the offline step before receiving the proof. First, note that com is anyway public and available to the verifier. We can thus turn the BARG language into an index language (akin to [CJJ22]), by simply Merkle hashing com into a hash value u, and then giving as part of the BARG witness $\tilde{w}_i$ the local opening for the right block of com that needs to be fetched at step i of the computation. Then,

- **Offline Phase.** In this phase, the verifier can compute the partial hash[6] of $x \| 0^S$. Furthermore, the partial hash of 0^S can be computed once and for all and this partial hash could be made part of the CRS. Therefore, the verifier only needs to read x in the offline phase (which includes the circuit C in the case where we use a Universal Turing machine) and compute the partial hash of $x \| 0^S$ in time $|C| \cdot \mathsf{poly}(\lambda)$. Therefore, the running time of the offline phase is at most $|C| \cdot \mathsf{poly}(\lambda)$.
- **Online Phase.** Once the verifier receives the proof, it can combine the partial hash of $x \| 0^S$ with w' to obtain $\mathsf{st}_0 = \mathsf{MDHash}(\mathsf{hk}', \mathsf{init}_0)$ in time $|w'| \cdot \mathsf{poly}(\lambda)$. After the verifier computes st_0, it runs Verify_H and Verify_B. Verify_H can be computed in time $\mathsf{poly}(\lambda)$ and Verify_B for the case of index BARG takes time $\mathsf{poly}(\log T, \lambda) \cdot \mathsf{poly}(s) = \mathsf{poly}(\lambda)$ (from Claim 1). Therefore, the total running time of the online verifier is $|w| \cdot \mathsf{poly}(\lambda)$.

5.2 Analysis

We now proceed with the analysis of our scheme.

Theorem 5 (Completeness)**.** *The NIZK construction presented above is complete given that the underlying fully-local extractable commitment is local opening complete, the dual-mode SEH is opening complete and the underlying NIZKs and BARG scheme are complete.*

Proof. Assume that $x \in \mathcal{L}$ and that w is the corresponding witness. We will show that $1 \leftarrow \mathsf{V}(\mathsf{crs}, x, \pi)$ where $\pi = (w', v, (\mathsf{st}_0, \rho_0), \mathsf{com}, \pi_B)$.

First, by the completeness of N', we have that $\mathsf{V}_{N'}(\mathsf{crs}_{N'}, x', \sigma) = 1$ where $x' = (\mathsf{hk}, v, 0, \mathsf{st}_0) \in \mathcal{L}'$ by construction.

By construction and opening completeness of the underlying extractable commitment and SEH, we have that $X_g = (\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0^*, \mathsf{com}, g, \mathbf{d}_i^g, \mathbf{d}_i^g, \mathbf{d}_j^g) \in \mathcal{L}_N$ by all $g \in [T]$. Let w_g be the corresponding witness. Then, $\mathsf{V}_N(\mathsf{crs}_N, X_g, \pi_g) = 1$ by the completeness of the underlying NIZK, where $\pi_g \leftarrow P_N(\mathsf{crs}_N, X_g, w_g)$. Moreover, by the completeness of the underlying (index) BARG, we have that $\mathsf{V}_B(\mathsf{crs}_B, (\mathsf{crs}_N, \mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0, \mathsf{com}), \pi_B) = 1$ where $\pi_B = \mathsf{P}_B(\mathsf{crs}_B, (\mathsf{crs}_N, \mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0, \mathsf{com}), (w_1, \ldots, w_T))$ for $w_i = \pi_i$. □

Theorem 6 (Argument of knowledge)**.** *The NIZK construction presented above is an argument of knowledge given that the underlying SEH is index hiding and SSB, the underlying BARG is index hiding and somewhere extractable, the underlying NIZK is an argument of knowledge and the underlying Merkle-Damgård hash function is collision resistant.*

Proof. Let $(\mathsf{Ext}_{N,1}, \mathsf{Ext}_{N,2})$ be the extractor of the underlying NIZK scheme. We will start by describing the extractor $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$.

[6] By partial hash, we mean computing all the internal nodes in the Merkle-Damgård hash tree that are "influenced" by a part of the input and then giving out the "highest" ancestors of those nodes. The size of the partial hash is at most one hash value for each level of the tree and hence, this size is at most $\mathsf{poly}(\lambda)$.

$\mathsf{Ext}_1(1^\lambda)$:

- $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$.
- $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$ and $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$.
- $(\mathsf{hk}_H, \mathsf{td}_H) \leftarrow \mathsf{ExtGen}_H(1^\lambda, N, T)$.
- Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
- $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, T)$.
- Output $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ and $\mathsf{td} = (\mathsf{td}_C, \mathsf{td}_N, \mathsf{td}_H, \mathsf{td}_B)$.

$\mathsf{Ext}_2(\mathsf{td}, x, \pi)$:

- Check if $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$. If not, output $\perp$.
- Else, parse π as $(\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
- Extract $w \leftarrow \mathsf{Extract}_C(\mathsf{td}_C, \mathsf{com})$.

We start by moving into a hybrid where both the extractable commitment and the SEH are moved from the statistially hiding mode into the extractable mode. The proof follows the following sequence of hybrids.

- Hyb_0 : This is the argument of knowledge game described in Definition 2.
- Hyb_1 : This hybrid is identical to the previous one except that we set $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$

  Claim 2. Hyb_0 *and* Hyb_1 *are indistinguishable given that the FLC is dual-mode indistinguishable.*

  The hybrids are identical except for the mode in which the setup of the FLC is set to (either statistically hiding mode or extractable mode). Hence, we could use an adversary that distinguishes the hybrids to break the mode indistinguishability of the underlying FLC.
- Hyb_2 : This hybrid is identical to the previous one except that we set $(\mathsf{hk}_H, \mathsf{td}_H) \leftarrow \mathsf{ExtGen}_H(1^\lambda, N, T)$.

  Claim 3. Hyb_1 *and* Hyb_2 *are indistinguishable given that the SEH is dual-mode indistinguishable.*

  The hybrids are identical except for the mode in which the setup of the SEH is set to (either statistically hiding mode or somewhere extractable mode). Hence, we could use an adversary that distinguishes the hybrids to break the mode indistinguishability of the underlying SEH.

To prove argument of knowledge, it is sufficient to prove the following statement for hybrid Hyb_2:

If output of Ext_2 it not $\perp$, then $\mathcal{R}(x, w) = 1$ except with negligible probability.

Let (w, k) be the witness extracted by the extractor described above. Let us define st_i that is computed as follows.

1. Set $\mathsf{init}_0 = x \| 0^S$.
2. For every g from 1 to i:
   (a) Let i' and j' be the input wires to g-th gate and let ℓ be the output wire.
   (b) Let $\alpha = \mathsf{init}_{g,i'}$ and $\beta = \mathsf{init}_{g,j'}$.
   (c) Compute $\gamma = \mathsf{NAND}(\alpha, \beta)$.
   (d) Let init_g be same as init_{g-1} except that ℓ-th entry is set to γ.

3. Compute $\mathsf{st}_i = \mathsf{MTHash}(\mathsf{hk}, \mathsf{init}_i)$.

Let $(\mathsf{Ext}_{N,1}, \mathsf{Ext}_{N,2})$ be the extractor of the underlying NIZK scheme. Additionally let $(\mathsf{Ext}_{N',1}, \mathsf{Ext}_{N',2})$ be the extractor for the NIZK for language $\mathcal{L}'$. We prove the following claim about the above computed st_i.

Lemma 4. *For any $i \in [N]$ and for any PPT adversary P^*, consider the following experiment.*

1. *Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where*
   - *(a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$*
   - *(b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$ and $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$,*
   - *(c) $(\mathsf{hk}_H, \mathsf{td}_H) \leftarrow \mathsf{ExtGen}_H(1^\lambda, N, i)$ where $N = (T+1) \cdot \lambda$,*
   - *(d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.*
   - *(e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.*
2. *$(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma_0), \pi_B)$*

Then, we have

$$\Pr\left[\mathsf{st}_i \neq \mathsf{st}_i^* \wedge \mathsf{V}(\mathsf{crs}, x, \pi) = 1 : \mathsf{st}_i^* \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)\right] \leq \mathsf{negl}(\lambda).$$

Proof. We will prove this claim via induction on i. We start by analyzing the base case where $i = 0$.

Basis $i = 0$. Note that if $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then it means that st_0 that is sent as part of π satisfies $\mathsf{st}_0 = \mathsf{MTHash}(\mathsf{hk}', x \| 0^S)$ and $\mathsf{Verify}_{N'}(\mathsf{crs}_{N'}, x', \sigma) = 1$. Using $\mathsf{Ext}_{N',2}$ we can extract ρ_0 such that (st_0, ρ_0) is a valid opening of v at location 0 (otherwise we could break the AoK of the NIZK for $\mathcal{L}'$). It now follows from the somewhere binding property of SEH that $\mathsf{st}_0^* = \mathsf{st}_0$ and hence, the claim holds at $i = 0$.

Induction step. We will now show that if the induction hypothesis holds for $i-1$ then it also holds for i. We will prove this via a hybrid argument.

- Hyb_0 : Consider the following experiment:
  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{ExtGen}_H(1^\lambda, N, i-1)$ where $N = (T+1) \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i-1)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. $\mathsf{st}_{i-1}^* \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
  4. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions hold.
     (a) $\mathsf{st}_{i-1} \neq \mathsf{st}_{i-1}^*$

  Note that by induction hypothesis, the probability that we output 1 in Hyb_0 is negligible.
- Hyb_1 : This hybrid is identical to the previous one except that we generate crs_B as $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$. Concretely,
  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$

(c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = (T+1)\cdot\lambda$,
(d) $\underline{(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)}$ where $k = T$.
(e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.

2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
3. $\mathsf{st}^*_{i-1} \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
4. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions hold.
   (a) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$

Claim 4. *Hybrids* Hyb_0 *and* Hyb_1 *are indistinguishable given that the BARG scheme is index hiding.*

We show from the index hiding property of the BARG scheme that the probability that we output 1 in Hyb_1 is negligible. Suppose for the sake of contradiction assume that the probability we output 1 in Hyb_1 is non-negligible. We show that we can break the index hiding property of BARG. We interact with this challenger and send i and $i-1$ as the challenge index locations. We receive crs_B from the challenger. We generate the rest of the components of the CRS as before and send this to the prover. After receiving the proof, we compute st^*_{i-1} and st_i as before. If the proof verifies and $\mathsf{st}^*_{i-1} \neq \mathsf{st}_i$, we output 1. Eles, we output 0. Note that if crs_B is generated with respect to location i, then the reduction mimics Hyb_1. Else, it mimics Hyb_0. Therefore, this reduction can break the index hiding property of the BARG scheme with non-negligible advantage and this is a contradiction.

- Hyb_2 : This hybrid is identical to the previous one, except that we extract $w^*_i \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$, where $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$. Additionally, output 1 if either $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$ or w^*_i is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$. Concretely,

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = T\cdot\lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\underline{\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)}$ and $\underline{w^*_i \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)}$.
  4. $\mathsf{st}^*_{i-1} \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
  5. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
     (a) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$
     (b) $\underline{w^*_i \text{ is not a valid witness for the instance of } \mathcal{L}_N \text{ given by } (\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)}$

Claim 5. *Hybrids* Hyb_1 *and* Hyb_2 *are indistinguishable given that the underlying BARG is somewhere extractable and the underlying NIZK is an argument of knowledge.*

We prove from the somewhere argument of knowledge property of BARG scheme and the knowledge extraction property of NIZK, the probability that we output 1 in Hyb_2 is negligible. We do this in two steps. We consider a sub-hybrid we we first extract π'_i using $\mathsf{Extract}_B$ and output 1 if the proof π verifies and either $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$ or if π'_i is not a valid witness for the index language with the instance being $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$. The probability that we output 1 in this sub-hybrid is negligible from the somewhere argument of knowledge property of the BARG scheme. Now, we use the extractor for the NIZK to extract w^*_i from π'_i and replace the second condition in the previous sub-hybrid to w^*_i being a valid witness for the instance $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$ of $\mathcal{L}_N$. The probability that we output 1 in this hybrid is negligible from the argument of knowledge property of the NIZK scheme. Note that the final hybrid is identical to Hyb_2.

- Hyb_3 : This hybrid is identical to the previous one except that we parse w_i^* as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$. Additionally, we output 1 if either $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$ or w_i^* is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$ or $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$. Concretely,

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w_i^* \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.
  4. Parse w_i^* as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$.
  5. $\mathsf{st}^*_{i-1} \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
  6. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
     (a) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$
     (b) w_i^* is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$
     (c) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.

Claim 6. *Hybrids* Hyb_2 *and* Hyb_3 *are indistinguishable given that the SEH is somewhere extractable.*

We prove using the somewhere binding property of the SEH that the probability that we output 1 in Hyb_3 is negligible. Assume for the sake of contradiction that the probability we output 1 in Hyb_3 is non-negligible. Since the probability we output 1 in Hyb_2 is negligible, we infer that the following event happens with non-negligible probability: (i) the proof π verifies, (ii) $\mathsf{st}_{i-1} = \mathsf{st}^*_{i-1}$, (iii) w_i^* is a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$, and (iv) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$. Since w_i^* is a valid witness, it follows that $(\mathsf{st}'_{i-1}, \rho'_{i-1})$ is a valid opening to v. If $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$, then this breaks the somewhere binding property of SEH and this is a contradiction.

- Hyb_4 : This hybrid is identical to the previous one except that we compute $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$ we output 1 if either i) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$ or ii) w_i^* is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$ or iii) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$ or iv) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$. Concretely,

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w_i^* \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.
  4. Parse w_i^* as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$.
  5. $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$
  6. $\mathsf{st}^*_{i-1} \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
  7. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.

(a) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$
(b) w^*_i is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$
(c) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.
(d) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$.

Claim 7. *Hybrids* Hyb_3 *and* Hyb_4 *are indistinguishable given that the underlying FLC has extraction correctness.*

Assume for the sake of contradiction that we output 1 in Hyb_4 is non-negligible probability. Since the probability we output 1 in Hyb_3 is negligible, the following event happens with non-negligible probability: (i) the proof π verifies, ii) $\mathsf{st}_{i-1} = \mathsf{st}^*_{i-1}$, iii) $\tilde{w}^*_i$ is a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$, iii) $\mathsf{st}'_{i-1} = \mathsf{st}^*_{i-1}$. Then if we output 1 in this hybrid it is because $c^g_i \neq \hat{w}_i$ or $c^g_j \neq \hat{w}_j$, which breaks the extraction correctness of the underlying fully-local extractable commitment.

- Hyb_5 : This hybrid is identical to the previous one except that we output 1 if either i) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$ or ii) $\tilde{w}^*_i$ is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$ or iii) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$ or iv) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$, or v) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$. Concretely,

1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
   (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
   (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
   (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = T \cdot \lambda$,
   (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
   (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w^*_i \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.
4. Parse w^*_i as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$.
5. Let $(b^g_{i'}, \rho^g_{i'})$, $(b^g_{j'}, \rho^g_{j'})$, (b^g_ℓ, ρ^g_ℓ) be the local openings to the i-th, j-th and the ℓ-th locations in st_{i-1}.
6. $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$
7. $\mathsf{st}^*_{i-1} \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
8. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
   (a) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$
   (b) w^*_i is not a valid witness for the instance of $\mathcal{L}_n$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$
   (c) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.
   (d) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$.
   (e) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$.

Claim 8. *Hybrids* Hyb_4 *and* Hyb_5 *are indistinguishable given that* MTHash *is collision-resistant.*

We prove using the collision-resistance of the MTHash that the probability we output 1 in Hyb_5 is negligible. Assume for the sake of contradiction that we output 1 in Hyb_5 is non-negligible probability. Since the probability we output 1 in Hyb_4 is negligible, the following event happens with non-negligible probability: (i) the proof π verifies, (ii) $\mathsf{st}_{i-1} = \mathsf{st}^*_{i-1}$, (iii) w^*_i is a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$, (iv) $\mathsf{st}'_{i-1} = \mathsf{st}^*_{i-1}$, and (v) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$. It follows from conditions (ii) and (iv) that $\mathsf{st}_{i-1} = \mathsf{st}'_{i-1}$. Since w^*_i is a valid witness, it follows that $(c^g_{i'}, \nu^g_{i'}), (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell)$ are valid openings to st_{i-1}. However, if condition (v) happens, then we have found a collision to the MTHash function family and this is a contradiction.

- Hyb_6 : This hybrid is identical to the previous one except that we output 1 if either i) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$ or ii) $\tilde{w}^*_i$ is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$ or iii) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$ or iv) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$, or v) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$, or vi) $\mathsf{st}'_i \neq \mathsf{st}_i$. Concretely,

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w^*_i \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.
  4. Parse w^*_i as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$.
  5. Let $(b^g_{i'}, \rho^g_{i'})$, $(b^g_{j'}, \rho^g_{j'})$, (b^g_ℓ, ρ^g_ℓ) be the local openings to the i-th, j-th and the ℓ-th locations in st_{i-1}.
  6. $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$
  7. $\mathsf{st}^*_{i-1} \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
  8. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
     (a) $\mathsf{st}_{i-1} \neq \mathsf{st}^*_{i-1}$
     (b) w^*_i is not a valid witness for the instance of $\mathcal{L}_n$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$
     (c) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.
     (d) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$.
     (e) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$.
     (f) $\underline{\mathsf{st}'_i \neq \mathsf{st}_i}$.

  Claim 9. *Hybrids* Hyb_5 *and* Hyb_6 *are indistinguishable by the correctness of* $\mathsf{MTWrite}$.

  Since w^*_i is a valid witness and $(b^g_i, b^g_j) = (c^g_i, c^g_j)$, it follows from the perfect extraction of com that the output of the gate i that is computed in w^*_i is same as the output computed as part of init_i. Since init_{i-1} and init_i only differ at the ℓ-th location, it follows from the correctness of $\mathsf{MTWrite}$ that $\mathsf{st}_i = \mathsf{st}'_i$. Hence, the probability that we output 1 in Hyb_6 is negligible.

- Hyb_7 : This hybrid is identical to the previous one except that we output 1 if either i) $\tilde{w}^*_i$ is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$ or ii) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$ or iii) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$, or iv) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$, or v) $\mathsf{st}'_i \neq \mathsf{st}_i$. Concretely,

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i-1)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w^*_i \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.

4. Parse w_i^* as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$.
5. Let $(b^g_{i'}, \rho^g_{i'})$, $(b^g_{j'}, \rho^g_{j'})$, (b^g_ℓ, ρ^g_ℓ) be the local openings to the i-th, j-th and the ℓ-th locations in st_{i-1}.
6. $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$
7. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
   (a) w_i^* is not a valid witness for the instance of $\mathcal{L}_n$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0^*, \mathsf{com}, i, \mathbf{d}^i_j, \mathbf{d}^j_\ell)$
   (b) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.
   (c) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$.
   (d) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$.
   (e) $\mathsf{st}'_i \neq \mathsf{st}_i$.

Claim 10. *Hybrids* Hyb_6 *and* Hyb_7 *are indistinguishable by the correctness of* MTWrite.

Notice that the only difference in Hyb_5 and Hyb_6 is that in Hyb_6, we no longer extract st^*_{i-1} from v and do not check if $\mathsf{st}_{i-1} \stackrel{?}{=} \mathsf{st}^*_{i-1}$. Hence, the probability that we output 1 in Hyb_6 is at most the probability that we output 1 in Hyb_5 and this is negligible.

- Hyb_8 : This hybrid is identical to the previous one except that we set hk to be $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i)$.

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w_i^* \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.
  4. Parse w_i^* as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c^g_{i'}, \nu^g_{i'}), \delta^g_{i'}, \delta^g_{j'}, (c^g_{j'}, \nu^g_{j'}), (c^g_\ell, \nu^g_\ell))$.
  5. Let $(b^g_{i'}, \rho^g_{i'})$, $(b^g_{j'}, \rho^g_{j'})$, (b^g_ℓ, ρ^g_ℓ) be the local openings to the i-th, j-th and the ℓ-th locations in st_{i-1}.
  6. $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$
  7. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
     (a) w_i^* is not a valid witness for the instance of $\mathcal{L}_n$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{st}_0^*, \mathsf{com}, i)$
     (b) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.
     (c) $c^g_i \neq \hat{w}_i$ if $i \in [m]$ or $c^g_j \neq \hat{w}_j$ if $j \in [m]$.
     (d) $(b^g_{i'}, \rho^g_{i'}) \neq (c^g_{i'}, \nu^g_{i'})$ or $(b^g_{j'}, \rho^g_{j'}) \neq (c^g_{j'}, \nu^g_{j'})$ or $(b^g_\ell, \rho^g_\ell) \neq (c^g_\ell, \nu^g_\ell)$.
     (e) $\mathsf{st}'_i \neq \mathsf{st}_i$.

Claim 11. *Hybrids* Hyb_7 *and* Hyb_8 *are indistinguishable given that the underlying SEH is index hiding.*

We show from the index hiding property of SEH that the probability that we output 1 in Hyb_8 is negligible. Assume for the sake of contradiction that the probability that we output 1 in Hyb_8 is negligible. We break the index hiding property of SEH. We interact with the challenger and provide $(i-1)$ and i as the challenge locations and obtain crs_H. We generate the rest of the components in the CRS as before and send to the prover. On obtaining the proof, we make the same checks as in Hyb_7 and output 1 if and only if the output of the experiment described in the hybrids outputs 1. Note that if crs_H is generated with respect to location i, the this reduction mimics Hyb_8. Else, it mimics Hyb_7. Therefore, this reduction contradicts the index hiding property of SEH.

- Hyb_9 : This hybrid is identical to the previous one except that we compute $\mathsf{st}_i^* \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$ and output 1 if either i) $\tilde{w}_i^*$ is not a valid witness for the instance of $\mathcal{L}_N$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0^*, \mathsf{com}, i, \mathbf{d}_j^i, \mathbf{d}_\ell^j)$ or ii) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$ or iii) $c_i^g \neq \hat{w}_i$ if $i \in [m]$ or $c_j^g \neq \hat{w}_j$ if $j \in [m]$, or iv) $(b_{i'}^g, \rho_{i'}^g) \neq (c_{i'}^g, \nu_{i'}^g)$ or $(b_{j'}^g, \rho_{j'}^g) \neq (c_{j'}^g, \nu_{j'}^g)$ or $(b_\ell^g, \rho_\ell^g) \neq (c_\ell^g, \nu_\ell^g)$, or v) $\mathsf{st}'_i \neq \mathsf{st}_i$, or vi) $\mathsf{st}'_i \neq \mathsf{st}_i$. Concretely,

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. Compute $\pi'_i \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w_i^* \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_i)$.
  4. Parse w_i^* as $((\mathsf{st}'_{i-1}, \rho'_{i-1}), (\mathsf{st}'_i \rho'_i), (c_{i'}^g, \nu_{i'}^g), \delta_{i'}^g, \delta_{j'}^g, (c_{j'}^g, \nu_{j'}^g), (c_\ell^g, \nu_\ell^g))$.
  5. Let $(b_{i'}^g, \rho_{i'}^g)$, $(b_{j'}^g, \rho_{j'}^g)$, (b_ℓ^g, ρ_ℓ^g) be the local openings to the i-th, j-th and the ℓ-th locations in st_{i-1}.
  6. $\hat{w} \leftarrow \mathsf{Ext}_C(\mathsf{td}_C, \mathsf{com})$
  7. $\underline{\mathsf{st}_i^* \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)}$.
  8. If $\mathsf{V}(\mathsf{crs}, x, \pi) = 1$, then we output 1 if any of the following conditions holds.
     (a) w_i^* is not a valid witness for the instance of $\mathcal{L}_n$ given by $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{st}_0^*, \mathsf{com}, i)$
     (b) $\mathsf{st}'_{i-1} \neq \mathsf{st}^*_{i-1}$.
     (c) $c_i^g \neq \hat{w}_i$ if $i \in [m]$ or $c_j^g \neq \hat{w}_j$ if $j \in [m]$.
     (d) $(b_{i'}^g, \rho_{i'}^g) \neq (c_{i'}^g, \nu_{i'}^g)$ or $(b_{j'}^g, \rho_{j'}^g) \neq (c_{j'}^g, \nu_{j'}^g)$ or $(b_\ell^g, \rho_\ell^g) \neq (c_\ell^g, \nu_\ell^g)$.
     (e) $\mathsf{st}'_i \neq \mathsf{st}_i$.
     (f) $\underline{\mathsf{st}_i^* \neq \mathsf{st}'_i}$.

Claim 12. *Hybrids* Hyb_8 *and* Hyb_9 *are indistinguishable given that the underlying SEH is is somewhere extractable.*

Observe that since w_i^* is a valid witness, it follows from the somewhere binding property of SEH that the probability that we output 1 in Hyb_9 is negligible. This argument is similar to Claim 6.

- $\mathsf{Hyb}_1 0$: This hybrid is identical to the previous one except that we output 1 if $\mathsf{st}_i^* \neq \mathsf{st}_i$.

  1. Generate $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{crs}_N, \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{hk}, \mathsf{hk}')$ where
     (a) $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{ExtSetup}_C(1^\lambda)$
     (b) $(\mathsf{crs}_N, \mathsf{td}_N) \leftarrow \mathsf{Ext}_{N,1}(1^\lambda)$, $(\mathsf{crs}_{N'}, \mathsf{td}_{N'}) \leftarrow \mathsf{Ext}_{N',1}(1^\lambda)$
     (c) $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N, i)$ where $N = T \cdot \lambda$,
     (d) $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, i)$ where $k = T$.
     (e) Sample $\mathsf{hk}' \leftarrow \mathsf{MTGen}(1^\lambda)$.
  2. $(x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, where $\pi = (\mathsf{com}, v, (\mathsf{st}_0, \sigma), \pi_B)$.
  3. $\mathsf{st}_i^* \leftarrow \mathsf{Extract}_H(\mathsf{td}_H, v)$.
  4. $\underline{\text{If } \mathsf{V}(\mathsf{crs}, x, \pi) = 1, \text{ then we output 1 if any of the following conditions holds.}}$
     (a) $\underline{\mathsf{st}_i^* \neq \mathsf{st}_i}$.

Claim 13. *Hybrids* Hyb_9 *and* Hyb_{10} *are indistinguishable.*

Observe that if $\mathsf{st}'_i = \mathsf{st}_i$ and $\mathsf{st}'_i = \mathsf{st}^*_i$ with overwhelming probability in Hyb_9, it follows that $\mathsf{st}_i = \mathsf{st}^*_i$ with overwhelming probability. As we are making fewer checks in $\mathsf{Hyb}_1 0$, the probability that we output 1 in $\mathsf{Hyb}_1 0$ is at most the probability that we output 1 in Hyb_9. Hence, the probability that we output 1 in $\mathsf{Hyb}_1 0$ is negligible. This completes the induction step.

This completes the proof of the Lemma. □

It remains to show that output of the final gate (i.e., the T-th gate) computed in init_T is 1 except with negligible probability if the proof π is accepting. Suppose this is not the case. Let st_T be the digest of init_T. Let $(0, \rho_T)$ be the opening to the T-th location in st_T. By applying Lemma 4 at location T, we have that st^*_T (extracted from v) is same as st_T except with negligible probability. Let $\pi'_T \leftarrow \mathsf{Extract}_B(\mathsf{td}_B, \pi_B)$ and $w^*_T \leftarrow \mathsf{Ext}_{N,2}(\mathsf{td}_N, \pi'_T)$ (where π_B is part of π).

It follows from somewhere soundness of the BARG and the knowledge extraction of NIZK that w^*_T is a valid witness to the language $\mathcal{L}_N$ with the instance being $(\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, T, \mathbf{d}^T_i, \mathbf{d}^T_j)$. Parse this witness as

$$(\mathsf{st}'_{T-1}, \rho'_{T-1}), (\mathsf{st}'_T \rho'_T), (c^g_{i'}, \nu^g_{i'}), (\mathbf{e}^g_{i'}, \sigma^g_{i'}), (c^g_{j'}, \nu^g_{j'}), (\mathbf{e}^g_{j'}, \sigma^g_{j'}), (c^g_\ell, \nu^g_\ell)$$

where (i', j') are the input wires to the T-th gate and ℓ is the output wire. It follows from somewhere binding property of SEH that $\mathsf{st}'_T = \mathsf{st}^*_T$ except with negligible probability. Since w^*_T is a valid witness, it follows that there is a valid opening $(1, \rho'_T)$ to the T-th location in st^*_T. However, this contradicts the collision resistance of the MTHash.

This completes the proof of the theorem. □

Finally, we show that the scheme has statistical zero-knowledge.

Theorem 7 (Zero-knowledge)**.** *The NIZK construction presented above is statistical zero-knowledge given that the underlying NIZKs have statistical zero-knowledge and the fully local extractable commitment is statistically hiding.*

Proof. We will first describe the simulator Sim. Let Sim_N be the simulator for the underlying NIZK scheme and $\mathsf{Sim}_{N'}$ be the simulator for the NIZK for language $\mathcal{L}'$.

$\mathsf{Sim}(1^\lambda, (C, x))$:

- Sample $(\mathsf{crs}_C, \mathsf{td}_C) \leftarrow \mathsf{Setup}_C(1^\lambda)$, $\mathsf{crs}_{N'} \leftarrow \mathsf{Setup}_{N'}(1^\lambda)$, $(\mathsf{hk}, \mathsf{td}_H) \leftarrow \mathsf{Gen}_H(1^\lambda, N)$, $\mathsf{hk}' \leftarrow \mathsf{MDGen}(1^\lambda)$, and $(\mathsf{crs}_B, \mathsf{td}_B) \leftarrow \mathsf{Gen}_B(1^\lambda, 1^k, 1^s, 1)$.
- Compute $(\mathsf{com}, \rho) \leftarrow \mathsf{Com}_C(\mathsf{crs}_C, 0^m)$.
- Set $\mathsf{init}_0 = x || 0^S$. Let $\mathsf{st}_0 = \mathsf{MDHash}(\mathsf{hk}', \mathsf{init}_0)$.
- For each $g \in [T]$,
  - Set init_g exactly as init_{g-1} except that ℓ-th coordinate is set to $d \leftarrow\$ \{0, 1\}$.
  - Compute $\mathsf{st}_g = \mathsf{MDHash}(\mathsf{hk}, \mathsf{init}_g)$.
- Compute $v = \mathsf{Hash}(\mathsf{hk}, \mathsf{st}_0 \| \ldots \| \mathsf{st}_T)$.
- Use Sim_N to compute $(\mathsf{crs}_N, \pi_1, \ldots, \pi_T) \leftarrow \mathsf{Sim}_N(1^\lambda, , X_1, \ldots, X_T)$ where each $X_g = (\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}^*_0, \mathsf{com}, g)$ for each $g \in [T]$.
- Compute π_B as in the real scheme and set $\pi = (w', v, (\mathsf{st}_0, \sigma), \pi_B)$ where st_0 and σ are set as in the real scheme. Set $\mathsf{crs} = (\mathsf{crs}_C, \mathsf{hk}, \mathsf{hk}', \mathsf{crs}_B, \mathsf{crs}_N, \mathsf{crs}_{N'})$.
- Output (crs, π).

We now show that for any (C, x) and $w \in \{0,1\}^m$ such that $C(w, x) = 1$, we have:

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

The proof follows from the following sequence of hybrids.

- Hyb_0 : This is the distribution of the honestly generated (crs, π).
- Hyb_1 : This hybrid is identical to the previous one except that we simulate the NIZK proofs $\pi_1, \ldots, \pi_T$ and crs_N using Sim_N.

  Claim 14. *Hybrids* Hyb_0 *and* Hyb_1 *are statistically indistinguishable given that the underlying NIZK for language* $\mathcal{L}_N$ *is statistical zero-knowledge.*

  Assume for the sake of contradiction that Hyb_0 and Hyb_1 are distinguishable with non-negligible advantage. We give a reduction that breaks the zero-knowledge property of the NIZK. The reduction generates $\mathsf{hk}, \mathsf{hk}', \mathsf{crs}_{N'}, \mathsf{crs}_B, \mathsf{crs}_C$ as in the previous hybrid. It then computes v, com as in Hyb_0. For each $g \in [T]$, it generates the witness w_g for the instance $X_g = (\mathsf{crs}_C, v, \mathsf{hk}, \mathsf{hk}', \mathsf{st}_0^*, \mathsf{com}, g, \mathbf{d}_i^g, \mathbf{d}_j^g)$ as in Hyb_0. It gives $(X_1, w_1), \ldots, (X_T, w_T)$ to the challenger. The challenger responds with $(\mathsf{crs}_N, \pi_1, \ldots, \pi_N)$. It uses this to compute crs and π as before. Observe that if the challenger generated the CRS and the proofs using the honest prover algorithm, then the output of the above reduction is identical to Hyb_0. Else, it is distributed identically to Hyb_1. Therefore, this reduction breaks the zero-knowledge property of the NIZK scheme and this is a contradiction.

- Hyb_2 : This hybrid is identical to the previous one except that we simulate $(\mathsf{crs}_{N'}, \sigma) \leftarrow \mathsf{Sim}_{N'}(1^\lambda, x')$ (instead of computing σ honestly).

  Claim 15. *Hybrids* Hyb_1 *and* Hyb_2 *are statistically indistinguishable given that the underlying NIZK for language* $\mathcal{L}'$ *is statistical zero-knowledge.*

  The proof of this claim is identical to the proof of Claim 14.

- Hyb_3 : This hybrid is identical to the previous one except that we set com to be $\mathsf{Com}_C(\mathsf{crs}_C, 0^m)$.

  Claim 16. *Hybrids* Hyb_2 *and* Hyb_3 *are statistically indistinguishable given that the underlying FLC is statistically hiding.*

  If Hyb_2 is distinguishable from Hyb_3 with non-negligible advantage, we show that this contradicts the statistical hiding of the commitment scheme. The reduction receives crs_C and the challenge com (either encrypting w or 0^m). Then it behaves exactly as in the previous hybrid to create a proof π and crs. If com is generated as a commitment to w, then the output of the reduction is identical to Hyb_2. Else, it is distributed identically to Hyb_3. Therefore, the reduction breaks the hiding property of the commitment scheme with non-negligible advantage and this is a contradiction.

- Hyb_4 : This hybrid is identical to the previous one except that for each $g \in [T]$, we set init_g to be same as init_{g-1} except that the ℓ-th coordinate is set to a uniformly chosen random bit.

  Claim 17. *Hybrids* Hyb_3 *and* Hyb_4 *are statistically indistinguishable given that the underlying dual-mode SEH is statistically hiding.*

  If Hyb_3 is distinguishable from Hyb_4 with non-negligible advantage, we show that this contradicts the statistical hiding of the dual-mode SEH scheme. The reduction receives hk and the challenge v (either hashing the honestly generated init_g or a uniformly generated init'_g). Then it behaves exactly as in the previous hybrid to create a proof π and crs. If v is generated as a hashing of init_g, then the output of the reduction is identical to Hyb_3. Else, it is distributed identically to Hyb_4.

- Hyb_5 : This hybrid is identical to the previous one except that we compute σ honestly. That is, we compute $\mathsf{crs}_{N'} \leftarrow \mathsf{Setup}_{N'}(1^\lambda)$ and σ as as in the real scheme.

  Claim 18. *Hybrids* Hyb_4 *and* Hyb_5 *are statistically indistinguishable given that the underlying NIZK for language* $\mathcal{L}'$ *has statistical zero-knowledge.*

  The proof of this claim is identical to the proof of Claim 14.

Note that the last hybrid corresponds to the simulation described above, and this concludes the proof of the theorem. □

References

[ADD+22] Divesh Aggarwal, Nico Döttling, Jesko Dujmovic, Mohammad Hajiabadi, Giulio Malavolta, and Maciej Obremski. Algebraic Restriction Codes and Their Applications. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 455–472, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 157–186, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BD24] Shany Ben-David. Probabilistically checkable arguments for all np. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 345–374, Cham, 2024. Springer Nature Switzerland.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[BDS23] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. A framework for statistically sender private OT with optimal rate. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 548–576, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.

[BDSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BDSZ24] Pedro Branco, Nico Döttling, Akshayaram Srinivasan, and Riccardo Zanotto. Rate-1 fully local somewhere extractable hashing from DDH. To appear in PKC, 2024.

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

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

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 474–482, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 738–767, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[BKP+23] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. Cryptology ePrint Archive, Paper 2023/754, 2023. `https://eprint.iacr.org/2023/754`.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[BWW23] Eli Bradley, Brent Waters, and David J. Wu. Batch arguments to NIZKs from one-way functions. Cryptology ePrint Archive, Paper 2023/1938, 2023. `https://eprint.iacr.org/2023/1938`.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090, Phoenix, AZ, USA, June 23–26, 2019. ACM Press.

[CGJ+23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 635–668, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.

[CGKS23] Matteo Campanelli, Chaya Ganesh, Hamidreza Khoshakhlagh, and Janno Siim. Impossibilities in succinct arguments: Black-box extraction and more. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023*, pages 465–489, Cham, 2023. Springer Nature Switzerland.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 394–423, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

[CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd FOCS*, pages 68–79, Denver, CO, USA, February 7–10, 2022. IEEE Computer Society Press.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press.

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

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd FOCS*, pages 1057–1068, Denver, CO, USA, October 31 – November 3, 2022. IEEE Computer Society Press.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany.

[DJJ24] Quang Dao, Aayush Jain, and Zhengzhong Jin. Non-interactive zero-knowledge from LPN and MQ. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 321–360. Springer, 2024.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

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

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

[GGI+15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, 28(4):820–843, October 2015.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

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

[GOS18] Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 515–544, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[GS18a] Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[GS18b] Sanjam Garg and Akshayaram Srinivasan. A simple construction of iO for Turing machines. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 425–454, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[GVW02] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1-2):1–53, 2002.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. SNARGs for P from sub-exponential DDH and QR. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 520–549, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 3–32, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany.

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

[KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and RAM delegation. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1545–1552. ACM, 2023.

[KNYY19] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 622–651, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Compact NIZKs from standard assumptions on bilinear maps. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 379–409, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1115–1124, Phoenix, AZ, USA, June 23–26, 2019. ACM Press.

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

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[PP22] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *63rd FOCS*, pages 1045–1056, Denver, CO, USA, October 31 – November 3, 2022. IEEE Computer Society Press.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[Wat24] Brent Waters. A new approach for non-interactive zero-knowledge from learning with errors. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 399–410. ACM, 2024.

[WW22] Brent Waters and David J. Wu. Batch arguments for sfNP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 433–463, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.