On Valiant's Conjecture

Impossibility of Incrementally Verifiable Computation from Random Oracles

Mathias Hall-Andersen[0000−0002−0195−6659]⋆ and Jesper Buus Nielsen[0000−0002−7074−0683]⋆⋆

Aarhus University

Abstract. In his landmark paper at TCC 2008 Paul Valiant introduced the notion of "incrementally verifiable computation" which enables a prover to incrementally compute a succinct proof of correct execution of a (potentially) long running process. The paper later won the 2019 TCC test of time award. The construction was proven secure in the random oracle model without any further computational assumptions. However, the overall proof was given using a non-standard version of the random-oracle methodology where sometimes the hash function is a random oracle and sometimes it has a short description as a circuit. Valiant clearly noted that this model is non-standard, but conjectured that the standard random oracle methodology would not suffice. This conjecture has been open for 14 years. We prove that if the proof system can receive a long witness as input in an incremental manner and is also zero-knowledge then the conjecture is true. Valiant's original construction does not have these properties but can easily be extended to have them in his model. We relate our result to recent possibility and impossibility results for SNARKs and incrementally verifiable computation.

Keywords: Idealized Models, Lower Bounds, Separations and Impossibility Results, Proof Systems, Zero-Knowledge

1 Introduction

Incrementally Verifiable Computation. In his landmark paper Paul Valiant [21] introduced the notion of "incrementally verifiable computation" (IVC) which enables a prover to incrementally compute a succinct proof of correct execution of a (potentially) long running process. At any time the prover can suspend the computation and return a proof of correct execution leading up to the present state. This paper inspired a lot of later constructions, including modern recursive SNARK constructions, and won the 2019 TCC test-of-time award.

⋆ Funded by the Concordium Foundation.

⋆⋆ Partially funded by The Concordium Foundation; The Danish Independent Research Council under Grant-ID DFF-8021-00366B (BETHE); The Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM).

The methodology applied by Valiant is incremental. The computation applies the same step function T a number of ℓ times. There is an initial state M_0 and $M_i = T(M_{i-1})$. There is also an initial proof π_0, the empty string say. To construct the proof π_i that $M_i = T^i(M_0)$ one constructs a proof of knowledge of (M_{i-1}, π_{i-1}) for which it holds that $M_i = T(M_{i-1})$ and that π_{i-1} verifies the statement $M_{i-1} = T^{i-1}(M_0)$.[1]

The proofs are *succinct* in the sense that their have a size depend only poly-logarithmically on the number of steps i. Verification time also depends only poly-logarithmically on i. So neither the prover nor the verifier can just rerun the computation from M_0. Note that some notion of succinctness must follow from any reasonable notion of incrementality. Otherwise each new proof could just be recomputed from M_0, which hardly qualifies as "incremental".

The soundness of the recursive proof system is proven in the random oracle model without any further computational assumptions. However, Valiant need to apply a non-standard version of the random oracle model. When proving soundness of the proof system extending a proof by one step it is assumed that the hash function is a random oracle. However, when recursively proving that $\pi_{\ell-1}$ verifies it is assumed that the hash function has a short description as a circuit. This gives a somewhat interesting model where the hash function at different times has contradicting properties. The paper is very up front about this and justifies it by the conjecture that it seems that the standard random oracle methodology is not enough:

> ... When we try to recursively embed this system the recursion breaks down because, even at the first level of recursion, we are no longer trying to prove statements about classical computation but rather statements of the form "M with oracle access to $\mathcal{O}$ accepts the following string..." *Thus standard applications of random oracles do not appear to help.* [our emphasis]. ...
>
> –Paul Valiant[21]

In [8] Chiesa and Liu show impossibility results for proofs in relativized worlds, i.e., proofs of exactly the form "M with oracle access to $\mathcal{O}$ accepts the following string..." They show that $\mathrm{DTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(o(t), o(t))^{\mathcal{O}}$ and $\mathrm{NTIME}(t)^{\mathcal{O}} \not\subseteq \mathrm{PCP}(\mathsf{poly}(t), o(t))^{\mathcal{O}}$, which can informally be interpreted as not *all* statements of the form "M with oracle access to $\mathcal{O}$ accepts the following string..." can have a non-trivial proof where not all the oracle queries of M are checked by the verifier. But if the verifier checks all oracle queries of the prover and each steps makes just one query then the verifier is not succinct. As noted in [8] this "gives strong evidence that Valiant's approach was in some sense justified." However, it does not conclusively rule out that Valiant's approach can be instantiated in the standard random oracle model. It cannot be ruled out

[1] As detailed later this description is oversimplified but will suffice for our discussion. The real recursive strategy is more involved to tame the complexity of knowledge extraction.

that a proof system can be constructed where the verifier is simple enough that it does not fall prey to the Chiesa-Liu results, as they only prove that not *all* statements have such a proof.

And even if we could rule out the explicitly *recursive* strategy, where we extend a proof by proving knowledge of an accepting sub-proof, then it might still be possible to do *incremental* proofs in the random oracle model using some other strategy. In particular, the end result of Valiant's approach is to give a proof about random oracle devoid computation, which is not ruled out by the Chiesa-Liu results. As already noted by Valiant:

> ... It remains an interesting question whether the goals of this paper may be attained in some other way using random oracles. ...
>
> –Paul Valiant[21]

In the present paper we show that Valiant was correct and that indeed the standard random oracle model is not sufficient for incremental proofs. We rule out not just explicitly recursive designs, but general incremental designs. As we discuss below we do not prove impossibility for the exact setting studied by Valiant: we need to assume two additional but natural properties of the proof system, which Valiant's construction can easily be extended to have.

Non-Deterministic Computation. The first additional assumption we need is that the ongoing computation can receive a long witness as input in an incremental manner. The verifier is assumed to only have access to a short instance. In a modern setting this could be a verifier knowing only the genesis block and a recent block of a blockchain and the prover wants to succinctly prove that the blockchain has some property, like the verifier having been paid a certain amount defined by the overall activity on the blockchain. Here the genesis block plus the recent block is the short instance and the blockchain is the long witness. It is a natural question whether the proof can be computed incrementally, say by consuming the blockchain block-by-block.

The original notion of IVC considers only deterministic computation: the verifier is provided with a Turing machine and the prover convinces the verifier that the provided state is reached after executing the Turning machine for some number of steps. Motivated by "distributed computation" Chiesa and Tromer [10] subsequently generalized IVC to the powerful notion of "Proof-Carrying Data" (PCD) in which the correct computation of a function taking multiple inputs can be proven given proofs of correctness for each of the inputs, e.g., the computation of $F(G_1(w_1), G_2(w_2))$ can be proven given $y_1 = G_1(w_1)$, $y_1 = G_2(w_2)$ and corresponding proofs-of-knowledge π_1, π_2 for w_1, w_2. For our impossibilities we use the abstraction of "non-deterministic incrementally verifiable computation", which is a special case of PCD with "fan-in" 1 with the same function applied in each step, or equivalently, a generalization of IVC where each step of the Turning machine may take a witness.

Zero-Knowledge via Reprogramming the Random Oracle. Our impossibility results also assume that the incremental proof is zero-knowledge. Succinct arguments already information theoretically hides most of the witness, as the proof is much shorter than the witness. For general PCD zero-knowledge is even a natural requirement: different steps of the computation may be performed by mutually distrustful parties which do not want to share their secrets.

The notion of ZK which we consider is as follows. The simulator is given a correct state and proof for step i and must then produce a proof for step $i+1$ without knowing the corresponding witness. This simulated proof should look indistinguishable to a PPT adversary. The simulator may inspect and reprogram the random oracle, but to make its job harder we give the adversary access to querying the random oracle *before* the simulation is made. This can be seen as giving a limited form of auxiliary information on the oracle to the adversary. It is discussed by Goldreich in [15] why auxiliary information is important for composability of ZK proofs. As discussed by Unruh in [20] it is also essential for composability to give the adversary auxiliary information on the random oracle. Otherwise the security assumption assumes the random oracle appears magically *after* the adversary specified its strategy. In this case each proof would need its own fresh random oracle even for sequential composition. For the case of IVC this would require that a fresh random oracle appears after each proof step, which is a very unnatural model.

We note that our impossibilities hold under any computational assumptions, as long as the ZK simulation proceeds only by reprogramming the random oracle. It therefore does not, e.g., rule out constructions from common reference strings where the simulation relies on the trapdoor of the CRS.

In This Paper. In this paper the main result is to show that succinct, zero-knowledge non-deterministic IVC from random oracles is impossible in the following two cases.

1. There exist collision intractable hash functions and the proof system has *knowledge soundness*. Knowledge soundness and zero-knowledge may depend on standard model computational assumptions including non-falsifiable assumptions.
2. There exist perfectly binding rerandomizable commitments and the proof system has *soundness*. Both soundness and zero-knowledge may depend on standard-model computation assumptions including non-falsifiable assumptions.

Universal Knowledge Soundness. For the first result we consider a notion of knowledge extraction with a universal extractor: we require that there exists a poly-time extractor which works for all poly-time adversaries. The extractor is given the code of the adversary as input, so it can still use *non-blackbox extraction*. However, quantifying the extractor before the adversary makes it hard to use for instance knowledge-of-exponent assumptions or any other assumption of the form "for all adversaries there exists an extractor such that ...". We

note that the first result still stands if one makes knowledge assumptions or, in general, any non-falsifiable assumptions. The only restriction we make is that our definition of universal knowledge soundness makes it harder to exploit these assumptions.

We now give an overview of our proof techniques and discuss the results in more detail, discuss generalisations, and compare to existing (im)possibility results for PCDs and SNARKs.

PCD via Recursion. Above we discussed recursive proofs as being simply sequential. To avoid confusion let us note that in Valiant's original paper [21] IVC is constructed using a tree of linear-time extractable CS proofs [19] in which the leafs each prove a step of the execution, while the parents (a CS proof) proves the correct execution of the verifier on the two children (CS proofs) which each cover half of the computation time. By maintaining just $\log(T)$ such proofs the computation can be extended in the obvious way. The tree structure is essential to ensuring polynomial-time extraction, since the linear-time knowledge extractor need only be applied $\log(T)$ times recursively to extract the entire computational trace. In later works [4,3] from zk-SNARKS the proofs are composed iteratively, which implies that the proof as far as we know only is sound for computation of constant depth. Lately, in practical schemes/deployments, the efficiency of the recursive extraction is largely ignored: instead showing that a single level of recursion is extractable [6,5]. Common to all known constructions is the non-blackbox use of (parts of) the verifier.

Incremental PCD. Our results apply not only to recursive proofs but to succinct incremental proofs in general. We look at incremental proofs produced by some ℓ number of succinct steps. By succinct we mean that the state of the prover passed on from one step to the next has size $\mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$, where $\mathcal{R}$ is the PPT relation checking that one step was computed correctly, λ is the security parameter, and ℓ is the number of steps. Each computation of a proof need not be state bounded, only the state passed on to the next step. We also require that the verifier has running time $\mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$.

Technical Overview We sketch the main ideas behind the impossibility results. For all results the witness for an ℓ-step proof is a long random vector $\vec{w} = (w_1, \ldots, w_\ell)$, where w_i is given (only) in step i. Each w_i is security parameter long. We first prove that no adversary (cheating prover) can produce an accepting proof if there is some index i such that we do not give it the witness w_i used in iteration i. We sketch why this is true.

For the case of collision intractable hash functions the computation computes a Merkle-Damgård hash of $\vec{w}$, consuming one w_i per step. If the prover could succeed in producing an accepting proof without using w_i, then we could apply the knowledge extractor to the accepting proof and recover w_i. It is easy to see that this can be used to violate collision intractability.

For the case of perfectly binding rerandomizable commitments the instance is a long sequence of commitments $c_0, c_1, \ldots$ where the claim is that the sequence

was produced as a sequence of rerandomisations of the previous commitment. Step i of the proof gets as input c_{i-1} and c_i and the witness is the randomness used to produce c_i as a rerandomisation of c_{i-1}. The commitment c_0, of step 1, is a commitment of 0. By perfect binding it follows that for true instances the commitment c_ℓ of step ℓ is also a commitment of 0. The missing witness will now be the randomness used for a rerandomisation in some step i. If the prover is not given this randomness it cannot distinguish a commitment c_i of 0 from a commitment of 1. Hence we can do a switch from a commitment c_{i-1} of 0 to a commitment c_i of 1. So if for a true instance the prover could succeed without w_i then it could also succeed for a false instance, breaking soundness.

We then finish the proofs by showing that if the verifier does not make $\Theta(\ell)$ queries to the random oracle then there exists an adversary producing an accepting proof and which does *not* use all witnesses, giving a contradiction.

This proof only uses that there is a zero-knowledge simulator in the random oracle model: it can simulate a given step if allowed to reprogram the random oracle. The indistinguishability of the real view and the simulated view may depend on other computational assumptions. For each step m and each step $n > m$ we use that the simulator works by programming the oracle to argue that the verifier of step n must make a check *related to* the proof of step m. To see this note that if we simulate step m and the simulator reprograms the points S_m then the verifier of step n must check a point $x \in S_m$. Namely, if we simulate step m then we do not need w_m. Therefore the proof must reject: we already argued that all successful provers use all witnesses. But if the verifier of step n does not query $x \in S_m$, then the reprogrammed random oracle will look like the real random oracle to this particular verifier and it must therefore accept the proof (as it accepts the proof when the oracle is reprogrammed, by definition of zero-knowledge).

Let $x_{m,n}$ denote a query by verifier n related to proof m. This is a random variable. The instances m and n range from 1 to ℓ, so there are $\Theta(\ell^2)$ of the random variable $x_{m,n}$. The main challenge of the proof is to prove that they are disjoint enough that we force some verifier to make $\Theta(\ell)$ queries, which is not allowed as we assume the verifier has running time in $\mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$. The main challenge in proving this is that we cannot make a world where we simulate all proofs, as some verifier will then surely check some reprogrammed point and reject. Also, we cannot easily define $x_{m,n}$ in the real world where step m is run honestly, as there is no notion of S_m. We therefore need to capture $x_{m,n}$ in the world where step m is simulated using some poly-time observable. The observable we use is essentially "x was not queried before step m and it got queried after step m". We show this captures $x_{m,n}$ when step m is simulated. The reason is that by our notion of zero-knowledge a reprogrammed point cannot have been queried before it was reprogrammed. And by arguments from above, some reprogrammed point must be queried by a verifier in the future. We then argue that this poly-time observable $x_{m,n}$ must exist in the real world too, or zero-knowledge was broken. We then argue that the definitions of the poly-time

observables are such that the $\theta(\ell^2)$ points $x_{m,n}$ are disjoint enough. This is done in Lemma 4.

Generalizations Our results apply directly to schemes which only rely on random oracles, like that of Valiant [21] (based on CS proofs) and recursive Fractal [9]. However, we emphasise that our results are not oracle separation results. We do not give the adversary access to for instance an NP oracle which can break all cryptography except the random oracle. As a result the impossibility results apply even in presence of additional computational assumptions.

Specifically, the zero-knowledge may depend on computational assumptions, as long as these are not phrased via relativized worlds extra to the random oracle model. The results therefore stand also if there exist for instance trapdoor permutations or indistinguishability obfuscation. Our results do not rule out constructions where zero-knowledge is proven in for instance the generic group model, as it is relativized. Similarly, in result 1 knowledge soundness, and in result 2 the soundness, may depend on computational assumptions, as long as these are not phrased via relativized worlds extra to the random oracle model.

Although we primarily focus on random oracles, the result can easily be generalized to $O(\mathsf{poly}(\lambda))$-local oracles, i.e., where responses to queries might be dependent in a bounded manner: All queries can be divided into disjoint sets P_i of size $|P_i| = O(\mathsf{poly}(\lambda))$ and replies to queries in different sets are independent. For a given verifier we can simply look at the one which if it queries $x \in P_i$ then it queries all $x' \in P_i$. This still gives it running time $O(\mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell))$. And we can now look at the proof system as using a 1-local oracle with larger replies. And it is easy to see that our results still apply to such 1-local oracles. Note that for instance oracles like "generic (bilinear) groups" are *not* $O(\mathsf{poly}(\lambda))$-local, as the group law correlates all replies.

1.1 Relation to other results.

The impossibility of Gentry-Wichs [14] for adaptively sound zk-SNARGs applies also to zero-knowledge, non-deterministic IVC, so one cannot hope to construct non-deterministic IVC from falsifiable assumptions. However, this does not rule out a construction of IVC in the RO model. In particular, unlike non-deterministic IVC, there *are* known constructions of zk-SNARKs in the random oracle model, e.g., classic CS proofs [19] from PCPs and the compiler of Ben-Sasson et al. [2] applied to round-by-round sound Holographic IOPs like Fractal [9] and zk-STARKs [1]. Below we compare to other results. The discussion is summarised in Fig. 1.

We note that while Gentry and Wichs [14] proved the impossibility of a security reduction, this paper proves impossibility a construction: Gentry-Wichs shows that any SNARG *cannot have a black-box reduction* to a game-based definition, while ours, shows that any construction of a zero-knowledge non-deterministic IVC in the random oracle model *has an efficient adversary breaking it.*

	CRS	RO	Non-BB RO
IVC (**P**)	✓ [17]	?	✓
Batch Arguments ($\mathbf{NP}^{\ell}$)	✓ [13]	✓	✓
Non-Adaptively Secure SNARGs (**NP**)	✓ [18]	✓	✓
Adaptively Secure SNARGs (**NP**)	✗ [14]	✓ [19]	✓
Non-Deterministic IVC (**NP**)	✗ [14]	✗ [Here]	✓ [21]

Fig. 1. An overview of known constructions (✓), impossibility results (✗) and open questions ? in the existing literature, in relation to our result (✗). If a cell has no citation it is implied by the value in another cell in the table, for brevity we only include one construction per cell. Note that Valiant's original construction [21] of IVC can easily be extended to the non-deterministic setting. CRS stands for the model with a common reference string and no RO. RO stands for the model with a standard RO and no CRS. Non-BB RO stands for the model with non-standard RO *a la* Valiant and no CRS.

Common Reference String. A number of recent results have probed the limits of the Gentry and Wichs separation [14] of adaptively secure SNARKs from falsifiable assumptions: Tauman Kalai, Paneth and Yang constructed [17] a delegation scheme for **P**, deterministic IVC, from falsifiable assumptions on bilinear pairings with a CRS. Choudhuri and Jain recently constructed [13] batch arguments for **NP** from standard assumptions and CRS. Lastly Lipmaa and Pavlyk [18] recently resolved an open problem in the Gentry and Wichs paper by proving that there exists a construction of non-adaptively sound SNARGs from falsifiable assumptions.

Random Oracle. Adaptively secure (zk)SNARKs has been widely constructed in the random oracle model (without a CRS) [19,12,9,11,1], including a recent tight lower bound on the number of random oracle queries [16]. We prove that similar positive results cannot be obtained for the *incremental equivalent* of zkSNARK: non-deterministic IVC in the random oracle model. We tackle the impossibility of *non-deterministic* IVC in the random oracle model, since proving the impossibility of *deterministic* IVC (in any model) must preclude the trivial scheme in which the oracle is not used and the poly-log verifier simply decides membership given a poly-log certificate computed by the prover. Impossibility of this seems closely related to proving $\mathrm{P} \not\subseteq \mathrm{NTIME}(O(\log^c n))$— which remains an open problem in complexity.

Non-Blackbox Random Oracle. Constructions of (non-deterministic) IVC relying on "non-blackbox" use of the random oracle exists in the literature [9,21]: in such schemes the security proof is in the random oracle model, however, the construction relies on the oracle having a short description. This is an interesting model where the scheme does not exist in the idealised model in which it is proven secure. Such use of the random oracle often arises implicitly [9,21] when a SNARK in the RO model is heuristically converted to a SNARK in the plain model, by replacing the random oracle with a concrete cryptographic hash

function, and used to prove the satisfiability of the verification circuit for the same SNARK.

Non-Deterministic IVC In Relativized Worlds. Non-deterministic IVC trivially exists in worlds with certain types of oracles, the question is how "complicate" this oracle needs to be: motivated both by theoretic curiosity and practical desire to heuristically instantiate the oracle in the standard model. Our results shows that to allow zero-knowledge, incremental PCD the oracle must be non-local.

As discussed above, Chiesa and Liu [8] showed that it is impossible to construct non-trivial PCPs of random oracle computation (e.g., circuits with RO gates). This rules out most hope constructing IVC by proving the correct execution of a verifier in the random oracle model but does not exclude that other design would allow for IVC in the RO model.

On the positive side, the original construction of Proof-Carrying Data (PCD) [10] (a generalization of non-deterministic IVC) by Chiesa and Tromer is in a world with a signed random oracle: a random oracle which additionally returns a signature on the (query, response) pair, this allows verifying the validity of oracle queries without need for oracle computation, by simply verifying the signature, this enables a recursive construction similar to Valiant but without contradictions. This oracle is non-local as all replies are signed with the same key. Recently Chen, Chiesa and Spooner [7] demonstrated that SNARKs exists for the relativized world of low-degree polynomial oracles using an accumulation scheme [6] for oracle query/response pairs. This scheme is non-local as replies are related by the polynomial.

1.2 Can we Drop the ZK Assumption?

Our impossibility result applies only to the setting where a large witness is consumed piecemeal and where the proof is zero-knowledge. Since the original construction of Valiant, and modern uses of recursive proofs in the RO model, easily generalises to have these properties the result seems pessimistic, but it keeps open the possibility of getting non-deterministic IVC in the random-oracle model which is *not* zero-knowledge. We prove a secondary result showing that there does not seem to be any easy way to construct this. Namely, the proof system would have to have an unnatural looking property that the proof system itself makes queries it cannot "remember" later. More specifically, we can show that non-deterministic IVC from random oracles is impossible in the following case:

3. If there exists collision intractable hash functions and the proof system has *blackbox knowledge soundness* and the proof system has a property informally stated as follows: it can with non-negligible probability be predicted for all queries made by the *prover* whether they are fresh or it made them before.

This result shows that even if we drop the assumption of zero-knowledge one cannot get incremental proofs, but now using an assumption that the freshness of queries can be determined with non-negligible probability. Note that this

assumption is non-trivial as the proof system is succinct, so it cannot just remember all queries of all previous steps. However, it seems hard to use forgotten queries in a constructive way. We discuss the assumption further in Section 5.

For result 3 we use a different proof approach. Here we observe that if the final verifier, of step ℓ, is succinct, then it makes a number of queries to its oracle essentially independent of ℓ. So by setting ℓ large enough we can create a polynomially long *stretch* from step p_1 to step p_2 such that no *fresh* query made by a proof in steps $[p_1, p_2]$ will be queried by the final verifier. A fresh query is one which was not also made before the stretch. We then create an adversary which picks the witnesses used in steps $[p_1, p_2]$ independent of the witnesses used outside the interval and independent of all *queries* made before steps $[p_1, p_2]$.

During the stretch we let the adversary use a *simulated oracle* instead of the real one for all fresh queries. It simply samples the oracle replies itself without asking the real oracle. This will still give an accepting proof as the final verifier does not make queries corresponding to fresh queries by the prover during the stretch. Hence the real oracle and the simulated one will look the same to the final verifier. Letting the adversary use a simulated oracle $\widetilde{\mathcal{O}}$ during the stretch ensures that the blackbox extractor gets no information on the stretch witnesses: the adversary makes no queries to its oracle during the stretch and is therefore opaque to the blackbox extractor.

Hence all the information that the extractor gets on the stretch witnesses is via queries made by the adversary to its oracle during the proofs *after* step p_2 in the main execution. Intuitively this information can be no larger than the state σ_2 of the prover after step p_2. We could give σ_2 to the extractor and let it finish the proof itself. If the proof system is succinct then we can pick $p_2 - p_1 > |\sigma_2|$ to ensure that σ_2 information theoretically cannot encode all the stretch witnesses. This shows that a blackbox extractor cannot compute the stretch witnesses from *blackbox* access to the adversary, violating knowledge soundness.

The above argument uses that we could give the state of the prover after the step p_2 to the adversary and let it finish the proof itself. But note that between steps p_1 and p_2 we used a simulated oracle $\widetilde{\mathcal{O}}$. To appeal to correctness of the proof system when we let the adversary finish the proof it must know $\widetilde{\mathcal{O}}$ and must be able to determine which queries to send to $\widetilde{\mathcal{O}}$ if they are made again by later steps in the proof. And we should give the adversary this ability by giving it concise information, or we might be leaking the stretch witness to it. We can implement $\widetilde{\mathcal{O}}$ as a pseudo-random oracle and just give the short seed to the adversary. However, we cannot give it the set of all queries made between p_1 and p_2 as the query points themselves might encode information about the stretch witnesses. This is why we need to assume that there is a concise mechanism to determine whether or not a query made by a later step in the proof is fresh, so we know whether to reply with the real random oracle or the simulated $\widetilde{\mathcal{O}}$. The mechanism need not be perfect. If it passes on a state which is some constant fraction shorter than the stretch witness and works with non-negligible probability we can still get a contradiction to extracting the stretch witnesses when the mechanism works, by making the stretch long enough.

2 Definitions

Formally our model of computation is repeated application of a Boolean circuit T which encodes the "transition function". Formally, we show impossibility of $\mathcal{O}$-IVC supporting particular sets of transition functions $\mathcal{T}$, in particular we show impossibility for schemes supporting all Boolean circuits.

Definition 1 (Transition Functions). *Let $\mathcal{T}$ be a set of Boolean circuits, $T \in \mathcal{T}$:*

$$T : \{0,1\}^{|M|} \times \{0,1\}^{|w|} \to \{0,1\}^{|M|}$$

Definition 2 (Repeated Application of T). *We denote by T^ℓ the function that applies T ℓ-times to a state M_0 with witnesses $w_1, \ldots, w_\ell$. Formally, let $T^0 = id$ (the identity function) and define T^ℓ for $\ell > 0$ recursively as:*

$$\begin{array}{l} T^\ell(M_0, \vec{w} = (w_1, \ldots, w_\ell)) \\ \hline 1: \quad M_{\ell-1} \leftarrow T^{\ell-1}(M, (w_1, \ldots, w_{\ell-1})) \\ 2: \quad \textbf{return } T(M_{\ell-1}, w_\ell) \end{array}$$

We define the relation/language defined by $\mathcal{T}$ as follows:

$$(x, \vec{w}) \in \mathcal{R}_\mathcal{T} \iff x = (T, M_0, M_\ell, \ell) \wedge M_\ell = T^\ell(M_0, \vec{w})$$

$$x = (T, M_0, M_\ell, \ell) \in \mathcal{L}_\mathcal{T} \iff \exists \vec{w} \text{ st. } (x, \vec{w}) \in \mathcal{R}_\mathcal{T}$$

Definition 3 (Non-Deterministic $\mathcal{O}$-IVC.). *A non-deterministic $\mathcal{O}$-IVC scheme for a set of transition functions $\mathcal{T}$ consists of two PPT $\mathcal{O}$-algorithms:*

$\mathbb{P}^\mathcal{O}(x_\ell = (T, M_0, M_\ell, \ell), w_\ell, \pi_\ell) \mapsto \pi_{\ell+1}$**.** *A PPT algorithm taking a description of the state transition T, the initial state M_0, the current state M_ℓ, the length of the computation ℓ, some additional input w_ℓ and an accepting proof π_ℓ of $x_\ell \in \mathcal{L}_{(T,\ell)}$. Then outputs a proof $\pi_{\ell+1}$ of $x_{\ell+1} = (T, M_0, M_{\ell+1}, \ell+1) \in \mathcal{L}_\mathcal{T}$ where $M_{\ell+1} = T(M_\ell, w_\ell)$. Note that the prover is not given the witness for $x_\ell \in \mathcal{L}_\mathcal{T}$.*

$\mathbb{V}^\mathcal{O}(x_\ell = (T, M_0, M_\ell, \ell), \pi_\ell) \mapsto \{\top, \bot\}$**.** *Verifies a proof π of the statement $(T, M_0, M_\ell, \ell) \in \mathcal{L}_\mathcal{T}$; i.e., there exists a sequence of witnesses $\vec{w}$ such that $M_\ell = T^\ell(M_0, \vec{w})$.*

We assume for notational convenience (and without loss of generality) that the proof for the trivial statement $x_0 = (T, M_0, M_0, 0)$ (i.e. application of T zero times to M_0 yields M_0) is $\pi_0 = \epsilon$ (the empty string). Additionally we require that $\mathbb{P}^\mathcal{O}$ and $\mathbb{V}^\mathcal{O}$ satisfy completeness:

(Perfect) Completeness: *Informally states that if a proof is produced correctly, it verifies.*
Formally, for all $(T, \vec{w}, M_0, \ell)$:

$$\Pr\left[\mathbb{V}^{\mathcal{O}}(x_\ell = (T, M_0, M_\ell, \ell), \pi_\ell) = \bot \;\middle|\; \begin{array}{c} \forall i \in [\ell] : \\ M_i = T(M_{i-1}, w_i); \\ x_i = (T, M_0, M_i, i) \\ \pi_i \leftarrow \mathbb{P}^{\mathcal{O}}(x_i, w_i, \pi_{i-1}) \end{array}\right] = 0$$

We assume perfect completeness for simplicity, however all our results easily generalize to the slightly weaker case where the scheme has a negligible probability of failure. We do not require that the prover can extend any accepting proof, only those honestly produced.

Remark 1. An alternative definition (similar to [6]) would instead have $\mathbb{P}^{\mathcal{O}}$ and $\mathbb{V}^{\mathcal{O}}$ take a description of an NP relation $\mathcal{R}$ rather than a description of a poly-time computable function T. In which case $\mathbb{P}$ proves knowledge of a w st. $(x = (M, M'), w) \in \mathcal{R}$ (rather than $M' = T(M, w)$). We note that these two definitions are trivially equivalent, but find the definition presented here simpler notationally: in particular the knowledge extractor does not need to explicitly extract a sequence of statements.

We employ both standard soundness and knowledge soundness definitions in different flavors of our impossibility results.

Definition 4 ((Computationally) Sound Non-Deterministic $\mathcal{O}$-IVC). *The probability of any PPT adversary producing an accepting proof of a false statement is negligible:*

$$\forall \mathcal{A}^{(\cdot)} : \Pr\left[\mathbb{V}^{\mathcal{O}}(x, \pi) = \top \wedge x \notin \mathcal{L} \;\middle|\; (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda);\right] \leq \mathsf{negl}(\lambda)$$

Many languages are trivial (i.e., every instance is in the language), in which case knowledge soundness is required for non-deterministic IVC to be non-trivial. We consider two standard variations: (1) knowledge soundness with a universal non-blackbox extractor (the weaker definition), in which the extractor is given access to the code of the adversary. (2) knowledge soundness with a blackbox extractor (the stronger definition), in which the extractor is given only blackbox (rewinding) access to the adversary.

Definition 5 (Universal Non-Blackbox Knowledge Soundness Non-Deterministic $\mathcal{O}$-IVC). *There exists a PPT algorithm $\mathbb{E}$ st. for all PPT $\mathcal{A}^{\mathcal{O}}$ when $\mathcal{A}^{\mathcal{O}}$ outputs an accepting proof, the extractor given a description of the adversary, recovers a valid witness $(w_1, \ldots, w_\ell)$ given $\mathcal{A}^{(\cdot)}$ except with negligible probability. Formally:*

$$\exists\ \mathbb{E}\ \textit{st.}\ \forall \mathcal{A}^{(\cdot)} :$$

$$\Pr\left[\begin{array}{c} \mathbb{V}^{\mathcal{O}}(x, \pi) = \top \\ \wedge\ T^\ell(M_0, \vec{w}) \neq M_\ell \end{array} \;\middle|\; \begin{array}{c} (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda); \vec{w} \leftarrow \mathbb{E}^{\mathcal{O}}(1^\lambda, x, \mathcal{A}^{(\cdot)}); \\ x = (T, M_0, M_\ell, \ell) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 6 (Blackbox Knowledge Sound Non-Deterministic $\mathcal{O}$-IVC). *There exists a PPT algorithm $\mathbb{E}$ st. for all PPT $\mathcal{A}^{\mathcal{O}}$ when $\mathcal{A}^{\mathcal{O}}$ outputs an accepting proof, the extractor given black-box (rewinding) access to the adversary $\mathcal{A}^{(\cdot)}$ recovers a valid witness $(w_1, \ldots, w_\ell)$, except with negligible probability. Formally:*

$$\exists\ \mathbb{E}\ \textit{st.}\ \forall \mathcal{A}^{(\cdot)}:$$
$$\Pr\left[\begin{array}{c} \mathbb{V}^{\mathcal{O}}(x,\pi)=\top \\ \wedge\ T^{\ell}(M_0,\vec{w})\neq M_\ell \end{array}\ \middle|\ \begin{array}{c}(x,\pi)\leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda);\vec{w}\leftarrow \mathbb{E}^{\mathcal{O},\mathcal{A}^{(\cdot)}}(1^\lambda,x);\\ x=(T,M_0,M_\ell,\ell)\end{array}\right]\leq \mathsf{negl}(\lambda)$$

Additionally we may require that the IVC scheme is zero-knowledge, which informally states that any step can be simulated by programming the oracle and that simulated proofs are indistinguishable from real proofs. Note that the statement to be simulated includes an accepting proof of correctness for M_ℓ.

Definition 7 ((Computational) Zero-Knowledge Non-Deterministic $\mathcal{O}$-IVC). *There exists a PPT (in $\lambda, |T|, \ell$) algorithm $\mathbb{S}^{(\cdot)}$ which for any $T \in \mathcal{T}$, $\ell = \mathsf{poly}(\lambda)$, $x = (T, M_0, M_\ell, \ell) \in \mathcal{L}_T$, w, and accepting π ($\mathbb{V}^{\mathcal{O}}(x,\pi) = \top$), $\mathbb{S}^{\mathcal{O}}$ outputs an accepting proof and a set of (re)programmings $\mathcal{Q} = \{(Q_i, R_i)\}_i$ for the oracle fooling any PPT adversary.*

$\exists \mathbb{S}^{\mathcal{O}}\, \forall \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)\, \forall\, T \in \mathcal{T}, x = (T, M_0, M_\ell, \ell) \in \mathcal{L}_{\mathcal{T}}, w, \pi$ *st.* $\mathbb{V}^{\mathcal{O}}(x,\pi) = \top$:

$$\Pr\left[b = b' \;\middle|\; \begin{array}{r} M_{\ell+1} = T(M_\ell, w) \\ \mathsf{h} \leftarrow \mathcal{A}_1^{\mathcal{O}}(1^\lambda, M_{\ell+1}, x, \pi) \\ \pi_0' \leftarrow \mathbb{P}^{\mathcal{O}}(x, \pi, w) \\ (\mathcal{Q}, \pi_1') \leftarrow \mathbb{S}^{\mathcal{O}}(M_{\ell+1}, x, \pi) \\ b \leftarrow\!\$\ \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_b}(1^\lambda, \mathsf{h}, \pi_b') \end{array}\right] - 1/2 \leq \mathsf{negl}(\lambda)$$

Where $\mathcal{O}_0 = \mathcal{O}$ and $\mathcal{O}_1 = [\mathcal{Q}, \mathcal{O}]$, where $[\mathcal{Q}, \mathcal{O}]$ is the oracle mapping $\mathfrak{q}$ to R_i if $(\mathfrak{q}, R_i) \in \mathcal{Q}$ and $\mathcal{O}(\mathfrak{q})$ otherwise. The probability is over $\mathcal{O}$, the random tape of $\mathcal{A}^{(\cdot)}$, $\mathbb{P}$ and $\mathbb{S}$. We allow the running time of the simulator to depend polynomially on the running time of the adversary.

Remark 2. An easy observation is that if $\mathcal{A}_1$ queried $\mathcal{O}$ at $\mathfrak{q}$ in $\mathsf{h} \leftarrow \mathcal{A}_1^{\mathcal{O}}(1^\lambda, M_{\ell+1}, x, \pi)$, then we can assume that, except with negligible probability, $\mathcal{O}_1(\mathfrak{q}) = \mathcal{O}(\mathfrak{q})$, i.e., the simulator does not reprogram on $\mathfrak{q}$. Namely, if $\mathcal{O}_1(\mathfrak{q}) \neq \mathcal{O}(\mathfrak{q})$ happens with non-negligible probability the adversary could remember all queries $\mathfrak{q}$ and replies made during the first step and redo them in the second step and guess $b = 1$ when $\mathcal{O}_1(\mathfrak{q}) \neq \mathcal{O}(\mathfrak{q})$ and $b = 0$ otherwise. This would break zero-knowledge. We call the property that the simulator only programs points that were never queried *fresh reprogramming* below.

Remark 3. We want to warn that our definitions were tailored for proving negative results. They might not be strong enough for positive applications. Namely, our soundness requires only that extension works for honestly generated proofs. So it might be possible to maliciously generate a proof π_{i-1} which accepts but extends into a non-accepting proof π_i. That means that in a proof carrying data context an honest party might end up producing and further extending an non-accepting proof π_i into a proof π_{i+1}. At the same time by our notion of zero-knowledge the proof π_{i+1} might not be zero-knowledge as zero-knowledge

only holds when starting from an accepting proof $\pi = \pi_i$. That means an honest party might end up producing an non-zero-knowledge proof π_{i+1}. However, the definitions are enough to prove our results. Starting from a weaker definition makes impossibility proofs stronger. For practical applications stronger definitions should be used.

2.1 Rerandomizable Commitments

Definition 8 (Rerandomizable Bit Commitments). *A rerandomizable bit commitment scheme consists of three algorithms:*

$\mathsf{Setup} : \{1\}^* \times \{0,1\}^* \to \mathcal{P}$ *a PPT algorithm which takes a unary representation of the security parameter* 1^λ *and produces public parameters, i.e.,* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda; r)$ *for a random tape* $r \in \{0,1\}^*$.

$\mathsf{Commit} : \mathcal{P} \times \{0,1\} \to \mathcal{C}$ *a deterministic algorithm which sends a bit to the commitment space, i.e.,* $\mathsf{c} = \mathsf{Commit}(\mathsf{pp}, b)$, $b \in \{0,1\}$.

$\mathsf{ReRand} : \mathcal{P} \times \mathcal{C} \times (\{0,1\}^*)^* \to \mathcal{C}$ *takes a commitment and produces a rerandomization of the same commitment (without knowing the opening).*

Note that we do not require the rerandomizable commitments to have succinct openings, in particular " Open*" can be constructed by simply re-executing all the rerandomizations of the original commitment, i.e.* $\mathsf{Open}(\mathsf{pp}, b, \mathsf{c}, \mathbf{r} = (r_1, \ldots, r_m))$ $:= \mathsf{c} \stackrel{?}{=} \mathsf{ReRand}^m(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, b); \mathbf{r}) = \mathsf{ReRand}(\ldots \mathsf{ReRand}(\mathsf{ReRand}(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, b); r_1); r_2), \ldots; r_m)$

$\mathtt{Game}^{(m)}_{\mathtt{Hiding}}(\mathcal{A}, \lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $((v^{(0)}, \vec{r}^{(0)}), (v^{(1)}, \vec{r}^{(1)}), \mathsf{st}) \leftarrow \mathcal{A}(\mathtt{find}, \mathsf{pp}, 1^\lambda)$
3 : $\mathsf{c}^{(0)} = \mathsf{ReRand}^m(\mathsf{Commit}(\mathsf{pp}, v^{(0)}); \vec{r}^{(0)})$
4 : $\mathsf{c}^{(1)} = \mathsf{ReRand}^m(\mathsf{Commit}(\mathsf{pp}, v^{(1)}); \vec{r}^{(1)})$
5 : $b \leftarrow\!\!\$\ \{0,1\}; \mathsf{c}' \leftarrow \mathsf{ReRand}(\mathsf{c}^{(b)})$
6 : $b' \leftarrow \mathcal{A}(\mathtt{guess}, \mathsf{st}, \mathsf{pp}, \mathsf{c}', 1^\lambda)$
7 : **return** $b \stackrel{?}{=} b'$

We require the rerandomizable commitment scheme to be perfectly binding and computationally hiding.

Definition 9 (Perfect Binding). *For every* pp *and number of rerandomizations* m*, the set of (rerandomized) commitments to* 0 *and* 1 *are disjoint, i.e.*

$$\forall m \geq 0, \forall \mathbf{r}^{(0)}, \mathbf{r}^{(1)} : \Pr\left[\mathsf{c}_0 = \mathsf{c}_1 \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{c}_0 = \mathsf{ReRand}^m\Big(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, 0), \mathbf{r}^{(0)}\Big) \\ \mathsf{c}_1 = \mathsf{ReRand}^m\Big(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, 1), \mathbf{r}^{(1)}\Big) \end{array}\right] = 0$$

We do not require this to hold if the two commitments are rerandomized a different number of times; which is weaker than the common definition.

Definition 10 (Computational Hiding). *For every* $m \geq 1$ *and PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\Big[\mathit{Game}^{(m)}_{\mathit{Hiding}}(\mathcal{A}, \lambda)\Big] - 1/2 \leq \mathsf{negl}(\lambda)$$

We do not require the scheme to hide the number (m) *of times a commitment has been rerandomized; which is weaker than the common definition.*

Remark 4 (Concrete Assumptions for Perfectly Binding Rerandomizable Commitments). Perfectly binding rerandomizable commitments can be obtained from decisional Diffie-Hellman using Elgamal encryption: which additionally hides the number (m) of rerandomizations; a property we do not require.

2.2 Collision Intractable Hashes

Definition 11 (Collision Intractable Hash Functions). *A family* $\mathcal{H}_\lambda = \{\mathsf{H}_\mathsf{k}\}_{\mathsf{k} \in \{0,1\}^\lambda}$ *of a set of PPT computable functions from* $\{0,1\}^*$ *to* $\{0,1\}^\lambda$ *indexed by the security* λ *is* collision intractable *if for every PPT adversary* $\mathcal{A}$*, there exist a negligible function* $\mathsf{negl}(\lambda)$ *st.*

$$\Pr\big[\mathsf{H}(x) = \mathsf{H}(x') \wedge x \neq x' \mid \mathsf{H} \leftarrow\!\$\, \mathcal{H}_\lambda; (x, x') \leftarrow \mathcal{A}(\mathsf{H}, 1^\lambda)\big] \leq \mathsf{negl}(\lambda)\ .$$

2.3 Basic Notation

Definition 12 (Stretch). *Let a length* ℓ *of a proof be fixed, i.e.,* ℓ *is the number of times the basic step function is run. We call* (p, q) *with* $1 \leq p$*,* $0 \leq q$ *and* $p + q \leq \ell$ *a* stretch *of length* q *with start position* p*.*

Definition 13 (Query Sets). *Consider a length* ℓ *and a run of a proof of length* ℓ*, which proceeds as follows. For* $i = 1, \ldots, \ell$ *compute* $M_i = T(M_{i-1}, w_i)$*, let* $\mathcal{P}^{(i)}_\downarrow$ *be the queries made to* $\mathcal{O}$ *in computing* $\pi_i = \mathbb{P}^\mathcal{O}(T, M_{i-1}, \pi_{i-1}, w_i; \rho_i)$ *and let* $\mathcal{V}^{(i)}_\downarrow$ *be the queries made to* $\mathcal{O}$ *in computing* $\mathbb{V}^\mathcal{O}(T, M_0, M_i, \pi_i)$*. For* $1 \leq i \leq k \leq \ell$*, let* $\mathcal{V}^{(i,k)}_\cup = \cup_{j=i}^k \mathcal{V}^{(j)}_\downarrow$ *and* $\mathcal{P}^{(i,k)}_\cup = \cup_{j=i}^k \mathcal{P}^{(j)}_\downarrow$*. Define the 'fresh' queries made at step* i *as* $\mathcal{V}^{(i)}_\Delta = \mathcal{V}^{(i)}_\downarrow \backslash \mathcal{V}^{(1,i-1)}_\cup$ *and* $\mathcal{P}^{(i)}_\Delta = \mathcal{P}^{(i)}_\downarrow \backslash \mathcal{P}^{(1,i-1)}_\cup$*. Finally define the fresh queries during stretches as* $\mathcal{V}^{(p,q)}_\Delta = \cup_{i=p}^{p+q-1} \mathcal{V}^{(i)}_\Delta$ *and* $\mathcal{P}^{(p,q)}_\Delta = \cup_{i=p}^{p+q-1} \mathcal{P}^{(i)}_\Delta$*.*

Definition 14 (Oracle Extension). *For a set of queries $\mathcal{Q}_1$ and two oracles $\mathcal{O}_1$ and $\mathcal{O}$ we define the oracle $[\mathcal{Q}_1 \mapsto \mathcal{O}_1, \mathcal{O}]$ as follows. On input $\mathfrak{q}$, if $\mathfrak{q} \in \mathcal{Q}_1$ then output $\mathcal{O}_1(\mathfrak{q})$. Otherwise output $\mathcal{O}(\mathfrak{q})$. In general, let*

$$[\mathcal{Q}_1 \mapsto \mathcal{O}_1, \ldots, \mathcal{Q}_\ell \mapsto \mathcal{O}_\ell, \mathcal{O}] = [\mathcal{Q}_1 \mapsto \mathcal{O}_1, [\mathcal{Q}_2 \mapsto \mathcal{O}_2, \ldots, \mathcal{Q}_\ell \mapsto \mathcal{O}_\ell, \mathcal{O}]] \ .$$

3 Theorem Statements

Having the definitions in place we give the formal theorem statements. For the statements we use the following step functions. Let T_{H} be the step function for repeated hashing of the witnesses, i.e., $T_{\mathsf{H}} \coloneqq \mathsf{H}(M \| w)$. Let T_{pp} be the step function for repeated rerandomization of a commitment using the witness as randomness, i.e., $T_{\mathsf{pp}}(M, w) \coloneqq \mathsf{ReRand}(\mathsf{pp}, M; w)$. The following statements is proven in Section 4.

Theorem 1 (Impossibility of Non-Trivial ZK Non-Deterministic $\mathcal{O}$-IVC). *The existence of collision intractable functions or perfectly binding rerandomizable commitments precludes the existance of (knowledge-sound) non-trivial zero-knowledge non-deterministic $\mathcal{O}$-IVC, more formally:*

- ***Collision Intractability Precludes Knowledge-Soundness.*** *Assuming the existence of a family of collision intractable functions $\mathcal{H}_\lambda$ (Definition 11), there exists a transition function T_{H} such that any zero-knowledge (Definition 7), knowledge-sound (Definition 5) $\mathcal{O}$-IVC scheme (Definition 3) for the step function T_{H} must have a verifier with running time linear in the number of steps ℓ.*
- ***Rerandomizable Commitments Precludes (Regular) Soundness.*** *Assuming the existence of perfectly binding rerandomizable commitment schemes (Definition 8), there exists transition functions T_{pp} such that any zero-knowledge (Definition 7) and computationally sound (Definition 4) $\mathcal{O}$-IVC scheme (Definition 3) for T_{pp} must have a verifier with running time linear in the number of steps ℓ.*

For the impossibility for black-box schemes we need to formalize the notion that one can recognize whether queries are fresh.

Definition 15 (Structured Oracle Queries). *We say that a proof system $(\mathcal{T}, \mathbb{P}, \mathbb{V})$ has structured oracle queries if there exists a PPT algorithm* used *for which the following holds for all PPT adversaries $\mathcal{A}$. For all $T \in \mathcal{T}$, all lengths ℓ, and all witnesses $(w_1, \ldots, w_\ell)$ let M_0 be an initial state, $\pi_0 = \epsilon$, $\pi_i = \mathbb{P}^{\mathcal{O}}(T, M_{i-1}, \pi_{i-1}, w_i, \rho_i)$, where ρ_i is the possible random tape of $\mathbb{P}$, $\mathcal{P}_{\downarrow}^{(i)}$ be the queries made by this i'th run of $\mathbb{P}$, $\mathcal{P}_{\cup}^{(1,i)} = \cup_{j=1}^{i} \mathcal{P}_{\downarrow}^{(j)}$, and let $\mathsf{used}_i = \mathsf{used}(T, M_{i-1}, \pi_{i-1}, w_i, \rho_i)$ be the description of a PPT predicate. Now compute $(i, \mathfrak{q}) = \mathcal{A}^{\mathcal{O}}(T, \vec{w}, M_0, \vec{\rho})$. We say that the adversary wins if $\mathsf{used}_i(\mathfrak{q}) = \top$ and $\mathfrak{q} \notin \mathcal{P}_{\cup}^{(1,i)}$ or $\mathsf{used}_i(\mathfrak{q}) = \bot$ and $\mathfrak{q} \in \mathcal{P}_{\cup}^{(1,i)}$. We say that the proof system is p_{STRUC}-SOQ if the probability that the adversary wins is $\leq 1 - p_{\text{STRUC}}$.*

Below we will assume that the proof system is $1/\lambda^{\gamma}$-SOQ for some constant $\gamma > 0$. This means we essentially just need a non-negligible probability that the queries are structured. Note that used_i is computed from the current state of the prover, so if the proof system is succinct then so is the state needed to compute used_i which will be basis for our impossibility result. The following theorem is proven in Section 5.

Theorem 2. *If there exist collision intractable hash functions then there does not exist succinct, non-deterministic IVC for the random oracle model (Definition 3) with blackbox knowledge soundness (Definition 6) which is $1/\lambda^{O(1)}$-SOQ (Definition 15) for the step function T_{H}. By succinct we mean that the size of a proof of an ℓ-iteration computation is $\mathsf{poly}(\lambda, \log \ell)$.*

4 Impossibility from Zero-Knowledge

In the following section we prove two impossibility results for the case where the $\mathcal{O}$-IVC is zero-knowledge. One is for the case where the proof system is knowledge sound and collision intractable functions exists. The other is for the case where the proof system has just soundness but under the assumption of perfectly binding rerandomizable commitments. We start by proving some lemmas and then put them together at the end of the section.

The following lemmas state that for certain transition functions no adversary can produce an accepting proof without knowing the witness for every step; without violating (knowledge) soundness of the $\mathcal{O}$-IVC scheme.

Let $\mathcal{U}_n^{\ell} = \mathcal{U}_n \times \cdots \times \mathcal{U}_n$ be the distribution of ℓ iid. uniform n bit strings and define $\vec{w}^{(\bar{m})} := (w_1, \ldots, w_{m-1}, \bot, w_{m+1}, \ldots, w_{\ell})$ (i.e., a sequence where the m'th witness is removed) for any sequence of witnesses $\vec{w}$. Impossibility of knowledge soundness follows from collision intractable functions:

Lemma 1 (All Witnesses Are Required for Knowledge Soundness). *For a (randomly sampled) collision intractable hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\lambda}$, consider the following step function $T_{\mathsf{H}}(M, w) := \mathsf{H}(M \| w)$. We now show that for any knowledge-sound $\mathcal{O}$-IVC scheme, PPT adversary $\hat{\mathcal{A}}$, $\ell = O(\mathsf{poly}(\lambda, |T_{\mathsf{H}}|))$ and $m \in [\ell]$, $\hat{\mathcal{A}}$ produces an accepting proof π of the T_{H}^{ℓ} execution given all witnesses except for step m, with only negligible probability. i.e., there exists a negligible function $\mathsf{negl}(\lambda)$ st.*

$$\Pr\left[\mathbb{V}^{\mathcal{O}}(x, \pi) = \top \;\middle|\; \begin{array}{c} \mathsf{H} \leftarrow\!\!\$\; \mathcal{H}_{\lambda}; \vec{w} \leftarrow\!\!\$\; \mathcal{U}_{2\lambda}^{\ell}; \\ M_0 = \epsilon;\ \textbf{for}\ i \in [\ell] : M_i = T_{\mathsf{H}}(M_{m-1}, w_m); \\ \vec{w}^{(\bar{m})} = (w_1, \ldots, w_{m-1}, \bot, w_{m+1}, \ldots, w_{\ell}); \\ \pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x = (T_{\mathsf{H}}, M_0, M_{\ell}, \ell), \vec{w}^{(\bar{m})}, M_m) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Proof. Since the $\mathcal{O}$-IVC scheme is (non-blackbox) extractable by assumption, there exists an extractor $\mathbb{E}$. Now, for any $\ell \geq 1$ and $m \in [\ell]$, consider the following adversary $\mathcal{A}$ for the collision game (see Definition 11):

$(v_1, v_2) \leftarrow \mathcal{A}(\mathsf{H})$

// Run $\hat{\mathcal{A}}$ to get a proof without the pre-image of M_m
1 : $\vec{w} \leftarrow\!\!\$\ \mathcal{U}_{2\lambda}^{\ell}$
2 : $M_0 = \epsilon$; **for** $i \in [\ell] : M_i = T_{\mathsf{H}}(M_{i-1}, w_i)$;
3 : $\vec{w}^{(\bar{m})} \coloneqq (w_1, \ldots, w_{m-1}, \bot, w_{m+1}, \ldots, w_\ell)$
4 : $x = (T_{\mathsf{H}}, \epsilon, M_\ell, \ell); \pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x, \vec{w}^{(\bar{m})}, M_m)$
// Run $\mathbb{E}$ to get preimages for each state.
5 : $\vec{w}' \leftarrow \mathbb{E}(x, \hat{\mathcal{A}}^{(\cdot)}(x, \vec{w}^{(\bar{m})}, M_m))$
6 : $M'_0 = \epsilon$; **for** $i \in [\ell] : M'_i = T_{\mathsf{H}}(M'_{i-1}, w'_i)$;
// Look for collision.
7 : **for** $i \in [\ell - 1]$:
8 : $\quad v_1 \coloneqq M_i \| w_{i+1};\ v_2 \coloneqq M'_i \| w'_{i+1}$
9 : $\quad$ **if** $M_{i+1} = M_{i+1} \wedge v_1 \neq v_2$
10 : $\quad\quad$ **return** (v_1, v_2)
11 : **return** $\bot$

Let $f : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be defined as $f(y) \mapsto \mathsf{H}(M_{m-1} \| y)$, note that the probability that there exists ≥ 2 preimages of $f(w_m)$ is overwhelming, since w_m is sampled uniformly at random. Hence the extractor given only $M_m \coloneqq f(w_m)$ recovers w'_m such that $w_m \neq w'_m$ with probability at least $1/2 - \mathsf{negl}(\lambda)$. This violates collision intractability of f and in particular of H. □

If we are willing to make the stronger assumption that perfectly binding and computationally hiding rerandomizable commitments exist we can strengthen the lemma to violate soundness of the $\mathcal{O}$-IVC scheme:

Lemma 2 (All Witnesses Are Required for Soundness). *For a perfectly binding rerandomizable commitment scheme, consider the following step function $T_{\mathsf{pp}}(M, w) \coloneqq \mathsf{ReRand}(\mathsf{pp}, M; w)$—repeated rerandomization of the commitment. We now show that for any computationally sound $\mathcal{O}$-IVC scheme, PPT adversary $\hat{\mathcal{A}}$, $\ell = O(\mathsf{poly}(\lambda, |T_{\mathsf{pp}}|))$ and $m \in [\ell]$, $\hat{\mathcal{A}}$ produces an accepting proof π of the T_{pp}^{ℓ} execution given all witnesses except for step m, with only negligible probability, i.e., there exists a negligible function $\mathsf{negl}(\lambda)$ st.*

$$\Pr\left[\mathbb{V}^{\mathcal{O}}(x, \pi) = \top \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda); \vec{w} \leftarrow\!\!\$\ \mathcal{U}_{\mathsf{poly}(\lambda)}^{\ell}; \\ M_0 = \mathsf{Commit}(\mathsf{pp}, 0); \\ \textbf{for } i \in [\ell] : M_i = T_{\mathsf{pp}}(M_{m-1}, w_m); \\ \vec{w}^{(\bar{m})} = (w_1, \ldots, w_{m-1}, \bot, w_{m+1}, \ldots, w_\ell); \\ \pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x = (T_{\mathsf{pp}}, M_0, M_\ell, \ell), \vec{w}^{(\bar{m})}, M_m) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Proof. Let p be the probability that $\hat{\mathcal{A}}$ outputs an accepting proof (in the original game), we assume for contradiction that p is non-negligible (in λ). Consider the

following PPT algorithm which we use to violate soundness of the $\mathcal{O}$-IVC scheme or break computational hiding of the commitment scheme:

$\mathcal{A}_{\mathsf{Hiding}}(\mathsf{find}, \mathsf{pp}, 1^\lambda)$

// Sample randomness / witnesses for ℓ steps.

$\vec{r} \leftarrow\!\!\$\ \mathcal{U}^{\ell}_{\mathsf{poly}(\lambda)}; \mathsf{st} = \vec{r}$

// Get rerandomisation of either 0 or 1.

$v^{(0)} = 0; v^{(1)} = 1$

$\vec{r}^{(0)} = \vec{r}^{(1)} = (r_1, \ldots, r_{m-1})$

return $((v^{(0)}, \vec{r}^{(0)}), (v^{(1)}, \vec{r}^{(1)}), \mathsf{st})$

$\mathcal{A}_{\mathsf{Hiding}}(\mathsf{guess}, \mathsf{st}, \mathsf{pp}, \mathsf{c}', 1^\lambda)$

$\vec{r} = \mathsf{st}$

$\vec{w}^{(\bar{m})} \coloneqq (r_1, \ldots, r_{m-1}, \perp, r_{m+1}, \ldots, r_\ell)$

$M_0 = \mathsf{Commit}(\mathsf{pp}, 0); M_m = \mathsf{c}'$

for $i \in [m+1, \ell] : M_i = T_{\mathsf{pp}}(M_{i-1}, w_i)$

$x = (T_{\mathsf{pp}}, M_0, M_\ell, \ell)$

$\pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x, \vec{w}^{(\bar{m})}, M_m)$

if $\mathbb{V}^{\mathcal{O}}(x, \pi) = 1$ **return** 0

else return 1

Observe that when $b = 0$ in the $\mathtt{Game}^{(m-1)}_{\mathtt{Hiding}}$ game $M_m = \mathsf{c}'$ is a rerandomization of the 0 commitment and hence M_ℓ is as well, therefore x is true and $\mathcal{A}_{\mathsf{Hiding}}$ correctly returns 0 with probability p by assumption on $\hat{\mathcal{A}}$. However, when $b = 1$, the commitment c' is a rerandomization of the 1 commitment and x is false, hence $\mathbb{V}^{\mathcal{O}}(x, \pi) = 0$ except with negligible probability $\mathsf{negl}(\lambda)$, otherwise computational soundness is violated. This implies that $\mathcal{A}_{\mathsf{Hiding}}$ wins the $\mathtt{Game}^{(m)}_{\mathtt{Hiding}}$ game with advantage at least $p - \mathsf{negl}(\lambda)/2$; which is non-negligible, a contradiction. □

We now show that proof systems with transition functions like the ones in Lemma 1 and Lemma 2 where all witnesses are needed will have the verifiers make many queries to the random oracle.

In the below we use some common definitions of honest and simulated experiments. For a given proof system we can define the honest experiment HonExp as follows.

Experiment HonExp:

1. Let M_0 be the start state and $\pi_0 = \epsilon$. Let $\vec{w}$ be a vector of witnesses.
2. For $i = 1, \ldots, \ell$ compute $M_i = T(M_{i-1}, w_i)$, $\pi_i = \mathbb{P}^{\mathcal{O}}(T, M_{i-1}, \pi_{i-1}, w_i)$, $\mathbb{V}^{\mathcal{O}}(T, M_0, M_i, \pi_i)$.

Let the query sets be defined as in Definition 13.

For any $1 \leq m \leq \ell$ we can define a simulation experiment SimExp_m where everything is defined as in the honest experiment except that we simulate in step m and then use the reprogrammed oracle from then on.

Experiment SimExp_m:

1. Let M_0 be the start state and $\pi_0 = \epsilon$. Let $\vec{w}$ be a vector of witnesses.
2. For $i = 1, \ldots, m-1$ compute $M_i = T(M_{i-1}, w_i)$, $\pi_i = \mathbb{P}^{\mathcal{O}}(T, M_{i-1}, \pi_{i-1}, w_i)$, $\mathbb{V}^{\mathcal{O}}(T, M_0, M_i, \pi_i)$.
3. Compute $M_m = T(M_{m-1}, w_m)$. Compute a simulated proof $(\mathcal{Q}, \pi_m) \leftarrow \mathbb{S}^{\mathcal{O}}(T, M_m, \pi_{m-1})$. Let $\mathcal{O}_1 = [\mathcal{Q}, \mathcal{O}]$. Let $S_m = \{\mathfrak{q} | \exists y\,((\mathfrak{q}, y) \in \mathcal{Q})\}$ be the set of query points on which $\mathcal{Q}$ programs. Compute $\mathbb{V}^{\mathcal{O}_1}(T, M_0, M_m, \pi_m)$. Note that we use the reprogrammed oracle from here on.
4. For $i = 1, \ldots, m+1$ compute $M_i = T(M_{i-1}, w_i)$, $\pi_i = \mathbb{P}^{\mathcal{O}_1}(T, M_{i-1}, \pi_{i-1}, w_i)$, $\mathbb{V}^{\mathcal{O}_1}(T, M_0, M_i, \pi_i)$.

We can show that if all steps are run honestly except that step m is simulated then *all* future verifiers must check one of the points that the simulator programmed in step m. More formally:

Lemma 3 (Must Check Programmed Points). *For transition functions T as described in Lemma 1 and Lemma 2 and for all m and ℓ with $1 \leq m \leq \ell$ it holds that* $\Pr\left[S_m \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset\right] = \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}(\lambda)$.

Proof. Towards contradiction, suppose there exists (m, ℓ) such that $\Pr\left[S_m \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset\right] = p$ where p is non-negligible in λ, then construct an adversary violating Lemma 1 and Lemma 2 as follows:

$\pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x, \vec{w}^{(\bar{m})}, M_m)$; produces a proof without w_m.

$x = (T, M_0, M_\ell, \ell); \pi_0 = \epsilon$

$\vec{w}^{(\bar{m})} = (w_1, \ldots, w_{m-1}, \bot, w_{m+1}, \ldots, w_\ell)$

// Start execution using first $m-1$ witnesses

for $j \in [1, m-1]$:

  $M_j \leftarrow T(M_{j-1}, w_j)$

  $\pi_j \leftarrow \mathbb{P}^{\mathcal{O}}(T, M_{j-1}, w_j, \pi_{j-1})$

// Simulate step m

$(\mathcal{Q}_m, \pi_m) \leftarrow \mathbb{S}^{\mathcal{O}}(T, M_m, \pi_{m-1})$

// Finish execution with reprogrammed oracle

for $j \in [m+1, \ell]$:

  $M_j \leftarrow T(M_{j-1}, w_j)$

  $\pi_j \leftarrow \mathbb{P}^{[\mathcal{Q}_m, \mathcal{O}]}(T, M_{j-1}, w_j, \pi_{j-1})$

return π_ℓ

Where T, M_0 are instantiated as in Lemma 1 and Lemma 2. To reach contradiction we now argue that $\mathbb{V}^{\mathcal{O}}(x, \pi) = \top$ with probability p: notice that when $S_m \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset$, then $\mathbb{V}^{[\mathcal{Q}_m, \mathcal{O}]}(x, \pi) = \mathbb{V}^{\mathcal{O}}(x, \pi)$ since the verifier makes no queries

in $\mathcal{Q}_m$ (S_m). Now, simply observe that $\mathbb{V}^{[\mathcal{Q}_m,\mathcal{O}]}(x,\pi) = \top$ follows from zero-knowledge—otherwise π_m could be distinguished from a real proof by extending it $\ell - m - 1$ times and running the verifier. Therefore $\mathbb{V}^{\mathcal{O}}(x,\pi)$ accepts with non-negligible probability contradicting Lemma 1 and Lemma 2. □

The above lemma intuitively implies that some query points "belonging" to step m must be checked by many future verifiers. If this was true for all m simultaneously and these query points were distinct then we would be done. Too many distinct points would need to be checked often in the future, so the query sets of the verifiers would have to get too big. It is, however, not straight forward to generalise the above lemma to show that the query sets of the verifiers must be large. If we simulate at many steps the set of reprogrammed points might grow so large that we cannot argue that the final verifier will not query a reprogrammed point and reject the proof. Note that the final verifier has access to the real random oracle, not the reprogrammed random oracle. And if the verifier rejects, then we do not get a contradiction to Lemma 1 or Lemma 2. We will therefore need a slightly more subtle strategy. We show that because Lemma 3 holds in SimExp_m we can carefully compute in HonExp a set of query points uniquely associated to step m which must be checked often in the future. In HonExp we can then sum over all m.

Lemma 4 (Too Large Verifier Query Set). *For transition functions T for which the property in Lemma 3 holds there exists i such that the set $\mathcal{V}_{\downarrow}^{(i)}$ sometimes has size at least $\frac{\ell-1}{4}$ in* HonExp.

Proof. We describe an adversary $\mathcal{B}$ (the "blocking adversary") which in each step n computes a set B_n, the "blocking set". For now, we assume this adversary knows witnesses for every step, i.e., $\vec{w}$ such that $T^{\ell}(M_0,\vec{w}) = M_\ell$. The "blocking sets" produced by $\mathcal{B}$ will satisfy:

Disjointness: The blocking sets are disjoint: $\forall i,j : i \neq j \implies B_i \cap B_j = \emptyset$.

Frequent Appearance: In a random run from step n until step ℓ the expected number of elements from B_n which occur in $\mathcal{V}_{\downarrow}^{(i)}$ for $i \geq n$ is at least $(\ell-n)/2$. Formally:

$$\mathbb{E}\left[\sum_{i=n}^{\ell}\left|B_n \cap \mathcal{V}_{\downarrow}^{(i)}\right|\right] > (\ell-n)/2 \ .$$

We first argue that if we can prove the two properties then we are done. Assume *disjointness* and *frequent appearance*. By linearity of expectation and Gauss' trick we get that

$$\mathbb{E}\left[\sum_{n=1}^{\ell}\sum_{i=n}^{\ell}\left|B_n \cap \mathcal{V}_{\downarrow}^{(i)}\right|\right] > \sum_{n=1}^{\ell}(\ell-n)/2 = \frac{\ell(\ell-1)}{4} \ .$$

By disjointness we get that

$$\sum_{n=1}^{\ell}\sum_{i=n}^{\ell}\left|B_n \cap \mathcal{V}_{\downarrow}^{(i)}\right| = \sum_{i=1}^{\ell}\sum_{n=1}^{i}\left|B_n \cap \mathcal{V}_{\downarrow}^{(i)}\right| = \sum_{i=1}^{\ell}\left|(\cup_{n=1}^{i}B_n) \cap \mathcal{V}_{\downarrow}^{(i)}\right| \leq \sum_{i=1}^{\ell}\left|\mathcal{V}_{\downarrow}^{(i)}\right| \ .$$

Combining the last two inequalities we get that

$$\mathbb{E}\left[\sum_{i=1}^{\ell} |\mathcal{V}_{\downarrow}^{(i)}|\right] > \frac{\ell(\ell-1)}{4} \ .$$

This shows that it happens with non-zero probability that

$$\sum_{i=1}^{\ell} |\mathcal{V}_{\downarrow}^{(i)}| > \frac{\ell(\ell-1)}{4} \ .$$

Therefore there must exist i such that it happens with non-zero probability that $|\mathcal{V}_{\downarrow}^{(i)}| > \frac{\ell(\ell-1)}{4\ell}$, which proves the lemma.

The Blocking Adversary. Let p be a polynomial which we specify below. The blocking adversary $\mathcal{B}$ runs as follows.

1. Let M_0 be the start state, $\pi_0 = \epsilon$, and $\vec{w}$ a witness vector
2. For $m = 1, \ldots, \ell$ compute $M_m = T(M_{m-1}, w_m)$
3. Let $B_{<1} = \emptyset$ and for $m = 1, \ldots, \ell$ do:
   (a) Query the random oracle on all points in $B_{<m} = \cup_{i=1}^{m-1} B_i$
   (b) Compute $\pi_m = \mathbb{P}^{\mathcal{O}}(T, M_{m-1}, \pi_{m-1}, w_m)$
   (c) Run $\mathbb{V}^{\mathcal{O}}(T, M_0, M_m, \pi_m)$ and name its queries $\mathcal{V}_{\downarrow}^{(m)}$
   (d) For $\iota = 1, \ldots, p$ let $\pi_m^{(\iota)} = \pi_m$ and for $f = m, \ldots, \ell$ do:
      i. Run $\mathbb{V}^{\mathcal{O}}(T, M_0, M_f, \pi_f^{(\iota)})$ and record its queries $\mathcal{V}_{\downarrow}^{(f,\iota)}$
      ii. Add $\mathcal{V}_{\downarrow}^{(f,\iota)} \setminus B_{<m}$ to B_m
      iii. If $f < \ell$ then compute $\pi_{f+1}^{(\iota)} = \mathbb{P}^{\mathcal{O}}(T, M_f, \pi_f^{(\iota)}, w_{f+1})$

Call the experiment HonExp. Note that the blocking sets are disjoint by construction. We now prove frequent appearance by appealing to zero-knowledge.

Likely/Unlikely Queries. For $m \geq 1$ let $\mathcal{B}^m$ be the adversary running as $\mathcal{B}$ except that it simulates $3(b)$ in iteration m instead of using the witness for this step and call it SimExp_m. Let S_m be the set of queries programmed by the simulator. We call a point $\mathfrak{q} \in S_m$ *likely* if when $\mathcal{B}_m$ runs forward from step m then $\mathfrak{q}$ appears in $\mathcal{V}_{\downarrow}^{(\geq m)} = \cup_{i=m}^{\ell} \mathcal{V}_{\downarrow}^{(i)}$ with probability at least $(q|S_m|)^{-1}$ for a polynomial q specified below. Let $L_m \subseteq S_m$ be the set of likely points. Conversely we call $U_m = S_m \setminus L_m$ the *unlikely* points.

Since $|U_m| \leq |S_m|$, it follows by a union bound that if we do a random run, then the probability that an unlikely point is verified is low. More precisely,

$$\Pr\left[U_m \cap \mathcal{V}_{\downarrow}^{(\geq m)} \neq \emptyset\right] \leq q^{-1} \ .$$

Collecting All Likely Queries. For all q we can set p to be a polynomial such that if $\mathcal{B}^m$ does p random runs from step m on, then except with negligible probability the likely points are included in the blocking set B_m, as described now. Consider any likely point $\mathfrak{q}$. In a random run it appears with probability at least $(q|S_m|)^{-1}$. So if we do $q|S_m|$ independent runs it appears in one of these except with probability about $1/e$ as $(1-1/n)^n \to 1/e$ with fast convergence. So if we run for instance λq times, it appears except with probability about $e^{-\lambda}$. Then use that negligible probabilities are maintained by polynomial union bounds and that the size of S_m is polynomial. This gives us that the likely point will appear in some $\mathcal{V}_{\downarrow}^{(f,\iota)}$. It will therefore be added to the blocking set when the adversary adds $\mathcal{V}_{\downarrow}^{(f,\iota)} \setminus B_{<m}$ to B_m. Namely, the query $\mathfrak{q}$ will not be in $B_{<m}$ as the adversary queried on all points in $B_{<m}$ before the simulation step was run. So by *fresh reprogramming* no element from S_m is in $B_{<m}$, and all likely points are in S_m by definition.

Next we argue that we can pick q large enough such that:

$$\mathbb{E}\left[\sum_{n\geq m}\left|U_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] \leq 1/2.$$

For $q > 2\ell|S_m|$ it holds that a given unlikely point appears with probability at most $1/q$, by definition, and that when it does it contributes at most ℓ to the sum (if it is in all verifier sets). So its contribution to the expected value is at most $\ell/q = 1/2|S_m|^{-1}$. Then use that $U_n \subset S_n$ to see that there are at most $|S_n|$ unlikely points and apply linearity of expectation.

Putting the pieces together. By Lemma 3 we have that

$$\mathbb{E}\left[\sum_{n\geq m}\left|S_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] \geq \ell - m - \mathsf{negl}(\lambda)\ .$$

Combining the above two inequalities and $L_m = S_n \setminus U_m$ we get that:

$$\mathbb{E}\left[\sum_{n\geq m}\left|L_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] \geq \ell - m - \mathsf{negl}(\lambda) - 1/2 \geq \ell - m - 1\ .$$

Using that $L_m \subset B_m$ we obtain:

$$\mathbb{E}\left[\sum_{n\geq m}\left|B_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] \geq \ell - m - 1\ .$$

This inequality holds in SimExp_m. Now run $\mathcal{B}$ (HonExp) instead of $\mathcal{B}_m$. Then by a reduction to zero-knowledge we easily get that:

$$\mathbb{E}\left[\sum_{n\geq m}\left|B_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] \geq \frac{\ell - m}{2}\ .$$

Namely, the value $\sum_{n \geq m} |B_m \cap \mathcal{V}_{\downarrow}^{(n)}|$ can be computed in poly-time in both experiments. So, if $\mathbb{E}\left[\sum_{n \geq m} \left|B_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] \geq \ell - m - 1$ in the real world and $\mathbb{E}\left[\sum_{n \geq m} \left|B_m \cap \mathcal{V}_{\downarrow}^{(n)}\right|\right] < \frac{\ell - m}{2}$ in the simulation we can easily make a distinguisher which computes $\sum_{n \geq m} |B_m \cap \mathcal{V}_{\downarrow}^{(n)}|$ and uses it to guess whether we simulated in step m or not. This completes the proof. □

Proof (Proof of Theorem 1). By combining Lemma 1, Lemma 3, and Lemma 4 we conclude that any proof system for T_{H}, we can pick the number of steps ℓ to be a large enough polynomial such that the proof system will have some verifier of some step i make at least $\frac{\ell-1}{4}$ queries to its random oracle. Therefore the verifier must have running time at least $\frac{\ell-1}{4} \notin \mathsf{poly}(|T_{\mathsf{H}}|, \lambda, \log \ell)$. This proves the first part of the theorem.

The second part is proven by combining Lemma 2, Lemma 3, and Lemma 4 similarly for T_{pp}. □

Remark 5 (Simulation in the presence of computational assumptions). Despite its simplicity the impossibility result above is quite general, in particular it applies to any non-deterministic $\mathcal{O}$-IVC scheme where the simulator works by only programming the random oracle—even in the presence of arbitrary computational assumptions. In particular it applies to interactive zero-knowledge arguments compiled in the random oracle model, like Fiat-Shamir transformations.

5 On Proving Impossibility without Zero-Knowledge

In the previous section we proved impossibility for proof systems which are zero-knowledge. We now explore what it would take to circumvent this result. Can we construct non-deterministic IVC in the random-oracle model which is *not* zero-knowledge. Towards this we prove impossibility of non-deterministic IVC in the random-oracle model with the following properties:

Knowledge Soundness The proof is knowledge sound.

Blackbox The knowledge extractor only has blackbox rewinding access to the prover.

Structured Queries When the prover makes a query x to the random oracle, then with good probability it knows whether the query x was made already or whether it is the first time the random oracle is queried on x.

Collision Intractability There exist collision intractable hash functions.

We know that it is possible to make blackbox knowledge sound proofs in the random oracle model, for instance Micali's CS proofs. It is hard to imagine a world where it is reasonable to assume a random oracle, but where a family of collision intractable hash functions does not exist. Our result can therefore be interpreted as saying that it is the succinctness plus structured queries that give the impossibility.

We discuss the structure assumption briefly. First of all, this is clearly not a reasonable assumption about any proof system. It says that the prover has to "remember" previous queries using a small state. It is easy to make proof systems that do not have this property. For instance, in iteration r flip a uniformly random bit and query r if and only if the bit is 1. On the other hand, it seems hard to exploit such forgotten queries in a constructive way. The result therefore hints that if we want to circumvent Theorem 1 we need to come up with completely new ways to use a random oracle. To see this, note that when querying random oracles in a proof system one typically makes two types of queries. One can make a query on a fresh point to get a fresh "challenge" that the prover is not in control over *a la* the Fiat-Shamir transform. In this case it is crucial that the queried point is fresh such that the challenge is unknown until the time of query. Typically *provers* makes this type of query. One can also make a query to check the validity of a previous query, for instance when recomputing a hash path in a Merkle tree in the CS proofs. In this case one knows that the point on which the queries are made are not fresh, at least in an honest run of the proof system. Typically *verifiers* make this type of query. However, in an iterative proof system we can imagine that also provers make such queries, possibly to check previous provers or verifiers. We leave it as an open problem to determine whether there are proof systems without structured queries which allow to circumvent Theorem 1.

We proceed to the proof. A simple, yet central, component in our proof is a simple lemma which states that for polynomially long computations there will be polynomially long "stretches" of proof steps where no fresh query made during the proofs in the stretch is checked by the final verifier.

Lemma 5 (Non-trivial $\mathcal{O}$-IVC Implies Unchecked Stretches). *Let the running time of $\mathbb{V}^{\mathcal{O}}(x,\pi)$ be bounded by a polynomial $\psi \in \mathsf{poly}(|\mathcal{R}|,\lambda,\log \ell)$. Then for all lengths $q \in \mathsf{poly}(|\mathcal{R}|,\lambda,\log \ell)$ there exist large enough $\ell \in \mathsf{poly}(|\mathcal{R}|,\lambda)$ and a position $p \in [1,\ldots,\ell]$ such that $\mathcal{P}_{\Delta}^{(p,q)} \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset$ with non-negligible probability. The position p may depend on λ.*

Proof. Let ℓ be a free variable for now, we fix it later. Since $|\mathcal{R}| \in \mathsf{poly}(\lambda)$ it is sufficient to consider any constants $a,b \in \mathbb{N}$ and thereby any polynomial $q = \lambda^b(\log \ell)^c$. We want to show that there exists d such that if we let $\ell = \lambda^d$ then there exists a position p (which might be a function of λ) such that

$$\Pr_{\lambda}[\mathcal{P}_{\Delta}^{(p,q)} \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset] = \mathsf{negl}(\lambda) \ ,$$

where $\Pr_\lambda$ denotes that the probability is taken over a random run with security parameter set to λ.

For any q as above we can consider $e = b+1$ and $q' = \lambda^e$. We have that $q' > q$ for large enough λ as $\ell = \mathsf{poly}(\lambda)$. So for large enough λ we have that $\mathcal{P}_{\Delta}^{(p,q)} \subset \mathcal{P}_{\Delta}^{(p,q')}$. It is therefore sufficient to consider any constant $e \in \mathbb{N}$ and thereby any polynomial $q' = \lambda^e$ and show that there exists d such that if we let

$\ell = \lambda^d$ then there exists a position p such that

$$\Pr_{\lambda}[\mathcal{P}_{\Delta}^{(p,q')} \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset] = \mathsf{negl}(\lambda) \ .$$

Now that q' does not depend on ℓ we can for any $\phi \in \mathsf{poly}(\lambda)$ set $\ell = q\phi$. Then we have ϕ disjoint stretches $(1, q), (q + 1, q), \ldots, (\ell - q + 1, q)$. This by definition gives disjoints sets $\mathcal{P}_{\Delta}^{(1,q)}, \mathcal{P}_{\Delta}^{(q+1,q)}, \ldots, \mathcal{P}_{\Delta}^{(\ell-q+1,q)}$.

Since the running time of $\mathbb{V}^{\mathcal{O}}(x, \pi)$ is bounded by $\mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$ it is also bounded by some $\phi \in \mathsf{poly}(\lambda)$ for large enough λ via the same arguments as above. So for large enough λ the verifier can make at most ψ queries to the oracle, i.e., $|\mathcal{V}_{\downarrow}^{(\ell)}| \leq \psi$. So if we set $\phi = 2\psi$, then in any given run at most half the sets $\mathcal{P}_{\Delta}^{(p,q)}$ enumerated above will contain an element from $\mathcal{V}_{\downarrow}^{(\ell)}$.

For each large enough λ this allows us to pick a fixed position p_λ such that for a random run $\mathcal{P}_{\Delta}^{(p,q)}$ will contain an element from $\mathcal{V}_{\downarrow}^{(\ell)}$ with probability at most $1/2$. For smaller λ simply let $p_\lambda = 1$. Now let $p(\lambda) = \lambda$. Then

$$\exists \lambda' \forall \lambda > \lambda_{\emptyset} \Pr_{\lambda}[\mathcal{P}_{\Delta}^{(p,q')} \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset] \geq \frac{1}{2}$$

which is non-negligible. □

Note that the function $p(\lambda)$ can be computed in non-uniform PPT in λ by a simple lookup table. This is enough for where we use the lemma as we consider non-uniform adversaries for simplicity. We could, however, also get impossibility for uniform adversaries. If we allow p to be randomized (and all subsequent proofs can handle a randomized p), then we can simply set ℓ as in the proof and do λ runs of the experiment. We can then let $p(\lambda)$ be any position where $\mathcal{P}_{\Delta}^{(p,q')} \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset$ happens with frequency at least $\frac{1}{2}$. It is easy to see that such a position exists and will have $\Pr_{\lambda}[\mathcal{P}_{\Delta}^{(p,q')} \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset] \neq \mathsf{negl}(\lambda)$ in a fresh run.

Before giving the full proof, we prove a warmup case (Lemma 6) to give the intuition of the proof up front. The lemma just says that if a long witness is hashed and then the witness extracted, then it is the original witnesses which is extracted, or collision intractability is broken. We then later show how to exploit this to get impossibility by showing that it cannot be the case that the original witness is extracted.

We describe a class of adversaries $\mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,\vec{w},\rho}$ in Fig. 2. Let $\mathcal{A}^{(\cdot)}_{\ell}$ denote the random variable describing $\mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,\vec{w},\rho}$, where H, $\vec{w}$ and ρ are sampled at random. And let $\vec{w}(\mathcal{A}^{\mathcal{O}}_{\ell})$ denote the witnesses used by this adversary when run with oracle $\mathcal{O}$.

Lemma 6. *There exists a PPT algorithm $\mathbb{E}$ such that when $\mathcal{O}$ is a random oracle and $(\mathsf{H}, M_\ell, \pi_\ell) \leftarrow \mathcal{A}^{\mathcal{O}}_{\ell}$ then $\mathbb{E}^{\mathcal{A}_\ell, \mathcal{O}} = \vec{w}(\mathcal{A}^{\mathcal{O}}_{\ell})$ except with negligible probability.*

Proof. Since $\mathcal{A}^{\mathcal{O}}_{\ell}$ internally runs an honest proof using $\mathbb{P}$ we have that $\mathbb{V}^{\mathcal{O}}(\mathsf{H}, M_\ell, \pi_\ell) = \top$ except with negligible probability. So, by knowledge soundness we have that there exists a PPT extractor $\mathbb{E}$ such that if we let $\vec{w}' = \mathbb{E}^{\mathcal{A}_\ell, \mathcal{O}}$ then $M_\ell =$

$\mathsf{H}^\ell(M_0, \vec{w}')$ except with negligible probability. Let $\vec{w} = \vec{w}(\mathcal{A}_\ell^{\mathcal{O}})$. We have by construction that $M_\ell = \mathsf{H}^\ell(M_0, \vec{w})$. This implies that $\vec{w}' = \vec{w}$ or $(\vec{w}, \vec{w}')$ is a collision for H. It is therefore enough to prove that $(\vec{w}, \vec{w}')$ is a collision for H with negligible probability. This follows from a simple reduction to collision intractability of H using the fact that $\mathbb{E}$ is a fixed PPT algorithm and H is chosen at random after $\mathbb{E}$ is fixed. □

The above simple case shows that if the proof system has knowledge soundness we can make the extractor extract the long witness $\vec{w}$ from blackbox interaction with the adversary. The only way the extractor learns information is via the queries of the adversary to the random oracle. We now show that it is possible for $\mathcal{A}$ to use a fake hardcoded oracle for a long stretch of the proof and still have the proof be accepted with good probability. This is because the verifier does not have queries enough to test a query from all proof steps. During this stretch the adversary will not query the real oracle $\mathcal{O}$. So there is no interaction with the extractor. Hence the extractor does not learn enough about the witness used during the stretch to be able to extract it. We now flesh out this intuition.

The adversary $\mathcal{A}^{\mathcal{O}}_{\mathsf{H},\ell,\vec{w},\rho}$ has the following values hard-coded. A hash function H, the number of steps ℓ, the witnesses $\vec{w}$, and a random tape $\rho = (\rho_1, \ldots, \rho_\ell)$ long enough to provide $\mathbb{P}$ with randomness ℓ times. Let $\mathcal{O}$ denote the oracle used by the adversary. The adversary proceeds as follows.

1. Let $M_0 = 0^\lambda$ and $\pi_0 = \epsilon$.
2. For $i = 1, \ldots, \ell$ compute $M_i = \mathsf{H}(M_{i-1}, w_i)$ and
$$\pi_i = \mathbb{P}^{\mathcal{O}}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i) \ .$$
3. Output $(\mathsf{H}, M_\ell, \pi_\ell)$.

Fig. 2. $\mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,\vec{w},\rho}$

The transition function we look at is simply collision intractable hashing. We describe the class of transition functions $\mathcal{T}$. We assume we have a family of collision intractable hash functions $\mathsf{H} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$. The witnesses are given by $\vec{w} \in (\{0,1\}^\lambda)^\ell$. The step function T is represented by a hash function H. We always let $M_0 = 0^\lambda$ and the step function is given by $M_i = T(M_{i-1}, w_i) = \mathsf{H}(M_{i-1}, w_i)$. Since M_0 is fixed we drop it from the notation below.

We describe a class of adversaries in Fig. 3. In the proof we will need to go through some hybrids. For simplicity we provide a single adversary with some parameters (two oracles and a binary switch) allowing to produce all the hybrids. For the same purpose we define in Fig. 4 an experiment with two binary switches a and b.

The adversary $\mathcal{A}^{\mathcal{O}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},b}$ has the following values hard-coded. A hash function H, the number of steps ℓ, a stretch (p,q), the *pre-stretch witnesses* $\vec{w}^{\text{PRE}} = (w_1,\ldots,w_{p-1})$, the *post-stretch witnesses* $\vec{w}^{\text{POST}} = (w_{p+q},\ldots,w_\ell)$, a random tape ρ long enough to provide $\mathbb{P}$ with randomness ℓ times, an oracle $\widetilde{\mathcal{O}} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ called the *stretch oracle*, an oracle $\widetilde{\mathcal{W}} : \{0,1\}^* \to (\{0,1\}^\lambda)^q$ called *the witness oracle*, and a switch $b \in \{0,1\}$. Let $\mathcal{O}$ denote the oracle which the adversary has oracle access to. The adversary proceeds as follows.

1. Let $M_0 = 0^\lambda$ and $\pi_0 = \epsilon$.
2. Let $(w_1,\ldots,w_{p-1}) = \vec{w}^{\text{PRE}}$.
3. For $i = 1,\ldots,p-1$ compute $M_i = \mathsf{H}(M_{i-1}, w_i)$ and
$$\pi_i = \mathbb{P}^{\mathcal{O}}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i) \ .$$
4. Let $\mathcal{P}^{\text{PRE}}_{\downarrow} = \mathcal{P}^{(1,p-1)}_{\cup}$ be the queries from $\mathbb{P}$ to $\mathcal{O}$ in the above step. Let $\mathcal{O}^{\text{STR}} = [\mathcal{P}^{\text{PRE}}_{\downarrow} \mapsto \mathcal{O}, \widetilde{\mathcal{O}}]$. For the i'th query $\mathfrak{q}_i \in \mathcal{P}^{\text{PRE}}_{\downarrow}$ in order of appearance in the execution let $y_i = \mathcal{O}(\mathfrak{q}_i)$ be the reply given by $\mathcal{O}$ to the query $\mathfrak{q}_i$ in the above step, let $h = |\mathcal{P}^{\text{PRE}}_{\downarrow}|$, and define the *query tag* $T = ((\mathfrak{q}_1, y_1),\ldots,(\mathfrak{q}_h, y_h))$.
5. Define the *stretch witnesses* $\vec{w}^{\text{STR}} = (w_p,\ldots,w_{p+q-1}) = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})$.
6. For $i = p,\ldots,p+q-1$ compute $M_i = \mathsf{H}(M_{i-1}, w_i)$ and
$$\pi_i = \mathbb{P}^{\mathcal{O}^{\text{STR}}}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i) \ .$$
7. Let $\mathsf{used}_{p+q} = \mathsf{used}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i)$.
8. Let $\mathcal{P}^{\text{STR}}_{\downarrow} = \mathcal{P}^{(p,p+q-1)}_{\cup} \setminus \mathcal{P}^{(1,p-1)}_{\cup}$ be the queries from $\mathbb{P}$ to $\widetilde{\mathcal{O}}$ in the above step.
9. Let $\mathcal{O}^{\text{POST}}_0 = [\mathcal{P}^{\text{PRE}}_{\downarrow} \mapsto \mathcal{O}, \mathcal{P}^{\text{STR}}_{\downarrow} \mapsto \widetilde{\mathcal{O}}, \mathcal{O}]$.
10. Let $\mathcal{O}^{\text{POST}}_1$ be the following oracle.
$$\mathcal{O}^{\text{POST}}_1(\mathfrak{q}) = \begin{cases} \widetilde{\mathcal{O}}(\mathfrak{q}) & \text{if } \mathsf{used}_{p+q}(\mathfrak{q}) = \top \wedge \mathfrak{q} \notin \mathcal{P}^{\text{PRE}}_{\downarrow} \\ \mathcal{O}(\mathfrak{q}) & \text{otherwise} \end{cases}$$
11. For $i = p+q,\ldots,\ell$ compute $M_i = \mathsf{H}(M_{i-1}, w_i)$ and
$$\pi_i = \mathbb{P}^{\mathcal{O}^{\text{POST}}_b}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i) \ .$$
12. Output $(\mathsf{H}, M_\ell, \pi_\ell)$.

Fig. 3. $\mathcal{A}^{\mathcal{O}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},b}$

Note that the oracle $\mathcal{O}^{\text{STR}} = [\mathcal{P}_{\downarrow}^{\text{PRE}} \mapsto \mathcal{O}, \widetilde{\mathcal{O}}]$ used during the stretch will send all fresh queries to the simulated oracle $\widetilde{\mathcal{O}}$ so it will make no new queries to the real oracle $\mathcal{O}$. Note that $\mathcal{O}_1^{\text{POST}}$ is the oracle which behaves like the real random oracle $\mathcal{O}$ except on queries which according to used_{p+q} were made while proving for the stretch witness. The queries $\mathsf{used}_{p+q}(\mathfrak{q}) = \top \wedge \mathfrak{q} \notin \mathcal{P}_{\downarrow}^{\text{PRE}}$ are those presumable made before the stretch ended and not before the stretch started. For such stretch queries it uses the simulated oracle $\widetilde{\mathcal{O}}$.

The experiment $\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$ runs as follows.

1. Pick $\mathsf{H}, \vec{w}^{\text{PRE}}, \vec{w}^{\text{POST}}, \rho$ at random.
2. Let $\widetilde{\mathcal{O}}_0$ and $\widetilde{\mathcal{W}}_0$ be uniformly random functions from their domain. Let $\widetilde{\mathcal{O}}_1$ and $\widetilde{\mathcal{W}}_1$ be pseudo-random functions over their domains, specified by uniformly random keys $O, W \in \{0,1\}^{\lambda}$.
3. Let $\mathcal{A}^{(\cdot)} = \mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}}_a,\widetilde{\mathcal{W}}_a,b}$.
4. Let $(\mathsf{H}, M_\ell, \pi_\ell) \leftarrow \mathcal{A}^{\mathcal{O}}$.
   - Let $\mathcal{P}_{\downarrow}^{\text{STR}}$ denote the set $\mathcal{P}_{\downarrow}^{\text{STR}}$ used inside $\mathcal{A}$. Similarly for other variables used by $\mathcal{A}$ like $\mathcal{O}_b^{\text{POST}}$, used_{p+q}, and $\mathcal{P}_{\cup}^{(i,k)}$.
   - Let NSF be the *no structure failure* event that it did *not* happen that $\mathcal{O}_b^{\text{POST}}$ was queried on an $\mathfrak{q}$ such that ($\mathsf{used}_{p+q}(\mathfrak{q}) = \top$ and $\mathfrak{q} \notin \mathcal{P}_{\cup}^{(1,p+1)}$) or ($\mathsf{used}_{p+q}(\mathfrak{q}) = \bot$ and $\mathfrak{q} \in \mathcal{P}_{\cup}^{(1,p+1)}$).
5. Let $\mathcal{O}_0 = \mathcal{O}$ and let $\mathcal{O}_1 = \mathcal{O}_b^{\text{POST}}$ be the oracle used in $\mathcal{A}^{\mathcal{O}}$.
6. For $c = 0, 1$ let $J_c = \mathbb{V}^{\mathcal{O}_c}(\mathsf{H}, M_\ell, \pi_\ell)$.
   - Let $\mathcal{V}_c = \mathcal{V}_{\downarrow}^{(\ell)}$ be the queries from $\mathbb{V}^{(\cdot)}(\mathsf{H}, M_\ell, \pi_\ell)$ to its oracle $\mathcal{O}_c$ and let QFS_c be the *query-free stretch* event that $\mathcal{V}_c \cap \mathcal{P}_{\downarrow}^{\text{STR}} = \emptyset$. Let $\mathsf{QFS} = \mathsf{QFS}_0$.
   - Let VER_c be the event that $J_c = \top$.
7. Let $\vec{v} = \mathbb{E}^{\mathcal{A}^{(\cdot)},\mathcal{O}}$ and let $\vec{v}^{\text{PRE}} \| \vec{v}^{\text{STR}} \| \vec{v}^{\text{POST}} = \vec{v}$, where $|\vec{v}^{\text{PRE}}| = p - 1$ and $|\vec{v}^{\text{STR}}| = q$.
   - Let XTF be the *extraction of full witness* event that $J_0 = \bot$ or $\vec{v} = \vec{w}$.
   - Let XTS be the *stretch extraction* event that $\vec{v}^{\text{STR}} = \vec{w}^{\text{STR}}$.

Fig. 4. $\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$

We are particularly interested in the event $\mathsf{XTS}_{a=0,b=1}$. This is the event that the random witness used during the stretch is correctly extracted when $a = 0$ (such that a uniformly random oracle and uniformly random witness are used) and $b = 1$ (such that $\mathcal{O}_1^{\text{POST}}$ is used for the post-stretch part of the proof in step 11). Below we prove two lemmas about $\mathsf{XTS}_{a=0,b=1}$, which allows us to give the following proof for Theorem 2.

Proof (Proof of Theorem 2). Under the premises of the theorem we can prove both Lemma 7 and Lemma 8, and these two lemmas are in contradiction. □

We first prove:

Lemma 7. *Under the premises of Theorem 2 the following holds. There exists a polynomial stretch length $q \in \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$ such that it is not possible to set the number of steps ℓ to a polynomial and set the stretch position p such that*

$$\Pr[\mathsf{XTS}_{a=0,b=1}] \geq 1/\lambda^{O(1)} .$$

Proof. Note that the stretch witness is picked uniformly at random using the oracle $\widetilde{\mathcal{W}}$. This means that it has entropy $q|w|$. The extractor needs to learn this many bits to extract the stretch witness. Let us be curious about from where it could learn this many bits of information.

Note that the stretch witness is computed as $\vec{w}^{\text{STR}} = (w_p, \ldots, w_{p+q-1}) = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})$, where (T, M_{p-1}, π_{p-1}) is the entire state of the proof up to step 5, including previous random-oracle queries and replies. This ensured that if the adversary is rewound behind step 5 then an independent, uniformly random stretch witness is picked. So we can ignore extractors rewinding behind step 5.

The extractor also learns no information *during* steps 5 and 6 as the prover queries the stretch oracle $\widetilde{\mathcal{O}}$ during the stretch, not the real oracle. Hence the adversary does not interact with the extractor at all during steps 5 and 6.

We finally argue that interacting with the adversary after step 6 cannot leak the entire stretch witness. To see this note that the adversary only needs the post-stretch witness and $\mathcal{O}_1^{\text{POST}}$ to run after step 6. The post-stretch witness is independent of the stretch witness, and we can represent $\mathcal{O}_1^{\text{POST}}$ such that it does not contain all information about the stretch witness. Namely, we could indistinguishably for the extractor switch to $a = 1$ such that $\widetilde{\mathcal{O}}$ is pseudo-random and specified by a short key O. Note that we can then compute $\mathcal{O}_1^{\text{POST}}$ given O, used_{p+q}, $\mathcal{P}_{\downarrow}^{\text{PRE}}$, and $\mathcal{O}$. The oracle $\mathcal{O}$ and the queries $\mathcal{P}_{\downarrow}^{\text{PRE}}$ were chosen before the stretch witness was chosen, so cannot depend on it. The key O is short so can only contain little information on the stretch witness. Finally, we assumed used_{p+q} can be computed from the state of the prover which we have assumed is $\mathsf{poly}(\lambda, \log \ell)$ for some fixed polynomial. So if we set q to some larger polynomial, then the state of the prover cannot contain the stretch witness. By making q large enough we can ensure that the stretch witness can be guessed only with negligible probability from the state of the adversary after step 6, which proves the lemma. □

We then prove the following contradicting lemma:

Lemma 8. *Under the premises of Theorem 2 the following holds. For all polynomials $q \in \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$ it is possible to set ℓ to a polynomial and set p such that*

$$\Pr[\mathsf{XTS}_{a=0,b=1}] \geq 1/\lambda^{O(1)} .$$

Proof. We will argue that when $a = 0$ and $b = 1$ then we can for all q set ℓ such that $J_0 = \top$ with polynomial probability. When this happens, then knowledge soundness gives us that the extraction $\vec{v} = \mathbb{E}^{\mathcal{A}^{(\cdot)}, \mathcal{O}}$ must be correct and therefore

$\mathsf{XTS}_{a=0,b=1}$ happened. Note that when $b = 1$ then the adversary's proof is constructed with $\mathcal{O}_1 = \mathcal{O}_1^{\text{POST}}$ but $\mathbb{E}^{\mathcal{A}^{(\cdot)},\mathcal{O}}$ extracts with $\mathcal{O}$. The key to the proof is to show that with polynomial probability it holds that 1) the proof constructed by the adversary verifies against $\mathcal{O}_1^{\text{POST}}$ and that 2) $\mathcal{O}_1^{\text{POST}}(\mathfrak{q}) = \mathcal{O}(\mathfrak{q})$ for all queries made by the verifier. The first part would give $J_1 = \top$ and the second part would give $J_0 = J_1$, and we would be done.

We first prove 2). Note that $\mathcal{O}_0^{\text{POST}}$ and $\mathcal{O}_1^{\text{POST}}$ only differ in which queries they send to the simulated oracle $\widetilde{\mathcal{O}}$. The oracle $\mathcal{O}_0^{\text{POST}}$ sends exactly the queries made for the first time in the stretch. The oracle $\mathcal{O}_1^{\text{POST}}$ tries to do the same but relies on used_{p+q} correctly identifying queries made before step $p + q$. Under the assumption that NSF happens we have that $\mathcal{O}_0^{\text{POST}} = \mathcal{O}_1^{\text{POST}}$ and by the assumption that the proof system is $1/\lambda^{O(1)}$-SOQ we can assume that NSF happens with polynomial probability. So let us proceed under the assumption that $\mathcal{O}_0^{\text{POST}} = \mathcal{O}_1^{\text{POST}}$. Then note that $\mathcal{O}_0^{\text{POST}} = \mathcal{O}$ unless queried on a query made by the prover during the stretch and use Lemma 5 to set ℓ large enough that the verifier with polynomial probability does not make such a query. This ensure that $\mathcal{O}_1(\mathfrak{q}) = \mathcal{O}(\mathfrak{q})$ for all queries made by the verifier, as desired.

To prove 1) we then need to argue that it is also the case that the proof made by the adversary verifies against $\mathcal{O}_1^{\text{POST}}$. But note that when $\widetilde{\mathcal{O}}$ is uniformly random, then $\mathcal{O}_0^{\text{POST}}$ is just another uniformly random oracle. And therefore $\mathcal{O}_1^{\text{POST}} = \mathcal{O}_0^{\text{POST}}$ is also a uniformly random oracle. So the proof of the adversary is a random, honestly generated proof relative to a uniformly random oracle $\mathcal{O}_1^{\text{POST}}$. Therefore, by completeness, it will also verify relative to $\mathcal{O}_1^{\text{POST}}$. □

References

1. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), https://eprint.iacr.org/2018/046
2. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. pp. 31–60 (2016). https://doi.org/10.1007/978-3-662-53644-5_2
3. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. pp. 276–294 (2014). https://doi.org/10.1007/978-3-662-44381-1_16
4. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. pp. 111–120 (2013). https://doi.org/10.1145/2488608.2488623
5. Bünz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. Cryptology ePrint Archive, Report 2020/1618 (2020), https://eprint.iacr.org/2020/1618
6. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. pp. 1–18 (2020). https://doi.org/10.1007/978-3-030-64378-2_1
7. Chen, M., Chiesa, A., Spooner, N.: On succinct non-interactive arguments in relativized worlds. pp. 336–366 (2022). https://doi.org/10.1007/978-3-031-07085-3_12

8. Chiesa, A., Liu, S.: On the impossibility of probabilistic proofs in relativized worlds. pp. 57:1–57:30 (2020). https://doi.org/10.4230/LIPIcs.ITCS.2020.57
9. Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. pp. 769–793 (2020). https://doi.org/10.1007/978-3-030-45721-1_27
10. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. pp. 310–331 (2010)
11. Chiesa, A., Yogev, E.: Subquadratic SNARGs in the random oracle model. pp. 711–741 (2021). https://doi.org/10.1007/978-3-030-84242-0_25
12. Chiesa, A., Yogev, E.: Tight security bounds for micali's SNARGs. pp. 401–434 (2021). https://doi.org/10.1007/978-3-030-90459-3_14
13. Choudhuri, A.R., Jain, A., Jin, Z.: Non-interactive batch arguments for NP from standard assumptions. pp. 394–423 (2021). https://doi.org/10.1007/978-3-030-84259-8_14
14. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. pp. 99–108 (2011). https://doi.org/10.1145/1993636.1993651
15. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems **7**(1), 1–32 (Dec 1994). https://doi.org/10.1007/BF00195207
16. Haitner, I., Nukrai, D., Yogev, E.: Lower bound on SNARGs in the random oracle model. Cryptology ePrint Archive, Report 2022/178 (2022), https://eprint.iacr.org/2022/178
17. Kalai, Y.T., Paneth, O., Yang, L.: How to delegate computations publicly. pp. 1115–1124 (2019). https://doi.org/10.1145/3313276.3316411
18. Lipmaa, H., Pavlyk, K.: Gentry-wichs is tight: a falsifiable non-adaptively sound SNARG. pp. 34–64 (2021). https://doi.org/10.1007/978-3-030-92078-4_2
19. Micali, S.: CS proofs (extended abstracts). pp. 436–453 (1994). https://doi.org/10.1109/SFCS.1994.365746
20. Unruh, D.: Random oracles and auxiliary input. pp. 205–223 (2007). https://doi.org/10.1007/978-3-540-74143-5_12
21. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. pp. 1–18 (2008). https://doi.org/10.1007/978-3-540-78524-8_1