Electronic Colloquium on Computational Complexity, Report No. 99 (2017)

Multi-Collision Resistance: A Paradigm for Keyless Hash Functions

Nir Bitansky* Yael Tauman Kalai† Omer Paneth‡

June 5, 2017

Abstract

We study *multi-collision-resistant hash functions* — a natural relaxation of collision-resistant hashing that only guarantees the intractability of finding many (rather than two) inputs that map to the same image. An appealing feature of such hash functions is that unlike their collision-resistant counterparts, *they do not necessarily require a key*. In the unkeyed setting we only require that the size of collisions an adversarial algorithm can find is not much larger than its description size, or non-uniform advice.

We show how to replace collision resistance with multi-collision resistance in several foundational applications. Relying on such unkeyed function, we improve on the best known round complexity for these applications. This includes:

- 3-message zero-knowledge arguments for **NP**.
- 3-message succinct arguments of knowledge for **NP**.
- 4-message ε-zero-knowledge proofs for **NP**.
- 5-message public-coin zero-knowledge arguments for **NP**.

These results are obtained in the standard model of non-uniform adversaries of arbitrary polynomial size. Our techniques can also be applied in the keyed setting, at the cost of adding another message. In this case, we weaken the known complexity assumptions for the last three applications, while still matching the state of the art in terms of round complexity.

The core technical contribution behind our results is a domain extension transformation from multi-collision-resistant hash functions for a fixed input length to ones with an arbitrary input length and a local opening property.

*MIT, email nirbitan@csail.mit.edu. Supported by NSF Grants CNS-1350619 and CNS-1414119, and the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

†Microsoft Research, email yael@microsoft.com.

‡MIT, email omerpa@mit.edu Supported by NSF Grants CNS-1350619 and CNS-1414119, and the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

ISSN 1433-8092

Contents

1 Introduction

Collision-resistant hash functions are central to cryptography. They are used everywhere to compress communication and storage, from simple applications such as *hash-and-sign*, through more advanced applications like reliable delegation of data and computation, and all the way to core foundational concepts such as, succinct proofs, non-black-box proofs of security, or hardness results in complexity theory [Mer89, Dam89, DPP93, Kil94, Bar01, MP91, KNY17].

In this work, we study a natural relaxation of collision resistance that we call *multi-collision resistance*. A hash function is multi-collision-resistant if finding *many* inputs that hash to the same output is intractable. We formalize the notion of multi-collision resistance, explore its applications and develop techniques for robust composition of such hash functions.

The main motivation for our study is the discrepancy between the theoretical modeling of hash functions as keyed families, and the practical applications of hashing that are based on unkeyed functions. This problem was termed by Rogaway [Rog06] *the foundations-of-hashing dilemma*. Next, we explain the dilemma and how it motivates the study of multi-collision resistance.

The Dilemma: to Key or Not to Key? Cryptographic hash functions such as MD5 or SHA2 have been studied and widely used in practice since the 80's. Such functions, however, cannot satisfy the formal definition of collision-resistance. Indeed, for any single shrinking function, there exist algorithms that can efficiently find collisions, by simply having such collisions hardwired in their code.

To resolve this problem, in the theoretical treatment of collision-resistance we consider *keyed families of hash functions*, requiring that efficient algorithms cannot find collisions when the key is chosen at random. This modeling however, comes at the price of a *trust assumption*: we have to ensure that the key is indeed chosen at random by a trusted entity, otherwise, collisions may be easy to find. While in some applications such trust is implicitly assumed, in others, it comes at a cost, for example, extra rounds of communication in which parties choose and announce a hash key.

Different approaches to circumvent this dilemma have been suggested. One approach is to consider unkeyed hash functions and compromise on weaker security. For example, restrict attention to *uniform* adversaries whose description size is shorter than the hash's input size. In practice, however, for any reasonable choice of hash function, the adversary's description may very well be larger than the input size. Rogaway suggests an approach that sidesteps the dilemma and unifies the treatment of keyed and unkeyed hashing — focus on establishing an *explicit reduction* from breaking the security of a cryptographic scheme to finding collisions in the hash function. Even in the unkeyd setting, such reductions provide confidence in the scheme's security rooted at the fact that some popular cryptographic hash functions have, so far, resisted humanity's efforts to find collisions.

This Work: Unkeyed Hash Functions Against Arbitrary Adversaries. We suggest a new paradigm for the treatment of hash functions that does not necessitate keys and considers adversaries with arbitrary (polynomial) non-uniform advice. The paradigm is centered around the notion of multi-collision resistance.

A shrinking hash

$$\left\{\mathsf{H} : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\lambda}\right\}_{\lambda\in\mathbb{N}} ,$$

where λ is a security parameter and $\ell(\lambda) \gg \lambda$, is K-collision resistant if a polynomial-time adversary cannot output K distinct inputs:

$$X_1, \ldots, X_K \qquad \textit{such that} \qquad \mathsf{H}(X_1) = \ldots = \mathsf{H}(X_K) \ .$$

Note that a collisions of size $2^{\ell(\lambda)-\lambda}$ always exists. While an unkeyed hash function can never be collision

resistant, it may satisfy multi-collision-resistance as long as we allow the number of collisions K that the adversary can find to scale with the adversary's description, including the size of its non-uniform advice. Crucially, K should not depend on the adversary's running time, which could be larger than K.

A bit more formally, for a function $K(\cdot)$, we will say that a (possibly unkeyed) hash function H is K-collision resistant, if for any two polynomials $\zeta(\cdot), T(\cdot)$, any adversary with non-uniform advice of size $\zeta(\lambda)$ and running-time $T(\lambda)$ can find at most $K(\zeta(\lambda))$ inputs in $\{0,1\}^{\ell(\lambda)}$ that map to the same image. That is, the same collision bound K holds regardless of the (polynomial) running time T, and depending only on the size ζ of non-uniform advice.

In the next section, we present our main results on the applications of multi-collision-resistant hash functions. We then discuss the notion of multi-collision resistance in further detail, including possible candidates and complexity-theoretic aspects. Finally, we give a technical overview of the main ideas behind our results.

1.1 Results

We show how to replace collision-resistance with multi-collision resistance in several foundational applications. In several cases, relying on such unkeyed functions, enables us to reduce the round complexity of several protocols.

We construct the following protocols:

- 3-message zero-knowledge arguments for **NP**. These require multi-collision-resistant hash functions with slightly-super (e.g. qausi) polynomial hardness, and additional standard assumptions (e.g., LWE).
- 3-message succinct arguments of knowledge for **NP**.
- 4-message ε-zero-knowledge proofs for **NP**.
- 5-message public-coin zero-knowledge arguments for **NP**. These require multi-collision-resistant hash functions with slightly-super-polynomial hardness.

We stress that the above results are obtained in the standard model of non-uniform adversaries of arbitrary polynomial size.

We next elaborate on each of these applications.

3-Message Zero-Knowledge Arguments.[1] While 4-message zero-knowledge arguments for NP are known based on one-way functions [FS89, BJY97], the existence of 3-message zero-knowledge (with negligible soundness error) has been a long standing open problem. It is the optimal round complexity a zero-knowledge argument could have, since 2-message zero-knowledge is impossible (for non-trivial languages) [GO94].

The difficulty and importance of this question stem from the fact that proving security (specifically, simulation) of 3-message protocols necessarily requires using the code of the adversary in a non-black-box way [GK96a]. While non-black-box techniques have been known since the work of Barak [Bar01], so far they have fallen short of solving this problem. In known constructions security is either heuristic, or based on auxiliary-input knowledge assumptions [HT98, BP04, CD09, BCC$^+$14], which are subject to implausibility [BCPR14], and also result in a non-explicit simulator. Alternatively, 3-message protocols have been shown in restricted adversarial models where either the verifier or the prover is assumed to be uniform [BCPR14, BBK$^+$16], or in models where the simulator can run in super-polynomial time [Pas03].

[1] We recall the distinction between *arguments* that are only computationally sound and *proofs* which are statistically sound.

Our protocol is in the plain model, secure against non-uniform verifiers and provers of arbitrary polynomial size, and has an explicit polynomial-time simulator. The simulator is (inherently) non-black-box in the code of the verifier. This is the first such protocol based on any simple plausible assumption.

3-Message Succinct Arguments. Succinct arguments are proof systems in which the verifier can certify the correctness of a long **NP** computation, independently of the computation time. Such arguments have been the subject of a long line of research and are strongly motivated by the problem of delegating computation. Kilian gave a 4-message succinct argument for NP based on collision-resistant hashing [Kil92] (later extended to a *universal argument* in [BG08]). So far, this round complexity was only improved under strong knowledge assumptions [DCL08, BCC$^+$14, BCCT13], with a non-explicit soundness reduction, or in the random oracle model [Mic00]. We give the first 3-message protocol for **NP**, which is not based on knowledge assumptions, and with an explicit soundness reduction (or witness extraction).

4-Message ε-Zero-Knowledge Proofs. When considering zero-knowledge proofs (rather than arguments), the state of the art in terms of round complexity is five messages [GK96a]. Here, similarly to the case of 3-message arguments, Katz showed that 4-message zero-knowledge proofs cannot have a black-box simulator, except for languages in $\mathbf{NP} \cap \mathbf{coAM}$ [Kat12]. The lower bound holds also for the slightly relaxed notion of ε-zero-knowledge where the simulator's running time may grow polynomially with the simulation accuracy (i.e., distinguishing gap). We give the first 4-message ε-zero-knowledge proof. Interestingly the non-black-box component in our proof of security is minimal, the simulator only utilizes the size of the adversary's code, but otherwise uses it as a black-box. As a building block, which may be of independent interest, we construct non-interactive statistically-hiding commitments that are *weakly binding* — analogously to multi-collision resistance, the adversary can only open to a few values proportional to the size of its non-uniform advice.

5-Message Public-Coin Zero-Knowledge Arguments. Constant-round public-coin zero-knowledge arguments are also subject to black-box lower bounds [GK96b]. Barak gave a 7-message protocol [Bar01], which was later improved to 6 messages [OV12]. We construct a 5-message public-coin zero-knowledge protocol. A downside of our 5-message protocol is that it requires slightly-super-polynomial hardness, whereas existing protocols can be based on polynomially-hard collision-resistance [BG08].

Reducing Assumptions in the Keyed Setting. Our results also have implications in the keyed setting. In the keyed setting, the applications of 3-message succinct arguments and 4-message ε-zero-knowledge proofs turn into 4-message, and 5-message, respectively. While this is identical to the state of the art in terms of round complexity, we improve on the assumption aspect — we replace (keyed) collision-resistant hash functions with the relaxed notion of (keyed) multi-collision resistance. Indeed, unlike unkeyed multi-collision-resistance, which may be incomparable to keyed collision-resistance, in the keyed setting, multi-collision resistance follows from collision-resistance, and a reverse implication is not known. Furthermore, (keyed) multi-collision-resistant hash functions have recently been constructed from complexity assumptions that are not known to imply collision-resistance, such as average-case hardness of the Ramsey problem [KNY17].

The Core Technical Result. The core technical contribution behind our results is a transformation from multi-collision-resistant hash functions for a fixed input length to ones with an arbitrary input length and a *local opening* property. For collision-resistant hash functions such domain extension and local opening properties are quite basic and have been known since the dawn of hash functions [Mer89, Dam89]. In contrast, developing analogous techniques for multi-collision resistance turns out to be quite challenging.

In a nutshell, while collision-resistant hash functions are *robust under composition*, this is not the case for multi-collision-resistant hash functions. For instance, in the canonical *hash tree* construction [Mer89], collision-resistance is easily argued by observing that a collision between two long inputs immediately gives rise to a collision in the underlying (fixed length) hash function. This construction also allows to locally

open any specific input bit and certify its uniqueness by providing only a small (logarithmic) number of hash values.

This simple construction completely fails when considering multi-collision resistance. Indeed, when instantiating the tree construction with a K-collision resistant function, collisions multiply — the number of potential openings for any specific leaf could be $(K-1)^d$, where d is the depth of the tree (and the number of repeated applications of the hash function). Furthermore, it is possible to "mix-and-match" the values of different leafs. For $K = 2$, corresponding to actual collision-resistance, this is not so bad, due to the miraculous fact that $1 \times 1 = 1$. However, for $K > 2$, this is devastating, as the overall number of openings for an n-leaf tree may be $(K-1)^{d \cdot n}$, which is exponential in the input length!

Nevertheless, we show how to convert any multi-collision-resistant hash function into one with arbitrarily large domain and with local opening. Our transformation is based on a combination of basic classical tree hashing techniques together with a new (information-theoretic) construction of certain *collision-free codes*. We refer the reader to the technical overview for further details.

1.2 Multi-Collision-Resistance: Discussion and Open Questions

We further discuss the notion of multi-collision resistance, considering both the case of keyed and unkeyed functions. In particular, while this work focuses on the power of multi-collision resistance, the notion raises several interesting questions and research directions that we highlight here.

The Collision Bound K. The strength of the notion is naturally related to the collision bound K, with the invariant that the larger K is the weaker the notion becomes. In the keyed setting, any K may be possible; in particular, unlike the unkeyed setting, the collision bound K does not have to grow with the adversary's non-uniform advice, it could be a fixed polynomial in the security parameter, or even a constant (where $K = 1$ corresponds to full-fledged collision-resistance).

We also note that while it is natural to think of the collision bound K as polynomial in the security parameter (or the size of the non-uniform advice), one may want to consider a setting where the size of hard-to-find collisions is super-polynomial $K = \lambda^{-\omega(1)}$. There are several possible ways of formalizing such a requirement. One is to allow the adversary also run super-polynomial time $K^{O(1)}$.

A somewhat more appealing way of formalizing such a requirement would be following the *inaccessible entropy* approach of [HRVW09]. Here we would require that a polynomial-time sampler cannot output preimages for any given image Y with too much entropy. In other words, for every hash output Y there exists a set S_Y of K values, so that the attacker can only output preimages of Y from the set S_Y, except with negligible probability. Such a definition coincides with the previous definition for polynomial collision-size K, and also allows to treat super-polynomial values of K. At this point, we do not know how to leverage such a definition for super-polynomial K in our applications.

Relations to Existing Primitives. Multi-collision-resistant hash functions (keyed or unkeyed) imply the existence one-way functions. In fact any multi-collision-resistant hash is already a *distributional one-way function*. That is, for a random image Y, it is computationally hard to sample (statistically-close-to) uniform preimages of Y.[2] Such distributional one-way functions are known to imply (plain) one-way functions [IL89].

As expected, keyed multi-collision-resistant hash functions follow from any assumption that implies collision-resistance. However, we do not know of any standard cryptographic primitive that implies unkeyed

[2]To see why this is the case, note that any such sampler could be used to find a multi-collision of arbitrary polynomial size, in particular much larger than the polynomial advice required by the sampler. This can be done just by sampling enough preimages for some random image Y.

multi-collision resistance. Indeed, as we explain next, unkeyed multi-collision-resistance seems to have a different complexity-theoretic nature than that of typical cryptographic primitives. In the converse direction, we do not know if unkeyed multi-collision resistance implies keyed collision-resistance or any cryptographic primitive other than one-way functions.

Unkeyed Hashing and Universal Hardness. The notion of hardness encapsulated in unkeyed multi-collision-resistance does not seem to be captured by our standard treatment of hard search problems. Indeed, we usually think of hard search problems as *instance based*. Worst-case hardness says that any algorithm fails to find a solution at least for some instances, whereas cryptography requires a stronger notion of average-case hardness, where all algorithms fail to find solutions for instances from some distribution.

In contrast, in unkeyed multi-collision-resistance *there are no instances* or if you'd like, the problem given by a hash function H has a *a single universal* sequence of instances $\left\{1^\lambda : \lambda \in \mathbb{N}\right\}$. The solutions for the instance 1^λ are all the multi-collisions

$$\left\{X_1, \ldots, X_k \in \{0,1\}^{\ell(\lambda)} : k \geq 2, X_i \neq X_j, \mathsf{H}(X_1) = \cdots = \mathsf{H}(X_i)\right\} \quad .$$

We can test that a given tuple is a solution in polynomial-time in the tuple size, but the length of solutions is not bounded by any fixed polynomial in the input size λ. We require more than worst/average-case hardness — *the same instance* 1^λ *is hard (in some sense) for all algorithms*. Unlike average-case hardness, we cannot expect that non-uniform algorithms with arbitrary polynomial advice completely fail to solve the problem. All that we can expect is that they fail to find *solutions that are much longer than their non-uniformity*.

Another possibly useful point of view on such universal problems is as *compression problems*. We are interested in problems where it is impossible to compress long solutions into short advice, so that they can be later efficiently decompressed.

We find that such universal problems may be of independent interest from a complexity-theoretic perspective. Understanding their implications and feasibility calls for further research.

Candidates. Keyed multi-collision-resistance directly follows from any hardness assumption that implies collision-resistance, and thus from a large set of algebraic problems such as hardness of discrete logs, factoring, or finding short vectors in lattices. Interestingly, concurrent and independent work by Komargodski, Naor, and Yogev gave a construction of keyed multi-collision-resistance from average-case hardness of a completely combinatorial problem in **TFNP**— the Ramsey problem [KNY17].

Turning to the unkeyed setting, investigating candidates for multi-collision-resistance is a fascinating question for future research. While it is not the focus of this work, we do point out to some limitations, plausible candidates, and possible approaches toward new candidates.

A first tempting thought toward obtaining unkeyed multi-collision resistance would be to take the keyed collision-resistant hash functions that we know and love, and derandomize the choice of key in some way (for instance, by taking the first λ digits of Pi as the key). This approach miserably fails when considering the typical algebraic constructions of collision-resistant hashing (e.g. based on discrete-logs or factoring). Indeed, in all of these hash functions *there exists a single fixed-length trapdoor* that enables to find many collisions, or even sample random ones. While this trapdoor is hard to find, given a random key, when fixing the key, this trapdoor can always be taken as non-uniform advice. Indeed, these hash functions are typically based on random-self-reducible problems with a *unique solution* (e.g. finding a discrete log), and are of an all or nothing nature — when finding this unique solution is hard, finding any collision is hard, whereas once it is known, all collisions can be found. Avoiding such unique-solution problems seems to be essential.

Diverging from algebraically structured hash functions, the most natural candidates for unkeyed multi-collision resistance are asymptotic versions of existing cryptographic hash functions, such as the SHA

family. We note that even in "old-generation" versions, such as SHA1, where specific collisions have been found, multi-collisions are not known. Unlike collision-resistance, which such functions could not possibly satisfy, multi-collision resistance is a simple assumption that they could and are, in fact, expected to satisfy (breaking it may deem these functions useless). As a sanity check, we show that random oracles satisfy multi-collision resistance in the model of *random oracles with auxiliary inputs* of Unruth [Unr07]. In this model, the adversary first obtains a full description of the random oracle and can store arbitrary (short) auxiliary information z about the random oracle. Then at the second stage, using only the stored z and black-box access to the oracle, it attempts to output a multi-collision. We show that (unbounded) adversaries with polynomially (or even exponentially) many queries cannot output collisions that are significantly larger than the size of the auxiliary input. See Appendix A for details.

Yet another place to look for candidates that may avoid trapdoors are combinatorial constructions of hash functions. In this regime, a class of candidates is given by Goldreich's one-way functions with small output locality. Here the prototype of a function is simply given by a bipartite expander specifying the dependence of each output bit on certain input bits (the neighbors), and a suitable predicate. While there has been much research on this in the context of pseudorandomness (see survey by [App16]), our knowledge on possible collision-resistance properties of such functions seems quite limited at this point [AM13]. Another possible combinatorial route toward finding a candidate is through the Ramsey-based construction of (keyed) multi-collision resistance [KNY17], by identifying candidates for universally hard Ramsey problems.

Stronger Notions of Multi-Collision Resistance. We conclude the discussion by pointing out a stronger notion of keyless hash functions that we simply call *strong multi-collision resistance*. According to this notion, it is not only hard to find K elements that are hashed to the same value, but rather it is hard to find arbitrary K distinct collisions $(X_1, X_1') \ldots (X_K, X_K')$ that may hash to different values $Y_1, \ldots, Y_K$.

Indeed, this notion clearly implies the previous notion of multi-collision resistance and seems stronger. In particular, it is not hard to see that such strong multi-collision resistance (keyed or unkeyed) implies (keyed) hash functions that are collision resistant in the standard sense. To see this, note that given such a hash function, say, $\mathsf{H} : \{0,1\}^{3\lambda} \to \{0,1\}^{\lambda}$, we can consider a new keyed family $\mathsf{H}'_{X_1} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ defined by $\mathsf{H}'_{X_1}(X_2) = \mathsf{H}(X_1 X_2)$, which must be fully collision resistant.

While stronger, we find that this notion is still quite natural and adheres to what we would expect from existing keyless hash functions (such as SHA). In addition, using this notion, it is possible to improve the parameters of some of our constructions.

1.3 Technical Overview

In this section, we overview the main ideas behind our results based on the existence of multi-collision resistant hash functions. To build intuition, we start with a simple application to commitment schemes. We then move to discuss the main technical challenge centered around the problems of domain extension and local opening. We describe our solution to this problem and how it is applied to obtain our results on round-efficient protocols.

Non-Interactive Commitments with Weak Binding. Commitments are a basic building block in cryptographic protocols such as zero-knowledge proofs. They consist of a commitment phase, where a sender commits to a value, and an opening phase, where the commitment is opened and the value is revealed. The commitment should be *hiding*: the receiver does not learn anything about the committed value before the opening phase, and *binding*: the sender cannot open the commitment to more than a single value.

Collision-resistant hashing has been essential in achieving two useful properties for commitment schemes. First, they are used to obtain *shrinking commitments*, where the communication in the commit phase is

shorter than the committed value. Second, they are used to construct constant-round *statistically-hiding commitments* where hiding is guaranteed against unbounded receivers [DPP93, HM96]. Since collision-resistant hash functions must be keyed, the corresponding commitments have a two-message commit phase where in the first message, the receiver specifies the hash key.

Replacing keyed collision-resistant hashing with unkeyed multi-collision-resistant hashing, we get a non-interactive (one commitment message) shrinking and statistically-hiding commitments with *weak binding*: the sender may be able to open the commitment to more than one value, but not to too many values (say, polynomial in the size of its non-uniform advice). We then observe that in many applications of commitments, weak binding is sufficient. We proceed to give examples.

Barak's Protocol. A first example is the public-coin constant-round zero-knowledge protocol of Barak [Bar01]. To prove an **NP** statement $x \in \mathcal{L}$, the protocol proceeds in two phases. In a preamble phase the prover sends a shrinking commitment c to the code of some (potentially long) program Π and the verifier responds with a random string r, much longer than the commitment c. Then, in a proof phase, the prover gives a succinct witness-indistinguishable argument of knowledge proving that either $x \in \mathcal{L}$ or that the committed program $\Pi(c)$ happens to output r. By committing to the code of the verifier itself, a non-black-box simulator is able to produce an accepting transcript without using the witness. Still, a cheating prover, who does not know the verifier's randomness r, can only commit to such a program with negligible probability.

We can shave one message from Barak's protocol by replacing the prover's commitment with a weakly-binding commitment. Roughly speaking, the resulting protocol is still sound because even if the prover can open its commitment to polynomially many programs, with overwhelming probability, none of these programs predicts r. Namely, the soundness error reduces by a factor of K, where K is the number of values that the prover can open.

Domain Extension. The above example is, in fact, oversimplified. To prove that Barak's protocol is zero-knowledge against verifiers of arbitrary (polynomial) size, the simulator must be able to commit to arbitrarily-long programs. Therefore, the hash H underlying the commitment scheme must shrink arbitrarily long input strings in $\{0,1\}^*$ to output strings of some fixed length, say in $\{0,1\}^\lambda$. Transforming a fixed-domain hash function $\mathsf{H} : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^\lambda$ to one over arbitrary inputs in $\{0,1\}^*$ is typically quite basic and is known as *domain extension* [Dam89, Mer89]. However, as we have already explained, while existing domain-extension techniques preserve collision resistance, they destroy multi-collision resistance — the bound on the collision size grows exponentially with the input length.

We give a new domain extension construction for multi-collision-resistant hash functions. The construction consists of a cryptographic component (based on multi-collision resistance) and an information-theoretic component. We now proceed to describe each of the two.

Component 1: Hash Trees. The first component is, in fact, a standard hash tree construction [Mer89], which we now describe more concretely. Given a basic hash H shrinking strings of length 2λ to strings of length λ, we hash a long string by splitting it into blocks of length λ and using H in the form of a binary tree, to hash all blocks down to a single root. We observe that if H is resistant against collisions of size K, then the tree construction provides a *local* multi-collision resistance guarantee: for every index i, it is hard to find many long strings $X \in \left(\{0,1\}^\lambda\right)^n$ that hash to the same root such that their i-th block $X[i] \in \{0,1\}^\lambda$ takes more than $(K-1)^d$ values, where $d = \log n$ is the depth of the tree. Starting from a stronger basic hash H that shrinks strings of say length λ^2 to strings of length λ, we can use a tree of arity λ and improve the bound on the collision size: for input strings of arbitrary polynomial length, it is hard to find a $\mathrm{poly}(K)$ collision for any individual block.

We note again that the guaranteed local multi-collision resistance is weaker than full multi-collision

resistance since it may be possible to "mix-and-match" collisions for different blocks. Accordingly, the bound on the number of global collisions still grows exponentially with the length of the input (rather than the depth of the tree).

Component 2: Rectangle-Evasive Codes. To go from the local multi-collision-resistance guarantee provided by the hash tree to full multi-collision-resistance, we force a certain global structure on the inputs by first encodings them with an appropriate code. Specifically, let $C : \left(\{0,1\}^\lambda\right)^\ell \to \left(\{0,1\}^\lambda\right)^n$ be a code that maps ℓ blocks to n blocks. We hash inputs X of length $\lambda \cdot \ell$, by first encoding them, and then hashing the encoded input with a hash tree. We require that the code does not have a large intersection with any relatively small rectangle. That is, for every sequence of sets $S_1, \ldots, S_n \subseteq \{0,1\}^\lambda$, each of size K, the number of codewords in the rectangle $S_1 \times \cdots \times S_n$ is at most $K' = \mathrm{poly}(K)$. We call such codes *rectangle evasive*. This construction of a hash tree (with an underlying multi-collision-resistant hash) over a rectangle-evasive code, gives a multi-collision-resistant hash function for an arbitrary polynomial-size domain.

Rectangle-evasive codes can be based on known constructions of unbalanced expanders [GUV09] and are closely related to the notion of list-recoverable codes [GI01].[3] Specifically, consider a degree n bipartite graph over $2^{\lambda \cdot \ell}$ nodes on the left, and 2^λ nodes on the right such that the n neighbors of a node x on the left are the n blocks of $C(x)$ on the right. Rectangle evasiveness now translates to a weak expansion property on the graph: The neighborhood of any K' nodes on the left must be of size at least K.

Succinct Arguments and Local Opening. Many applications of collision-resistant hashing require domain extension with stronger properties such as efficient local opening. For example, in Kilian's succinct argument system for NP, the prover constructs a PCP proof of the statement and commits to the long proof with a hash tree. The verifier then attempts to verify the PCP proof by asking the prover to open a few random locations of the committed proof. By exploiting the tree structure of the hash, the prover can open the requested locations and prove that the values are consistent with the committed root without opening the entire tree. It only needs to provide the hash values on the paths from the opened locations to the root (along with their siblings).

Formalizing the local opening in the setting of multi-collision resistance requires more care. Simply requiring that every subset of input locations can be opened in a small number of different ways is not sufficient in applications such as Kilian's arguments. Instead, we make a stronger requirement, which intuitively says that there exists a small number of global assignments, such that any opened subset of values is consistent with one of them. A bit more accurately, we require that given any (randomized) adversary that produces a hash value and then successfully opens some subset of locations, it is possible to efficiently extract at most K global inputs, such that any subset of locations opened by the adversary is consistent with one of the global inputs with overwhelming probability.

We emphasize that when multiple locations are opened simultaneously they are required to be consistent with the same global input. Therefore, the opening of different locations must be correlated. We cannot simply open every location independently as in the case of collision-resistant hash trees.

Kilian's argument can be instantiated based on any hash with local opening that satisfies the above notion of multi-collision resistance. Given any cheating prover that tries to prove a false statement, we can extract K full PCP proofs. By the (negligible) soundness of the PCP system, none of the K proofs is convincing with overwhelming probability. Since the prover must always answer any subset of queries consistently with one of these proofs, soundness is guaranteed.

Achieving Local Opening. The domain extension construction for multi-collision resistance, based on

[3] List recoverable codes were also shown to be useful for parallel domain extension for collision-resistant hash functions in [HIOS15].

rectangle-evasive codes, does not support local opening in general — to open even a single location of the input, one must first open all the leaves of the hash tree, check that the result is a valid codeword, and only then decode it.

The first idea toward achieving local opening is to use a rectangle-evasive code C that is also *locally decodable*. Here it suffices to consider the error-less case, where the goal is to locally decode from codewords without any errors. To open one location of the input, select a small set of locations D that can be used for local decoding and open the hash-tree leaves that contain the locations in D. This approach however does not guarantee (global) multi-collision resistance — even if every rectangle contains at most K' global codewords, there could be much more than K' codewords that are consistent with the rectangle on a small set of locations such as D.

Collision-Free Codes. We design a new type of code, which we call *collision free*, that will suffice for constructing a multi-collision-resistant hash with local opening. The code has the following local decoding flavor. To decode a location i of the input word, we read a small set of locations D of the codeword together with a small random set of *test locations* T. The test locations are independent of i and intuitively, are used to synchronize the decoding of different locations. In addition to decoding the value of the i-th location of the input word, we also verify consistency between the values in the locations given by D and those given by T.

Collision freeness says roughly the following: for every rectangle $\mathbf{S} = S_1 \times \cdots \times S_n$ of small (say polynomial) cardinality, with high probability over the choice of T, for any location i and set D used to decode the location i, there are no partial codewords C and C' that collide in the following sense:

- Both C and C' are consistent with the rectangle $\mathbf{S}$.[4]
- In both C and C', the values in locations T and D satisfy the consistency test.
- C and C' agree on the locations T, but not on D.

In other words, with high probability, an assignment to the test locations, completely fixes how any location is decoded, provided that the symbols read are always taken from the rectangle $\mathbf{S}$.

Based on collision-free codes, the construction of multi-collision-resistant hash with local opening is as follows: to open a set of locations $i_1, \ldots i_k$ of the input, sample a set of test locations T and sets $D_1, \ldots, D_k$ for decoding the locations $i_1, \ldots i_k$ of the input. Open the leaves of the hash tree that contain these locations and verify the consistency of the values in locations T and D_j for every $j \in [k]$. If all values are consistent, decode the input locations $i_1, \ldots i_k$.

By the local multi-collision-resistance of the hash tree, there is some rectangle $\mathbf{S} = S_1, \ldots, S_n \subseteq \{0,1\}^\lambda$ where each S_i is of size at most K, such that every opening for the locations T and $\{D_j\}$ is consistent with $\mathbf{S}$. It follows that the locations T can take at most $K^{|T|}$ combinations of values. By the collision freeness of the code, the values for locations T fix some global word input x, and the decoded values in locations $i_1, \ldots i_k$ are consistent with x. The collision bound of the final construction is therefore $K^{|T|}$. To minimize this bound, we design a collision-free code where the size of the set T is small (for a natural setting of parameters, it will be a constant).

Collision-Free Polynomial Code. We construct a collision-free polynomial code C based on low-degree multivariate polynomials (which can be seen as an over-redundant variant of Reed-Muller codes). For every m, the code maps strings of length $\ell = \lambda^m$ to a codeword that consists of $n = \mathrm{poly}(\ell)$ blocks. We first

[4] In more detail, by a partial codeword C, we mean that $C = (C_i \mid i \in U)$ is a partial assignment for a subset $U \subseteq [n]$ of the blocks. C is consistent with the rectangle $\mathbf{S} = S_1 \times \cdots \times S_n$ if for any $i \in U$, $C_i \in S_i$.

describe the code for $m = 2$ and then generalize the construction to arbitrary m. Let $\mathbb{F}$ be a field of size $\text{poly}(\lambda)$, and let $H \subseteq \mathbb{F}$ be a subset of size λ. To encode an input $x \in \{0,1\}^{\lambda^2} \simeq \{0,1\}^{H^2}$, we first compute the low-degree extension P_x of x. That is, $P_x : \mathbb{F}^2 \to \mathbb{F}$ is a bivariate polynomial of (individual) degree $\lambda - 1$ whose evaluations on the square H^2 encode x. The codeword $C(x)$ consists of the restrictions of P_x to all horizontal and vertical lines. That is, the codeword consists of $n = 2|\mathbb{F}|$ blocks, each describing a degree $\lambda - 1$ univariate polynomial.

We consider a rectangle $S_1 \times \cdots \times S_n$ where $|S_i| \leq K$. The set T consists of τ random vertical lines, where τ is chosen such that $(|\mathbb{F}|/\lambda)^\tau > 2K^2 \cdot |\mathbb{F}|$. For example, if $K = \text{poly}(\lambda)$ then $|T| = \tau$ can be constant.[5] To decode the i-th input location, we read the horizontal line that encodes x_i and check that it agrees with all the vertical lines in T on their intersection points.

To show that the code is collision free, we note that fixing the values on τ random vertical lines fixes the value on τ random points for every horizontal line. We then argue that this is enough to fix a single value for each horizontal line, and thus a unique decoded value. In a bit more detail, let restrict attention to some specific horizontal line. Then by consistency with the rectangle $\mathbf{S}$, there are at most K polynomials (each of degree $\lambda - 1$) that the line may take. By Schwartz-Zippel, each pair of these polynomials agrees on the τ random intersection points with the vertical lines in T with probability at most $(\lambda/\mathbb{F})^\tau$. Thus the probability that any such pair among the K polynomials agrees on these points is at most $K^2 \cdot (\lambda/\mathbb{F})^\tau < 1/2|\mathbb{F}|$. Furthermore, with probability $1/2$ this will be the case for all $|\mathbb{F}|$ restrictions of P_x to horizontal lines. In the actual construction, this collision probability is reduced by standard amplification.

We extend this construction recursively for higher values of m. Specifically to encode an input $x \in \{0,1\}^\ell$ for $\ell = \lambda^m$, we extend x to an m-variate degree-$(\lambda - 1)$ polynomial $P_x : \mathbb{F}^m \to \mathbb{F}$. The codeword contains the restrictions of P_x to all $n = m|\mathbb{F}|^{m-1}$ axis-parallel lines. To decode the i-th input location, we read one such axis-parallel line γ that encodes x_i. We then sample τ random $m - 1$ dimensional hyperplanes orthogonal to γ. For every hyperplane, we recursively decode the intersection point of the hyperplane with γ. This will result in a set of test indexes T of size τ^m. For an input of size $\text{poly}(\lambda)$, $|T| = \tau$ is again constant.

3-Message Zero-Knowledge Arguments. Our starting point is the 3-message zero-knowledge argument of Bitansky et al. [BBK+16] in *the global hash model*. In this model, the prover and verifier agree on a hash function before interacting. The soundness of the protocol is reduced to finding collisions in the hash. We get a protocol, in the plain model, relying on an unkeyed multi-collision-resistant hash.

The protocol of Bitansky et al. relies on a *memory delegation* scheme. In memory delegation schemes, the prover sends a short digest of a long memory string. The verifier then sends the description of a computation to be executed on the memory, together with a cryptographic challenge, and the prover responds with the computation's output and a proof of correctness. The protocol is sound in the sense that the prover cannot convince the verifier that the same computation has two different outputs with respect to the same memory digest. Based on memory delegation, Bitansky et al. give a 3-message version of Barak's protocol. They essentially use memory delegation construct a succinct 3-message witness-indistinguishable argument.

In their protocol, the collision-resistant hash is used both to instantiate the memory delegation protocol, and in the transformation from memory delegation to zero-knowledge. In this overview, we focus on achieving the former based on multi-collision resistance, which is more essential (in particular, the latter can avoid the use of collision resistance assuming ZAPs [DN07].) We first consider the task of memory delegation in an intermediate model where, instead of sending a digest of the memory, the prover publishes

[5] When we use the code to construct a hash with local opening, the actual value of K is not known ahead of time. We thus apply the code for all values $\tau \leq \bar{\tau}$, for a slightly super-constant function $\bar{\tau} = \omega(1)$. Then, only in the analysis we restrict attention to the relevant τ.

an encoding of the entire memory as an oracle. The verifier delegates a computation to be executed on the memory and can verify the proof of correctness by making only a few queries to the encoded memory given by the oracle. We also require that the verifier's queries depend only on its private coins and not on the proof. We note that such *oracle memory delegation* follows from the protocol of [KRR14].

We then go from oracle-based memory delegation to standard memory delegation in two steps. First, the prover commits to the encoded memory oracle with a multi-collision-resistant hash with local opening. Then, together with its challenge, the verifier sends its queries to the oracle under fully-homomorphic encryption. The prover responds with the proof (in the clear) and the oracle answers (under the encryption). We prove, based on the multi-collision-resistance of the commitment and semantic security of the encryption, that the memory delegation scheme remains sound even when the verifier's encrypted queries are given to the prover.

Since the oracle commitment relies on multi-collision-resistant hash, we are only guaranteed that the oracle's answers are consistent with one of a small number of oracles. We, therefore, get a weaker soundness guarantee from the memory delegation — the prover cannot convince the verifier that the same computation has *many* different outputs with respect to the same memory digest. Just like in the 5-message version of Barak's protocol described above, such weak soundness is sufficient to prove the soundness of the zero-knowledge protocol, since with overwhelming probability, the random string r is not among the outputs for which the prover can generate a convincing proof.

4-Message ε-Zero-Knowledge Proofs. We show how to modify the 5-message zero-knowledge proof system of Goldrich and Kahan [GK96a] to get a 4-message ε-zero-knowledge proof based on unkeyed multi-collision-resistant hash. In ε-zero-knowledge the simulator's running time may grow polynomially with the required simulation accuracy (distinguishing gap) ε.

In the protocol of [GK96a], the verifier first sends a statistically-hiding commitment to a random string r. The prover and verifier then execute a 3-message public-coin proof with negligible soundness [GMW91], where the verifier opens r as its random challenge. The 3-message proof has the property that for every r, a proof with challenge r can be efficiently simulated given r. Therefore the entire protocol can be simulated as follows: interact with the verifier until it opens the challenge r, and then rewind the verifier and simulate the proof using r. The simulation is successful since the verifier is bound to open the same value of r in every interaction.

We replace the verifier's two-message statistically-hiding commitment, with a non-interactive weakly-binding commitment. If the verifier's commitment is only weakly binding, when rewound, the verifier may open the commitment to a different value. Therefore, we have the simulator repeatedly rewind the verifier until it again opens the commitment to r. The expected time of this naive simulation strategy is exponential since the verifier may open the commitment to any value r' with some negligible probability, which will result in a super-polynomial number of rewinding attempts. Instead, we change the simulation to abort after $1/\varepsilon$ rewinding attempts. Using the weak binding of the verifier's commitment, we show that the verifier only aborts with probability roughly ε.

We note that by the weak binding of the commitment, there exists a list of K values such that the verifier cannot open the commitment to any value outside the list, except with negligible probability. If we change our simulator to abort when the verifier opens a value outside this list, we would get expected polynomial time simulation and negligible distinguishing gap. However, we do not know how to efficiently extract this list from the cheating verifier (the list may even depend on the statement being proved). Our actual solution can be seen as using an approximate list, where values outside the list may be opened with probability at most ε.

2 Preliminaries

We rely on the standard computational concepts:

- A PPT is a probabilistic polynomial-time algorithm.
- We model efficient adversaries as PPTs with arbitrary non-uniform advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$.
- We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for all constants $c > 0$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$. We sometimes denote negligible functions by negl.
- We say that a function $f : \mathbb{N} \to \mathbb{R}$ is noticeable if there exists a constant $c > 0$ and $N \in \mathbb{N}$ such that for all $n > N$, $f(n) \geq n^{-c}$.
- If $\mathcal{X}^{(b)} = \{X_\lambda^{(b)}\}_{\lambda\in\mathbb{N}}$ for $b \in \{0, 1\}$ are two ensembles of random variables indexed by $\lambda \in \mathbb{N}$, we say that $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ are ε-computationally indistinguishable for a function $\varepsilon(\lambda)$, if for all polynomial-size distinguishers $\mathcal{D}$, and all large enough λ,

$$\left|\Pr[\mathcal{D}(X_\lambda^{(0)}) = 1] - \Pr[\mathcal{D}(X_\lambda^{(1)}) = 1]\right| \leq \varepsilon(\lambda).$$

  We denote this by $\mathcal{X}^{(0)} \approx_\varepsilon \mathcal{X}^{(1)}$. We say that the ensembles are simply **computational indistinguishable** if they are ε-indistinguishable for evert noticeable function $\varepsilon(\lambda) = \lambda^{-O(1)}$.

Fact 2.1. *Let D be a distribution, π be a predicate, f be a function on the support of D, and $t \in \mathbb{N}$ be an integer. Let $S_0 = \emptyset$, and consider a random process where for every $i \in [t]$, we sample $x_i \leftarrow D$ and if $\pi(x_i) = 1$, add $f(x_i)$ to the previous set $S_i := S_{i-1} \cup \{f(x_i)\}$. Let p be the probability that an additional sample $x_{t+1} \leftarrow D$ satisfies the predicate and is not covered by S_n, namely, $\pi(x_{t+1}) = 1$ but $f(x_{t+1}) \notin S_t$. Then,*

$$p \leq \frac{\mathbb{E}\left[|S_t|\right]}{t} \quad .$$

Proof. The fact that S_t grows with probability p given a random sample x_{t+1}, implies that for every i, the probability that S_{i-1} grows in step i is at least p. Thus, the expected number size of S_t is at least tp. □

2.1 Zero-Knowledge Protocols

In what follows, we denote by $\langle \mathsf{P} \leftrightarrows \mathsf{V}\rangle$ a protocol between two parties P and V. For input w for P, and common input x, we denote by $\langle \mathsf{P}(w) \leftrightarrows \mathsf{V}\rangle(x)$ the output of V in the protocol. For honest verifiers this output will be a single bit indicating acceptance (or rejection), whereas we assume (without loss of generality) that malicious verifiers outputs their entire view. Throughout, we assume that honest parties in all protocols are uniform PPT algorithms.

Definition 2.1 ([GMR89])**.** *A protocol $\langle \mathsf{P} \leftrightarrows \mathsf{V}\rangle$ for an **NP** relation $\mathcal{R}(x, w)$ is a zero-knowledge proof if it satisfies:*

Completeness: *For any $\lambda \in \mathbb{N}, x \in \mathcal{L}(\mathcal{R}) \cap \{0, 1\}^\lambda$, $w \in \mathcal{R}(x)$:*

$$\Pr\left[\langle \mathsf{P}(w) \leftrightarrows \mathsf{V}\rangle(x) = 1\right] = 1 \quad .$$

Soundness: *For any prover* P^* *there exists a negligible function* μ*, such that for any* $x \in \{0,1\}^\lambda \setminus \mathcal{L}(\mathcal{R})$*,*

$$\Pr\left[\langle \mathsf{P}^* \leftrightarrows \mathsf{V}\rangle(x) = 1\right] \leq \mu(\lambda) \ .$$

The system is **computationally sound** *if the above is restrict to PPT provers* P^* *with polynomial-size advice* $\{z_\lambda\}_{\lambda\in\mathbb{N}}$*. In this case, the protocol is said to be* **an argument***.*

Computational Zero Knowledge: *For every PPT verifier* V^* *with polynomial-size advice* $\{z_\lambda\}_{\lambda\in\mathbb{N}}$*, there exists a (uniform) PPT simulator* S *such that:*

$$\{\langle \mathsf{P}(w) \leftrightarrows \mathsf{V}^*(x; z_\lambda)\rangle\}_{\substack{(x,w)\in\mathcal{R} \\ |x|=\lambda}} \approx_c \{\mathsf{S}(x; z_\lambda)\}_{\substack{(x,w)\in\mathcal{R} \\ |x|=\lambda}} \ .$$

3 Multi-Collision-Resistant Hash Functions

In this section, we define the notion of multi-collision-resistant hash functions. We start with the standard formulation of this notion, and then define a relaxed version geared toward the setting of fixed (unkeyed) hash functions.

Syntax: A hash function is associated with an input length function $\ell(\lambda) > \lambda$ and polynomial-time algorithms $\mathsf{H} = (\mathsf{H.Gen}, \mathsf{H.Hash})$ with the following syntax:

- $\mathsf{H.Gen}(1^\lambda)$ is a probabilistic algorithm that takes the security parameter 1^λ and outputs a key hk. In the unkeyed setting, this algorithm is deterministic and outputs a fixed key $\mathsf{hk} \equiv 1^\lambda$.
- $\mathsf{H.Hash}(\mathsf{hk}, X)$ is a deterministic algorithm that takes the key hk and an input $X \in \{0,1\}^{L(\lambda)}$ and outputs a hash $Y \in \{0,1\}^\lambda$.

Definition 3.1 (K-Collision Resistance)**.** *Let* $K(\cdot)$ *be a function. We say that* H *is* K*-collision-resistant if for any PPT* $\mathcal{A}$ *and sequence of polynomial-size advice* $\{z_\lambda\}_{\lambda\in\mathbb{N}}$*, there is a negligible function* μ*, such that for any* $\lambda \in \mathbb{N}$*, letting* $K = K(\lambda)$*,*

$$\Pr\left[\begin{array}{c} Y_1 = \cdots = Y_K \\ \forall i \neq j : X_i \neq X_j \end{array} \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{H.Gen}(1^\lambda) \\ (X_1, \ldots, X_K) \leftarrow \mathcal{A}(\mathsf{hk}; z_\lambda) \\ \forall i : Y_i = \mathsf{H.Hash}(\mathsf{hk}, X_i) \end{array}\right] \leq \mu(\lambda) \ .$$

Remark 3.1 (Compression). We note that for the above definition to be non-trivial it must be that there necessarily exists K-collisions. Typically, we will consider input length $L(\lambda) \geq 2\lambda$, output length λ, and $K(\lambda) = 2^{o(\lambda)}$, in which case almost any input is part of a K-collision.

The above definition of multi-collision-resistance is clearly unachievable in the unkeyed setting where H is a fixed hash function. Indeed, the advice z_λ may always include some K-collision in the function H. This motivates our relaxed definition, where the adversary may be able to find collisions of size proportional to its advice, but not significantly larger. That is, the number of collisions K now depends on (and could be bigger than) the size of the advice z_λ.

We note that while the definition is geared toward the unkeyed setting it also serves as a meaningful relaxation in the keyed setting. We accordingly formulate it in the more general keyed setting.

Definition 3.2 (Weak K-Collision Resistance)**.** *Let $K(\cdot,\cdot)$ be a function. We say that H is weakly K-collision-resistant if for any PPT $\mathcal{A}$ and any sequence of polynomial-size advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$, there is a negligible function μ, such that for any $\lambda\in\mathbb{N}$, letting $K = K(\lambda,|z_\lambda|)$,*

$$\Pr\left[\begin{array}{c|l} Y_1=\cdots=Y_K & \mathsf{hk}\leftarrow\mathsf{H.Gen}(1^\lambda) \\ \forall i\neq j: X_i\neq X_j & (X_1,\ldots,X_K)\leftarrow\mathcal{A}(\mathsf{hk};z_\lambda) \\ & \forall i: Y_i=\mathsf{H.Hash}(\mathsf{hk},X_i)\end{array}\right]\leq\mu(\lambda)\ .$$

Remark 3.2 (Super-Polynomial Running Time). For some of our applications, we will require a strengthening of the above two definitions that allows the adversary $\mathcal{A}$ to run in some (usually slight) super-polynomial time. Accordingly, for a function $\gamma(\lambda)$ (possibly $\gamma=\lambda^{\omega(1)}$), we will say that H is (weakly) (K,γ)-collision resistant if the guarantee of Definitions 3.1 and 3.2 holds against any probabilistic $\gamma^{O(1)}$-time $\mathcal{A}$ with arbitrary polynomial-size advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$.

Remark 3.3 (The Collision Size K). Naturally, we shall think of the collision size parameter K as polynomial in the security parameter λ. In some cases, however, we may want to think also of super-polynomial K. In those cases, for the above definition to be meaningful, we consider (K,γ)-collision resistance for γ that may be large than K, e.g. an arbitrary polynomial $\mathrm{poly}(K)$.

4 Multi-Collision-Resistant Hash with Local Opening

In this section, we define and construct a multi-collision-resistant hash with local opening, which is an analog of the concept of hash tress from the literature [Mer89], with a relaxed collision resistance requirement.

Syntax: A multi-collision-resistant hash with local opening is associated with polynomial-time algorithms

$$\mathsf{HLO}=(\mathsf{HLO.Gen},\mathsf{HLO.Hash},\mathsf{HLO.Chal},\mathsf{HLO.Auth},\mathsf{HLO.Ver})\ ,$$

with the following syntax:

- $\mathsf{hk}\leftarrow\mathsf{HLO.Gen}(1^\lambda)$ is a probabilistic algorithm that takes the security parameter 1^λ and outputs a key hk. In the unkeyed setting, this algorithm is deterministic and outputs a fixed key $\mathsf{hk}\equiv 1^\lambda$.
- $\mathsf{dig}\leftarrow\mathsf{HLO.Hash}(\mathsf{hk},X)$ is a deterministic algorithm that takes the key hk and an input $X\in\{0,1\}^L$ of length $L\leq 2^\lambda$. It outputs a digest $\mathsf{dig}\in\{0,1\}^\lambda$.
- $\mathsf{ch}\leftarrow\mathsf{HLO.Chal}(1^\lambda,1^\rho)$ is a probabilistic algorithm that takes the security parameter 1^λ and an opening-size parameter 1^ρ. It outputs a challenge ch.
- $\Pi\leftarrow\mathsf{HLO.Auth}(\mathsf{hk},X,I,\mathsf{ch})$ is a deterministic algorithm that takes the key hk, input $X\in\{0,1\}^L$, an index set $I\subseteq[L]$ and a challenge ch. It outputs a proof Π that $X|_I=(X_i\mid i\in I)$ is consistent with the digest dig.
- $b\leftarrow\mathsf{HLO.Ver}(\mathsf{hk},L,\mathsf{dig},I,A,\mathsf{ch},\Pi)$ is a deterministic algorithm that takes the key hk, an input length $L\in\mathbb{N}$, the digest $\mathsf{dig}\in\{0,1\}^\lambda$, the index set I, and an assignment $A: I\to\{0,1\}$, as well as a challenge ch and a corresponding proof Π. It outputs a bit.

Remark 4.1 (The Challenge). Differently from the standard notion of Merkle tree, where the opening phase is non-interactive and includes a single message from the opening party, in the definition considered here the opening phase consists of a random challenge from the receiver, followed by the response message. The

challenge itself depends on the security parameter λ as well as an opening size ρ, which intuitively specifies the maximal number of locations that can be simultaneously opened, while guaranteeing consistency with the digest (in terms of completeness, any number of locations may be opened regardless of ρ).

In the above definition, verification is public in the sense that the challenge algorithm does not produce any private state. We may further require a *public-coin* challenge algorithm, where the challenger simply outputs its random coins. Indeed, our construction will satisfy this stronger requirement.

Definition 4.1 (Multi-Collision-Resistant Hash with Local Opening)**.** *A multi-collision-resistant hash with local opening* $\mathsf{HLO} = (\mathsf{HLO.Gen}, \mathsf{HLO.Hash}, \mathsf{HLO.Chal}, \mathsf{HLO.Auth}, \mathsf{HLO.Ver})$ *satisfies:*

Correctness: *The verifier accepts in any honest execution. That is, for every parameters* $\lambda, \rho \in \mathbb{N}$*, integer* $L \leq 2^\lambda$*, string* $X \in \{0,1\}^L$*, and set* $I \subseteq [L]$*,*

$$\Pr\left[\mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, X|_I, \mathsf{ch}, \Pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda) \\ \mathsf{dig} = \mathsf{HLO.Hash}(\mathsf{hk}, X) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ \Pi = \mathsf{HLO.Auth}(\mathsf{hk}, X, I, \mathsf{ch}) \end{array}\right] = 1 \ .$$

Succinctness: *There exists a fixed polynomial* poly*, such that the length of the proof in the above (honest) experiment is* $|\Pi| = \mathrm{poly}(\lambda, \rho, |I|)$.

K-Collision Resistance for Length Bound $\bar{L}(\lambda)$: *There exists a PPT extractor* Ext *with the following guarantee. For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, any polynomial-size advice sequence* $\{z_\lambda\}_{\lambda \in \mathbb{N}}$*, any noticeable function* $\varepsilon(\lambda)$*, any length* $L(\lambda) = \bar{L}^{O(1)}$*, for all but finitely many security parameters* λ *and every opening size* $\rho \leq L$*, letting* $K = K(\lambda, |z_\lambda|, L)$*,*

$$\Pr\left[\begin{array}{l} \\ |I| \leq \rho \\ \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi) = 1 \\ A \notin \{X|_I : X \in S\} \\ \\ \end{array} \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda) \\ (\mathsf{dig}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ (I, A, \Pi) \leftarrow \mathcal{A}_2(\mathsf{ch}; \mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2(\cdot;\mathsf{st})}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{1/\varepsilon}) \end{array}\right] \leq \varepsilon \ .$$

Furthermore, in the above experiment Ext *always outputs a set* S *of size at most* K.

Remark 4.2 (The Extracted-List Size K). While it is natural to think of the size K of the extracted list as polynomial in the security parameter λ, we will also consider a weaker guarantee where K is super polynomial. We note that even when K is super polynomial we may address polynomial-size adversaries $\mathcal{A}$, and only the size of the extractor scales with K.

Remark 4.3 (The Input-Length Bound $\bar{L}$). We note that while in terms of functionality, we will always support hashing strings of any length $\leq 2^\lambda$, in terms of security we may be restricted to smaller values of $\bar{L}$. Indeed, in our constructions the achieved parameters will depend on $\bar{L}$, and out scheme will be able to tolerate a bound $\bar{L}$ of at most $2^{o(\lambda)} \ll 2^\lambda$ (assuming appropriate collision-resistance).

We now state the main theorems proved in this section regarding the existence of multi-collision-resistant hash with local opening (according to the above definition) based on weak multi-collision-resistant hash functions (Definition 3.2). Theorem 4.1 is a polynomial version that guarantees security for inputs of arbitrary polynomial length, based on polynomial assumptions. Theorem 4.2 is a super-polynomial version that guarantees security even for slightly super-polynomial input length, relying on slightly super-polynomial assumptions. This second theorem will be useful in applications where an apriori polynomial bound on the input length may not be known.

Theorem 4.1 (Multi-Collision-Resistant Hash with Local Opening for Polynomial-Length Input)**.** *Assuming a weakly K-collision-resistant hash for $K(\lambda,\zeta)=\mathrm{poly}(\lambda,\zeta)$, there exists a $K^{O(1)}$-collision-resistant hash with local opening for any input-length bound $\bar{L}(\lambda)=\lambda^{O(1)}$.*

Theorem 4.2 (Multi-Collision-Resistant Hash with Local Opening for Super-Polynomial-Length Input)**.** *For any (arbitrarily small) $\tau(\lambda)=\omega(1)$, there exists $\bar{L}(\lambda)=\lambda^{\omega(1)}$ such that assuming a weakly (K,γ)-collision-resistant hash for $K(\lambda,\zeta)=\mathrm{poly}(\lambda,\zeta)$ and $\gamma(\lambda)=\lambda^{\tau}$, there exists a K^{τ}-collision-resistant hash with local opening for input-length bound $\bar{L}$.*

Remark 4.4 (Super-Polynomial Collision Bound). In the above, we concentrate on a natural setting of parameters, where we assume the collision bound K is polynomial. We can consider a collision parameter that may be some super-polynomial function at the cost of assuming (K,γ)-collision-resistance for an appropriate $\gamma \gg K$ hardness parameter.

In the next sections, we describe the construction behind the theorems, and its building blocks. Eventually, the theorems above are derived as corollaries of a more general Theorem 4.5 proven in Sections 4.3,4.4.

4.1 Ingredient I: Hash Trees

As a building block toward the construction of hash with local opening, we define and construct a hash tree. Roughly speaking, our hash tree allows to commit to a string $X \in \{0,1\}^{\lambda \times L}$ consisting of L blocks in $\{0,1\}^{\lambda}$. Unlike multi-collision-resistant hash with local opening, the hash tree is only *locally binding* in the sense that for any single location $i \in [L]$, specifying *a block of bits* the adversary may successfully open only a small number of values K from $\{0,1\}^{\lambda}$ for this block. However, when opening many locations $I = \{i_1,\ldots,i_{|I|}\}$ simultaneously, the adversary can "mix-and-match" values. That is, the number of possible openings overall may be as large as $K^{|I|} \gg \mathrm{poly}(K)$.[6]

Syntax: A hash tree is associated with polynomial-time algorithms

$$\mathsf{HT} = (\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Auth}, \mathsf{HT.Ver})$$

with the following syntax:

- $\mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^{\lambda})$: is a probabilistic algorithm that takes the security parameter 1^{λ} and outputs a key hk. In the unkeyed setting, the algorithm is deterministic and outputs a fixed key $\mathsf{hk} \equiv 1^{\lambda}$.
- $\mathsf{dig} \leftarrow \mathsf{HT.Hash}(\mathsf{hk}, X)$: is a deterministic algorithm that takes a key $\mathsf{hk} \in \{0,1\}^{\lambda}$ and an input $X \in \{0,1\}^{\lambda \times L}$ where $L \leq 2^{\lambda}$. It outputs a digest $\mathsf{dig} \in \{0,1\}^{\lambda}$.
- $\Pi \leftarrow \mathsf{HT.Auth}(\mathsf{hk}, X, i)$: is a deterministic algorithm that takes a key hk, an input $X \in \{0,1\}^{\lambda \times L}$ and an index $i \in [L]$. It outputs a proof Π that X_i is consistent with the digest dig.
- $b \leftarrow \mathsf{HT.Ver}(\mathsf{hk}, L, \mathsf{dig}, i, A, \Pi)$: is a deterministic algorithm that takes the key hk, an input length $L \in \mathbb{N}$, the digest $\mathsf{dig} \in \{0,1\}^{\lambda}$, the index i, a block assignment $A \in \{0,1\}^{\lambda}$, and a proof Π. It outputs a bit.

[6] Note that for such a definition, considering blocks rather than individual bits is necessary. Indeed, we consider blocks that may potentially have an exponential number of values, making the restriction to polynomial meaningful, whereas for an individual bit a restriction to any more than a single value becomes meaningless.

Definition 4.2 (Multi-Collision-Resistant Hash Tree)**.** *A multi-collision-resistant hash tree* $\mathsf{HT} = (\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Auth}, \mathsf{HT.Ver})$ *satisfies:*

Correctness: *The verifier accepts in any honest execution. That is, for every security parameter* $\lambda \in \mathbb{N}$*, integer* $L \leq 2^\lambda$*, string* $X \in \{0,1\}^{\lambda \times L}$*, and index* $i \in [L]$*,*

$$\Pr\left[\mathsf{HT.Ver}(\mathsf{hk}, L, \mathsf{dig}, i, X_i, \Pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda) \\ \mathsf{dig} = \mathsf{HT.Hash}(\mathsf{hk}, X) \\ \Pi = \mathsf{HT.Auth}(\mathsf{hk}, X, i) \end{array}\right] = 1 \ .$$

Succinctness: *There exists a fixed polynomial* poly*, such that the length of the proof in the above (honest) experiment is* $|\Pi| = \mathrm{poly}(\lambda)$*.*

K-Collision Resistance for Input-Length Bound $\bar{L}(\lambda)$ and Accuracy Bound $\underline{\varepsilon}(\lambda)$: *There exists a PPT extractor* Ext *with the following guarantee. For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, any polynomial-size advice sequence* $\{z_\lambda\}_{\lambda \in \mathbb{N}}$*, any* $\varepsilon(\lambda) > \underline{\varepsilon}^{\Omega(1)}$*, any length* $L \leq \bar{L}^{O(1)}$*, for all but finitely many security parameters* λ*, and for every* $i \in [L]$*, letting* $K = K(\lambda, |z_\lambda|, L)$*,*

$$\Pr\left[\begin{array}{l} \mathsf{HT.Ver}(\mathsf{hk}, L, \mathsf{dig}, i, A, \Pi) = 1 \\ A \notin S \end{array} \;\middle|\; \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda) \\ (\mathsf{dig}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (A, \Pi) \leftarrow \mathcal{A}_2(\mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2(\mathsf{st})}(i, 1^K, 1^{1/\varepsilon}) \end{array}\right] \leq \varepsilon \ .$$

Furthermore, in the above experiment Ext *always outputs a set* S *of size at most* K*.*

Remark 4.5. In the above definition, $\mathcal{A}_2(\mathsf{st})$ has no further input. The second part of the experiment is thus defined over the coins tosses of $\mathcal{A}_2(\mathsf{st})$.

The Construction. The construction of HT:

$$\mathsf{HT} = (\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Auth}, \mathsf{HT.Ver})$$

is based on a standard hash-tree with arity $\alpha = \alpha(\lambda)$ for $2 \leq \alpha(\lambda) \leq \lambda$. The basic building block is accordingly a hash function $\mathsf{H} = (\mathsf{H.Gen}, \mathsf{H.Hash})$ shrinking $\lambda \times \alpha$ bits to λ bits.

- $\mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda)$**:**
  Runs $\mathsf{H.Gen}(1^\lambda)$ and outputs the corresponding key hk.

- $\mathsf{dig} \leftarrow \mathsf{HT.Hash}(\mathsf{hk}, X)$**:**
  Constructs a hash tree as follows. Let L be the number of blocks in the input $X \in \{0,1\}^{\lambda \times L}$ and assume w.l.o.g that $L = \alpha^d$ for some $d \in \mathbb{N}$. We consider a corresponding α-array, depth-d, tree where each node is indexed by string $\sigma \in [\alpha]^i$ for $0 \leq i \leq d$, and is associated with a label $X_\sigma \in \{0,1\}^\lambda$ computed as follows:

  - For each leaf $\sigma \in \{0,1\}^d$,
    $$X_\sigma = X_\sigma \ .$$
    Namely, the label is the σ-th input block, where we naturally interpret σ as an integer in $[L] \cong [\alpha]^d$.
  - For each intermediate node $\sigma \in \{0,1\}^i$, where $0 \leq i < d$,
    $$X_\sigma = \mathsf{H.Hash}(\mathsf{hk}, X_{\sigma 1} \ldots X_{\sigma \alpha}) \ .$$
    Namely, the label corresponding to σ is the hash of the parent labels.

The output digest dig is then set to be the root label X_ε, where ε is the empty string.

- $\Pi \leftarrow \mathsf{HT.Auth}(\mathsf{hk}, X, \sigma)$**:**
  interprets $\sigma \in [L]$ as a string in $[\alpha]^d$, and outputs as the proof Π all the nodes on the path from σ to the root along with their siblings:

$$\Pi = (X_{\sigma_1 \ldots \sigma_i 1}, \ldots, X_{\sigma_1 \ldots \sigma_i \alpha} \mid 0 \leq i < d-1) \quad .$$

- $b \leftarrow \mathsf{HT.Ver}(\mathsf{hk}, L, X_\varepsilon, \sigma, A, \Pi)$**:**
  given a proof

$$\Pi = \left(X^*_{\sigma_1 \ldots \sigma_i 1}, \ldots, X^*_{\sigma_1 \ldots \sigma_i \alpha} \;\middle|\; 0 \leq i < d-1\right) \quad ,$$

  and letting $X^*_\varepsilon := X_\varepsilon$, it checks the consistency of the path:

$$X^*_{\sigma_1 \ldots \sigma_i} = \mathsf{H.Hash}(\mathsf{hk}, X^*_{\sigma_1 \ldots \sigma_i 1}, \ldots, X^*_{\sigma_1 \ldots \sigma_i \alpha}) \text{ for all } 0 \leq i < d \quad ,$$

  and the consistency with the given assignment:

$$X^*_{\sigma_1 \ldots \sigma_d} = A \quad .$$

The following proposition shows that if the underlying hash function is K-collision resistant, then the above tree is locally K^d-collision resistant, where d is the depth of the tree.

Theorem 4.3. *Let:*

- $K(\lambda, \zeta)$ *be polynomial in* ζ*.*
- $\bar{L} = \bar{L}(\lambda)$ *be an input-length bound and let* $\underline{\varepsilon} = \underline{\varepsilon}(\lambda)$ *be an accuracy bound.*
- H *be a weak* (K, γ)*-collision resistant hash where* $\gamma(\lambda) = K^{\bar{d}}(\lambda, \lambda)/\underline{\varepsilon}$*, and* $\bar{d} = \log_\alpha \bar{L}$*.*
- $K'(\lambda, \zeta, L) = K^d(\lambda, \zeta)$ *be a collision bound where* $d = \log_\alpha L$*.*

Then HT*, with arity* α *is* K'*-collision-resistant with input-length bound* $\bar{L}$ *and accuracy bound* $\underline{\varepsilon}$*.*

Remark 4.6 (Parameters). Throughout most of this work, we will set $\alpha = \lambda$, which will ensure that when $\bar{L}$ is polynomial so is K'. This comes at the account of assuming that the underlying hash function is polynomially compressing, i.e. maps λ^2 (or more generally $\lambda^{1+\Omega(1)}$) bits to λ bits. We can also deal with say $\alpha = 2$, which will result in weaker collision-resistance of the hash tree $K' \approx \mathrm{quasipoly}(K)$ and require stronger collision-resistance from the underlying hash (K, γ)-resistance for $\gamma \gg \mathrm{quasipoly}(K)$.

Also, the assumption above that the dependence of $K(\lambda, \zeta)$ on ζ is polynomial is for the sake of simplicity. The same statement also holds for any dependence $2^{\zeta^{o(1)}}$. (Even worst dependence can be tolerated, at the account of degrading the resulting collision resistance.)

Proof. The correctness and succinctness properties hold readily (just as in the basic construction of hash trees from the literature [Mer89]). We focus on proving multi-collision-resistance.

We describe the extractor $\mathsf{Ext}^{\mathcal{A}_2(\mathsf{st})}(i, 1^{K'}, 1^{1/\varepsilon})$ and prove that it satisfies the required guarantee.

The extractor does the following:

- **Initialize:** Creates an (initially empty) list of blocks S.

- **Sample:** Obtains from $\mathcal{A}_2(\mathsf{st})$ a total of $2K'/\varepsilon$ samples of the form (A, Π). For every such sample where Π is valid add it: $S = S \cup \{A\}$.
- **Output:** If $|S| > K'$ output `fail`, otherwise output S.

We first note that the running time of the extractor is $\mathrm{poly}(K', \varepsilon^{-1})$. Next, we prove that the extractor satisfies the local K'-collision-resistance guarantee. For this purpose, fix any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, accuracy $\varepsilon(\lambda) > \underline{\varepsilon}^{\Omega(1)}$, input length $L(\lambda) \leq \bar{L}^{O(1)}$, and $\sigma \in [L] \cong [\alpha]^d$. We first bound the failure probability.

Claim 4.1. *There exists a negligible μ, such that for all $\lambda \in \mathbb{N}$*

$$\Pr\left[\begin{array}{l|l} \mathsf{Ext}\ \textit{outputs}\ \mathtt{fail} & \begin{array}{l}\mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda) \\ (\mathsf{dig}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2(\mathsf{st})}(\sigma, 1^K, 1^{1/\varepsilon})\end{array}\end{array}\right] \leq \mu(\lambda) \ .$$

Proof. We prove this based on the (K, γ)-collision resistance of H. For this, it is enough to show that whenever the exactor fails, we can efficiently extract a K-collision in H from its transcript. Indeed, whenever the extractor fails, the list S contains at least $K' = K^d$ distinct block assignments A for σ, where each was sampled with a valid proof Π_A of consistency.

We argue that for some node $\sigma_1 \ldots \sigma_j$, along the path from the root to $\sigma = \sigma_1 \ldots \sigma_d$, the extractor obtains a label $X^*_{\sigma_1 \ldots \sigma_j}$ together with K distinct preimages:

$$\left\{X^*(t) := \left(X^*_{\sigma_1 \ldots \sigma_j 1}(t), \ldots, X^*_{\sigma_1 \ldots \sigma_j \alpha}(t)\right) \;\middle|\; t \in [K]\right\}$$
$$\textit{such that}$$
$$X^*_{\sigma_1 \ldots \sigma_j} = \mathsf{H.Hash}(\mathsf{hk}, X^*(1)) = \cdots = \mathsf{H.Hash}(\mathsf{hk}, X^*(K)) \ .$$

Indeed, if for all exhibited label $X^*_{\sigma_1 \ldots \sigma_j}$, there are fewer than K preimages, then since the depth of the tree is d, the overall number of exhibited leafs X^*_σ is smaller than K^d, in contrast to our assumption.

Thus, we can find such collisions with the same probability as that of failure. Furthermore, the time to find such a collision is proportional to the extractor's running time which is bounded by

$$\mathrm{poly}(K', \varepsilon^{-1}) \leq \mathrm{poly}(\gamma) \ ,$$

since $K' \leq K^d$, for $d \leq \bar{d}$, $K(\lambda, |z_\lambda|) = K(\lambda, \lambda^{O(1)}) = K^{O(1)}(\lambda, \lambda)$, and $\gamma(\lambda) = K^{\bar{d}}(\lambda, \lambda)/\underline{\varepsilon}$.

The claim now follows from the (K, γ)-collision resistance of H. □

To complete the proof of the proposition, we bound the probability that the adversary answers inconsistently with the extracted list.

Claim 4.2. *For all $\lambda \in \mathbb{N}$,*

$$p := \Pr\left[\begin{array}{l|l} \mathsf{HT.Ver}(\mathsf{hk}, L, \mathsf{dig}, \sigma, A, \Pi) = 1 \\ A \notin S & \begin{array}{l}\mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda) \\ (\mathsf{dig}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (A, \Pi) \leftarrow \mathcal{A}_2(\mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2(\mathsf{st})}(\sigma, 1^{K'}, 1^{1/\varepsilon})\end{array}\end{array}\right] \leq \varepsilon/2 \ .$$

Proof. We rely on Fact 2.1. The extractor makes $t = \frac{2K'}{\varepsilon}$ samples (A, Π) to construct the set S, and adds to S the values A with an accepting proof Π. By Fact 2.1 and using the previous claim:

$$p \leq \frac{\mathbb{E}[S]}{t} \leq \frac{K' + \Pr\left[S > K'\right] \cdot t}{t} \leq \varepsilon/2 + \Pr\left[\mathsf{Ext} \text{ outputs } \texttt{fail}\right] \leq \varepsilon/2 + \mu \leq \varepsilon \ ,$$

where the last inequality holds for large enough λ. □

□

This concludes the proof of the proposition.

4.2 Ingredient II: Collision-Free Code

The second building block used in the construction of hash with local opening is a new coding scheme that has a certain collision-freeness property.

The code has the following local decoding flavor. To decode a location i of the input word, we read a small set of locations D of the codeword together with a small random set of *test locations* T. The test locations are independent of i and are only used to synchronize the decoding of different locations. In addition to decoding the value of the i-th location of the input word, we also verify consistency between the values in the locations given by D and those given by T. Collision freeness says the following: for every rectangle $\mathbf{S} = S_1 \times \cdots \times S_n$ of small enough cardinality, with high probability over the choice of T, for any location i and set D used to decode the location i, there are no partial codewords C and C' that collide in the following sense:

- Both C and C' are contained in the rectangle $\mathbf{S}$.
- In both C and C', the values in locations T and D satisfy the consistency test.
- C and C' agree on the locations T, but not on D.

In other words, with high probability an assignment to the test locations, completely fixes how any location is decoded, provided that the symbols read are always taken from the rectangle $\mathbf{S}$.

We now proceed to define and construct the code.

Syntax: A collision-free code is parameterized by

- An output alphabet Σ.
- An input length $L \in \mathbb{N}$ and output length $N \in \mathbb{N}$.
- A index set size function $\Phi(\cdot)$.
- A rectangle size function $\Delta(\cdot)$.

The code is associated with polynomial-time algorithms

$$\mathsf{CFC} = (\mathsf{CFC.Code}, \mathsf{CFC.Chal}, \mathsf{CFC.TestInd}, \mathsf{CFC.DecInd}, \mathsf{CFC.Dec}, \mathsf{CFC.Test})$$

with the following syntax:

- $C \leftarrow \mathsf{CFC.Code}(X)$: is a deterministic algorithm that takes a word $X \in \{0,1\}^L$ and outputs a codeword $C \in \Sigma^N$.

- $R \leftarrow \mathsf{CFC.Chal}(1^\tau)$: is a randomize algorithm that takes a challenge length parameter 1^τ and samples a challenge R.
- $T \leftarrow \mathsf{CFC.TestInd}(R)$: is a deterministic algorithm that takes the challenge R and outputs a set of indices $T \subseteq [N]$ of size $\Phi(\tau)$.
- $D \leftarrow \mathsf{CFC.DecInd}(R, i)$: is a deterministic algorithm that takes the challenge R and an index $i \in [L]$ and outputs a set of indices $D \subseteq [N]$ of size $\Phi(\tau)$.
- $b \leftarrow \mathsf{CFC.Test}(a)$: is a deterministic algorithm that takes an assignment $a : U \to \Sigma$ where $U \subseteq [N]$, and outputs a bit.
- $b \leftarrow \mathsf{CFC.Dec}(a, i)$: is a deterministic algorithm that takes an assignment $a : D \to \Sigma$ where $D \subseteq [N]$ and an index $i \in [L]$ and outputs a bit.

Definition 4.3 (Collision-Free Code)**.** *A collision-free code* $\mathsf{CFC} = (\mathsf{CFC.Code}, \mathsf{CFC.Chal}, \mathsf{CFC.TestInd}, \mathsf{CFC.DecInd}, \mathsf{CFC.Dec}, \mathsf{CFC.Test})$ *satisfies:*

Correctness of Decoding: *For every $\tau \in \mathbb{N}$, every word $X \in \{0,1\}^L$ and every index $i \in [L]$,*

$$\Pr\left[\mathsf{CFC.Dec}\left(\mathsf{CFC.Code}(X)|_D, i\right) = X_i \;\middle|\; \begin{array}{r} R \leftarrow \mathsf{CFC.Chal}(1^\tau) \\ D \leftarrow \mathsf{CFC.DecInd}(R, i) \end{array}\right] = 1 \ .$$

Correctness of Testing: *For every word $X \in \{0,1\}^L$ and every set of indices $U \subseteq [N]$*

$$\mathsf{CFC.Test}\left(\mathsf{CFC.Code}(X)|_U\right) = 1 \ .$$

Collision Freeness: *For every $\tau \in \mathbb{N}$ and every set $S \subset \Sigma$ of size at most $\Delta(\tau)$,*

$$\Pr\left[\begin{array}{l} \exists i \in [L] \ \exists a, a' : U \to S : \\ \quad D = \mathsf{CFC.DecInd}(R, i) \\ \quad D \cup T \subseteq U \\ \quad a|_T = a'|_T \\ \quad \mathsf{CFC.Dec}(a, i) \neq \mathsf{CFC.Dec}(a', i) \\ \quad \mathsf{CFC.Test}(a) = 1 \wedge \mathsf{CFC.Test}(a') = 1 \end{array} \;\middle|\; \begin{array}{r} \\ \\ R \leftarrow \mathsf{CFC.Chal}(1^\tau) \\ T = \mathsf{CFC.TestInd}(R) \\ \\ \\ \\ \end{array}\right] \leq \frac{1}{2} \ .$$

4.2.1 Construction

We prove the following theorem:

Theorem 4.4. *For any two integers $m \in \mathbb{N}, h \geq 2$ there exists a collision-free code*

$$\mathsf{CFC}_{m,h} = (\mathsf{CFC.Code}_{m,h}, \mathsf{CFC.Chal}_{m,h}, \mathsf{CFC.TestInd}_{m,h}, \mathsf{CFC.DecInd}_{m,h}, \mathsf{CFC.Dec}_{m,h}, \mathsf{CFC.Test}_{m,h}) \ .$$

With the following parameters

$$|\Sigma| \leq h^{3h} \quad , \quad L = h^m \quad , \quad N \leq m \cdot h^{3(m-1)} \quad , \quad \Phi(\tau) = m \cdot \tau^m \quad , \quad \Delta(\tau) = h^{\tau/2} \ .$$

The Code $C \leftarrow \mathsf{CFC.Code}(X)$:
For any two integers $m \in \mathbb{N}, h \geq 2$, we construct a code $\mathsf{CFC}_{m,h}$ as follows. Let $\mathbb{F}$ be a field such that $2h^2 \leq |\mathbb{F}| \leq h^3$. We set

$$\Sigma = \mathbb{F}^h \quad , \quad L = h^m \quad , \quad N = m \cdot |\mathbb{F}|^{m-1} \quad , \quad \Phi(\tau) = m \cdot \tau^m \quad , \quad \Delta(\tau) = h^{\tau/2} \ .$$

The algorithm $\mathsf{CFC.Code}_{m,h}$ takes as input a word $X \in \mathbb{F}^L$, and computes a codeword as follows:

- **Low-Degree Extension:** Let $H \subseteq \mathbb{F}$ be a subset of size h. Recalling that $L = h^m$, we identify the set of indices $[L]$ with the cube H^m. Accordingly, the input word is associated with a function $X : H^m \to \{0,1\}$.

  Let $P_X : \mathbb{F}^m \to \mathbb{F}$ be the low-degree extension of X to $\mathbb{F}^m$. That is, P_X is the m-variate polynomial of individual degree $h-1$ such that $P_X|_{H^m} = X$.

- **Restriction to Axis-Parallel Lines:** For every $v \in \mathbb{F}^m$ and $j \in [m]$ let $\gamma_v^j : \mathbb{F} \to \mathbb{F}^m$ be the line passing through v parallel to the standard basis vector e_j:

  $$\gamma_v^j(\beta) = (v_1, \ldots, v_{j-1}, \beta, v_{j+1}, \ldots, v_m) \ .$$

  Let $\Gamma = \left\{ \gamma_v^j \;\middle|\; v \in \mathbb{F}^m, j \in [m] \right\}$ be the set of all axis parallel lines. Note that each line in Γ has $\mathbb{F}$ different representations. The total size of Γ is $m \cdot |\mathbb{F}|^{m-1}$. We identify the set of indices $[N]$ with the set Γ. The code outputs the restrictions of P_X to lines in Γ.

  $$\mathsf{CFC.Code}_{m,h}(X) = (P_X(\gamma) \mid \gamma \in \Gamma) \ .$$

  Note the every restriction $P_X(\gamma)$ is a univariate polynomial of degree $h-1$ and therefore, can be described by an element of $\Sigma = \mathbb{F}^h$, for example, by listing the evaluations of the polynomial on the set H. We identify the set Σ with the set of all univariate polynomial of degree $h-1$.

We now describe the other associated algorithms:

- $R \leftarrow \mathsf{CFC.Chal}_{m,h}(1^\tau)$**:**
  takes a parameter 1^τ, samples τ independently uniform elements $\beta_1, \ldots, \beta_\tau \leftarrow \mathbb{F}$ and outputs the challenge $R = \{\beta_1, \ldots, \beta_\tau\}$.

- $T \leftarrow \mathsf{CFC.TestInd}_{m,h}(R)$**:**
  takes the challenge $R \subseteq \mathbb{F}$ and outputs the following set of $\Phi(\tau)$ indices in $[N] \cong \Gamma$:

  $$T = \left\{ \gamma_v^j \;\middle|\; v \in R^m, j \in [m] \right\} \ .$$

- $D \leftarrow \mathsf{CFC.DecInd}_{m,h}(R, u)$**:**
  takes the challenge $R \subseteq \mathbb{F}$ and an index $u \in H^m \cong [L]$. For $v \in R^m$, $j \in [m]$ let $\gamma_{v,u}^j$ denote the line

  $$\gamma_{v,u}^j = \gamma_{v_1,\ldots,v_j,u_{j+1},\ldots,u_m}^j \ .$$

  The algorithm outputs the set of $\Phi(\tau)$ indices

  $$D = \left\{ \gamma_{v,u}^j \;\middle|\; v \in R^m, j \in [m] \right\} \ .$$

- $b \leftarrow \mathsf{CFC.Dec}_{m,h}(a,u)$**:**
  The algorithm takes an assignment $a : D \to \Sigma$ and an index $u \in H^m \cong [L]$. If $\gamma_u^1 \in D$ and $a(\gamma_u^1)(u_1) \in \{0,1\}$ the algorithm outputs $a(\gamma_u^1)(u_1)$. Otherwise, the algorithm fails.

- $b \leftarrow \mathsf{CFC.Test}_{m,h}(a)$**:**
  The algorithm takes an assignment $a : U \to \Sigma$ for $U \subseteq \Gamma$. If there exists $\gamma, \gamma' \in U$ and $\beta, \beta' \in \mathbb{F}$ such that
  $$\gamma(\beta) = \gamma'(\beta') \quad \wedge \quad a(\gamma)(\beta) \neq a(\gamma')(\beta') \ ,$$
  the algorithm outputs 0, otherwise it outputs 1.

Remark 4.7 (Public Coins). The challenge algorithm $\mathsf{CFC.Chal}$ is public-coin — it simply outputs random field elements. The output of both $\mathsf{CFC.TestInd}(R), \mathsf{CFC.DecInd}_{m,h}(i,R)$ is deterministically fixed given the coins R.

It is straightforward to verify that the construction satisfies the correctness properties. Next we prove collision-freeness.

Collision Freeness. To prove this, we show that whenever collision freeness is violated with respect to a small set $S \subset \Sigma$, there must exist two distinct polynomials in S that agree on all challenge points in $R \subset \mathbb{F}$:

Claim 4.3. *Fix an index $u \in H^m$, a set $S \subset \Sigma$, and challenge $R \subset \mathbb{F}$. Let*

$$T \leftarrow \mathsf{CFC.TestInd}(R) \quad , \quad D \leftarrow \mathsf{CFC.DecInd}(R,u) \ ,$$

and let $a, a' : U \to S$ be a pair of assignments such that $T \cup D \subseteq U$ and

$$a|_T = a'|_T \tag{1}$$
$$\mathsf{CFC.Dec}(a,u) \neq \mathsf{CFC.Dec}(a',u) \tag{2}$$
$$\mathsf{CFC.Test}(a) = 1 \wedge \mathsf{CFC.Test}(a') = 1 \tag{3}$$

Then there exists $\gamma \in U$ such that the univariate polynomials $a(\gamma), a'(\gamma) \in S$ are distinct and agree on R.

Before proving the claim, we show that it indeed implies collision freeness. Recall that $\mathsf{CFC.Chal}_{m,h}(1^\tau)$ samples a challenge set R that contains τ independently random elements in $\mathbb{F}$, and that Σ contains univariate polynomials of degree $h-1$. Thus any two distinct polynomials in $S \subseteq \Sigma$ agree on R with probability at most $\left(\frac{h-1}{|\mathbb{F}|}\right)^\tau \leq (h/2)^{-\tau}$. Assuming the set S is of size at most $\Delta(\tau)$, we can take a union bound over all such pairs to deduce collision freeness:

$$\Pr\left[\begin{array}{l|r} \exists i \in [L] \ \exists a, a' : U \to S : & \\ \quad D = \mathsf{CFC.DecInd}(R,i) & \\ \quad D \cup T \subseteq U & \\ \quad a|_T = a'|_T & R \leftarrow \mathsf{CFC.Chal}(1^\tau) \\ \quad \mathsf{CFC.Dec}(a,i) \neq \mathsf{CFC.Dec}(a',i) & T = \mathsf{CFC.TestInd}(R) \\ \quad \mathsf{CFC.Test}(a) = 1 \wedge \mathsf{CFC.Test}(a') = 1 & \end{array}\right] \leq \Delta^2(\tau) \cdot \left(\frac{h}{2}\right)^{-\tau} \leq 2^{-\tau} \leq \frac{1}{2} \ .$$

It is left to prove Claim 4.3.

Proof of Claim 4.3. Recall that

$$T = \left\{\gamma_v^j\right\}_{v\in R^m, j\in[m]} \quad , \quad D = \left\{\gamma_{v,u}^j = \gamma_{v_1,\ldots,v_j,u_{j+1},\ldots,u_m}^j\right\}_{v\in R^m, j\in[m]} \ .$$

For every $v \in R^m$, we have by definition that $\gamma_{v,u}^m = \gamma_v^m$. By the fact that a and a' agree on T (Equation (1)), we deduce that $a(\gamma_{v,u}^m) = a'(\gamma_{v,u}^m)$. Since decoding a, a' at u gives different results (Equation (2)), and since $\gamma_u^1 = \gamma_{v,u}^1$ by definition, we have that $a(\gamma_{v,u}^1) \neq a'(\gamma_{v,u}^1)$. Therefore, there exists $j \in [m-1]$ and $w \in R^m$ such that

$$a(\gamma_{w,u}^j) \neq a'(\gamma_{w,u}^j) \quad \wedge \quad \forall v \in R^m : a(\gamma_{v,u}^{j+1}) = a'(\gamma_{v,u}^{j+1}) \ . \tag{4}$$

For every $\beta \in R$ let $w_\beta = \gamma_v^j(\beta) \in R^m$. Then by definition,

$$\gamma_{w,u}^j(\beta) = \gamma_{w_\beta,u}^{j+1}(u_{j+1}) \ .$$

Therefore, since the assignments a, a' are valid (Equation (3)) and combining with Equation (4), we have

$$a(\gamma_{w,u}^j)(\beta) = a(\gamma_{w_\beta,u}^{j+1})(u_{j+1}) = a'(\gamma_{w_\beta,u}^{j+1})(u_{j+1}) = a'(\gamma_{w,u}^j)(\beta) \ .$$

That is, $a(\gamma_{w,u}^j)$ and $a'(\gamma_{w,u}^j)$ are distinct and agree on every $\beta \in R$. □

4.3 Construction

We now move on to construct a multi-collision-resistant hash with local opening:

$$\mathsf{HLO} = (\mathsf{HLO.Gen}, \mathsf{HLO.Hash}, \mathsf{HLO.Chal}, \mathsf{HLO.Auth}, \mathsf{HLO.Ver}) \ .$$

We use the following two building blocks:

- A hash tree (as defined and constructed in Section 4.1):

  $$\mathsf{HT} = (\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Auth}, \mathsf{HT.Ver}) \ .$$

- A collision-free code (as defined and constructed in Section 4.2):

  $\mathsf{CFC}_{m,h} = (\mathsf{CFC.Code}_{m,h}, \mathsf{CFC.Chal}_{m,h}, \mathsf{CFC.TestInd}_{m,h}, \mathsf{CFC.DecInd}_{m,h}, \mathsf{CFC.Dec}_{m,h}, \mathsf{CFC.Test}_{m,h})$.

  (The parameters m, h will be set below.)

We now describe the required algorithms:

- $\mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda)$**:**
  takes the security parameter 1^λ and outputs a key for a hash tree $\mathsf{hk} \leftarrow \mathsf{HT.Gen}(1^\lambda)$. (We assume throughout that hk includes the security parameter 1^λ in the clear.)

- $\mathsf{dig} \leftarrow \mathsf{HLO.Hash}(\mathsf{hk}, X)$**:**
  takes the key hk and an input $X \in \{0,1\}^L$. It proceeds as follows

– Let $h = \lambda^{.9}, m = \log_h L$. Recall that the code $\mathsf{CFC}_{m,h}$ maps words in $\{0,1\}^{h^m} = \{0,1\}^L$ to codewords in Σ^N where:

$$|\Sigma| \leq h^{3h} = 2^{o(\lambda)} \quad , \quad N \leq m \cdot h^{3(m-1)} = o(L^3) \ .$$

– Parse the codeword $C = \mathsf{CFC.Code}_{m,h}(X) \in \Sigma^N$ as a sequence of N blocks in $\{0,1\}^\lambda$,[7] hash the codeword and output the digest $\mathsf{dig} = \mathsf{HT.Hash}(\mathsf{hk}, C)$.

- $\mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho)$**:**
takes the security parameter 1^λ and the opening size 1^ρ. For every $j \in [\lambda \cdot \rho]$ and every $\tau \in [\bar{\tau}]$ it samples $R_{j,\tau} \leftarrow \mathsf{CFC.Chal}_{m,h}(1^\tau)$. It outputs the challenge

$$\mathsf{ch} = \{R_{j,\tau}\}_{j \in [\lambda \cdot \rho], \tau \in [\bar{\tau}]} \ .$$

Here $\bar{\tau} = \bar{\tau}(\lambda)$ is a parameter that bounds the challenge length. We show how to exactly set this parameter later on.

- $\Pi \leftarrow \mathsf{HLO.Auth}(\mathsf{hk}, X, I, \mathsf{ch})$**:**
takes the key hk, input $X \in \{0,1\}^L$, an index set $I \subseteq [L]$ and a challenge $\mathsf{ch} = \{R_{j,\tau}\}$. It proceeds as follows

– Obtains the codeword

$$C = \mathsf{CFC.Code}_{m,h}(X) \in \{0,1\}^{\lambda \times N} \ .$$

– For every $j \in [\lambda \cdot \rho]$, every $\tau \in [\bar{\tau}]$ and every $i \in I$ it obtain the index sets

$$T_{j,\tau} = \mathsf{CFC.TestInd}_{m,h}(R_{j,\tau}) \quad , \quad D_{i,j,\tau} = \mathsf{CFC.DecInd}_{m,h}(R_{j,\tau}, i) \ .$$

– Let U be the set of all indexes obtained

$$U = \bigcup_{\substack{i \in I, j \in [\lambda \cdot \rho], \\ \tau \in [\bar{\tau}]}} T_{j,\tau} \cup D_{i,j,\tau} \ .$$

– Output a proof Π that contains the set U, the restriction $C|_U$, and proofs of consistency with the digest dig.

$$\Pi = \left(U, C|_U, \{\Pi_u = \mathsf{HT.Auth}(\mathsf{hk}, C, u)\}_{u \in U}\right) \ .$$

- $b \leftarrow \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi)$**:**
takes the key hk, an input length $L \in \mathbb{N}$, the digest $\mathsf{dig} \in \{0,1\}^\lambda$, the index set I, and an assignment $A : I \to \{0,1\}$, as well as a challenge $\mathsf{ch} = \{R_{j,\tau}\}$ and a corresponding proof $\Pi = \left(U, C|_U, \{\Pi_u\}_{u \in U}\right)$. It accepts if the following conditions are satisfied:

1. *Hash tree consistency:* for every $u \in U$: $\mathsf{HT.Ver}(\mathsf{hk}, N, \mathsf{dig}, u, C|_u, \Pi_u) = 1$.
2. *Code consistency:* for every $R_{j,\tau} \in \mathsf{ch}$ and every $i \in I$:
   (a) Let $T = \mathsf{CFC.TestInd}_{m,h}(R_{j,\tau})$ and $D = \mathsf{CFC.DecInd}_{m,h}(R_{j,\tau}, i)$
   (b) $T \cup D \subseteq U$.

[7] Note that $|\Sigma| \leq 2^\lambda$ and can thus be efficiently represented by λ-bit strings.

(c) $\mathsf{CFC.Test}_{m,h}(C|_{T\cup D}) = 1$.
(d) $\mathsf{CFC.Dec}_{m,h}(C|_D, i) = A(i)$.

In the next sections, we prove the following theorem:

Theorem 4.5 (From Hash Tree to Hash with Local Opening)**.** *Let:*

- $\bar{L} = \bar{L}(\lambda)$ *be an input-length bound.*
- HT *be* K*-collision resistant with input-length bound* $\bar{L}$ *and accuracy bound* $\underline{\varepsilon}(\lambda) = 1/\bar{L}$.
- $K'(\lambda, \zeta, L) \geq \lambda^{(\ell+k)^{5\ell}}$ *be a collision bound where* $\ell = \log_\lambda L$ *and* $k = \log_\lambda K(\lambda, \zeta, L)$.
- $\bar{\tau}(\lambda) = 7(\bar{\ell} + \bar{k})$ *be a bound on the challenge length, where* $\bar{\ell} = \log_\lambda \bar{L}$ *and* $\bar{k} = \log_\lambda K(\lambda, \lambda^{\omega(1)}, \bar{L})$.
- *Assume* $(\bar{\ell} + \bar{k})^{\bar{\ell}} \leq \lambda$.

Then HLO *with challenge length bound* $\bar{\tau}$ *is* K'*-collision resistance with length bound* $\bar{L}$.

Remark 4.8 (Parameters). In the definition of $\bar{k}$, the function $\omega(1)$ is an arbitrarily small super-constant, so that throughout $\bar{k} = \log_\lambda K(\lambda, \lambda^{\omega(1)}, \bar{L})$ is a bound on $k = \log_\lambda K(\lambda, \zeta, L)$ for any polynomial advice-size $\zeta(\lambda) = \lambda^{O(1)}$ and length $L(\lambda) \leq \bar{L}(\lambda)$.

Combining Theorem 4.5 and Theorem 4.3 (regarding the existence of hash trees based on multi-collision resistant hash functions), we obtain:

Corollary 4.1 (From Weak Multi-Collision-Resistant Hash to Hash with Local Opening)**.** *Let:*

- $\bar{L} = \bar{L}(\lambda)$ *be a bound on the input length.*
- $K(\lambda, \zeta)$ *be a collision bound, polynomial in* ζ.
- H *be a weakly* (K, γ)*-collision-resistant hash such that* $\gamma(\lambda) = \lambda^{\bar{k}\cdot\bar{\ell}}$ *for* $\bar{\ell} = \log_\lambda \bar{L}$ *and* $\bar{k} = \log_\lambda K(\lambda, \lambda)$.
- $K''(\lambda, \zeta, L) \geq \lambda^{(\ell k)^{10\ell}}$ *be a collision bound where* $\ell = \log_\lambda L$ *and* $k = \log_\lambda K(\lambda, \zeta)$.
- *Assume* $\left(\bar{k}\bar{\ell}\right)^{2\bar{\ell}} \leq \lambda$.

Then there exists a multi-collision-resistant hash with local opening that is K''*-collision resistant with length bound* $\bar{L}$.

Proof. By Theorem 4.3, given a multi-collision resistant H as above, there exists a hash tree HT (of arity $\alpha = \lambda$) that is K'-collision resistant for collision bound $K'(\lambda, \zeta, L) = \lambda^{k\cdot\ell}$, input-length bound $\bar{L}$, and accuracy bound $1/\bar{L}$.

Letting $k' = \log_\lambda K'(\lambda, \zeta, L) = k\cdot\ell$ and $\ell' = \ell$, and noting that $(\ell k)^{2\ell} \geq (\ell' + k')^{\ell'}$, it follows from Theorem 4.5 that there exists a K''-collision resistant hash with local opening for $K''(\lambda, \zeta, L) \geq \lambda^{(\ell k)^{10\ell}} \geq \lambda^{(\ell'+k')^{5\ell'}}$.

□

4.4 Proof of Theorem 4.5

In this section, we prove Theorem 4.5. We start by noting that the construction has the required correctness and succinctness, and then move on to the main part of the proof which is demonstrating and analysing an extractor.

The correctness of the construction follows readily from that of the hash tree and the collision-free code. We now note that the construction has the required succinctness.

Succinctness. We need to show that the proof size is $|\Pi| = \mathrm{poly}(\lambda, \rho, |I|)$. Recall that proof is of the form

$$\Pi = \left(U, C|_U, \{\Pi_u = \mathsf{HT.Auth}(\mathsf{hk}, C, u)\}_{u \in U}\right) \ ,$$

where

$$U = \bigcup_{\substack{i \in I, j \in [\lambda \cdot \rho], \\ \tau \in [\bar{\tau}]}} U_{i,j,\tau} \qquad \text{and} \qquad U_{i,j,\tau} = T_{j,\tau} \cup D_{i,j,\tau} \ .$$

Fix any i, j, τ. Then

$$|U_{i,j,\tau}, C|_{U_{i,j,\tau}}| \leq O(\Phi(\tau) \cdot \log|\Sigma|) \leq O(m \cdot \bar{\tau}^m \cdot \lambda) \leq \frac{10\bar{\ell}}{9} \cdot (7\bar{\ell} + 7\bar{k})^{\frac{10\bar{\ell}}{9}} \cdot O(\lambda) \leq (\bar{\ell} + \bar{k})^{3\bar{\ell}} \cdot O(\lambda)$$

and

$$|\{\mathsf{HT.Auth}(\mathsf{hk}, C, u)\}_{u \in U_{i,j,\tau}}| = |C|_{U_{i,j,\tau}}| \cdot \mathrm{poly}(\lambda) \leq (\bar{\ell} + \bar{k})^{3\bar{\ell}} \cdot \mathrm{poly}(\lambda) \ .$$

So overall the size of the proof is

$$|I| \cdot \lambda \cdot \rho \cdot \bar{\tau} \cdot |U_{i,j,\tau}, C|_{U_{i,j,\tau}}|, |\{\mathsf{HT.Auth}(\mathsf{hk}, C, u)\}_{u \in U_{i,j,\tau}}| \leq \mathrm{poly}(\lambda, |I|, \rho) \cdot (\bar{\ell} + \bar{k})^{4\bar{\ell}} \leq \mathrm{poly}(\lambda, |I|, \rho) \cdot \lambda^4 \ .$$

4.4.1 The Extractor

We now demonstrate an appropriate extractor as per Definition 4.1.

The extractor $\mathsf{Ext}^{\mathcal{A}_2(\cdot, \mathsf{st})}(1^\lambda, 1^\rho, 1^L, 1^{K'}, 1^{1/\varepsilon'})$ first derives the following parameters:

- N - the output length of the code.
- $\ell = \log_\lambda L$.
- K - the collision resistance parameter of the underlying hash tree HT.
- $k = \log_\lambda K$.

Step 1: Extracting a Rectangle. In the first step, the extractor obtains a (small) rectangle $\mathbf{S}^{\mathsf{HT}} = S_1^{\mathsf{HT}} \times \cdots \times S_N^{\mathsf{HT}}$ such that the answers of $\mathcal{A}_2$ are contained in $\mathbf{S}^{\mathsf{HT}}$ with high probability. (This part relies on the hash tree.)

Concretely, for every $i \in [N]$, the extractor first constructs an adversary $\mathcal{A}_{2,i}$ that produces openings for the i-th block of the hash tree. $\mathcal{A}_{2,i}$, given oracle access to $\mathcal{A}_2(\cdot;\mathsf{st})$, samples a challenge $\mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho)$, queries $\mathcal{A}_2(\cdot;\mathsf{st})$ with ch, and obtains an opening (I, A, Π). If the proof Π is not of the correct form $(U, C|_U, \{\Pi_u\})$, or if $i \notin U$, $\mathcal{A}_{2,i}$ fails and outputs $\perp$. Otherwise, $\mathcal{A}_{2,i}$ outputs $(C|_i, \Pi_i)$.

Let $\mathsf{HT.Ext}$ be the extractor for the hash tree HT guaranteed by Definition 4.2. Ext invokes:

$$S_i^{\mathsf{HT}} \leftarrow \mathsf{HT.Ext}^{\mathcal{A}_{2,i}}(i, 1^K, 1^\varepsilon) \ ,$$

with the adversary $\mathcal{A}_{2,i}$, collision bound K, and accuracy parameter $\varepsilon = \varepsilon'/2N$. It obtains a set $S_i^{\mathsf{HT}} \subseteq \Sigma$ of size at most K. The extracted rectangle is the product set

$$\mathbf{S}^{\mathsf{HT}} = S_1^{\mathsf{HT}} \times \cdots \times S_N^{\mathsf{HT}} \subseteq \Sigma^N \ .$$

In what follows, we say that an assignment $a : U \to \Sigma$ is consistent with $\mathbf{S}^{\mathsf{HT}}$, if for all $i \in U$, $a(i) \in S_i$.

Step 2: Extracting the List of Words. In the second step, given the rectangle $\mathbf{S}^{\mathsf{HT}}$, the extractor obtains a corresponding list of (global) words satisfying Definition 4.1. (This part relies on the collision-free code.)

For this purpose, we define a procedure $\mathsf{CFC.Ext}(R, \mathbf{S}^{\mathsf{HT}})$ that given a challenge R sampled by $\mathsf{CFC.Chal}_{m,h}$ and the rectangle $\mathbf{S}^{\mathsf{HT}}$, outputs a set of words $S_R \subseteq \{0,1\}^L$. Intuitively, this procedure considers assignments $a : U \to \Sigma$ that are consistent with the rectangle $\mathbf{S}^{\mathsf{HT}}$, and attempts to extend them to (partial) words, by decoding. These words are then added to S_R.

$\mathsf{CFC.Ext}(R, \mathbf{S}^{\mathsf{HT}})$ proceeds as follows:

- Let $S_R = \emptyset$ and let $T = \mathsf{CFC.TestInd}_{m,h}(R)$.
- For every assignment $a : T \to \Sigma$ that is consistent with $\mathbf{S}^{\mathsf{HT}}$:
  - For every $i \in [L]$, let $D = \mathsf{CFC.DecInd}_{m,h}(R, i)$.
    Mark the index i as consistent with the bit $b \in \{0,1\}$ if
    - There exists an assignment $a' : (T \cup D) \to \Sigma$ that is consistent with $\mathbf{S}^{\mathsf{HT}}$,
    - $a'|_T = a$,
    - $\mathsf{CFC.Test}_{m,h}(a') = 1$,
    - $\mathsf{CFC.Dec}_{m,h}(a'|_D, i) = b$.
  - For every $i \in [L]$ that has not been marked as consistent with either 0 or 1, mark i as consistent with 0 (here 0 is an arbitrary choice).
  - If for every $i \in [L]$, there exists a bit $b_i \in \{0,1\}$ such that i is marked as consistent with b_i, but not with $1 - b_i$, then add the word $b_1 \ldots b_L$ to S_R.
  - Otherwise (there exists $i \in [L]$ that is marked as consistent with both 0 and 1), mark that the challenge R has a collision.
- If R has a collision, output $\perp$. Otherwise, output the set S_R.

The extractor Ext now proceeds as follows:

- Let $\tau = 7(\ell + k)$.
- For every $j \in [\lambda \cdot \rho]$, sample a challenge R_j and extract a corresponding set of words:

$$R_j \leftarrow \mathsf{CFC.Chal}(1^\tau) \quad , \quad S_{R_j} = \mathsf{CFC.Ext}(R_j, \mathbf{S}^{\mathsf{HT}}) \ .$$

- Let J be the set of indexes j such that the challenge R_j did not have a collision:

$$J = \left\{ j \in [\lambda \cdot \rho] : S_{R_j} \neq \perp \right\} \quad .$$

- Output the union set $S = \bigcup_{j \in J} S_{R_j}$.

4.4.2 Analysis

In this section we prove that Ext satisfies Definition 4.1.

Bounding the Output Size and Running Time of the Extractor. We first show that Ext outputs a set S of size at most $K' := \lambda^{(\ell+k)^{5\ell}}$. We assume that $\ell \geq 2$. Consider the execution of any $\mathsf{CFC.Ext}(R_j, \mathbf{S}^{\mathsf{HT}})$. By the properties of the code $\mathsf{CFC}_{m,h}$, we have that for $T = \mathsf{CFC.TestInd}_{m,h}(R)$:

$$|T| \leq \Phi(\tau) = m \cdot \tau^m \leq \frac{10\ell}{9} \cdot (7\ell + 7k)^{\frac{10\ell}{9}} \leq (\ell + k)^{3\ell} \quad .$$

Since the sets $S_1^{\mathsf{HT}}, \ldots, S_N^{\mathsf{HT}}$ are each of size at most K, the number of assignments $a : T \to \Sigma$ that are consistent with $\mathbf{S}^{\mathsf{HT}}$ is at most

$$K^{|T|} \leq \left(\lambda^k\right)^{(\ell+k)^{3\ell}} \leq \lambda^{(\ell+k)^{4\ell}} \quad . \tag{5}$$

This is also a bound on the output set S_{R_j}, since for every such assignment, $\mathsf{CFC.Ext}$ adds at most one word to S_{R_j}. Ext outputs the union of at most $\lambda \cdot \rho \leq \lambda \cdot L = \lambda^{\ell+1}$ such sets, and therefore

$$|S| \leq \lambda^{\ell+1} \cdot \lambda^{(\ell+k)^{4\ell}} \leq \lambda^{(\ell+k)^{5\ell}} = K' \quad .$$

Finally, by inspection, one can see that the running time of the extractor is polynomial in the size of $|S|$ and its other inputs. Having already established that $|S| \leq K'$, it follows that the extractor runs in polynomial time in all of its inputs.

The Extractor is Successful. Fix a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, a polynomial-size advice sequence $\{z_\lambda\}_{\lambda \in \mathbb{N}}$, a noticeable function $\varepsilon' = \varepsilon'(\lambda)$, a length $L \leq \bar{L}^{O(1)}$, a large enough security parameter λ, and an opening size $\rho \leq L$.

The Experiment $\mathcal{E}$. We consider the following randomized experiment $\mathcal{E}$:

$$\begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda) \\ (\mathsf{dig}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ (I, A, \Pi) \leftarrow \mathcal{A}_2(\mathsf{ch}; \mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2(\cdot;\mathsf{st})}(1^\lambda, 1^\rho, 1^L, 1^{K'}, 1^{1/\varepsilon'}) \end{array} \quad .$$

We need to show that the adversary answers consistently with the extracted set:

$$\Pr_{\mathcal{E}} \left[\begin{array}{l} |I| \leq \rho \\ \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi) = 1 \\ A \notin \{X|_I : X \in S\} \end{array} \right] \leq \varepsilon' \quad . \tag{6}$$

In what follows:

- $\Pi = (U, C|_U, \{\Pi_u\})$ is the proof output by the adversary.
- $\mathbf{S}^{\mathsf{HT}} = S_1^{\mathsf{HT}} \times \cdots \times S_N^{\mathsf{HT}}$ is the extracted rectangle in the first step of the extraction.
- $S = \bigcup_{j \in J} S_j$ is the output of the extractor Ext.

The Adversary Respects the Rectangle $\mathbf{S}^{\mathsf{HT}}$. For every $i \in [N]$, invoking the K-collision resistance of the hash tree HT for the adversary the adversary $(\mathcal{A}_1, \mathcal{A}_{2,i}{}^{\mathcal{A}_2})$, we have that

$$\Pr_{\mathcal{E}}\left[\begin{array}{l} i \in U \\ \mathsf{HT.Ver}(\mathsf{hk}, N, \mathsf{dig}, i, C|_i, \Pi_i) = 1 \\ C|_i \notin S_i^{\mathsf{HT}} \end{array}\right] \leq \varepsilon = \frac{\varepsilon'}{2N} \ .$$

Note that $N = o(L^3) \leq \bar{L}^{O(1)}$ and $\varepsilon \geq \bar{L}^{-O(1)} = \underline{\varepsilon}^{\Omega(1)}$ as required to guarantee the above extraction.

Since the verifier $\mathsf{HLO.Ver}$ checks hash tree consistency (item 1 in the definition of $\mathsf{HLO.Ver}$), and taking a union bound, we have that

$$\Pr_{\mathcal{E}}\left[\begin{array}{l} \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi) = 1 \\ C|_U \text{ is not consistent with } \mathbf{S}^{\mathsf{HT}} \end{array}\right] \leq \frac{\varepsilon'}{2} \ .$$

Therefore, to establish Equation (6), and deduce successful extraction, it suffices to show:

$$\Pr_{\mathcal{E}}\left[\begin{array}{l} |I| \leq \rho \\ \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi) = 1 \\ C|_U \text{ is consistent with } \mathbf{S}^{\mathsf{HT}} \\ A \notin \{X|_I : X \in S\} \end{array}\right] \leq \frac{\varepsilon'}{2} \ . \tag{7}$$

Fixing the Set I. The number of sets $I \subseteq [L]$ such that $|I| \leq \rho$ is at most L^ρ. Therefore, to prove Equation (7) it suffices to show that for every fixed set $I \subseteq [L]$

$$\Pr_{\mathcal{E}}\left[\begin{array}{l} \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi) = 1 \\ C|_U \text{ is consistent with } \mathbf{S}^{\mathsf{HT}} \\ A \notin \{X|_I : X \in S\} \end{array}\right] \leq \frac{\varepsilon'}{2} \cdot L^{-\rho} \ . \tag{8}$$

For the rest of the proof, fix the set I.

Consistency of Assignment A with Extraction Relative to Challenge ch. We now show that had we performed extraction relative to the (code) challenges $\{R\}$ given by the challenge ch, then the adversary's assignment A would be consistent with the extracted set. We will then prove that except with small probability if the assignment is consistent with this (hypothetical) extracted set, it would also be consistent with the real extracted set (computed independently of the specific challenge ch.

In what follows:

- Let $\mathsf{ch} = \{R_{j,\tau}\}_{j \in [\lambda \cdot \rho], \tau \in [\bar{\tau}]}$ be the challenge in the experiment $\mathcal{E}$.
- Let $A : I \to \{0,1\}$ be the assignment and let $\Pi = (U, C|_U, \{\Pi_u\})$ be the proof, both produced by the adversary in $\mathcal{E}$.
- Fix $\tau = 7(\ell + k)$ (as fixed by the extractor Ext), and note that $\tau = 7(\ell + k) \leq 7(\bar{\ell} + \bar{k}) = \bar{\tau}$.

- For $j \in [\lambda \cdot \rho]$, let $S_j = \mathsf{CFC.Ext}(R_{j,\tau}, \mathbf{S}^{\mathsf{HT}})$ be the set of extracted words for the challenge $R_{j,\tau}$.
- Let $J = \{j \in [\lambda \cdot \rho] : S_j \neq \perp\}$ be the set of indexes j for which there is no collision.
- For every set S_j denote by $S_j|_I = \{X|_I : X \in S_j\}$ the same set of words when restricted to the locations I.

(Note that the extractor Ext in the experiment $\mathcal{E}$ does not necessarily invoke $\mathsf{CFC.Ext}(R_{j,\tau}, \mathbf{S}^{\mathsf{HT}})$. This is only for the sake of the analysis.)

Claim 4.4.

$$\Pr_{\mathcal{E}}\left[\begin{array}{l} \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, A, \mathsf{ch}, \Pi) = 1 \\ C|_U \textit{ is consistent with } \mathbf{S}^{\mathsf{HT}} \\ \exists j \in J : A \notin S_j|_I \end{array}\right] = 0 \ .$$

Proof. Assume toward contradiction that the above condition is violated for some $j \in J$. Let $T = \mathsf{CFC.TestInd}_{m,h}(R_{j,\tau})$. Since the verifier $\mathsf{HLO.Ver}$ checks code consistency (item 2 in the definition of $\mathsf{HLO.Ver}$) we have that $T \subseteq U$. Let $a = C|_T : T \to \Sigma$ and note that a is consistent with S^{HT}, since $C|_U$ is.

In the execution of $\mathsf{CFC.Ext}(R_{j,\tau}, \mathbf{S}^{\mathsf{HT}})$, consider the iteration corresponding to the assignment a. Recall that $j \in J$, and thus $R_{j,\tau}$ has no collisions imply that for every $i \in [L]$, there exists a bit $b_i \in \{0,1\}$ such that i is marked consistent with b_i but not with $1 - b_i$. The word $b_1 \ldots b_L$ is then added to the output set S_j. Since $A \notin S_j|_I$ (by the assumption toward contradiction), there exists $i \in I$ such that $A(i) \neq b_i$. Let $D = \mathsf{CFC.DecInd}_{m,h}(R, i)$. Again by the fact that $\mathsf{HLO.Ver}$ check code consistency (item 2 in its definition) we have that $D \subseteq U$ and

$$\mathsf{CFC.Test}_{m,h}(C|_{T \cup D}) = 1 \quad , \quad \mathsf{CFC.Dec}_{m,h}(C|_D, i) = A(i) \ ,$$

contradicting the fact that i was not marked as consistent with $A(i)$. □

We now show that, with high probability, the same consistency holds with respect to the actual extracted list.

In what follows:

- Let $R'_1, \ldots, R'_{\lambda \cdot \rho}$ be the challenges sampled by the extractor in the experiment $\mathcal{E}$.
- Let $S'_1, \ldots, S'_{\lambda \cdot \rho}$ be the corresponding extracted sets of words.
- Let J' is the set of indexes j such that the challenge R'_j did not have a collision.

Then by Claim 4.4, to prove Equation (8) and conclude the proof, it suffices to show that

$$\Pr_{\mathcal{E}}\left[\begin{array}{l} \forall j \in J : A \in S_j|_I \\ \forall j \in J' : A \notin S'_j|_I \end{array}\right] \leq \frac{\varepsilon'}{2} \cdot L^{-\rho} \ . \tag{9}$$

The Sets J and J' Are Large. Toward establishing the latter, we first bound from below the size of the sets J and J':

Claim 4.5.

$$\Pr_{\mathcal{E}}\left[\ |J| \geq \tfrac{\lambda \cdot \rho}{4} \quad \bigwedge \quad |J'| \geq \tfrac{\lambda \cdot \rho}{4} \ \right] \geq 1 - 2^{-\Omega(\lambda \cdot \rho)} \ .$$

Proof. Any assignment $a : U \to \Sigma$ that is consistent with $\mathbf{S}^{\mathsf{HT}}$ is supported on the set $\bigcup_{i\in[N]} S_i^{\mathsf{HT}}$ which is of size at most

$$N \cdot K \leq L^3 \cdot K = \lambda^{3\ell+k} \leq \left(\lambda^{.9}\right)^{7(\ell+k)/2} = h^{\tau/2} = \Delta(\tau) \ .$$

Therefore, for any $j \in [\lambda \cdot \rho]$, by the collision freeness property of the code $\mathsf{CFC}_{m,h}$

$$\Pr_{\mathcal{E}}\left[j \notin J\right] = \Pr_{\mathcal{E}}\left[\begin{array}{l} \exists i \in [L] \ \ \exists a, a' : U \to \Sigma : \\ \quad a, a' \text{ are consistent with } S^{\mathsf{HT}} \\ \quad T = \mathsf{CFC.TestInd}(R_{j,\tau}) \\ \quad D = \mathsf{CFC.DecInd}(R_{j,\tau}, i) \\ \quad D \cup T \subseteq U \\ \quad a|_T = a'|_T \\ \quad \mathsf{CFC.Dec}(a, i) \neq \mathsf{CFC.Dec}(a', i) \\ \quad \mathsf{CFC.Test}(a) = 1 \wedge \mathsf{CFC.Test}(a') = 1 \end{array}\right] \leq \frac{1}{2} \ .$$

Similarly, $\Pr_{\mathcal{E}}\left[j \notin J'\right] \leq 1/2$. Since the random variables $R_{1,\tau} \ldots, R_{\lambda\cdot\rho,\tau}$ and $R'_1 \ldots, R'_{\lambda\cdot\rho}$ are all independent the claim follows by a Chernoff bound. □

To prove Equation (9), we rely on the following basic fact.

Fact 4.1. *Let $X_1 \ldots, X_n$ and $X'_1 \ldots, X'_{n'}$ be independent and identically distributed random variables, where each one represents a subset from a universe U, and let $m = \min(n, n')$. Then*

$$\Pr\left[\exists x \forall j \in [m] : x \in X_j \wedge x \notin X'_j\right] \leq |U| \cdot 2^{-2m} \ .$$

Proof. Fix any $x \in U$, and let $p_x := \Pr[x \in X_1]$. Then

$$\Pr\left[\forall j \in [m] : x \in X_j \wedge x \notin X'_j\right] = (p_x(1-p_x))^m \leq 2^{-2m} \ .$$

The statement follows by a union bound over all $x \in U$. □

Now, consider the random variables

$$\left\{S_j|_I : j \in J\right\} \quad , \quad \left\{S'_j|_I : j \in J'\right\} \ .$$

For any $n, n' \in [\lambda \cdot \rho]$, conditioned on $|J| = n, |J'| = n'$, the above random variables are independent and idenitically distributed as follows

$$\left\{ S|_I \;\middle|\; \begin{array}{c} R \leftarrow \mathsf{CFC.Chal}(1^\tau) \\ S = \mathsf{CFC.Ext}(R, S^{\mathsf{HT}}) \\ S \neq \perp \end{array} \right\} \ .$$

Furthermore, these sets consists of elements from the universe $U = 2^I$, which is of size at most 2^ρ. Combining the above with Claim 4.5, we deduce:

$$\begin{aligned} &\Pr_{\mathcal{E}}\left[\begin{array}{l} \forall j \in J : A \in S_j|_I \\ \forall j \in J' : A \notin S'_j|_I \end{array}\right] \leq \\ &\Pr_{\mathcal{E}}\left[\begin{array}{l} \forall j \in J : A \in S_j|_I \\ \forall j \in J' : A \notin S'_j|_I \end{array} \;\middle|\; \min\left(|J|, |J'|\right) \geq \frac{\lambda\cdot\rho}{4}\right] + \Pr_{\mathcal{E}}\left[\min\left(|J|, |J'|\right) < \frac{\lambda\cdot\rho}{4}\right] \leq \\ &2^\rho \cdot 2^{-\lambda\cdot\rho/4} + 2^{-\Omega(\lambda\cdot\rho)} \leq \\ &2^{-\Omega(\lambda\cdot\rho)} \leq \lambda^{-O(1)} \cdot 2^{-\ell\rho} = \frac{\varepsilon'}{2} \cdot L^{-\rho} \ , \end{aligned}$$

where the last inequality holds since $\ell \leq \bar{\ell} = o(\lambda)$ (recall that $\bar{\ell}^{\bar{\ell}} \leq \lambda$).

This complete the analysis of the extraction procedure.

5 3-Message Succinct Arguments for NP

In this section, we show how to use multi-collision-resistant hash functions with local opening, to construct succinct argument systems for non-deterministic computations, and in particular for **NP**. In Section 5.1, we recall the definition of such argument systems. In Section 5.2, we recall probabilistically-checkable proofs, which are used in the construction. The construction itself is described and analyzed in Section 5.3.

5.1 Succinct Arguments for Non-Deterministic Computations

We recall the definition of succinct arguments for non-deterministic computations [Kil92, BG08].

The Universal Relation. The universal relation $\mathcal{R}_{\mathcal{U}}$ consists of pairs (u, w) where $u = (M, x, t)$, and M is a description of a Turing machine such that $M(x, w)$ accepts within t steps.

Definition 5.1 (Succinct Arguments)**.** *A succinct argument system for the universal relation $\mathcal{R}_{\mathcal{U}}$ is given by the pair of interactive PPT algorithms (P, V) satisfying the following requirements:*

Efficient Verifier and Relatively-Efficient Prover: *There exists a universal polynomial $p(\cdot)$ such that for every $u = (M, x, t) \in \{0,1\}^{\lambda} \cap \mathcal{L}(\mathcal{R}_{\mathcal{U}})$, where $t \leq 2^{\lambda}$, and every $w \in \mathcal{R}_{\mathcal{U}}(u)$:*

- *The prover $\mathsf{P}(u, w)$ runs in time $p(\lambda, t)$.*
- *The verifier $\mathsf{V}(u)$ runs in time $p(\lambda)$.*

Completeness: *For every $u = (M, x, t) \in \{0,1\}^{\lambda} \cap \mathcal{L}(\mathcal{R}_{\mathcal{U}})$, where $t \leq 2^{\lambda}$, and every $w \in \mathcal{R}_{\mathcal{U}}(u)$:*

$$\Pr\left[\langle \mathsf{P}(w) \leftrightarrows \mathsf{V} \rangle(x) = 1\right] = 1 \ .$$

Proof of Knowledge for Computation-Time Bound $\bar{t}(\lambda)$ with $\tau(\lambda)$-Time Extractor: *There exists an extractor E such that for any noticeable function $\varepsilon(\lambda) = \lambda^{-O(1)}$, any PPT prover P^* with polynomial-size advice $\{z_\lambda\}_\lambda$, any $t(\lambda) \leq \bar{t}^{O(1)}$, any large enough $\lambda \in \mathbb{N}$, and any (M, x) such that $u = (M, x, t) \in \{0,1\}^{\lambda}$,*

$$\text{if} \quad \Pr\left[\langle \mathsf{P}^* \leftrightarrows \mathsf{V} \rangle(u)\right] \geq \varepsilon, \quad \text{then} \quad \Pr\left[\mathcal{R}_{\mathcal{U}}(u) \ni w \leftarrow \mathsf{E}^{\mathsf{P}^*}(u; 1^{1/\varepsilon}, |z_\lambda|)\right] \geq 1 - 2^{-\Omega(\lambda)} \ .$$

The extractor runs in time $\tau(\lambda, t, \varepsilon^{-1}, |z_\lambda|)$.

We now state the main theorem proved in this section regarding the existence of succinct arguments (according to the above definition) based on weak multi-collision-resistant hash functions (Definition 3.2). The theorem has two parts. The first is a polynomial version that guarantees security for computations of arbitrary polynomial time, based on polynomial assumptions. The second is a super-polynomial version that guarantees security even for slightly super-polynomial computations, relying on slightly super-polynomial assumptions.

Theorem 5.1 (Succinct Arguments for Non-Deterministic Computations)**.**

- *Assuming a weakly K-collision-resistant hash for $K(\lambda,\zeta) = \mathrm{poly}(\lambda,\zeta)$, there exist succinct arguments for any computation-time bound $\bar{t}(\lambda) = \lambda^{O(1)}$, with polynomial extraction time $\tau(\lambda) = \lambda^{O(1)}$.*
- *For any (arbitrary small) $\tau(\lambda) = \omega(1)$, there exists $t(\lambda) = \lambda^{\omega(1)}$ such that assuming a weakly (K,γ)-collision-resistant hash for $K(\lambda,\zeta) = \mathrm{poly}(\lambda,\zeta)$ and $\gamma(\lambda) = \lambda^{\tau}$, there exist succinct arguments for computation-time bound $\bar{t}$, with extraction time $\tau(\lambda) = \gamma^{O(1)}$.*

If the underlying hash functions are unkeyed, the argument has 3 messages (and 4 otherwise).

In the next sections, we describe the construction behind the theorem, and its building blocks. Eventually, the theorem is derived as a corollary of a more general Theorem 5.2 proven in Section 5.3.

5.2 Probabilistically-Checkable Proofs

A (verifier-efficient) *probabilistically checkable proof* (PCP) for the universal relation $\mathcal{R}_{\mathcal{U}}$ consists of polynomial-time algorithms $(\mathsf{PCP.P}, \mathsf{PCP.V})$ with the following syntax:

- $\pi \leftarrow \mathsf{PCP.P}(u,w)$: takes a pair $(u,w) \in \mathcal{R}_{\mathcal{U}}$ and outputs a proof string π.
- $b \leftarrow \mathsf{PCP.V}^{\pi}(u)$: takes the instance u and makes queries into the string π. It output a bit b.

Definition 5.2 (PCP)**.** *A PCP for the universal relation $\mathcal{R}_{\mathcal{U}}$ $\mathsf{PCP} = (\mathsf{PCP.P}, \mathsf{PCP.V})$ satisfies:*

Efficient Verifier and Relatively-Efficient Prover: *There exists a universal polynomial $p(\cdot)$ such that for every $(u,w) = ((M,x,t),w) \in \mathcal{R}_{\mathcal{U}}$:*

- *The prover $\mathsf{PCP.P}(u,w)$ runs in time $p(|u|,t)$.*
- *The verifier $\mathsf{PCP.V}^{(\cdot)}(u)$ runs in time $p(|u|)$.*

Completeness: *For every $(u,w) \in \mathcal{R}_{\mathcal{U}}$, and $\pi \leftarrow \mathsf{PCP.P}(u,w)$:*

$$\Pr\left[\mathsf{PCP.V}^{\pi}(u) = 1\right] = 1 \ .$$

Witness Decoding: *There exists a decoder $\mathsf{PCP.D}$ such that for any π^* and any $u = (M,x,t)$ if*

$$\Pr\left[\mathsf{PCP.V}^{\pi^*}(u)\right] \geq 2^{-|u|} \ ,$$

then $\mathsf{PCP.D}(u,\pi)$ outputs a witness $w \in \mathcal{R}_{\mathcal{U}}(u)$ in time $\mathrm{poly}(|u|,t)$.

PCPs as above exist in the literature (see for instance [BG08] and references within).

5.3 Construction

We now describe the protocol based on multi-collision-resistant hashing with local opening. The protocol follows the standard recipe of combining PCPs with hashing with local openings [Kil92, Mic00, BG08]. The essential difference is that instead of the absolute binding guarantee that is typically provided by completely collision-resistant hash tress, we only have the weak binding guarantee of multi-collision resistance as defined and constructed in Section 4.

We turn to describe the construction. In what follows:

- $\mathsf{PCP} = (\mathsf{PCP.P}, \mathsf{PCP.V})$ is a PCP system (as defined in Section 5.2).
- $\mathsf{HLO} = (\mathsf{HLO.Gen}, \mathsf{HLO.Hash}, \mathsf{HLO.Chal}, \mathsf{HLO.Auth}, \mathsf{HLO.Ver})$ is a hash with local opening (as defined in Section 4).

Notation. We will use the following notation:

- When we want to be explicit about the PCP verifier's randomness r, we write $\mathsf{PCP.V}^{\pi}(u; r)$.
- We denote by $\rho = \rho(\lambda)$ the number of queries that $\mathsf{PCP.V}$ makes, for $u \in \{0,1\}^{\lambda}$.
- For a string π, and verifier randomness r, we denote by $I^{\pi}_{u,r}$ the set of queries that the verifier makes:

$$I^{\pi}_{u,r} := \left\{ i_1, \ldots, i_\rho \in [|\pi|] \;\middle|\; i_j \text{ is the } j\text{-th query made by } \mathsf{PCP.V}^{\pi^*}(u; r) \right\} \quad .$$

- We denote by $L = L(\lambda, t)$ the length of a PCP proof generated by $\mathsf{PCP.P}$ for any instance $u = (M, x, t) \in \{0,1\}^{\lambda} \cap \mathcal{L}(\mathcal{R}_{\mathcal{U}})$.

Protocol 1

Common Input: an instance $u = (M, x, t) \in \mathcal{L}(\mathcal{R}_{\mathcal{U}}) \cap \{0,1\}^{\lambda}$, for security parameter λ.

Auxiliary Input of P: a witness $w \in \mathcal{R}_{\mathcal{U}}(x)$.

1. V samples a key $\mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^{\lambda})$ and sends hk to P.
   In Unkeyed Setting: this message is not sent. $\mathsf{hk} \equiv 1^{\lambda}$ throughout the protocol.
2. P computes a proof $\pi \leftarrow \mathsf{PCP.P}(u, w)$ and a digest $\mathsf{dig} = \mathsf{HLO.Hash}(\mathsf{hk}, \pi)$.
   It sends dig to V.
3. V samples random coins r for $\mathsf{PCP.V}^{(\cdot)}(u)$ and a challenge $\mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^{\lambda}, 1^{\rho})$.
   It sends (r, ch) to P.
4. P computes the set of verifier queries $I := I^{\pi}_{u,r}$ and a proof $\Pi = \mathsf{HLO.Auth}(\mathsf{hk}, \pi, I, \mathsf{ch})$.
   It sends $(I, \pi|_I, \Pi)$ to V.
5. V verifies consistency of the opened proof bits with the digest dig by running $\mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, \pi|_I, \mathsf{ch}, \Pi)$. It also verifies the PCP, and that $I = I^{\pi}_{u,r}$, by running $\mathsf{PCP.V}^{\pi|_I}(u; r)$. It accepts if and only if all checks pass.

Figure 1: A Succinct Argument for Non-Deterministic Computations.

We will now show that the above protocol is a succinct argument assuming multi-collision resistance of the underlying hash with local opening. In the following, let $\bar{t}(\lambda)$ be any time bound function and let $\bar{L}(\lambda) = L(\lambda, \bar{t}(\lambda))$ be the bound on the length of corresponding PCP proofs.

Theorem 5.2. *Assume* HLO *is* K*-collision resistant for input-length bound* $\bar{L}$*. Then Protocol 1 is a succinct argument for* $\bar{t}$*-bounded computations. If* HLO *is unkeyed, then the protocol has* 3*-messages.*

Furthermore, the witness extractor runs in time:

$$\tau(\lambda, t, \varepsilon^{-1}, \zeta) = \mathrm{poly}(\lambda, t, \varepsilon^{-1}, K) \ ,$$

for $K = K(\lambda, \zeta, t^{O(1)})$.

Proof. We first note that completeness as well as verifier and prover efficiency requirements follow directly from that of the underlying PCP and the hash with local opening.

From hereon, we focus on establishing the proof of knowledge property. We first describe the extractor. Fix any PPT prover P^* with polynomial-size advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$, any sequence of inputs $u = (M, x, t) \in \{0,1\}^\lambda$, and any noticeable convincing probability $\varepsilon(\lambda) = \lambda^{-O(1)}$. As before, we let $L = L(\lambda, t)$ denote the length of PCP proofs and let $\rho = \rho(\lambda)$ be the number of queries that $\mathsf{PCP.V}$ makes for instances u. Also, we let $K = K(\lambda, |z_\lambda|, L)$, where K is the collision bound of HLO.

The Extractor $\mathsf{E}^{\mathsf{P}^*(\cdot;z_\lambda)}(u; 1^{1/\varepsilon}, |z_\lambda|)$**:**

1. Sample a key $\mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda)$, feed hk to P^*, who provides a digest dig.
   In Unkeyed Setting: $\mathsf{hk} \equiv 1^\lambda$ is fed to P^*.

2. Let $\mathsf{P}^*_2(\cdot;\mathsf{st})$ capture how P^* authenticates in the second message. That is, $\mathsf{P}^*_2(\cdot;\mathsf{st})$ given a challenge ch, samples PCP randomness r for $\mathsf{PCP.V}^{(\cdot)}(u)$ on its own, and runs $\mathsf{P}^*(r, \mathsf{ch}; \mathsf{st})$, where st is the state of P^* right after it produced the digest dig in the previous step.

   Obtain a set S of candidate PCPs from the HLO extractor:

   $$S \leftarrow \mathsf{Ext}^{\mathsf{P}^*_2(\cdot;\mathsf{st})}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon'}) \ .$$

3. For every $\pi^* \in S$ decode a candidate witness $w^* = \mathsf{PCP.D}(u, \pi^*)$. If $w^* \in \mathcal{R}_\mathcal{U}(u)$, output w^*.

4. The extractor repeats the above three steps for at most λ/ε times, and aborts if no witness is found.

Running Time. As required, the extractor runs in time

$$\tau(\lambda, t, \varepsilon^{-1}, |z_\lambda|) = \lambda \cdot \varepsilon^{-1} \cdot \mathrm{poly}(\lambda, L, \varepsilon^{-1}, K(\lambda, |z_\lambda|, L)) = \mathrm{poly}(\lambda, t, \varepsilon^{-1}, K(\lambda, |z_\lambda|, t^{O(1)})) \ .$$

Successful Witness Extraction. We now analyze the extractor. We first recall that the fact that P^* convinces the verifier V with probability ε, which implies the following:

$$\Pr\left[\begin{array}{l} \\ I = I^{\pi^*}_{u,r} \\ \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, \pi^*|_I, \mathsf{ch}, \Pi) = 1 \\ \mathsf{PCP.V}^{\pi^*|_I}(u; r) = 1 \\ \\ \end{array} \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda) \\ (\mathsf{dig}, \mathsf{st}) \leftarrow \mathsf{P}^*(\mathsf{hk}; z_\lambda) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ r \leftarrow \textit{randomness for } \mathsf{PCP.V}^{(\cdot)}(u) \\ (I, \pi^*|_I, \Pi) \leftarrow \mathsf{P}^*(r, \mathsf{ch}; \mathsf{st}) \end{array}\right] \geq \varepsilon \ .$$

Next, by the HLO extraction guarantee, we know that except with small probability, the prover always opens consistently with the extracted set of strings:

$$\Pr\left[\begin{array}{l|l} \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, \pi^*|_I, \mathsf{ch}, \Pi) = 1 & \mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda) \\ \pi^*|_I \notin \{\pi|_I : \pi \in S\} & (\mathsf{dig}, \mathsf{st}) \leftarrow \mathsf{P}^*(\mathsf{hk}; z_\lambda) \\ & \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ & r \leftarrow \textit{randomness for } \mathsf{PCP.V}^{(\cdot)}(u) \\ & (I, \pi^*|_I, \Pi) \leftarrow \mathsf{P}^*(r, \mathsf{ch}; \mathsf{st}) \\ & \overline{S \leftarrow \mathsf{Ext}^{\mathsf{P}^*_2(\cdot;\mathsf{st})}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon})} \end{array}\right] \leq \varepsilon/2 \ .$$

It follows that with noticeable probability the verifier V accepts an opening that is consistent with the extracted set of strings:

$$\Pr\left[\begin{array}{l|l} I = I^{\pi^*}_{u,r} & \mathsf{hk} \leftarrow \mathsf{HLO.Gen}(1^\lambda) \\ \mathsf{PCP.V}^{\pi^*|_I}(u; r) = 1 & (\mathsf{dig}, \mathsf{st}) \leftarrow \mathsf{P}^*(\mathsf{hk}; z_\lambda) \\ \pi^*|_I \in \{\pi|_I : \pi \in S\} & \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ & r \leftarrow \textit{randomness for } \mathsf{PCP.V}^{(\cdot)}(u) \\ & (I, \pi^*|_I, \Pi) \leftarrow \mathsf{P}^*(r, \mathsf{ch}; \mathsf{st}) \\ & \overline{S \leftarrow \mathsf{Ext}^{\mathsf{P}^*_2(\cdot;\mathsf{st})}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon})} \end{array}\right] \geq \varepsilon/2 \ .$$

We shall call $(\mathsf{hk}, (\mathsf{dig}, \mathsf{st}), \mathsf{ch}, S)$ good if conditioned on these values the above occurs with good probability:

$$\Pr\left[\begin{array}{l|l} I = I^{\pi^*}_{u,r} & r \leftarrow \textit{randomness for } \mathsf{PCP.V}^{(\cdot)}(u) \\ \mathsf{PCP.V}^{\pi^*|_I}(u; r) = 1 & (I, \pi^*|_I, \Pi) \leftarrow \mathsf{P}^*(r, \mathsf{ch}; \mathsf{st}) \\ \pi^*|_I \in \{\pi|_I : \pi \in S\} & \end{array}\right] \geq \varepsilon/4 \ .$$

First, by averaging, we know that the probability that $(\mathsf{hk}, (\mathsf{dig}, \mathsf{st}), \mathsf{ch}, S)$ are good is at least $\varepsilon/4$. Furthermore, for any such good tuple, by the fact that the extracted set is always small $|S| \leq K$, there exists $\pi \in S$ such that

$$\Pr\left[\begin{array}{l|l} I = I^{\pi^*}_{u,r} & r \leftarrow \textit{randomness for } \mathsf{PCP.V}^{(\cdot)}(u) \\ \mathsf{PCP.V}^{\pi^*|_I}(u; r) = 1 & (I, \pi^*|_I, \Pi) \leftarrow \mathsf{P}^*(r, \mathsf{ch}; \mathsf{st}) \\ \pi^*|_I = \pi|_I & \end{array}\right] \geq \varepsilon/4K \ .$$

In particular, for such π,

$$\Pr_r\left[\mathsf{PCP.V}^\pi(u; r) = 1\right] \geq \varepsilon/4K > 2^{-\lambda} \ ,$$

in which case, $\mathsf{PCP.D}(u, \pi)$ outputs a valid witness w.

It follows that whenever the extractor samples a good tuple it finds a witness. Since good tuples occur with probability $\varepsilon/4$, after λ/ε attempts, the extractor will find a witness except with probability $2^{-\lambda/4}$.

This completes the proof of the theorem. □

6 3-Message Zero Knowledge via Weak Memory Delegation

In this section, we construct a 3-message zero-knowledge argument for **NP**, which can be based on weak multi-collision resistance and fully-homomorphic encryption, both with slight super-polynomial hardness. We also show how similar (but simplified) ideas can be used to obtain a 5-message argument that has a *public-coin verifier*.

The main tool behind these constructions is *weak memory delegation*. We start by defining this notion and explaining why it is sufficient for the goal of 3-message zero knowledge. Then (through most of this section) we concentrate on constructing weak memory delegation schemes based on multi-collision resistant hashing with local opening. Finally, we explain how to achieve 5-message public-coin protocols based on similar ideas.

6.1 Weak Memory Delegation

In a two-message memory delegation scheme [CKLR11], an untrusted server provides the client a short commitment or *digest* dig of a large memory D. The client can then delegate any arbitrary deterministic computation M to be executed over the memory. The server responds with the computation's output y, as well as a short proof of correctness that can be verified by the client in time that is independent of that of the delegated computation and the size of the memory.

In the common definition of memory delegation, the soundness requirement says that having provided the digest dig, the prover should not be able to prove that a given computation M results in more than a single outcome y. Naturally, this requires that the short digest fixes (in a computational sense) at most a single underlying memory, which is usually achieved based on collision-resistant hashing. With the aim of replacing collision-resistance with multi-collision resistance (and to potentially avoid an extra setup step), we consider a weaker soundness requirement. The requirement roughly says that the attacker should not be able to prove consistency with *too many* outcomes y. There are several conceivable ways to capture this requirement. For simplicity, we give a somewhat restricted form of this intuition that only says that that the attacker should not be able to prove the correctness of an outcome $y \leftarrow Y$, sampled at random from a sufficiently entropic distribution Y, except with small probability. We proceed to present the formal syntax and definition.

Syntax: A two-message *memory delegation scheme* consists of algorithms:

$$(\mathsf{MD.Gen}, \mathsf{MD.Mem}, \mathsf{MD.Query}, \mathsf{MD.Prove}, \mathsf{MD.Ver}) \ ,$$

with the following syntax:

- $\mathsf{pp} \leftarrow \mathsf{MD.Gen}(1^\lambda)$: a randomized polynomial-time algorithm that given the security parameter 1^λ, outputs public parameters pp. In the unkeyed setting, this algorithm is deterministic and outputs fixed parameters $\mathsf{pp} \equiv 1^\lambda$.
- $\mathsf{dig} \leftarrow \mathsf{MD.Mem}(\mathsf{pp}, D)$: a deterministic polynomial-time algorithm that given pp and a memory D, outputs a digest dig of the memory.
- $(\mathsf{q}, \mathsf{vst}) \leftarrow \mathsf{MD.Query}(1^\lambda)$: a randomized polynomial-time algorithm that given the security parameter 1^λ, outputs a query q and a secret state vst.
- $\pi \leftarrow \mathsf{MD.Prove}(\mathsf{pp}, D, (M, t, y), \mathsf{q})$: a deterministic algorithm that takes the public parameters pp, a memory string D, a (deterministic) Turing machine M, an output string y, and time bound t, such that $(M, t, y) \in \{0,1\}^\lambda$ and $|D| \leq t \leq 2^\lambda$. It outputs a proof π that $M(D)$ outputs y within t steps.
- $b \leftarrow \mathsf{MD.Ver}(\mathsf{pp}, \mathsf{dig}, (M, t, y), \mathsf{vst}, \pi)$: a deterministic polynomial time oracle algorithm that takes the public parameters pp, a digest dig, a (deterministic) Turing machine M, a time bound t, and an output string y, such that $(M, t, y) \in \{0,1\}^\lambda$, together with a pair (vst, π), and outputs an acceptance bit b.

Definition 6.1 (Entropic Distribution Ensemble)**.** *We say that an efficiently samplable distribution ensemble* $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ *is entropic if*

$$H_\infty(Y_\lambda) := -\log \max_{y\in\mathrm{supp}(Y_\lambda)} \Pr[Y_\lambda = y] = \Omega(\lambda) \ .$$

Definition 6.2 (Weak Memory Delegation)**.** *A two-message delegation scheme* $\mathsf{MD} = (\mathsf{MD.Gen}, \mathsf{MD.Mem}, \mathsf{MD.Query}, \mathsf{MD.Prove}, \mathsf{MD.Ver})$ *satisfies:*

Efficient Verifier and Relatively-Efficient Prover: *There exists a universal polynomial* $p(\cdot)$ *such that for every* $(M, t, y) \in \{0,1\}^\lambda$*, and every* D *such that* $M(D)$ *outputs* y *within* t *steps, and* $|D| \le t \le 2^\lambda$*:*

- *The prover* $\mathsf{MD.Prove}(\mathsf{pp}, D, (M, t, y), \mathsf{q})$ *runs in time* $p(\lambda, t)$*.*
- *The verifier* $\mathsf{MD.Ver}(\mathsf{pp}, \mathsf{dig}, (M, t, y), \mathsf{vst}, \pi)$ *runs in time* $p(\lambda)$*.*

Correctness: *For every security parameter* $\lambda \in \mathbb{N}$*, every* $(M, t, y) \in \{0,1\}^\lambda$*, and every* D *such that* $M(D)$ *outputs* y *within* t *steps, and* $|D| \le t \le 2^\lambda$*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{MD.Ver}(\mathsf{pp}, \mathsf{dig}, (M, t, y), \mathsf{vst}, \pi) = 1 & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ \mathsf{dig} \leftarrow \mathsf{MD.Mem}(\mathsf{pp}, D) \\ (\mathsf{q}, \mathsf{vst}) \leftarrow \mathsf{MD.Query}(1^\lambda) \\ \pi \leftarrow \mathsf{MD.Prove}(\mathsf{pp}, D, (M, t, y), \mathsf{q}) \end{array} \end{array}\right] = 1 \ .$$

Weak Soundness for Computation-Time Bound $\bar{t}(\lambda)$***:*** *For every pair of PPT adversaries* $(\mathcal{A}_1, \mathcal{A}_2)$ *and polynomial-size advice* $\{z_\lambda\}_{\lambda\in\mathbb{N}}$*, there exists a negligible function* μ*, such that for every ensemble of samplable entropic distributions* $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$*, every* $t(\lambda) \le \bar{t}^{O(1)}$*, every* $\lambda \in \mathbb{N}$*, letting* $K = K(\lambda, |z_\lambda|, t)$*,*

$$\Pr\left[\begin{array}{l|l} \mathsf{MD.Ver}(\mathsf{pp}, \mathsf{dig}, (M, t, y), \mathsf{vst}, \pi) = 1 & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}; z_\lambda) \\ (\mathsf{q}, \mathsf{vst}) \leftarrow \mathsf{MD.Query}(1^\lambda) \\ y \leftarrow Y_\lambda \\ \pi \leftarrow \mathcal{A}_2(\mathsf{q}, y; \mathsf{st}) \end{array} \end{array}\right] \le \mu(\lambda) \ .$$

We now state the main theorem proved in this section regarding the existence of two-message memory delegation (according to the above definition) based on weak multi-collision-resistant hash functions (Definition 3.2). The theorem has two parts. The first is a polynomial version that guarantees security for computations of arbitrary polynomial time, based on polynomial multi-collision resistance and quasipolynomially-secure fully-homomorphic encryption. The second is a super-polynomial version that guarantees security even for slightly super-polynomial computations, relying on slightly super-polynomial multi-collision resistance.

Theorem 6.1 (Two-Message Delegation)**.**

- *Assuming a weakly* K*-collision-resistant hash for* $K(\lambda, \zeta) = \mathrm{poly}(\lambda, \zeta)$*, and* $\mathrm{quasipoly}(\lambda)$*-secure fully-homomorphic encryption, there exists a two-message memory-delegation scheme with weak soundness for any computation-time bound* $\bar{t}(\lambda) = \lambda^{O(1)}$*.*
- *For any (arbitrary small)* $\tau(\lambda) = \omega(1)$*, there exists* $\bar{t}(\lambda) = \lambda^{\omega(1)}$ *such that assuming a weakly* (K, γ)*-collision-resistant hash, for* $K(\lambda, \zeta) = \mathrm{poly}(\lambda, \zeta)$ *and* $\gamma(\lambda) = \lambda^\tau$*, and* $\mathrm{quasipoly}(\lambda)$*-secure fully-homomorphic encryption, there exists a two-message memory-delegation scheme with weak soundness for computation-time bound* $\bar{t}$*.*

If the underlying hash functions are unkeyed, the scheme does not require trusted public parameters.

In the next sections, we describe the construction behind the theorem, and its building blocks. Eventually, the theorem is derived as a corollary of a more general Theorem 6.4 proven in Section 6.3.

From Weak Memory Delegation to 3-Message Zero Knowledge. As explained in the intro, in [BBK+16] a 3-message zero knowledge protocol is constructed in the *global hash model*, where we assume the existence of a collision-resistant hash function as a public parameter generated in a setup phase. We show how to replace the collision-resistant hash with an unkeyed weakly multi-collision-resistant hash, to deduce:

Theorem 6.2 (Corollary of Theorem 6.1 and [BBK+16])**.** *For any function $\gamma(\lambda) = \lambda^{-\omega(1)}$, assume a weakly (K, γ)-collision-resistant hash, for $K(\lambda, \zeta) = \mathrm{poly}(\lambda, \zeta)$, $\mathrm{quasipoly}(\lambda)$-secure fully-homomorphic encryption, circuit-private 1-hop homomorphic encryption, and non-interactive perfectly-binding commitments. Then there exists a 3-message zero-knowledge argument for any language in* ***NP****.*

Remark 6.1. We refer the reader to Appendix B.2 for the definition of a circuit-private 1-hop homomorphic encryption, but mention that it can be constructed based on many standard cryptographic assumptions, and in particular, based on the learning with errors assumption.

We refer the reader to Appendix B.1 for the construction of our 3-round zero-knowledge protocol.

Proof Sketch. In [BBK+16], the collision-resistant hash function is used for two purposes:

- As the public parameters for a two-message memory delegation scheme.
- In the construction of a 3-message witness-indistinguishable proof of knowledge with first-message-dependent instances (see definition in Appendix B.1). We note that this use is somewhat less essential and can be avoided assuming, in addition, ZAPs [DN07].

We now argue that weak multi-collision-resistant is sufficient for both.

For the first one, we recall that the soundness (or proof-of-knowledge) guarantee of the [BBK+16] protocol is shown by a reduction to the soundness of memory delegation. Concretely, the reduction [BBK+16, Section 3.1] transforms any convincing prover into an attacker for the memory delegation scheme. We observe that the constructed attacker is, in fact, strong enough to also break multi-collision resistance. Specifically, the constructed attacker samples a digest dig and a computation M such that with noticeable probability ε (in [BBK+16], $\eta^{\Omega(1)}$) over a random output y, it successfully produces a convincing proof consistently with (M, y), which is enough to break weak soundness of memory delegation.

The second use of collision-resistance is in [BBK+16, Section 2.5], where the 3-message witness-indistinguishability protocol of Lapidot and Shamir [LS90a] is transformed into one where the first message has a fixed length, independently of the statement to be proven. There, collision-resistance is used to hash the first (commitment) message in the LS protocol, where the preimage commitment is revealed at the end. To prove that this system is an argument of knowledge, one relies on the fact that when rewinding an ε-convincing prover, to obtain different openings, the prover always opens the same commitment (hash preimage), or it could break collision-resistance. Here we can replace the collision-resistant hash with a K-collision-resistant hash, we still have the guarantee that one of at most K commitments is opened with high probability $\varepsilon^{\Omega(1)}/K$ with different challenges. □

6.2 Oracle Memory Delegation

Toward the construction of (weak) memory delegation we define a weaker notion called *oracle memory delegation*, which will be used as a building block in our construction. Roughly speaking, this is a memory delegation scheme in a hybrid model, where rather than providing a short digest to a memory. The server provides the memory in full *as an oracle*. The client, however, only accesses a small part of this oracle (similar to the interactive PCP model of [KR08]).

Syntax: A two-message oracle memory delegation scheme consists of algorithms:

$$(\mathsf{OMD.Mem}, \mathsf{OMD.Query}, \mathsf{OMD.Prove}, \mathsf{OMD.Ver}) \ ,$$

with the following syntax:

- $\widehat{D} \leftarrow \mathsf{OMD.Mem}(1^\lambda, D)$: a deterministic polynomial-time algorithm that given the security parameter 1^λ, and a memory D, outputs an encoding $\widehat{D}$ of the memory.
- $(\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda)$: a randomized polynomial-time algorithm that given the security parameter 1^λ outputs a query q, a set $I \subseteq \mathbb{N}$ of oracle queries, and a secret state vst.
- $\pi \leftarrow \mathsf{OMD.Prove}(D, (M, t, y), \mathsf{q})$: a deterministic algorithm that takes a memory string D, a (deterministic) Turing machine M, an output string y, and time bound t, such that $(M, t, y) \in \{0,1\}^\lambda$ and $|D| \leq t \leq 2^\lambda$. It outputs a proof π that $M(D)$ outputs y within t steps.
- $b \leftarrow \mathsf{MD.Ver}^{(\cdot)}((M, t, y), \mathsf{vst}, \pi)$: a deterministic polynomial time oracle-aided algorithm that takes a (deterministic) Turing machine M, a time bound t, and an output string y, such that $(M, t, y) \in \{0,1\}^\lambda$, together with a pair (vst, π), and outputs an acceptance bit b.

Definition 6.3. *A two-message oracle-memory delegation scheme* $\mathsf{OMD} = (\mathsf{OMD.Mem}, \mathsf{OMD.Query}, \mathsf{OMD.Prove}, \mathsf{OMD.Ver})$ *satisfies:*

Efficient Verifier and Relatively-Efficient Prover: *There exists a universal polynomial $p(\cdot)$ such that for every $(M, t, y) \in \{0,1\}^\lambda$, and every D such that $M(D)$ outputs y within t steps, and $|D| \leq t \leq 2^\lambda$:*

- *The prover* $\mathsf{OMD.Prove}(D, (M, t, y), \mathsf{q})$ *runs in time $p(\lambda, t)$.*
- *The verifier* $\mathsf{MD.Ver}^{(\cdot)}((M, t, y), \mathsf{vst}, \pi)$ *runs in time $p(\lambda)$.*

Correctness: *For every security parameter $\lambda \in \mathbb{N}$, every $(M, t, y) \in \{0,1\}^\lambda$, and every D such that $M(D)$ outputs y within t steps, and $|D| \leq t \leq 2^\lambda$:*

$$\Pr\left[\mathsf{OMD.Ver}^{\widehat{D}|_I}((M, t, y), \mathsf{vst}, \pi) = 1 \;\middle|\; \begin{array}{l} \widehat{D} \leftarrow \mathsf{OMD.Mem}(1^\lambda, D) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \pi \leftarrow \mathsf{OMD.Prove}(D, (M, t, y), \mathsf{q}) \end{array} \right] = 1 \ .$$

γ-Soundness for Computation-Time Bound $\bar{t}(\lambda)$: *for every pair of probabilistic $\gamma^{O(1)}$-time adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ and polynomial-size advice $\{z_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function μ, such that for every $t(\lambda) \leq \bar{t}^{O(1)}$, every $\lambda \in \mathbb{N}$, letting $K = K(\lambda, |z_\lambda|, t)$,*

$$\Pr\left[\begin{array}{l} y \neq y' \\ \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M, t, y), \mathsf{vst}, \pi) = 1 \\ \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M, t, y'), \mathsf{vst}, \pi') = 1 \end{array} \;\middle|\; \begin{array}{l} (\widehat{D}^*, M, y, y', \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ (\pi, \pi') \leftarrow \mathcal{A}_2(\mathsf{q}; \mathsf{st}) \end{array} \right] \leq \mu(\gamma(\lambda)) \ .$$

Theorem 6.3 (Follows from [KRR14]). *For any functions $\bar{t}(\lambda), \gamma \leq 2^{\lambda}$, assuming the existence of levelled fully homomorphic encryption (*FHE*) that is $\gamma \cdot \mathrm{quasipoly}(\bar{t})$-secure, there exists a 2-message oracle-memory delegation scheme that is γ-sound for time bound $\bar{t}$.*

Proof Sketch. Kalai, Raz, and Rothblum [KRR14] show that a levelled FHE that is $\gamma \cdot \mathrm{quasipoly}(\bar{t})$-secure implies a γ-sound 2-message delegation scheme (for all deterministic computations) for $\bar{t}$-time computations. Furthermore, they prove that the verifier does not need to read the input, but rather only needs to read a random point in the low-degree extension of the input. This point can be generated non-adaptively aged of time. Also, the complexity of the verifier does not depend on the input length, and is a fixed in the security parameter (provided that the input is shorter than 2^{λ}.)

This suggests the existence of an oracle-memory delegation scheme where the memory D is treated as the input, and its encoding $\widehat{D}$ is simply its low-degree extension. The only gap is that in [KRR14], it is assumed that $\widehat{D}$ is an honest low-degree extension, whereas in our case, the adversary may output an arbitrary encoding. This gap is naturally bridged by adding a low-degree test. We note that the queries for a low degree test can be generated non-adaptively ahead of time, and the complexity of the test only depends on the security parameter. □

6.3 Construction

We now construct a weak memory delegation scheme. We essentially show how to compile any oracle-memory-delegation scheme into a weak (non-oracle) memory delegation scheme based on multi-collision-resistant hashing with local opening.

We turn to describe the construction. In what follows:

- $\mathsf{OMD} = (\mathsf{OMD.Mem}, \mathsf{OMD.Query}, \mathsf{OMD.Prove}, \mathsf{OMD.Ver})$ is an oracle-memory delegation scheme (as defined in Section 6.2).
- $\mathsf{HLO} = (\mathsf{HLO.Gen}, \mathsf{HLO.Hash}, \mathsf{HLO.Chal}, \mathsf{HLO.Auth}, \mathsf{HLO.Ver})$ is a hash with local opening (as defined in Section 4).
- $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ is a secret-key (leveled) fully homomorphic encryption scheme [Gen09].

Notation. We will use the following notation:

- We denote by $\rho = \rho(\lambda)$ be the number of oracle queries generated by $\mathsf{OMD.Query}(1^{\lambda})$.
- We denote by $L = L(\lambda, t)$ the maximal length of an encoded oracle $\widehat{D}$ generated by $\mathsf{OMD.Mem}(1^{\lambda}, D)$ for any memory D of size at most t.

We now describe the scheme's algorithms:

$$\mathsf{MD} = (\mathsf{MD.Gen}, \mathsf{MD.Mem}, \mathsf{MD.Query}, \mathsf{MD.Prove}, \mathsf{MD.Ver}) \ .$$

- $\mathsf{MD.Gen}(1^{\lambda})$:
  - Sample a key $\mathsf{HLO.Gen}(1^{\lambda})$.
  - Output $\mathsf{pp} = \mathsf{hk}$. In the unkeyed setting, $\mathsf{pp} \equiv 1^{\lambda}$.

- $\mathsf{MD.Mem}(\mathsf{hk}, D)$:
  - Encode the memory $\widehat{D} = \mathsf{OMD.Mem}(D)$.
  - Output the digest $\mathsf{dig} = \mathsf{HLO.Hash}(\mathsf{hk}, \widehat{D})$.
- $\mathsf{MD.Query}(1^\lambda)$:
  - Sample $(\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda)$ and
  - Sample a secret key $\mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda)$ and encrypt the oracle queries $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(I)$.
  - Sample a challenge $\mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho)$, where recall that ρ is the size of query set I.
  - Output the query $\mathsf{q}' = (\mathsf{q}, \mathsf{ct}, \mathsf{ch})$ and state $\mathsf{vst}' = (\mathsf{vst}, \mathsf{ch}, \mathsf{sk})$.
- $\mathsf{MD.Prove}(\mathsf{hk}, D, (M, t, y), (\mathsf{q}, \mathsf{ct}, \mathsf{ch}))$:
  - Compute a proof $\pi \leftarrow \mathsf{OMD.Prove}(D, (M, t, y), \mathsf{ct})$,
  - Let $A(I)$ be the function that computes $\Pi = \mathsf{HLO.Auth}(\mathsf{hk}, \widehat{D}, I, \mathsf{ch})$, and outputs $(\widehat{D}|_I, \Pi)$.
  - Homomorphically compute $\widehat{\mathsf{ct}} = \mathsf{FHE.Eval}(\mathsf{ct}, A(\cdot))$.
  - Output as the proof $\pi' = (\pi, \widehat{\mathsf{ct}})$.
- $\mathsf{MD.Ver}(\mathsf{hk}, \mathsf{dig}, (M, t, y), (\mathsf{vst}, \mathsf{ch}, \mathsf{sk}), (\pi, \widehat{\mathsf{ct}}))$:
  - Decrypt $(\widehat{D}|_I, \Pi) = \mathsf{FHE.Dec}_{\mathsf{sk}}(\widehat{\mathsf{ct}})$.
  - Verify consistency with digest $\mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, \widehat{D}|_I, \mathsf{ch}, \Pi)$.
  - Run the oracle verifier $\mathsf{OMD.Ver}^{\widehat{D}|_I}((M, t, y), \mathsf{vst}, \pi)$.
  - Accept if and only if both checks pass.

We will now show that the above scheme is a weak memory delegation scheme assuming multi-collision resistance of the underlying hash with local opening. In the following, let $\bar{t}(\lambda)$ be any time bound function and let $\bar{L}_{\bar{t}}(\lambda) = L(\lambda, \bar{t}(\lambda))$ be the bound encoded memories $\widehat{D}$ for $|D| \leq \bar{t}(\lambda)$. Also, in what follows, $K = K(\lambda, \zeta, L)$ is such that $K(\lambda, \lambda, \bar{L}_{\bar{t}}) \leq 2^{o(\lambda)}$ and $\gamma(\lambda) = K(\lambda, \lambda, \bar{L}_{\bar{t}}) \leq 2^{o(\lambda)}$.

Theorem 6.4. *Assume that* HLO *is K-collision resistant for input-length bound $\bar{L}_{\bar{t}}$,* OMD *is γ-sound for computation-time-bound $\bar{t}$, and* FHE *is γ-secure. Then the scheme* MD *is a memory delegation scheme with K-weak soundness for computation-time bound $\bar{t}$. If* HLO *is unkeyed, then scheme does not require trusted public parameters.*

Proof. We first note that correctness as well as verifier and prover efficiency requirements follow directly from that of the underlying oracle-memory delegation and the hash with local opening.

From hereon, we focus on proving weak soundness. Fix any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ any polynomial-size advice $\{z_\lambda\}_{\lambda \in \mathbb{N}}$, any entropic distribution ensemble $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ and any noticeable function $\varepsilon(\lambda) = \lambda^{-O(1)}$. As before, we let $L = L(\lambda, t)$ denote the length of any encoded memory $\widehat{D}$ for $|D| \leq t$, and let $\rho = \rho(\lambda)$ be the number of oracle queries that $\mathsf{OMD.Query}(1^\lambda)$ generates. Also, we let $K = K(\lambda, |z_\lambda|, L)$, where K is the collision bound of HLO.

Assume toward contradiction that for infinitely many $\lambda \in \mathbb{N}$, and $t(\lambda) \leq \bar{t}^{O(1)}$, the adversary $\mathcal{A}$ violates K-weak soundness:

$$\Pr\left[\begin{array}{l|l} \mathsf{MD.Ver}(\mathsf{hk}, \mathsf{dig}, (M,t,y), \mathsf{vst}', \pi') = 1 & \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}', \mathsf{vst}') \leftarrow \mathsf{MD.Query}(1^\lambda) \\ y \leftarrow Y_\lambda \\ \pi' \leftarrow \mathcal{A}_2(\mathsf{q}, y; \mathsf{st}) \end{array} \end{array}\right] \geq \varepsilon \ .$$

Spelling this out according to the construction,[8]

$$\Pr\left[\begin{array}{l|l} \begin{array}{l} \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, \widehat{D}^*|_I, \mathsf{ch}, \Pi) = 1 \\ \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M,t,y), \mathsf{vst}, \pi) = 1 \\ \textit{where } (\widehat{D}^*|_I, \Pi) := \mathsf{FHE.Dec}_{\mathsf{sk}}(\widehat{\mathsf{ct}}) \end{array} & \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(I) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ y \leftarrow Y_\lambda \\ (\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y; \mathsf{st}) \end{array} \end{array}\right] \geq \varepsilon \ . \tag{10}$$

Our next step is to show that we can extract from $\mathcal{A}$ a set of encoded memories $S = \left\{\widehat{D}\right\}$ of size at most K, which $\mathcal{A}$ is almost always consistent with. We define $\mathcal{A}_2^{\mathsf{HLO}}(\cdot; \mathsf{st})$ which captures how $\mathcal{A}_2$ authenticates its answers.

Adversary $\mathcal{A}_2^{\mathsf{HLO}}(\cdot; \mathsf{st})$: given the state st produced by $\mathcal{A}_1$ after it created a digest dig, does the following:

- It samples on its own:
  - $(\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda)$,
  - $\mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda)$,
  - $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(I)$,
  - $\mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho)$,
  - $y \leftarrow Y_\lambda$.
- Obtains $(\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y; \mathsf{st})$.
- Decrypts $(\widehat{D}^*|_I, \Pi) = \mathsf{FHE.Dec}_{\mathsf{sk}}(\widehat{\mathsf{ct}})$.
- Outputs $(I, \widehat{D}^*|_I, \Pi)$.

Next, by the HLO extraction guarantee, there exists an extractor Ext, such that by our definition of $\mathcal{A}_2^{\mathsf{HLO}}$,

[8] The part of the experiment colored in gray will not change throughout proof.

except with small probability, the actual adversary $\mathcal{A}_2$ opens consistently with the extracted set of strings:

$$\Pr\left[\begin{array}{l} \mathsf{HLO.Ver}(\mathsf{hk}, L, \mathsf{dig}, I, \widehat{D}^*|_I, \mathsf{ch}, \Pi) = 1 \\ \widehat{D}^*|_I \notin \left\{\widehat{D}|_I : \widehat{D} \in S\right\} \end{array} \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(I) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ y \leftarrow Y_\lambda \\ (\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y; \mathsf{st}) \\ (\widehat{D}^*|_I, \Pi) = \mathsf{FHE.Dec}_{\mathsf{sk}}(\widehat{\mathsf{ct}}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2^{\mathsf{HLO}(\cdot;\mathsf{st})}}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon}) \end{array}\right] \leq \frac{\varepsilon}{2} \ . \quad (11)$$

Combining Equations 10 and 11, it follows that with noticeable probability the oracle-memory delegation verifier accepts consistently with the extracted set of strings:

$$\Pr\left[\begin{array}{l} \exists \widehat{D}^*|_I \in \left\{\widehat{D}|_I : \widehat{D} \in S\right\}, \\ \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M, t, y), \mathsf{vst}, \pi) = 1 \end{array} \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(I) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ y \leftarrow Y_\lambda \\ (\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y; \mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2^{\mathsf{HLO}(\cdot;\mathsf{st})}}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon}) \end{array}\right] \geq \frac{\varepsilon}{2} \ .$$

Next, we observe that by the semantic security of the FHE scheme, the same holds when $\mathcal{A}_2$ obtains an encryption that is independent of the oracle queries I, except with negligible probability $\lambda^{-O(1)} \ll \varepsilon/4$:

$$\Pr\left[\begin{array}{l} \exists \widehat{D}^*|_I \in \left\{\widehat{D}|_I : \widehat{D} \in S\right\}, \\ \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M, t, y), \mathsf{vst}, \pi) = 1 \end{array} \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(0^\rho) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ y \leftarrow Y_\lambda \\ (\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y; \mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2^{\mathsf{HLO}(\cdot;\mathsf{st})}}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon}) \end{array}\right] \geq \frac{\varepsilon}{4} \ .$$

Since $|S| \leq K$, it follows that the above event occurs for a random choice of $\widehat{D}^* \leftarrow S$ with good probability:

$$\Pr\left[\begin{array}{l} \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M,t,y),\mathsf{vst},\pi)=1 \end{array} \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(0^\rho) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ y \leftarrow Y_\lambda \\ (\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q},\mathsf{ct},\mathsf{ch}), y; \mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2^{\mathsf{HLO}(\cdot;\mathsf{st})}}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon}) \\ \widehat{D}^* \leftarrow S \end{array}\right] \geq \frac{\varepsilon}{4K} \ . \quad (12)$$

By a standard averaging argument, we know that with high probability the above also occurs simultaneously for two independent choices of $y, y' \leftarrow Y_\lambda$:

$$\Pr\left[\begin{array}{l} \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M,t,y),\mathsf{vst},\pi)=1 \\ \mathsf{OMD.Ver}^{\widehat{D}^*|_I}((M,t,y'),\mathsf{vst},\pi')=1 \end{array} \middle| \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda) \\ (\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda) \\ (\mathsf{q}, I, \mathsf{vst}) \leftarrow \mathsf{OMD.Query}(1^\lambda) \\ \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(0^\rho) \\ \mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho) \\ y, y' \leftarrow Y_\lambda \\ (\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q},\mathsf{ct},\mathsf{ch}), y; \mathsf{st}) \\ (\pi', \widehat{\mathsf{ct}}') \leftarrow \mathcal{A}_2((\mathsf{q},\mathsf{ct},\mathsf{ch}), y'; \mathsf{st}) \\ \hline S \leftarrow \mathsf{Ext}^{\mathcal{A}_2^{\mathsf{HLO}(\cdot;\mathsf{st})}}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon}) \\ \widehat{D}^* \leftarrow S \end{array}\right] \geq \Omega\left(\frac{\varepsilon^3}{K^3}\right) \ .$$

This immediately implies a breaker $\mathcal{A}^{\mathsf{OMD}} = (\mathcal{A}_1^{\mathsf{OMD}}, \mathcal{A}_2^{\mathsf{OMD}})$ for the oracle-memory delegation:

- $\mathcal{A}_1^{\mathsf{OMD}}(1^\lambda; z_\lambda)$ samples:
  - $\mathsf{hk} \leftarrow \mathsf{MD.Gen}(1^\lambda)$,
  - $(\mathsf{dig}, M, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{hk}; z_\lambda)$,
  - $S \leftarrow \mathsf{Ext}^{\mathcal{A}_2^{\mathsf{HLO}(\cdot;\mathsf{st})}}(1^\lambda, 1^\rho, 1^L, 1^K, 1^{2/\varepsilon})$.
  - $\widehat{D}^* \leftarrow S$.
  - $y, y' Y$.
- Output $(\widehat{D}^*, M, y, y', (\mathsf{st}, y, y'))$.
- $\mathcal{A}_2^{\mathsf{OMD}}(\mathsf{q}; (\mathsf{st}, y, y'))$ samples:
  - $\mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda)$,
  - $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(0^\rho)$,
  - $\mathsf{ch} \leftarrow \mathsf{HLO.Chal}(1^\lambda, 1^\rho)$,

– $(\pi, \widehat{\mathsf{ct}}) \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y; \mathsf{st})$.
$(\pi', \widehat{\mathsf{ct}}') \leftarrow \mathcal{A}_2((\mathsf{q}, \mathsf{ct}, \mathsf{ch}), y'; \mathsf{st})$.

- It outputs (π, π').

It follows directly from Equation 12 that $\mathcal{A}^{\mathsf{OMD}}$ breaks the underlying oracle-memory delegation scheme with probability at least $\Omega(\frac{\varepsilon^3}{K^3}) - 2^{-\Omega(\lambda)}$, where the $2^{-\Omega(\lambda)} = 2^{-H_\infty(Y_\lambda)}$ accounts for the probability that y, y' collide.

This completes the proof of the theorem. □

7 1-Message Statistically-Hiding Commitments with Weak Binding

In this section, we define and construct weakly-binding statistically-hiding commitments. The essential difference between such commitments and standard statistically-hiding commitments is in the binding guarantee. Whereas in standard commitments an efficient adversary should not be able to open a commitment to two distinct plaintexts, now we only require that it cannot open a commitment to more than K plaintexts. Analogously to the case of collision-resistant hash functions, under this definition, it is possible to consider such commitments that are completely non-interactive (without any key). Here the number of possible openings may scale with the adversary's non-uniform advice.

In Section 7.1, we define the notion and show how to obtain it from multi-collision-resistant hash functions. In Section 8, we show how to use such commitments to construct ε-zero-knowledge proofs from such commitments. Relying on unkeyed commitments, the resulting protocols improve the known round-complexity for these tasks.

7.1 Definition

We define weakly-binding statistically-hiding commitments and then describe a construction.

Syntax: A commitment scheme is associated with an input length function $L(\lambda)$ and polynomial-time algorithms $\mathsf{SHC} = (\mathsf{SHC.Gen}, \mathsf{SHC.Com})$ with the following syntax:

- $\mathsf{hk} \leftarrow \mathsf{SHC.Gen}(1^\lambda)$: a probabilistic algorithm that takes the security parameter 1^λ and outputs a key $\mathsf{hk} \in \{0,1\}^\lambda$. In the unkeyed setting, this algorithm is deterministic and outputs a fixed key $\mathsf{hk} \equiv 1^\lambda$.
- $c \leftarrow \mathsf{SHC.Com}(X; \mathsf{hk})$: a probabilistic algorithm that takes the key hk and an input $X \in \{0,1\}^{L(\lambda)}$ and outputs a commitment $c \in \{0,1\}^\lambda$. When we want to be explicit about the randomness r used by the algorithm, we may write $\mathsf{SHC.Com}(X; \mathsf{hk}, r)$.

Definition 7.1 (Weakly-Binding Statistically-Hiding Commitments)**.** *A weakly-binding statistically-hiding commitment* $\mathsf{SHC} = (\mathsf{SHC.Gen}, \mathsf{SHC.Com})$ *satisfies:*

Statistical Hiding: *For any two plaintexts* $X, X' \in \{0,1\}^{L(\lambda)}$*, and any key* $\mathsf{hk} \in \{0,1\}^\lambda$*, the corresponding commitments are statistically close:*

$$\mathsf{SHC.Com}(X; \mathsf{hk}) \approx_s \mathsf{SHC.Com}(X'; \mathsf{hk}) \ .$$

Weak K-Binding: *For any PPT $\mathcal{A}$ and any sequence of polynomial-size advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$, there is a negligible function μ, such that for any $\lambda \in \mathbb{N}$, letting $K = K(\lambda, |z_\lambda|)$,*

$$\Pr\left[\begin{array}{c|l} c_1 = \cdots = c_K & \mathsf{hk} \leftarrow \mathsf{SHC.Gen}(1^\lambda) \\ \forall i \neq j : X_i \neq X_j & ((X_1, r_1) \ldots, (X_K, r_K)) \leftarrow \mathcal{A}(\mathsf{hk}; z_\lambda) \\ & \forall i : c_i = \mathsf{SHC.Com}(X_i; \mathsf{hk}, r) \end{array}\right] \leq \mu(\lambda) \ .$$

Remark 7.1 (Plaintext Size). While in standard commitments we typically restrict attention to bit plaintexts, weakly-binding commitments only become meaningful for plaintexts from a super-polynomial message space $\{0,1\}^{L(\lambda)}$ for $L(\lambda) = \omega(\log \lambda)$.

7.2 Construction

We next describe a simple construction based on K-collision resistance. At high-level, the construction is similar to the construction of statistically-hiding commitments from collision-resistant hash functions [DPP93, HM96], only that collision-resistant hash functions are replaced with K-collision resistance. We describe it here for completeness.

Let L be the input length, and let $\mathsf{H} = (\mathsf{H.Gen}, \mathsf{H.Hash})$ be a K-collision-resistant hash function mapping $L(\lambda) + 2\lambda$ bits to λ bits. Let $\mathcal{H} = \{\mathcal{H}_\lambda\}$ be a family of pairwise independent hash functions mapping $L(\lambda) + 2\lambda$ bits to $L(\lambda)$ bits.

- $\mathsf{SHC.Gen}(1^\lambda)$: applies $\mathsf{H.Gen}(1^\lambda)$ and outputs the produced key hk. (In the unkeyed setting $\mathsf{hk} \equiv 1^\lambda$).
- $\mathsf{SHC.Com}(X; \mathsf{hk})$: samples a pairwise independent $h \leftarrow \mathcal{H}_\lambda$ and randomness $r \leftarrow \{0,1\}^{L(\lambda)+2\lambda}$. It computes $Y_r = \mathsf{H.Hash}(\mathsf{hk}, r)$, $z_r = h(r)$ and outputs: $c = (h, Y_r, z_r \oplus X)$.

Proposition 7.1. *The scheme is statistically-hiding and K-binding.*

The proof is a natural extension of the proof of statistically-hiding commitments from collisions-resistance. Below we give a short sketch. (The proof for statistical hiding will in fact be much simpler than in [DPP93, HM96], by relying on the notion of average min entropy and corresponding generalization of the leftover hash lemma [DORS08].)

Proof Sketch. To prove K-binding, note that for any $c = (h, Y, z)$ and randomness r there could be at most a single X such that $\mathsf{SHC.Com}(X; \mathsf{hk}, r) = c$. Thus opening c to K plaintexts X, implies exhibiting K preimages of Y under $\mathsf{H.Hash}(\mathsf{hk}, \cdot)$.

To prove statistical hiding, note that the average min-entropy [DORS08] of $r|Y_r$ is at least

$$H_\infty(r|Y_r) := \mathbb{E}_{Y \leftarrow Y_r} H_\infty(r|Y_r = Y) = L(\lambda) + 2\lambda - |Y_r| = L(\lambda) + \lambda \ .$$

By the generalized leftover hash lemma [DORS08],

$$(h, Y_r, z_r \oplus X) \approx_\varepsilon (h, Y_r, u) \ ,$$

for a uniformly random and independent $u \leftarrow \{0,1\}^{L(\lambda)}$ and $\varepsilon = O(\sqrt{2^{-H_\infty(r|Y_r)} \cdot 2^{L(\lambda)}}) \leq O(2^{-\lambda/2})$. □

Remark 7.2 (Shrinkage). In the above commitment scheme, the size of the commitment is proportional to the size of the plaintext. The commitment can be made shrinking by another application of the hash function over the commitment. This weakens the binding property quadratically.

8 4-Message ϵ-Zero Knowledge Proofs

In this section, we show how to use weakly-binding statistically-hiding commitments to get ε-zero-knowledge proofs [DNRS03], a relaxation of zero-knowledge proof where to achieve simulation accuracy ε the simulator may run in time $1/\varepsilon$. Relying on unkeyed (weakly-binding) commitments, the resulting proof system has four messages, which crosses a previous black-box lower bound of Katz [Kat12] that shows that (even such relaxed) zero-knowledge proofs cannot be achieved in four messages if the simulator completely treats the verifier as a black box. Indeed, our simulator will only be semi black-box — it will use the verifier as a black-box, but will depend on the size of the verifier's non-uniform advice.

Previously, the most round-efficient zero-knowledge proof was that of Goldreich and Kahan [GK96a], which had five messages and could be based on binding (not weakly) statistically-hiding commitments.[9] In contrast, the proof of Goldreich and Kahan satisfies a stronger zero-knowledge guarantee — it has an expected polynomial-time simulator with a negligible simulation error.

8.1 Definition

We start by recalling ε-zero-knowledge proofs.

Definition 8.1. *A protocol $\langle \mathsf{P} \leftrightarrows \mathsf{V} \rangle$ for an **NP** relation $\mathcal{R}(x,w)$ is an ε-zero-knowledge proof if it satisfies the standard completeness and soundness requirements (Definition 2.1), and the following augmented zero knowledge requirement.*

*ε **Zero-Knowledge:** For every PPT verifier V^* with polynomial-size advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a (uniform) PPT simulator S such that for every noticeable function $\varepsilon(\lambda) = \lambda^{-O(1)}$:*

$$\{\langle \mathsf{P}(w) \leftrightarrows \mathsf{V}^*(x; z_\lambda)\rangle\}_{\substack{(x,w)\in\mathcal{R}\\ |x|=\lambda}} \approx_\varepsilon \left\{\mathsf{S}(x; z_\lambda, 1^{1/\varepsilon(\lambda)})\right\}_{\substack{(x,w)\in\mathcal{R}\\ |x|=\lambda}} .$$

8.2 Construction

To present the protocol, we will use the abstraction of *Sigma Protocols*, which are a public-coin three-message proof system that only give a very weak zero-knowledge guaranteed.

Definition 8.2. *A sigma protocol for an **NP** relation $\mathcal{R}$ is a three-message protocol (α,β,γ) between a prover $\Sigma.\mathsf{P}$ and a public-coin verifier $\Sigma.\mathsf{V}$, with the following properties:*

***Completeness:** for any $(x,w)\in\mathcal{R}$, $\langle \Sigma.\mathsf{P}(w) \leftrightarrows \Sigma.\mathsf{V}\rangle(x) = 1$.*

***Soundness:** for any $x \in \{0,1\}^\lambda \setminus \mathcal{L}$ and unbounded prover $\Sigma.\mathsf{P}^*$,*

$$\Pr\left[\langle \Sigma.\mathsf{P}^* \leftrightarrows \Sigma.\mathsf{V}\rangle(x) = 1\right] \le \lambda^{-\omega(1)} .$$

***Special Zero-Knowledge:** The exists a PPT simulator $\Sigma.\mathsf{S}$ such that*

$$\{(\alpha,\gamma) \leftarrow \Sigma.\mathsf{P}(x,w;\beta)\}_{\substack{(x,w)\in\mathcal{R}\\ x,\beta\in\{0,1\}^\lambda}} \approx_c \{(\alpha,\gamma) \leftarrow \Sigma.\mathsf{S}(x,\beta)\}_{\substack{(x,w)\in\mathcal{R}\\ x,\beta\in\{0,1\}^\lambda}} ,$$

where $\Sigma.\mathsf{P}(x,w;\beta)$ is the prover's message distribution when the verifier's challenge is fixed to β.

[9] We note that zero-knowledge arguments (that are only computationally sound) are known in four rounds.

The next claim follows from the zero-knowledge requirement, and will be used throughout the analysis.

Claim 8.1 (First-Message Indistinguishability)**.** *In every Σ protocol:*

$$\{\alpha \leftarrow \Sigma.\mathsf{P}_1(x,w)\}_{\substack{(x,w)\in\mathcal{R}\\ x,\beta\in\{0,1\}^\lambda}} \approx_c \{\alpha \leftarrow \Sigma.\mathsf{S}_1(x,\beta)\}_{\substack{(x,w)\in\mathcal{R}\\ x,\beta\in\{0,1\}^\lambda}} \approx_c \left\{\alpha \leftarrow \Sigma.\mathsf{S}_1(x,0^\lambda)\right\}_{\substack{(x,w)\in\mathcal{R}\\ x,\beta\in\{0,1\}^\lambda}},$$

where $\Sigma.\mathsf{P}_1(x,w)$ and $\Sigma.\mathsf{S}_1(x,\beta)$ are the distributions on the first message of the prover and simulator.

Proof Sketch. Note that the first prover message, and likewise the first simulator message, are computed independently of the verifier's message; namely, $\Sigma.\mathsf{P}_1(x,w) \equiv \Sigma.\mathsf{P}_1(x,w;\beta)$. The claim now follows by the zero-knowledge guarantee. □

Sigma protocols are known to follow from classical zero-knowledge proof systems such as the (parallel repetition) of the 3-coloring protocol [GMW91], based on non-interactive statistically-binding commitments. A slight relaxation where all the parties and the simulator obtain a global common random string can be obtained from one-way functions, by using Naor's commitments [Nao91]. For simplicity of exposition, we shall rely on the above formulation, without a common random string.

From Sigma Protocols to Zero-Knowledge Proofs. Sigma protocols provide a weak form of Zero-Knowledge against verifier's whose message β is known ahead of time. The proof system of Goldreich and Kahan [GK96a] can be seen as general transformation that boosts this weak guarantee to full-fledged zero knowledge relying on perfectly (or statistically) hiding commitments.

We naturally augment their transformation by using weakly-binding statistically-hiding commitments. the resulting protocol is essentially the same, with the exception that weakly-binding commitments can be unkeyed, in which case we get a 4-message proof rather than a 5-message one. The main difference is in the analysis of zero knowledge. Our revised analysis will show that indeed weak binding is sufficient.

We proceed to describe the protocol and then analyze it.

8.3 Analysis

Proposition 8.1. *The protocol is an ε-zero-knowledge proof.*

Proof. As in [GK96a], the fact that the protocol is (statistically) sound follows directly from the statistical soundness of the Sigma protocol and the statistical hiding of the commitments.

Henceforth, we concentrate on proving zero knowledge. We first describe the zero-knowledge simulator. Our simulator will be a black-box simulator in the code of the verifier, with the exception that it depends on the size of the verifier's advice. Fix any verifier V^* with polynomial-size advice $\{z_\lambda\}_{\lambda\in\mathbb{N}}$, and fix a noticeable simulation accuracy function $\varepsilon(\lambda) = \lambda^{-O(1)}$. We assume w.l.o.g that the verifier is deterministic (any randomness can be embedded in the advice z_λ.) Also, throughout, we let $K = K(\lambda, |z_\lambda|)$, where K is the weak-binding parameter of the commitment scheme SHC.

The Simulator $\mathsf{S}^{\mathsf{V}^*(\cdot;z_\lambda)}(x, 1^K, 1^{1/\varepsilon})$:

1. Sample a key $\mathsf{hk} \leftarrow \mathsf{SHC.Gen}(1^\lambda)$ and feed hk to V^*.
   In Unkeyed Setting: this step is skipped. $\mathsf{hk} \equiv 1^\lambda$ throughout.
2. Obtain a commitment c from V^*.
3. Sample a dummy first message $\alpha \leftarrow \Sigma.\mathsf{S}_1(x, 0^\lambda)$, and feed it to V.

Protocol 2

Common Input: an instance $x \in \mathcal{L}(\mathcal{R}) \cap \{0,1\}^\lambda$, for security parameter λ.

Auxiliary Input of P: a witness $w \in \mathcal{R}(x)$.

1. P samples a key $\mathsf{hk} \leftarrow \mathsf{SHC.Gen}(1^\lambda)$ and sends hk to V.
   In Unkeyed Setting: this message is not sent. $\mathsf{hk} \equiv 1^\lambda$ throughout the protocol.
2. V samples a random $\beta \leftarrow \{0,1\}^\lambda$, computes a commitment $c \leftarrow \mathsf{SHC.Com}(\beta; \mathsf{hk})$. It sends c to P and stores the commitment randomness r it used.
3. P starts emulating $\Sigma.\mathsf{P}(x, w)$, obtains a message α and sends α to V.
4. V opens the commitment by sending (β, r) to P.
5. P completes emulating $\Sigma.\mathsf{P}(x, w)$ with verifier message β, obtains message γ and sends it to V.
6. V runs $\Sigma.\mathsf{V}(x, \alpha, \beta, \gamma)$ and accepts if and only if $\Sigma.\mathsf{V}$ accepts.

Figure 2: A ZK Proof for **NP** relation $\mathcal{R}$.

4. If V^* successfully opens its commitment c to a challenge message β^*, proceed to the next step. Otherwise, abort and output the transcript so far.
5. Repeat the following $t = 3K/\varepsilon$ times:
   (a) Rewind V^* two Step 2.
   (b) Sample simulated $(\alpha, \gamma) \leftarrow \Sigma.\mathsf{S}(x, \beta^*)$, and feed α to V^*.
   (c) If V^* responds with β^*, complete the simulation with γ.
6. If the latter step failed, abort and output `timeout`.

Running Time. The simulator's's running time is $\varepsilon^{-1} \cdot K \cdot \mathrm{poly}(\lambda) = \mathrm{poly}(\lambda)$.

Zero Knowledge. To prove zero knowledge, we consider a hybrid simulator hS that is given also the witness as the input and always samples messages using $\Sigma.\mathsf{P}(x, w)$ instead of $\Sigma.\mathsf{S}$; that is:

- In Step 3, instead of sampling $\alpha \leftarrow \Sigma.\mathsf{S}_1(x, 0^\lambda)$, sample $\alpha \leftarrow \Sigma.\mathsf{P}_1(x, w)$.
- In Step 5, instead of sampling $(\alpha, \gamma) \leftarrow \Sigma.\mathsf{S}(x, \beta^*)$, sample $(\alpha, \gamma) \leftarrow \Sigma.\mathsf{P}(x, w; \beta^*)$.

Claim 8.2. *The view generated by the hybrid simulator is indistinguishable from that generated by the original simulator:*

$$\mathsf{S}^{\mathsf{V}^*(\cdot; z_\lambda)}(x, 1^K) \approx_c \mathsf{hS}^{\mathsf{V}^*(\cdot; z_\lambda)}(x, 1^K) \ .$$

Proof Sketch. The claim follows by the special zero-knowledge property of the Sigma protocol (or first-message indistinguishability), and a standard hybrid argument. □

We now prove that the simulator hS does not abort due to `timeout`, except with small probability.

Claim 8.3. *For sufficiently large $\lambda \in \mathbb{N}$,*

$$\Pr\left[\textit{simulator } \mathsf{hS} \textit{ outputs } \texttt{timeout}\right] \leq \varepsilon(\lambda)/2 \ .$$

Proof Sketch. The proof is by a reduction to the weak K-binding of the commitment scheme SHC and is similar to the proof of Theorem 4.3. Denote by B the set of challenges β that the verifier opens in Step 5. The simulator hS outputs `timeout` only if when sampling $\Sigma.\mathsf{P}_1(x,w)$ $t = 3K/\varepsilon$ times in Step 5, and adding any valid opening β to B, the $t+1$-st sample β^* is valid and is not covered by B. Thus, by Fact 2.1:

$$\Pr\left[\text{simulator } \mathsf{hS} \text{ outputs } \texttt{timeout}\right] \leq \frac{\mathbb{E}[B]}{t} \leq \frac{K + \Pr\left[|B| > K\right] \cdot t}{t} \leq \varepsilon/3 + \Pr\left[|B| > K\right] \ .$$

Furthermore, since hS runs in time $\mathrm{poly}(\lambda)$, it holds by the weak K-binding of SHC that

$$\Pr\left[|B| > K\right] \leq \lambda^{-\omega(1)} \ .$$

This completes the proof of the claim. □

Next, consider an augmented version hS' of hS that repeatedly performs Step 5 until success (rather than aborting after t failed attempts. By the last Claim 8.3, the statistical distance between the views generated by these two simulators is at most $\varepsilon/2$:

$$\mathsf{hS}^{\mathsf{V}^*(\cdot;z_\lambda)}(x,1^K) \approx_{\varepsilon/2} \mathsf{hS}'^{\mathsf{V}^*(\cdot;z_\lambda)}(x,1^K) \ .$$

It is left to note that the view generated by hS' is identically distributed to the view generated in an interaction:

$$\mathsf{hS}'^{\mathsf{V}^*(\cdot;z_\lambda)}(x,1^K) \equiv \langle \mathsf{P}(w) \leftrightarrows \mathsf{V}^*(z_\lambda)\rangle(x) \ .$$

Indeed, hS' first samples β^* from the same distribution as in the real protocol, and then samples (α,γ) from this distribution conditioned on β^*, by rejection sampling. □

Acknowledgements

We thank Zvika Brakerski, Ran Raz, and Vinod Vaikuntanathan for their collaboration in the early stage of this project. We also thank Benny Applebaum, John Steinberger, and Avi Wigderson for valuable discussions.

References

[AM13] Benny Applebaum and Yoni Moses. Locally computable UOWHF with linear shrinkage. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 486–502, 2013.

[App16] Benny Applebaum. Cryptographic hardness of random local functions - survey. *Computational Complexity*, 25(3):667–722, 2016.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 106–115, 2001.

[BBK+16] Nir Bitansky, Zvika Brakerski, Yael Tauman Kalai, Omer Paneth, and Vinod Vaikuntanathan. 3-message zero knowledge against human ignorance. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, pages 57–83, 2016.

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 505–514, 2014.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[BJY97] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 280–305, 1997.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Proceedings of the 24th Annual International Cryptology Conference*, pages 273–289, 2004.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In *TCC*, pages 595–613, 2009.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 151–168, 2011.

[Dam89] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.

[DCL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP proofs from an extractability assumption. In *Proceedings of the 4th Conference on Computability in Europe*, pages 175–185, 2008.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *J. ACM*, 50(6):852–921, 2003.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DPP93] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 250–265, 1993.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.

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

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *i*-hop homomorphic encryption and rerandomizable yao circuits. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 155–172, 2010.

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 658–667, 2001.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

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

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009.

[HIOS15] Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 173–190, 2015.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 201–215, 1996.

[HRVW09] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 611–620, 2009.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *Proceedings of the 18th Annual International Cryptology Conference*, pages 408–423, 1998.

[HV16] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 397–429, 2016.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989.

[Kat12] Jonathan Katz. Which languages have 4-round zero-knowledge proofs? *J. Cryptology*, 25(1):41–56, 2012.

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

[Kil94] Joe Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 466–477, 1994.

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:15, 2017.

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 536–547, 2008.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014.

[LS90a] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365, 1990.

[LS90b] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 353–365, 1990.

[Mer89] Ralph C. Merkle. A certified digital signature. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 218–238, 1989.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[OV12] Rafail Ostrovsky and Ivan Visconti. Simultaneous resettability from collision resistance. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:164, 2012.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In *Progressin Cryptology - VIETCRYPT 2006, First International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, pages 211–228, 2006.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 205–223, 2007.

A Multi-Collision Resistance in the Auxiliary-Input Random Oracle Model

In this appendix, we show that multi-collision-resistant hashing (with very good parameters) exists in hte model of *random oracles with auxiliary inputs* of Unruth [Unr07]. In this model, the adversary $\mathcal{A}$ consists of two parts $(\mathcal{A}_1, \mathcal{A}_2)$. First $\mathcal{A}_1(\mathcal{R})$, who is completely unbounded, obtains a full description of a (shrinking) random oracle $\mathcal{R}$ and outputs some (short) auxiliary information z about the random oracle. Then at the second stage $\mathcal{A}_2^{\mathcal{R}}(z)$ obtains this auxiliary input, as well as oracle access to $\mathcal{R}$, and attempts to output an K-collision in $\mathcal{R}$. We show that $\mathcal{A}$ cannot output collisions that are significantly larger than the size of the auxiliary input. Specifically, a random oracle is (K, γ)-collision resistant for $K(\lambda, \zeta) = O(\zeta/\lambda)$ and $\gamma(\lambda) = 2^{(1-\Omega(1))\lambda}$.

Proposition A.1. *For $\ell > \lambda$, let $\mathcal{R}$ denote a random function from the set of functions $\{0,1\}^\ell \to \{0,1\}^\lambda$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2^{(\cdot)})$, where $\mathcal{A}_1$ is an unbounded algorithm that outputs $z \in \{0,1\}^\zeta$, and $\mathcal{A}_2^{(\cdot)}$ is an unbounded algorithm that makes T oracle queries and outputs $(X_1, \ldots, X_K) \in \{0,1\}^{\ell \times K}$. Then*

$$\Pr_{\mathcal{R},\mathcal{A}}\left[\begin{array}{c|c} \forall i \neq j : X_i \neq X_j & z \leftarrow \mathcal{A}_1(\mathcal{R}) \\ \mathcal{R}(X_1) = \cdots = \mathcal{R}(X_K) & (X_1, \ldots, X_K) \leftarrow \mathcal{A}_2^{\mathcal{R}}(z) \end{array}\right] \leq 2^{-K(\lambda - \log T) + \zeta + \lambda} \quad .$$

Proof. We assume w.l.o.g that $\mathcal{A}$ is deterministic and that the oracle queries made by $\mathcal{A}_2$ are always distinct. Let $\mathcal{S}$ be the set of oracles $\mathcal{R}$ for which $\mathcal{A}$ successfully finds an K-collision. We show that

$$|\mathcal{S}| \leq 2^{\lambda \cdot 2^\ell - K(\lambda - \log T) + \zeta + \lambda} \quad ,$$

which suffices as the total number of oracles $\mathcal{R}$ is $2^{\lambda \cdot 2^\ell}$.

Concretely, we show how to uniquely represent any $\mathcal{R} \in \mathcal{S}$ using $\lambda \cdot 2^\ell - K(\lambda - \log T) + \zeta + \lambda$ bits. Fix any such $\mathcal{R} \in \mathcal{S}$, and consider a corresponding execution of $\mathcal{A}$. Let z be the resulting auxiliary input, let $\mathbf{X} = \{X_1, \ldots, X_K\}$ be the resulting multi-collision, and let $\mathbf{Q} = \{Q_1, \ldots, Q_T\}$ be the set of oracle queries that $\mathcal{A}_2^{\mathcal{R}}(z)$ makes. We represent $\mathcal{R}$ as follows:

- z,
- $L_{\mathbf{Q}\cap\mathbf{X}} = (i \in [T] \mid Q_i \in \mathbf{X})$,
- $\mathcal{R}(X_1)$,
- $L_{\mathbf{Q}\setminus\mathbf{X}} := (\mathcal{R}(Q_i) \mid Q_i \notin \mathbf{X})$,
- $L_{\overline{\mathbf{Q}\cup\mathbf{X}}} := (\mathcal{R}(X) \mid X \notin \mathbf{X} \cup \mathbf{Q})$.

Note that the auxiliary input size is ζ, the second set $L_{\mathbf{Q}\cap\mathbf{X}}$ can be represented by at most $|\mathbf{X}| \cdot \log T$, the second by λ bits, and the last two sets $L_{\mathbf{Q}\setminus\mathbf{X}}$ and $L_{\overline{\mathbf{Q}\cup\mathbf{X}}}$ by $\lambda \cdot (2^\ell - |\mathbf{X}|)$ bits (together). In sum, this representation costs $\lambda \cdot 2^\ell - K(\lambda - \log T) + \zeta + \lambda$ as required.

To see that this representation is unique, note that it allows to reconstruct $\mathcal{R}$ as follows. First emulate $\mathcal{A}_2^{\mathcal{R}}(z)$. When it makes its ith query Q_i, if $i \in L_{\mathbf{Q}\cap\mathbf{X}}$, answer with $\mathcal{R}(X_1)$. Otherwise answer from $L_{\mathbf{Q}\setminus\mathbf{X}}$, and keep track of the current location in the list. Finally, obtain all of $\mathbf{X}$. At this point, we have all the pairs $(X, \mathcal{R}(X))$ such that $X \in \mathbf{Q} \cup \mathbf{X}$, and we can complete $L_{\overline{\mathbf{Q}\cup\mathbf{X}}}$ to a full description of the function $\mathcal{R}$. □

B Construction of 3-Round Zero-Knowledge Argument

In this appendix, and for the sake of completeness, we provide the details of the 3-message zero-knowledge argument from Section 6, which are taken almost verbatim from [BBK+16].

B.1 Witness Indistinguishability with First-Message-Dependent Instances

We define 3-message WI proofs of knowledge where the choice of statement and witness may depend on the first message in the protocol. In particular, the first message is generated independently of the statement and witness. Also, while we do allow the content of the message to depend on the length ℓ of the statement, the message length should be of fixed to λ (this allows to also deal with statements of length $\ell > \lambda$). The former requirement was formulated in several previous works (see, e.g., [HV16]) and the latter requirement was defined in [BCPR14].

Definition B.1 (WIPOK with first-message-dependent instances)**.** *Let $\langle \mathsf{P} \leftrightarrows \mathsf{V} \rangle$ be a 3-message argument for $\mathcal{L}$ with messages $(\mathsf{wi}_1, \mathsf{wi}_2, \mathsf{wi}_3)$; we say that it is a WIPOK with first-message-dependent instances if it satisfies:*

1. **Completeness with first-message-dependent instances:** *For any instance choosing function X, and $\ell, \lambda \in \mathbb{N}$,*

$$\Pr\left[\mathsf{V}(x, \mathsf{wi}_1, \mathsf{wi}_2, \mathsf{wi}_3; r') = 1 \;\middle|\; \begin{array}{r} \mathsf{wi}_1 \leftarrow \mathsf{P}(1^\lambda, \ell; r) \\ (x, w) \leftarrow X(\mathsf{wi}_1) \\ x \in \mathcal{L}, w \in \mathcal{R}_{\mathcal{L}}(x) \\ \mathsf{wi}_2 \leftarrow \mathsf{V}(\ell, \mathsf{wi}_1; r') \\ \mathsf{wi}_3 \leftarrow \mathsf{P}(x, w, \mathsf{wi}_1, \mathsf{wi}_2; r) \end{array}\right] = 1 \ ,$$

   where $r, r' \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)}$ are the randomness used by P and V.

   The honest prover's first message wi_1 is of length λ, independent of the length ℓ of the statement x.

2. **Adaptive witness-indistinguishability:** *For any polynomial $\ell(\cdot)$, non-uniform PPT verifier $\mathsf{V}^* = \{\mathsf{V}^*_\lambda\}_{\lambda\in\mathbb{N}} \in \mathbb{P}$ and all $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathsf{V}^*_\lambda(x,\mathsf{wi}_1,\mathsf{wi}_2,\mathsf{wi}_3) = b \;\middle|\; \begin{array}{r}\mathsf{wi}_1 \leftarrow \mathsf{P}(1^\lambda,\ell(\lambda);r)\\ x,w_0,w_1,\mathsf{wi}_2 \leftarrow \mathsf{V}^*_\lambda(\mathsf{wi}_1)\\ \mathsf{wi}_3 \leftarrow \mathsf{P}(x,w_b,\mathsf{wi}_1,\mathsf{wi}_2;r)\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda) \ ,$$

where $b \leftarrow \{0,1\}$, $r \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)}$ is the randomness used by P, $x \in \mathcal{L} \cap \{0,1\}^{\ell(\lambda)}$ and $w_0, w_1 \in \mathcal{R}_\mathcal{L}(x)$.

3. **Adaptive proof of knowledge:** *there is a uniform PPT extractor $\mathcal{E}$ such that for any polynomial $\ell(\cdot)$, all large enough $\lambda \in \mathbb{N}$, and any deterministic prover P^*:*

$$\textit{if } \Pr\left[\mathsf{V}(\mathsf{tr};r') = 1 \;\middle|\; \begin{array}{r}\mathsf{wi}_1 \leftarrow \mathsf{P}^*\\ \mathsf{wi}_2 \leftarrow \mathsf{V}(\ell(\lambda),\mathsf{wi}_1;r')\\ x,\mathsf{wi}_3 \leftarrow \mathsf{P}^*(\mathsf{wi}_1,\mathsf{wi}_2)\\ \mathsf{tr} = (x,\mathsf{wi}_1,\mathsf{wi}_2,\mathsf{wi}_3)\end{array}\right] \geq \varepsilon \ ,$$

$$\textit{then } \Pr\left[\begin{array}{l}\mathsf{V}(\mathsf{tr};r') = 1\\ w \leftarrow \mathcal{E}^{\mathsf{P}^*}(1^{1/\varepsilon},\mathsf{tr})\\ w \notin \mathcal{R}_\mathcal{L}(x)\end{array} \;\middle|\; \begin{array}{l}\mathsf{wi}_1 \leftarrow \mathsf{P}^*\\ \mathsf{wi}_2 \leftarrow \mathsf{V}(\ell(\lambda),\mathsf{wi}_1;r')\\ x,\mathsf{wi}_3 \leftarrow \mathsf{P}^*(\mathsf{wi}_1,\mathsf{wi}_2)\\ \mathsf{tr} = (x,\mathsf{wi}_1,\mathsf{wi}_2,\mathsf{wi}_3)\end{array}\right] \leq \mathrm{negl}(\lambda) \ ,$$

where $x \in \{0,1\}^{\ell(\lambda)}$, and $r' \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)}$ is the randomness used by V.

Instantiation. Protocols with first-message-dependent instances follow directly from the WIPOK protocol constructed in [BCPR14], assuming ZAPs and non-interactive commitments (there, the first message is taken from a fixed distribution that is completely independent of the instance).

Next, we sketch how such a protocol can be constructed without ZAPs, but assuming keyless multi-collision-resistant hash functions.

The Lapidot-Shamir protocol. As observed in [OV12], the Lapidot-Shamir variant of the 3-message (honest verifier) zero-knowledge protocol for Hamiltonicity [LS90b] is such that the first and second messages only depend on the size of the instance $|x| = \ell$, but not on the instance and witness themselves. The protocol, in particular, supports instances up to size ℓ that depend on the prover's first message. However, the size of the first message wi_1 in the protocol is $|\mathsf{wi}_1| > \ell$. We, on the other hand, would like to allow the instance x to be of an arbitrary polynomial size in $|\mathsf{wi}_1|$, and in particular such that $|\mathsf{wi}_1| < \ell$.

We now sketch a simple transformation from any such protocol where, in addition, the verifier's message is independent of the first prover message, into a protocol that satisfies the required first-message dependence of instances. Indeed, the verifier message in the Lapidot-Shamir protocol is simply a uniformly random string, and hence the transformation can be applied here.

The Transformation. Let $\ell(\lambda) > \lambda$ be any polynomial function and let $\mathcal{H}$ be a keyless multi-collision-resistant hash function from $\{0,1\}^{\ell(\lambda)}$ to $\{0,1\}^\lambda$. In the new protocol $(\mathsf{P}_{\mathsf{new}}, \mathsf{V}_{\mathsf{new}})$, the prover computes the first message mes_1 for instances of length $\ell(\lambda)$. Then, rather than sending mes_1 in the clear, the prover $\mathsf{P}_{\mathsf{new}}$ sends $y = \mathcal{H}_\lambda(\mathsf{mes}_1) \in \{0,1\}^\lambda$. The verifier proceeds as in the previous protocol (P,V) (note that mes_1 is not required for it to compute mes_2). Finally the prover $\mathsf{P}_{\mathsf{new}}$ answers as in the original protocol, and also sends mes_1 in the clear. The verifier $\mathsf{V}_{\mathsf{new}}$ accepts, if it would in the original protocol and mes_1 is a preimage of y under $\mathcal{H}_\lambda$.

We first note that now the size of the instance ℓ can be chosen to be an arbitrary polynomial in the length $\lambda = |\mathsf{wi}_1|$ of the first WI message. In addition, we note that the protocol is still WI, as the view of the verifier $\mathsf{V}_{\mathsf{new}}$ in the new protocol can be perfectly simulated from the view of the verifier V in the old protocol, by hashing the first message on its own.

Finally, we observe that any prover $\mathsf{P}^*_{\mathsf{new}}$ that convinces the verifier in the new protocol of accepting with probability ε, can be transformed into a prover P^* that convinces the verifier of the original protocol, or to a collision-finder that finds many collisions.

B.2 1-Hop Homomorphic Encryption

A *1-hop homomorphic encryption scheme* [GHV10] allows a pair of parties to securely evaluate a function as follows: the first party encrypts an input, the second party homomorphically evaluates a function on the ciphertext, and the first party decrypts the evaluation result. (We do not require any compactness of post-evaluation ciphertexts.)

Definition B.2. *A scheme* $(\mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$*, where* $\mathsf{Enc}, \mathsf{Eval}$ *are probabilistic and* Dec *is deterministic, is a semantically-secure, circuit-private, 1-hop homomorphic encryption scheme if it satisfies the following properties:*

- **Perfect correctness:** *For any* $\lambda \in \mathbb{N}$*,* $x \in \{0,1\}^n$ *and circuit* C*:*

$$\Pr\left[\begin{array}{c}(\mathsf{ct},\mathsf{sk}) \leftarrow \mathsf{Enc}(x) \\ \widehat{\mathsf{ct}} \leftarrow \mathsf{Eval}(\mathsf{ct}, C) \\ \mathsf{Dec}_{\mathsf{sk}}(\widehat{\mathsf{ct}}) = C(x)\end{array}\right] = 1 \ .$$

  where the probability is over the coin tosses of Enc *and* Eval*.*

- **Semantic security:** *For any non-uniform PPT* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}} \in \mathbb{P}$*, every* $\lambda \in \mathbb{N}$*, and any pair of inputs* $x_0, x_1 \in \{0,1\}^{\mathrm{poly}(\lambda)}$ *of equal length,*

$$\Pr_{\substack{b\leftarrow\{0,1\} \\ (\mathsf{ct},\cdot)\leftarrow\mathsf{Enc}(x_b)}}[\mathcal{A}_\lambda(\mathsf{ct}) = b] \leq \frac{1}{2} + \mathrm{negl}(\lambda) \ .$$

- **Circuit privacy:** *The randomized evaluation procedure,* Eval*, should not leak information on the input circuit* C*. This should hold even for malformed ciphertexts. Formally, let* $\mathcal{E}(x) = \mathsf{Supp}(\mathsf{Enc}(x))$ *be the set of all legal encryptions of* x*, let* $\mathcal{E}_\lambda = \cup_{x\in\{0,1\}^n}\mathcal{E}(x)$ *be the set legal encryptions for strings of length* n*, and let* $\mathcal{C}_\lambda$ *be the set of all circuits on* λ *input bits. There exists a (possibly unbounded) simulator* $\mathcal{S}_{\mathsf{1hop}}$ *such that:*

$$\{C, \mathsf{Eval}(c,C)\}_{\substack{n\in\mathbb{N}, C\in\mathcal{C}_\lambda \\ x\in\{0,1\}^n, c\in\mathcal{E}(x)}} \approx_c \left\{C, \mathcal{S}_{\mathsf{1hop}}(c, C(x), 1^{|C|})\right\}_{\substack{n\in\mathbb{N}, C\in\mathcal{C}_\lambda \\ x\in\{0,1\}^n, c\in\mathcal{E}(x)}}$$

$$\{C, \mathsf{Eval}(c,C)\}_{\substack{n\in\mathbb{N} \\ C\in\mathcal{C}_\lambda, c\notin\mathcal{E}_\lambda}} \approx_c \left\{C, \mathcal{S}_{\mathsf{1hop}}(c, \perp, 1^{|C|})\right\}_{\substack{n\in\mathbb{N} \\ C\in\mathcal{C}_\lambda, c\notin\mathcal{E}_\lambda}} \ .$$

B.3 A 3-Round Zero-Knowledge Argument

Ingredients and notation:

- A two-message weak memory delegation scheme $(\mathsf{MD.Gen}, \mathsf{MD.Mem}, \mathsf{MD.Query}, \mathsf{MD.Prove}, \mathsf{MD.Ver})$ for γ-bounded computations.
- A semantically secure and circuit-private, 1-hop homomorphic encryption scheme $(\mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ as in Definition B.2.
- A 3-message WIPOK for **NP** with first-message-dependent instances as in Definition B.1. We denote its messages by $(\mathsf{wi}_1, \mathsf{wi}_2, \mathsf{wi}_3)$.
- A non-interactive perfectly-binding commitment scheme Com.
- For some $\mathsf{wi}_1, \mathsf{cmt}$, denote by $\mathcal{M}_{\mathsf{wi}_1,\mathsf{cmt}}$ a Turing machine that given memory $D = \mathsf{V}^*$ parses V^* as a Turing machine, runs V^* on input $(\mathsf{wi}_1, \mathsf{cmt})$, parses the result as $(u, \mathsf{wi}_2, \mathsf{q}, \widehat{\mathsf{ct}}_\tau)$, and outputs u.
- Denote by $\mathcal{V}_{\mathsf{param}}$ a circuit that has the string param hard-coded and operates as follows. Given as input a verification state vst for the delegation scheme:
  - parse $\mathsf{param} = (\mathsf{wi}_1, \mathsf{cmt}, \mathsf{q}, u, \mathsf{dig}, t, \pi)$,
  - return 1 ("accept") if either of the following occurs:
    - the delegation verifier accepts: $\mathsf{MD.Ver}(\mathsf{dig}, \mathcal{M}_{\mathsf{wi}_1,\mathsf{cmt}}, t, u, \mathsf{vst}, \pi) = 1$,
    - the query is inconsistent: $(\mathsf{q}, \mathsf{vst}) \notin \mathsf{MD.Query}(1^\lambda)$.[10]

  In words, $\mathcal{V}_{\mathsf{param}}$, given the verification state vst, first verifies the proof π that "$\mathcal{M}_{\mathsf{wi}_1,\mathsf{cmt}}(D) = (u, \cdots)$" where D is the database corresponding to the digest dig. In addition, it verifies that q is truly consistent with the coins vst. If the query is consistent, but the proof is rejected $\mathcal{V}_{\mathsf{param}}$ also rejects.
- Denote by $\mathbf{1}$ a circuit of the same size as $\mathcal{V}_{\mathsf{param}}$ that always returns 1.

We describe our 3-round zero-knowledge protocol in Figure 3.

[10] We can assume without loss of generality that vst consists of the random coins of $\mathsf{MD.Query}$ and thus can assume that it is easy to test if q is consistent with vst or not.

Protocol 3

Common Input: an instance $x \in \mathcal{L} \cap \{0,1\}^\lambda$, for security parameter λ.

P: a witness $w \in \mathcal{R}_\mathcal{L}(x)$.

1. P computes
   - wi_1, the first message of the WIPOK for statements of length $\ell_\Psi(\lambda)$, where ℓ_Ψ is the length of the statement Ψ defined in Step 3 below,
   - $\mathsf{cmt} \leftarrow \mathsf{Com}(0^\lambda, 0^{\log \gamma(\lambda)})$, a commitment to the all zero string,

   and sends $(\mathsf{wi}_1, \mathsf{cmt})$.
2. V computes
   - wi_2, the second message of the WIPOK.
   - $(\mathsf{q}, \mathsf{vst}) \leftarrow \mathsf{MD.Query}(1^\lambda)$, where we assume w.l.o.g that vst consists of the coins of MD.Query.
   - $(\mathsf{ct}_{\mathsf{vst}}, \mathsf{sk}) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(\mathsf{vst})$, an encryption of the verification state,
   - $u \leftarrow \{0,1\}^\lambda$, a uniformly random string,

   and sends $(u, \mathsf{wi}_2, \mathsf{q}, \mathsf{ct}_{\mathsf{vst}})$.
3. P computes
   - $\widehat{\mathsf{ct}} \leftarrow \mathsf{Eval}(\mathbf{1}, \mathsf{ct}_{\mathsf{vst}})$, an evaluation of the constant one function,
   - wi_3, the third WIPOK message for the statement $\Psi = \Psi_1(x) \vee \Psi_2(\mathsf{wi}_1, \mathsf{cmt}, \mathsf{q}, u, \mathsf{ct}_{\mathsf{vst}}, \widehat{\mathsf{ct}})$ of length $\ell_\Psi(\lambda)$ given by:

$$\left\{ \exists w \;\middle|\; (x,w) \in \mathcal{R}_\mathcal{L} \right\} \bigvee$$

$$\left\{ \exists \begin{array}{l} \mathsf{dig}, \pi, r_{\mathsf{cmt}} \in \{0,1\}^\lambda \\ t \leq \gamma(\lambda) \end{array} \;\middle|\; \begin{array}{l} \mathsf{cmt} = \mathsf{Com}(\mathsf{dig}, t; r_{\mathsf{cmt}}) \\ \mathsf{param} = (\mathsf{wi}_1, \mathsf{cmt}, \mathsf{q}, u, \mathsf{dig}, t, \pi) \\ \widehat{\mathsf{ct}} = \mathsf{Eval}(\mathcal{V}_{\mathsf{param}}, \mathsf{ct}_{\mathsf{vst}}) \end{array} \right\},$$

   using the witness $w \in \mathcal{R}_\mathcal{L}(x)$ for Ψ_1,

   and sends $(\widehat{\mathsf{ct}}, \mathsf{wi}_3)$.
4. V verifies the WIPOK proof $(\mathsf{wi}_1, \mathsf{wi}_2, \mathsf{wi}_3)$ for the statement Ψ and that $\mathsf{Dec}_{\mathsf{sk}}(\widehat{\mathsf{ct}}) = 1$.

Figure 3: A 3-message ZK argument of knowledge for **NP**.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il