On Valiant's Conjecture: Impossibility of Incrementally Verifiable Computation from Random Oracles

Mathias Hall-Andersen[*], Jesper Buus Nielsen[†]

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 under some mild extra assumptions on the proof system the conjecture is true: the standard random-oracle model does not allow incrementally verifiable computation without making computational assumptions. Two extra assumptions under which we can prove the conjecture are 1) the proof system is also zero-knowledge or 2) when the proof system makes a query to its random oracle it can know with non-negligible probability whether the query is fresh or was made by the proof system earlier in the construction of the proof.

1 Intro

Incrementally Verifiable Computation. In his landmark paper Paul Valiant [Val08] 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

[*]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).

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.

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 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[Val08]

In [CL20] Chiesa and Liu show impossibility results for proofs in relativized worlds, i.e., proof of 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. As noted in [CL20] 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 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 rule out that not *all* statements have such a proof. And the end result of Valiant's approach is to give non-trivial PCPs about random oracle devoid computation, which is not ruled out by the Chiesa-Liu results either.

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* PCPs in the random oracle model using some other strategy, as already noted by Valiant.

[1]This description is oversimplified but will suffice for our discussion. The real recursive strategy is m ore involved to tame the complexity of knowledge extraction.

... It remains an interesting question whether the goals of this paper may be attained in some other way using random oracles. ...

–Paul Valiant[Val08]

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 additional but natural properties of the proof system. We discuss these assumptions below.

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. 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 [CT10] 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. Note in particular that PCD implies "non-deterministic" computation: where the witness can be provided piecemeal during the computation.

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 a generalization of IVC where each step of the Turning machine may take a witness.

Note that for the setting of proof carrying data it makes sense to require that each proof is zero-knowledge in itself: different steps of the computation could be performed by mutually distrustful parties which do not want to share their secrets. As discussed in more detail later, one of our additional assumptions will be that the proof is zero-knowledge.

Our impossibility result applies only to the setting where a large witness is consumed piecemeal. Since the original construction of Valiant easily generalises to give a construction for "non-deterministic" computation this seems like a mild relaxation of the impossibility result. And it certainly applies to many modern uses of incremental proofs.

PCD via Recursion. In Valiant's original paper [Val08] IVC is constructed using a tree of linear-time extractable CS proofs [Mic94]: 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). 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 [BCCT13] [BCTV14] from zk-SNARKS the proofs are composed iteratively, which implies that the proof is only 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 [BCMS20a, BCL$^+$20]. 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 $\mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$.

In This Paper. In this paper we show that succinct non-deterministic IVC from random oracles is impossible in the following three cases.

1. There exist collision resistant hash functions and the proof system has *knowledge soundness* (possibly with non-blackbox extraction) and is *zero-knowledge*. Knowledge soundness and zero-knowledge may depend on standard-model falsifiable computation assumptions.

2. There exist perfectly binding rerandomizable commitments and the proof system has *soundness* and is *zero-knowledge*. Both soundness and zero-knowledge may depend on standard-model falsifiable computation assumptions.

3. There exists collision resistant 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 were made before.

The first two results assume that the incremental proof is also zero-knowledge. This seems like a mild additional assumption. Succinct arguments already information theoretically hides most of the witness, as the proof is much shorter than the witness. It it hard to imagine techniques allowing this information theoretic partial hiding but not full zero-knowledge. For general PCD zero-knowledge is even a natural requirement. The third result complements the first two results by showing 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. We believe that the freshness assumption is not needed but is an artefact of our proof techniques.

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 $\vec{w} = (w_1, \dots, w_\ell)$, where w_i is given (only) in step i. Each w_i is security parameter long.

Results 1 and 2. In the first two results 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 w_i. We sketch why this is true.

For the case of collision resistant 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 break collision resistance.

For the case of perfectly binding rerandomizable commitments each step gets as input an in-commitment c_{i-1} and and out-commitment c_i and the witness is the randomness used to produce c_i as a rerandomisation of c_{i-1}. The in-commitment c_0, of step 1, is a commitment of 0. By perfect binding it follows that for true instances the out-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 an out-commitment c_i of 0 from an out-commitment of 1. Hence we can do a switch from an in-commitment c_{i-1} of 0 to an out-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 of the first two results by showing that if the verifier does not make $\Theta(\ell)$ queries 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 this query by verifier n related to proof m. There are $\Theta(\ell^2)$ of these. 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. We then argue that this $x_{m,n}$ 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 5 using a so-called blocking set argument.

Result 3. For result 3 we use a different 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 (pseudo-random) oracle instead of the real one for all fresh queries. 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 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.

Furthermore, the stretch witnesses are picked on-the-fly by the adversary using a pseudo-random function applied to the transcript of the proof up to position p_1. This ensures that the extractor cannot use rewinding to get information on the stretch witnesses used in the main execution which needs to be extracted. In particular, if the adversary is rewound and given different replies to oracle queries then it will use other witnesses during the stretch $[p_1, p_2]$ than the ones used in the main execution which needs to be extracted.

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 proof system after step p_2. We could give p_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 uses poly-time incompressibility of pseudo-random strings. This shows that a blackbox extractor cannot compute the stretch witnesses from *blackbox* access to the adversary, violating knowledge soundness.

While it seems intuitively clear that if we have a random witness of length $p_2 - p_1$ and the only information passed to the extractor about it is of size $|\sigma_2| < p_2 - p_1$, then the extractor cannot compute the witness, the proof is more subtle. In particular, when we do the proofs from step p_2 till ℓ, we need that queries identical to fresh queries made during the stretch are answered using the simulated oracle used during the stretch. Otherwise we cannot appeal to correctness of the proof system and conclude that the final proof must be accepting. And we need that the proof is accepting to be able to apply the extractor. We cannot *a priori* pass the set of fresh queries on to future proofs: It might be so large that it could encode the stretch witnesses. We *can* pass on a description of the simulated oracle, as it it independent of the stretch witnesses. However, the queries themselves might encode information about the stretch witnesses. We therefore 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 one. The mechanism need not be perfect. If it 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.

Generalizations Our results apply directly to schemes which only rely on random oracles, like that of Valiant [Val08] (based on CS proofs) and recursive Fractal [COS20]. 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 in results 1 and 2 the zero-knowledge may depend on computational assumptions, as long as these are not phrased via relativized worlds extra to the random oracle model. Results 1 and 2 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

Result 3 also stands if knowledge soundness is proven using additional computational assumptions, as long as these are blackbox and are not phrased via a relativized world. This still forces all information on the stretch witness to pass to the extractor via random oracle queries. Technically we quantify the extractor before the adversary and only give the extractor blackbox access to the adversary, so extraction of information need to pass via the queries to the random oracle. However, the proof that the extraction strategy works is allowed to depend on additional computational assumptions like existence of collision resistant hash function. Technically the proof may also depend on non-falsifiable assumptions, but they are probably hard to utilise when the extractor is quantified before the adversary.

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.

It is interesting to investigate how simple an oracle may allow incremental PCD, and our results show that we must look for non-local oracles.

Relation to Other Impossibilities. Chiesa and Liu [CL20] 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. Since in particular the impossibility of Gentry-Wichs [GW11] for (adaptively sound) zk-SNARGs applies, one could not 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 [Mic94] from PCPs and the compiler of Ben-Sasson et al. [BCS16] applied to round-by-round sound Holographic IOPs like Fractal [COS20] and zk-STARKs [BBHR18].

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:*

```
$T^\ell(M_0, \vec{w} = (w_1, \ldots, w_\ell))$
1 :   $M_{\ell-1} \leftarrow T^{\ell-1}(M, (w_1, \ldots, w_{\ell-1}))$
2 :   return $T(M_{\ell-1}, w_\ell)$
```

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:*

$\mathbb{P}^{\mathcal{O}}(T, M, w, \pi) \mapsto \pi'$**.** *Takes a description of the state transition T, the current state M, some additional input w and a proof π of $(M_0, M) \in \mathcal{L}_{(T,\ell)}$. Let $M' = T(M, w)$, the algorithm returns a proof π' of $(M_0, M') \in \mathcal{L}_{(T,\ell+1)}$.*

$\mathbb{V}^{\mathcal{O}}(x = (T, M_0, M_\ell, \ell), \pi) \mapsto \{\top, \bot\}$**.** *Verifies a proof π of the statement $(M_0, M_\ell) \in \mathcal{L}_{(T,\ell)}$; i.e., there exists a sequence of witnesses $\vec{w}$ which takes M_0 to M_ℓ in ℓ steps.*

We assume for notational convenience (and without loss of generality) that the proof for the trivial statement $x = (T, M_0, M_0, 0)$ (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 $(T, \vec{w}, M_0, \ell)$*

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

We assume perfect completeness for simplicity, however all our results generalize to the slightly weaker case where the scheme has a negligible probability of failure.

Remark 1. *An alternative definition (similar to [BCMS20b]) 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 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 (Non-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 a description of the adversary, recovers a valid witness $(w_1, \ldots, w_\ell)$ given $\mathcal{A}^{(\cdot)}$ except with negligible probability. Formally:*

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

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

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:*

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

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

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 $T \in \mathcal{T}$, $\ell = \mathsf{poly}(\lambda)$, $x = (T, \ell, M_0, M_\ell) \in \mathcal{L}_T$, 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 st. for all PPT $(\mathcal{A}_1, \mathcal{A}_2)$:*

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

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

Where $\mathcal{O}_0 = \mathcal{O}$ and $\mathcal{O}_1 = [\mathcal{Q}, \mathcal{O}]$, where $[\mathcal{Q}, \mathcal{O}]$ is the oracle mapping $\P$ to R_i if $(\P, R_i) \in \mathcal{Q}$ and $\mathcal{O}(\P)$ otherwise. The probability is over $\mathcal{O}$, the random tape of $\mathcal{A}^{(\cdot)}$, $\mathbb{P}$ and $\mathbb{S}$.

Remark 2. *Note that if $\mathcal{A}$ queried on $\P$ in $(T, x, \pi, w) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)$, then we can assume that except with negligible probability $\mathcal{O}_1(\P) = \mathcal{O}(\P)$, i.e., the simulator does not reprogram on $\P$. Namely, if $\mathcal{O}_1(\P) \neq \mathcal{O}(\P)$ happens with non-negligible probability the adversary could remember all queries $\P$ and replies made during the first step and redo them in the second step and guess $b = 1$ when $\mathcal{O}_1(\P) \neq \mathcal{O}(\P)$ and $b = 0$ otherwise. This would break zero-knowledge.*

```
$\mathtt{Game}^{(m)}_{\mathtt{Hiding}}(\mathcal{A}, \lambda)$
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 $b \overset{?}{=} b'$
```

We call the property that the simulator only programs points that were never queried fresh reprogramming *below.*

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.,* $\mathsf{c} = \mathsf{Commit}(\mathsf{pp}, b)$, $b \in \{0,1\}$.

$\mathsf{ReRand} : \mathcal{P} \times \mathcal{C} \times (\{0,1\}^*)^* \to \mathcal{P}$ *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.* $\mathsf{Open}(\mathsf{pp}, b, \mathsf{c}, \mathbf{r} = (r_1, \ldots, r_\ell)) \coloneqq \mathsf{c} \overset{?}{=} \mathsf{ReRand}^m(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, b); \mathbf{r})$ $= \mathsf{ReRand}(\ldots \mathsf{ReRand}(\mathsf{ReRand}(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, b); r_1); r_2), \ldots; r_\ell)$

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.*

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

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:*

$$\Pr\left[\mathit{Game}^{(m)}_{\mathit{Hiding}}(\mathcal{A}, \lambda)\right] - {}^1\!/\!_2 \leq \mathsf{negl}(\lambda)$$

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.

2.2 One-Way Functions and Collision Intractable Hashes

Definition 11 (One Way Functions)**.** *A family of functions where:*

$$\Pr\left[\mathsf{OWF}(\mathsf{x}') = \mathsf{y} \;\middle|\; \begin{array}{c}\mathsf{OWF} \leftarrow \mathsf{Gen}(1^\lambda); \mathsf{x} \leftarrow\$\, \{0,1\}^\lambda \\ \mathsf{y} \leftarrow \mathsf{OWF}(\mathsf{x}); \mathsf{x}' \leftarrow \mathcal{A}(\mathsf{OWF}, \mathsf{y}, 1^\lambda)\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 12 (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.*

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

2.3 Basic Notation

Definition 13 (Stretch)**.** *Let a length ℓ of a proof be fixed, i.i., 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 14 (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 $\mathcal{V}^{(i)}_{\Delta} = \mathcal{V}^{(i)}_{\downarrow} \setminus \mathcal{V}^{(1,i-1)}_{\cup}$ and $\mathcal{P}^{(i)}_{\Delta} = \mathcal{P}^{(i)}_{\downarrow} \setminus \mathcal{P}^{(1,i-1)}_{\cup}$. 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 15 (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 ¶, if $\text{¶} \in \mathcal{Q}_1$ then output $\mathcal{O}_1(\text{¶})$. Otherwise output $\mathcal{O}(\text{¶})$. In general, let $[\mathcal{Q}_1 \mapsto \mathcal{O}_1, \ldots, \mathcal{Q}_\ell \mapsto \mathcal{O}_\ell, \mathcal{O}] = [\mathcal{Q}_1 \mapsto \mathcal{O}_1, [\mathcal{Q}_2 \mapsto \mathcal{O}_2, \ldots, \mathcal{Q}_\ell \mapsto \mathcal{O}_\ell, \mathcal{O}]]$.*

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

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.

3 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 resistant functions exists. The other is for the case where the proof system has just soundness but under a stronger computational assumption. We start with the theorem statement and a proof appealing to future lemmas.

Theorem 1 (Impossibility of Non-Trivial ZK Non-Deterministic $\mathcal{O}$-IVC)**.** *The existance 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 Intractablity Precludes Knowledge-Soundness.*** *Assuming the existance of a family of collision intractable functions $\mathcal{H}_\lambda$, there exists a transition functions T_H such that any zero-knowledge, knowledge-sound $\mathcal{O}$-IVC scheme for T_H must have a verifier with running time linear in ℓ.*
- ***Rerandomizable Commitments Precludes Soundness.*** *Assuming the existance of perfectly binding rerandomizable commitment schemes, there exists transition functions T_pp such that any zero-knowledge and computationally sound $\mathcal{O}$-IVC scheme for T_pp must have a verifier with running time linear in ℓ.*

Proof. By combining Lemma 2, Lemma 4, and Lemma 5 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 3, Lemma 4, and Lemma 5 similarly for T_pp. $\square$

The following lemmas states 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 $\vec{w}^{(\bar{m})} := (w_1, \ldots, w_{m-1}, \perp, w_{m+1}, \ldots, w_\ell)$ (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 2 (All Witnesses Are Required for Knowledge Soundness)**.** *For a (randomly sampled) collision resistant 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 <u>*except for step m*</u>*, with only negligible probability. i.e., there exists a negligible function* $\mathsf{negl}(\lambda)$ *st.*

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

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 12):

```
$(v_1, v_2) \leftarrow \mathcal{A}(\mathsf{H})$
        // 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 :    $M_0 = \epsilon; \textbf{for } i \in [\ell] : M_i = T_{\mathsf{H}}(M_{i-1}, w_i);$
 3 :    $\vec{w}^{(\bar{m})} := (w_1, \dots, w_{m-1}, \perp, w_{m+1}, \dots, w_\ell)$
 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 :    $M'_0 = \epsilon; \textbf{for } i \in [\ell] : M'_i = T_{\mathsf{H}}(M'_{i-1}, w'_i);$
        // Look for collision.
 7 :    $\textbf{for } i \in [\ell - 1] :$
 8 :       $v_1 := M_i \| w_{i+1};\ v_2 := M'_i \| w'_{i+1}$
 9 :       $\textbf{if } M_{i+1} = M_{i+1} \wedge v_1 \neq v_2$
10 :          $\textbf{return } (v_1, v_2)$
11 :    $\textbf{return } \perp$
```

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 randomly. Hence the extractor given only $M_m := f(w_m)$ recovers w'_m st. $w_m \neq w'_m$ with probability at least $1/2 - \mathsf{negl}(\lambda)$: violating 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 3 (All Witnesses Are Required for Soundness)**.** *For a perfectly binding rerandomizable commitment scheme, consider the following step function* $T_{\mathsf{pp}}(M, w) := \mathsf{ReRand}(\mathsf{pp}, M; w)$

– 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^{ℓ}_{pp} execution given all witnesses $\underline{\text{except for step } m}$, with only negligible probability, i.e., there exists a negligible function $\mathsf{negl}(\lambda)$ st.

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

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:

```
A_Hiding(find, pp, 1^λ)
    // Sample randomness / witnesses for ℓ steps.
1 : $\vec{r} \leftarrow\!\$\, \mathcal{U}^{\ell}_{\mathsf{poly}(\lambda)}; \mathsf{st} = \vec{r}$
    // Get rerandomisation of either 0 or 1.
2 : $v^{(0)} = 0; v^{(1)} = 1$
3 : $\vec{r}^{(0)} = \vec{r}^{(1)} = (r_1, \dots, r_{m-1})$
4 : return $((v^{(0)}, \vec{r}^{(0)}), (v^{(1)}, \vec{r}^{(1)}), \mathsf{st})$
```

```
A_Hiding(guess, st, pp, c', 1^λ)
1 : $\vec{r} = \mathsf{st}$
2 : $\vec{w}^{(\bar{m})} := (r_1, \dots, r_{m-1}, \perp, r_{m+1}, \dots, r_{\ell})$
3 : $M_0 = \mathsf{Commit}(\mathsf{pp}, 0); M_m = \mathsf{c}'$
4 : for $i \in [m+1, \ell] : M_i = T_{\mathsf{pp}}(M_{i-1}, w_i)$
5 : $x = (T_{\mathsf{pp}}, M_0, M_{\ell}, \ell)$
6 : $\pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x, \vec{w}^{(\bar{m})}, M_m)$
7 : if $\mathbb{V}^{\mathcal{O}}(x, \pi) = 1$ return 0
8 : 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 2 and Lemma 3 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 14.

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 be the set of query points on which $\mathcal{Q}$ programs, i.e., $S_m = \{\mathtt{q} | \exists y\, ((\mathtt{q}, y) \in \mathcal{Q})\}$. 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 4 (Must Check Programmed Points)**.** *For transition functions T as described in Lemma 2 and Lemma 3 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) st. $\Pr\left[S_m \cap \mathcal{V}_{\downarrow}^{(\ell)} = \emptyset\right] = p$ where p is non-negligible in λ, then construct an adversary violating Lemma 2 and Lemma 3 as follows:

```
$\pi \leftarrow \hat{\mathcal{A}}^{\mathcal{O}}(x, \vec{w}^{(\bar{m})}, M_m)$; produces a proof without $w_m$.
1 :  $x = (T, M_0, M_\ell, \ell); \pi_0 = \epsilon$
2 :  $\vec{w}^{(\bar{m})} = (w_1, \ldots, w_{m-1}, \bot, w_{m+1}, \ldots, w_\ell)$
     // Start execution using first $m-1$ witnesses
3 :  for $j \in [1, m-1]$ :
4 :    $M_j \leftarrow T(M_{j-1}, w_j)$
5 :    $\pi_j \leftarrow \mathbb{P}^{\mathcal{O}}(T, M_{j-1}, w_j, \pi_{j-1})$
     // Simulate step $m$
6 :  $(\mathcal{Q}_m, \pi_m) \leftarrow \mathbb{S}^{\mathcal{O}}(T, M_m, \pi_{m-1})$
     // Finish execution with reprogrammed oracle
7 :  for $j \in [m+1, \ell]$ :
8 :    $M_j \leftarrow T(M_{j-1}, w_j)$
9 :    $\pi_j \leftarrow \mathbb{P}^{[\mathcal{Q}_m, \mathcal{O}]}(T, M_{j-1}, w_j, \pi_{j-1})$
10 : return $\pi_\ell$
```

Where T, M_0 are instantiated as in Lemma 2 and Lemma 3. 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 2 and Lemma 3. □

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 verifier will not query a reprogrammed point and reject the proof. So we do not get a contradiction to to Lemma 2 or Lemma 3. We will therefore need a slightly more subtle strategy. We show that because Lemma 4 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 5 (Too Large Verifier Query Set). *For transition functions T for which the property in Lemma 4 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., $\vec{w}$ st. $T^\ell(M_0, \vec{w}) = M_\ell$. 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. The adversary $\mathcal{B}$ runs all ℓ steps honestly as in HonExp. The adversary $\mathcal{B}$ will compute the blocking set B_n as follows in step n. First it queries the random oracle on all points in $B_{<n} = \cup_{i=1}^{n-1} B_i$. Then for a polynomial p which we specify below it will run the proof forward honestly p times. We assume that $\mathcal{B}$ knows the witnesses, so it can do this. In each step $m \geq n$ it computes $\mathcal{V}_{\downarrow}^{(m)}$ and adds $\mathcal{V}_{\downarrow}^{(m)} \setminus B_{<n}$ to B_n. 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 step m instead of using the witness for this step (as in 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 be 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 a random run 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 $\P$. 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 $\mathcal{V}_{\downarrow}^{(m)}$. It will therefore be added to the blocking set when the adversary adds $\mathcal{V}_{\downarrow}^{(m)} \setminus B_{<n}$ to B_n. Namely, the query $\P$ will not in $B_{<n}$ as the adversary queried on all points in $B_{<n}$ before the simulation step was run. So by *fresh reprogramming* no element from S_n is in $B_{<n}$, and all likely points are in S_n 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 4 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 use that $L_m = S_n \setminus U_m$ 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. $\square$

Remark 3 (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.*

4 Impossibility of Blackbox Construction

In this section we prove impossibility of space bounded proofs 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 Resistance There exist collision resistant 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 collision resistance hash function cannot be assumed to 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. 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. The Structured Queries assumption implies that future proof steps can resolve the type of a query before it is made. This is in principle easy to do: simply pass along the set of all queries which were already made. However, this might break the succinctness of the proof system as a growing set is passed along in each step. The Structured Queries assumption further says that we assume that future proof steps can resolve the the type of a query *using only a succinct state*. It turns out we only need that it can be done with non-negligible probability to get impossibility. We leave it as an open problem whether succinctness is feasible using a proof without structured queries.

The transition function we look at is simply collision resistant hashing. We describe the class of transition functions $\mathcal{T}$. We assume we have a family of collision resistant 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.

Before giving the full proof, we prove a warmup case 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 original witnesses which is extracted, or collision resistance 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 $\mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,\vec{w},\rho}$.

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 $(\mathsf{H}, M_\ell, \pi_\ell)$.

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 reductoin to collision resistance 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.

We first formalize the notion of structured queries.

Definition 16 (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 $(i, ¶) = \mathcal{A}^{\mathcal{O}}(T, \vec{w}, M_0, \vec{\rho})$. We say that the adversary wins if $\mathsf{used}_i(¶) = \top$ and $¶ \notin \mathcal{P}_{\cup}^{(1,i)}$ or $\mathsf{used}_i(¶) = \bot$ and $¶ \in \mathcal{P}_{\cup}^{(1,i)}$. 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 out impossibility result.

We describe a class of adversaries. 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.

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 $(\mathsf{H}, M_\ell, \pi_\ell)$.

For the ensuring proof we define an experiment with three binary switches a, b, c.

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 $\mathcal{P}^{\text{STR}}_{\downarrow}$ denote the set $\mathcal{P}^{\text{STR}}_{\downarrow}$ used inside $\mathcal{A}$. Similarly for other variables used by $\mathcal{A}$ like $\mathcal{O}^{\text{POST}}_b$, used_{p+q}, and $\mathcal{P}^{(i,k)}_{\cup}$.
   - Let NSF be the *no structure failure* event that it did *not* happen that $\mathcal{O}^{\text{POST}}_b$ was queried on an $\mathfrak{q}$ such that ($\mathsf{used}_{p+q}(\mathfrak{q}) = \top$ and $\mathfrak{q} \notin \mathcal{P}^{(1,p+1)}_{\cup}$) or ($\mathsf{used}_{p+q}(\mathfrak{q}) = \bot$ and $\mathfrak{q} \in \mathcal{P}^{(1,p+1)}_{\cup}$).
5. Let $\mathcal{O}_0 = \mathcal{O}$ and let $\mathcal{O}_1 = \mathcal{O}^{\text{POST}}_b$ 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 $\mathcal{V}_c = \mathcal{V}^{(\ell)}_{\downarrow}$ 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 $\mathcal{V}_c \cap \mathcal{P}^{\text{STR}}_{\downarrow} = \emptyset$. 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 $\vec{v}^{\text{STR}} = \vec{w}^{\text{STR}}$.

To avoid confusion, we note that XTS is the unconditioned event that we extract the stretch witness, whereas XTF is the event to extract the *full* witness *conditioned* on the proof verifying. This double asymmetry is on purpose.

We now prove some technical lemmas on relationships between the probabilities that certain events happen in the experiment depending on the setting of switches. We set up some notation for this. An event E is just a predicate on traces of runs of $\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$. For brevity we use $\mathrm{E}_{a,b,c}$ to denote that the event E happened in a random run of $\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$. In particular,

$$\Pr[\mathrm{E}_{a,b,c}] = \Pr_{\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}}[\mathrm{E}] \ .$$

Lemma 7 (a-switch)**.** *For all* $\mathrm{E} \in \{\mathsf{NSF}, \mathsf{QFS}, \mathsf{VER}_0, \mathsf{VER}_1, \mathsf{XTS}, \mathsf{XTF}\}$ *it holds that*

$$\Pr[\mathrm{E}_{0,b,c}] \approx \Pr[\mathrm{E}_{1,b,c}] \ .$$

Proof. For two oracles $\widetilde{\mathcal{O}}$ and $\widetilde{\mathcal{W}}$ let $\mathcal{A}^{\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},(\cdot)}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,b}$ be the adversary $\mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},b}$ but where $\widetilde{\mathcal{O}}$ and $\widetilde{\mathcal{W}}$ are accessed via oracle queries instead of being hardcoded. The main part of the proof is to observe that by construction the adversary only accesses its hardcoded oracles in a blackbox manner, so $\mathcal{A}^{\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},(\cdot)}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,b}$ is well-defined. Clearly

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

where $\equiv$ denotes behavioural equivalence under blackbox access.

Let $\textsc{ExtExp}^{\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},\mathcal{O}}_{\ell,(p,q),b}$ denote the experiment $\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$, where we skip Step 2 and where in Step 3 we let

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

Define $\widetilde{\mathcal{O}}_a$ and $\widetilde{\mathcal{W}}_a$ as in Step 2 in $\textsc{ExtExp}$. Clearly

$$\Pr_{\textsc{ExtExp}^{\widetilde{\mathcal{O}}_a,\widetilde{\mathcal{W}}_a,\mathcal{O}}_{\ell,(p,q),b}}[\mathrm{E}] = \Pr_{\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}}[\mathrm{E}] , \tag{1}$$

as the two experiments are perfectly equivalent modulo how $\mathcal{A}$ accesses $\widetilde{\mathcal{O}}_a$ and $\widetilde{\mathcal{W}}_a$. Now observe that $\textsc{ExtExp}^{\widetilde{\mathcal{O}}_a,\widetilde{\mathcal{W}}_a,\mathcal{O}}_{\ell,(p,q),b,c}$ can be computed in poly-time and by inspecting the trace of a run of $\textsc{ExtExp}^{\widetilde{\mathcal{O}}_a,\widetilde{\mathcal{W}}_a,\mathcal{O}}_{\ell,(p,q),b,c}$ we can for all $\mathrm{E} \in \{\mathsf{NSF}, \mathsf{QFS}, \mathsf{VER}_0, \mathsf{VER}_1, \mathsf{XTS}, \mathsf{XTF}\}$ compute in poly-time whether E occurred. From this it follows by a simple reduction to the pseudorandomness of $\widetilde{\mathcal{O}}_1$ and $\widetilde{\mathcal{W}}_1$ that

$$\Pr_{\textsc{ExtExp}^{\widetilde{\mathcal{O}}_0,\widetilde{\mathcal{W}}_0,\mathcal{O}}_{\ell,(p,q),b}}[\mathrm{E}] \approx \Pr_{\textsc{ExtExp}^{\widetilde{\mathcal{O}}_1,\widetilde{\mathcal{W}}_1,\mathcal{O}}_{\ell,(p,q),b}}[\mathrm{E}] . \tag{2}$$

Combining (1) and (2) we get that

$$\Pr_{\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),0,b}}[\mathrm{E}] \approx \Pr_{\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),1,b}}[\mathrm{E}] . \tag{3}$$

From (2) we have by definition that $\Pr[\mathrm{E}_{0,b,c}] \approx \Pr[\mathrm{E}_{1,b,c}]$, as desired. □

Let event $\mathsf{NSF}_{a,b,c}$ clearly does not depend on c. It can be seen by inspection that it can be computed whether NSF occurred before c is used. We therefore denote the event simply by $\mathsf{NSF}_{a,b}$

Lemma 8 (b-switch)**.** $\mathsf{NSF}_{a,b} \Longrightarrow \mathcal{O}^{\text{POST}}_b = \mathcal{O}^{\text{POST}}_0$.

Proof. The conclusion is trivially true when $b = 0$, so assume $b = 1$. Assume $\mathsf{NSF}_{a,1}$. By construction

$$\mathcal{O}^{\text{POST}}_0(\mathfrak{q}) = \begin{cases} \widetilde{\mathcal{O}}(\mathfrak{q}) & \text{if } \mathfrak{q} \in \mathcal{P}^{\text{STR}}_{\downarrow} \\ \mathcal{O}(\mathfrak{q}) & \text{otherwise} , \end{cases}$$

$$\mathcal{O}^{\text{POST}}_1(\mathfrak{q}) = \begin{cases} \widetilde{\mathcal{O}}(\mathfrak{q}) & \text{if } \mathsf{used}_{p+q}(\mathfrak{q}) = \top \wedge \mathfrak{q} \notin \mathcal{P}^{\text{PRE}}_{\downarrow} \\ \mathcal{O}(\mathfrak{q}) & \text{otherwise} . \end{cases}$$

It is therefore enough to prove that $\text{¶} \in \mathcal{P}_{\downarrow}^{\text{STR}}$ if and only if $\mathsf{used}_{p+q}(\text{¶}) = \top \wedge \text{¶} \notin \mathcal{P}_{\downarrow}^{\text{PRE}}$. When $\mathsf{NSF}_{a,1}$ then for all queries ¶ to the oracle $\mathcal{O}_1^{\text{POST}}(\text{¶})$ it holds that

$$\text{¶} \in \mathcal{P}_{\cup}^{(1,p+q)} \iff \mathsf{used}_{p+q}(\text{¶}) = \top \ .$$

Since $\mathcal{P}_{\downarrow}^{\text{PRE}} = \mathcal{P}_{\cup}^{(1,p)}$ and $\mathcal{P}_{\downarrow}^{\text{STR}} = \mathcal{P}_{\cup}^{(p,p+q-1)} \setminus \mathcal{P}_{\cup}^{(1,p-1)}$ we have that

$$\mathcal{P}_{\cup}^{(1,p+q-1)} = \mathcal{P}_{\downarrow}^{\text{PRE}} \dot\cup \mathcal{P}_{\downarrow}^{\text{STR}} \ .$$

This implies that

$$\text{¶} \in \mathcal{P}_{\downarrow}^{\text{PRE}} \dot\cup \mathcal{P}_{\downarrow}^{\text{STR}} \iff \mathsf{used}_{p+q}(\text{¶}) = \top$$

which in turn implies that

$$\text{¶} \in \mathcal{P}_{\downarrow}^{\text{STR}} \iff \left(\text{¶} \in \mathcal{P}_{\downarrow}^{\text{PRE}} \dot\cup \mathcal{P}_{\downarrow}^{\text{STR}}\right) \wedge \text{¶} \notin \mathcal{P}_{\downarrow}^{\text{PRE}} \iff \left(\mathsf{used}_{p+q}(\text{¶}) = \top\right) \wedge \text{¶} \notin \mathcal{P}_{\downarrow}^{\text{PRE}} \ ,$$

as desired. □

We now define an event NSFQFS_0 which does not depend on b. Specifically we define $\mathsf{NSFQFS}_{0,b,c}$ for all b and c and then show that when defined over the same probability space then $\mathsf{NSFQFS}_{0,0,0} = \mathsf{NSFQFS}_{0,0,1} = \mathsf{NSFQFS}_{0,1,0} = \mathsf{NSFQFS}_{0,1,1}$. We can therefore consider them a single event NSFQFS_0. Let

$$\mathsf{NSFQFS}_{0,b,c} \iff \mathsf{NSF}_{0,b} \wedge \mathsf{QFS}_{0,b,c}$$

and let

$$\mathsf{NSFQFS}_0 = \mathsf{NSFQFS}_{0,0,0} \ .$$

Lemma 9. *For all b and c it holds that $\mathsf{NSFQFS}_0 \iff \mathsf{NSFQFS}_{0,b,c}$. Furthermore, $\mathsf{QFS}_{0,0,0} \iff \mathsf{QFS}_{0,0,1}$.*

Proof. Note that NSF is the event that there is no structure failure, which is true from the beginning of the execution and once it becomes false it remains false. Likewise, QFS is the assumption that the verifier does not make a query which was also made in the stretch. This too is true from the beginning and once false remains false. Therefore $\mathsf{NSFQFS}_{0,b,c}$ has this monotonicity property too. It is then sufficient to prove that as long as $\mathsf{NSFQFS}_{0,0,0}$ is true in an execution it holds that if $\mathsf{NSFQFS}_{0,0,0}$ becomes violated then when executed over the same randomness space, all the events $\mathsf{NSFQFS}_{0,b,c}$ would become violated too.

We have by definition that

$$\mathsf{NSFQFS}_0 \iff \mathsf{NSFQFS}_{0,0,0} \iff \mathsf{NSF}_{0,0} \wedge \mathsf{QFS}_{0,0,0} \ .$$

It is therefore enough to prove that while $\mathsf{NSFQFS}_{0,b,c}$ holds true, then

$$\mathsf{NSF}_{0,b} \iff \mathsf{NSF}_{0,0} \tag{4}$$
$$\mathsf{QFS}_{0,b,c} \iff \mathsf{QFS}_{0,0,0} \ . \tag{5}$$

So assume that we are at a point in the execution where $\mathsf{NSFQFS}_{0,b,c}$ is true. This gives us $\mathsf{QFS}_{0,b,c}$ and $\mathsf{NSF}_{0,b}$. The only place where b is used is to select between $\mathcal{O}_0^{\text{POST}}$ and $\mathcal{O}_1^{\text{POST}}$ From Lemma 8 we have that when $\mathsf{NSF}_{0,b}$ then $\mathcal{O}_1^{\text{POST}} = \mathcal{O}_0^{\text{POST}}$. Therefore, while $\mathsf{NSFQFS}_{0,b,c}$ the events $\mathsf{QFS}_{0,b,c}$ and $\mathsf{NSF}_{0,b}$ are independent of b. This gives us $\mathsf{QFS}_{0,b,c} \Leftrightarrow \mathsf{QFS}_{0,0,c}$ and $\mathsf{NSF}_{0,b} \Leftrightarrow \mathsf{NSF}_{0,0}$. This already proves (4). Furthermore, to prove (5) it is now sufficient to prove that $\mathsf{QFS}_{0,0,c} \Leftrightarrow \mathsf{QFS}_{0,0,0}$. This is trivially true when $c = 0$, so it is sufficient to prove that when $\mathsf{NSFQFS}_{0,b,c}$ then

$$\mathsf{QFS}_{0,0,1} \iff \mathsf{QFS}_{0,0,0} \ . \tag{6}$$

Note that c is only used to select the oracle $\mathcal{O}_c$ used by $\mathbb{V}$. By construction $\mathcal{O}_1 = \mathcal{O}_b^{\text{POST}}$, and when $\mathsf{NSF}_{0,b}$ then $\mathcal{O}_1^{\text{POST}} = \mathcal{O}_0^{\text{POST}}$ by Lemma 8, so

$$J_1 = \mathbb{V}^{\mathcal{O}_0^{\text{POST}}}(\mathsf{H}, M_\ell, \pi_\ell) \ .$$

By construction $\mathcal{O}_0 = \mathcal{O}$, so

$$J_0 = \mathbb{V}^{\mathcal{O}}(\mathsf{H}, M_\ell, \pi_\ell) \ .$$

Then note that $\mathcal{O}$ and $\mathcal{O}_0^{\text{POST}} = [\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}_0, \mathcal{O}]$ are identical when not queried on $\mathfrak{q} \in \mathcal{P}_\downarrow^{\text{STR}}$. Specifically, as long as $\mathbb{V}^{\mathcal{O}}$ and $\mathbb{V}^{[\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}_0, \mathcal{O}]}$ did not query on $\mathfrak{q}$ from $\mathcal{P}_\downarrow^{\text{STR}}$ they both get the reply $\mathcal{O}(\mathfrak{q})$ and hence run identically. During this period both $\mathsf{QFS}_{0,0,0}$ and $\mathsf{QFS}_{0,0,1}$ are true. At the first point where they query on $\mathfrak{q}$ from $\mathcal{P}_\downarrow^{\text{STR}}$ the events $\overline{\mathsf{QFS}_{0,0,0}}$ and $\overline{\mathsf{QFS}_{0,0,1}}$ happened. Therefore, at all times $\mathsf{QFS}_{0,0,0} \Leftrightarrow \mathsf{QFS}_{0,0,1}$. This proves (6) and concludes the proof. □

Lemma 10. $\Pr[\mathsf{VER}_{0,1,0}] \geq \Pr[\mathsf{VER}_{0,0,1}] - \Pr[\overline{\mathsf{QFS}_{0,0,0}}] - \Pr[\overline{\mathsf{NSF}_{0,0}}]$.

Proof. In the proof of Lemma 9 we showed that when NSFQFS_0 then $\mathcal{O}_0 = \mathcal{O}_0^{\text{POST}}$ and $\mathcal{O}_1 = \mathcal{O}$ and

$$\mathbb{V}^{\mathcal{O}_0^{\text{POST}}}(\mathsf{H}, M_\ell, \pi_\ell) = \mathbb{V}^{\mathcal{O}}(\mathsf{H}, M_\ell, \pi_\ell)$$

which gives us that $\mathsf{VER}_{0,b,1} \Leftrightarrow \mathsf{VER}_{0,b,0}$. For $b = 0$ this gives us that

$$\Pr[\mathsf{VER}_{0,0,0} \,|\, \mathsf{NSFQFS}_0] = \Pr[\mathsf{VER}_{0,0,1} \,|\, \mathsf{NSFQFS}_0] \ .$$

Furthermore, since $\mathcal{O}_b^{\text{POST}} = \mathcal{O}_0^{\text{POST}}$ until $\overline{\mathsf{NSF}_{0,b}}$ happens we have that

$$\Pr[\mathsf{VER}_{0,1,0} \,|\, \mathsf{NSFQFS}_0] = \Pr[\mathsf{VER}_{0,0,0} \,|\, \mathsf{NSFQFS}_0] \ .$$

Combining these we get that

$$\Pr[\mathsf{VER}_{0,1,0} \,|\, \mathsf{NSFQFS}_0] = \Pr[\mathsf{VER}_{0,0,1} \,|\, \mathsf{NSFQFS}_0] \ .$$

Using elementary probability theory this gives us that

$$|\Pr[\mathsf{VER}_{0,1,0}] - \Pr[\mathsf{VER}_{0,0,1}]| \leq \Pr[\overline{\mathsf{NSFQFS}_0}] \ .$$

Using that $\overline{\mathsf{NSFQFS}_0} = \overline{\mathsf{NSF}_{0,0}} \cup \overline{\mathsf{QFS}_{0,0,0}}$ and a union bound it follows that

$$\Pr[\mathsf{VER}_{0,1,0}] \geq \Pr[\mathsf{VER}_{0,0,1}] - \Pr[\overline{\mathsf{QFS}_{0,0,0}}] - \Pr[\overline{\mathsf{NSF}_{0,0}}] \ .$$

□

Lemma 11. *For all $q \in \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$ and $p_{\text{QFS}} \in \mathsf{poly}(\lambda)$ we can set $\ell \in \mathsf{poly}(\lambda)$ and a position p such that* $\Pr[\mathsf{QFS}_{0,0,1}] > 1 - p_{\text{QFS}}$.

Proof. When $a = 0$ then $\widetilde{\mathcal{O}}_a$ is a uniformly random oracle. It is easy to see that this means that when $b = 0$ then

$$\mathcal{O}_b^{\text{POST}} = \mathcal{O}_0^{\text{POST}} = [\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}_a, \mathcal{O}]$$

is also a uniformly random oracle. Furthermore, the proof

$$(\mathsf{H}, M_\ell, \pi_\ell) \leftarrow \mathcal{A}^{\mathcal{O}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}}_a,\widetilde{\mathcal{W}}_a,b}$$

is computed correctly inside the adversary using this oracle $\mathcal{O}_b^{\text{POST}}$. Note that we look at the case $c = 1$ and that c is used to select the oracle for $\mathbb{V}$. Note then that by construction $\mathcal{O}_1 = \mathcal{O}_b^{\text{POST}}$, so

$$J_1 = \mathbb{V}^{\mathcal{O}_1}(\mathsf{H}, M_\ell, \pi_\ell) = \mathbb{V}^{[\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}_a, \mathcal{O}]}(\mathsf{H}, M_\ell, \pi_\ell)$$

is also computed using this same uniformly random oracle.

It then follows from Lemma 1 that for all polynomials q we can find a polynomial $\ell(\lambda)$ such that there exists p such that (p, q) is a QRS except with probability p_{QFS}. □

We now prove the version of the warm up lemma where we see that extraction still works when a fake witness is used during the stretch, at least if we assume that the proof accepts (which is part of the definition of XTF).

Lemma 12. $\Pr[\mathsf{XTF}_{1,b}] = 1 - \mathsf{negl}(\lambda)$.

Proof. The only way that $\mathsf{XTF}_{1,b} = \bot$ is that $J_0 = \top$ and $\vec{v} \neq \vec{w}$. So it is enough to prove that when $a = 1$, then it happens with negligible probability that $J_0 = \top$ and $\vec{v} \neq \vec{w}$. Since $\mathcal{O}_0 = \mathcal{O}$ it follows from $J_0 = \top$ that

$$\mathbb{V}^{\mathcal{O}}(\mathsf{H}, M_\ell, \pi_\ell) = \top \ .$$

When $a = 1$ then by construction of Step 3 we have that

$$\mathcal{A}^{(\cdot)} = \mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}}_1,\widetilde{\mathcal{W}}_1,b} \ .$$

Since the hardcoded pseudo-random oracles $\widetilde{\mathcal{O}}_1$ and $\widetilde{\mathcal{W}}_1$ have a poly-sized description and can be computed in poly-time, $\mathcal{A}^{(\cdot)}$ is a legal adversary for the knowledge extraction game. Therefore except with negligible probability it holds that when

$$(\mathsf{H}, M_\ell, \pi_\ell) \leftarrow \mathcal{A}^{\mathcal{O}}$$

(as is the case in the experiement) and $\mathbb{V}^{\mathcal{O}}(\mathsf{H}, M_\ell, \pi_\ell) = \top$ (which holds by assumption), then

$$\vec{v} = \mathbb{E}^{\mathcal{A}^{(\cdot)},\mathcal{O}}$$

is a valid witness, i.e.,

$$M_\ell = \mathsf{H}^\ell(M_0, \vec{v}) \ .$$

By construction

$$M_\ell = \mathsf{H}^\ell(M_0, \vec{w}) \ ,$$

which implies that $\vec{v} = \vec{w}$ except with negligible probability by collision resistance. □

Lemma 13. $\mathsf{VER}_{a,b,0} \wedge \mathsf{XTF}_{a,b} \Longrightarrow \mathsf{XTS}_{a,b}$.

Proof. If $\mathsf{VER}_{a,b,0}$ then $J_{a,b,0} = \top$. If $\mathsf{XTF}_{a,b}$ then $J_{a,b,0} = \bot$ or $\vec{v} = \vec{w}$. So, we have that $\vec{v} = \vec{w}$ which implies $\vec{v}^{\text{STR}} = \vec{w}^{\text{STR}}$ which implies $\mathsf{XTS}_{a,b}$. □

We now show that the proof indeed verifies when computed using a fake oracle during the stretch, at least if the same oracle is used by the verifier.

Lemma 14. $\Pr[\mathsf{VER}_{0,0,1}] = 1 - \mathsf{negl}(\lambda)$.

Proof. When $a = 0$ then $\widetilde{\mathcal{O}}_a$ is a uniformly random oracle. It is easy to see that this means that when $b = 0$ then

$$\mathcal{O}_b^{\text{POST}} = \mathcal{O}_0^{\text{POST}} = [\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}_a, \mathcal{O}]$$

is also a uniformly random oracle. Furthermore, the proof

$$(\mathsf{H}, M_\ell, \pi_\ell) \leftarrow \mathcal{A}^{\mathcal{O}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}}_a,\widetilde{\mathcal{W}}_a,b}$$

is computed honestly inside the adversary honestly using this oracle $\mathcal{O}_b^{\text{POST}}$. Note then that by construction $\mathcal{O}_1 = \mathcal{O}_b^{\text{POST}}$, so

$$J_1 = \mathbb{V}^{\mathcal{O}_1}(\mathsf{H}, M_\ell, \pi_\ell) = \mathbb{V}^{[\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}_a, \mathcal{O}]}(\mathsf{H}, M_\ell, \pi_\ell)$$

is also computed using this same uniformly random oracle. By completeness, a proof honestly generated and verified using the same uniformly random oracle will verify. Therefore $J_1 = \top$ except with negligible probability. Then use that $\mathsf{VER}_{0,0,1} = J_1$. □

Lemma 15. *There exists a polynomial* $p_{\text{STRUC}} = 1/\lambda^{O(1)}$ *such that* $\Pr[\mathsf{NSF}_{0,0}] > p_{\text{STRUC}}$.

Proof. We have assume that the proof system is $1/\lambda^{O(1)}$-SOQ. This essentially means that if a proof is computed honestly using a uniformly random oracle $\mathcal{O}$, then with probability $p_{\text{STRUC}} = 1/\lambda^{O(1)}$ it happens that $\mathsf{used}_{p+q}(¶) = \mathcal{O}'(¶)$ for all ¶ that an adversary can compute. When $a = 0$ and $b = 0$ then the proof computed by $\mathcal{A}$ is computed honestly using the uniformly random oracle

$$\mathcal{O}' = [\mathcal{P}_\downarrow^{\text{PRE}} \mapsto \mathcal{O}, \mathcal{P}_\downarrow^{\text{STR}} \mapsto \widetilde{\mathcal{O}}, \mathcal{O}] \ .$$

If $\overline{\mathsf{NSF}_{0,0}}$ occurs, we can by construction easily compute ¶ such that $\mathsf{used}_{p+q}(¶) = \top$ and $¶ \notin \mathcal{P}_\cup^{(1,p+q)}$ or $\mathsf{used}_{p+q}(¶) = \bot$ and $¶ \in \mathcal{P}_\cup^{(1,p+q)}$. This allows allows us to win the SOQ game. □

We now put the above lemmas together to show that

Lemma 16. *For all polynomial $q \in \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$ it is possible to set ℓ to a polynomial and set p such that*

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

Proof. We have from Lemma 13 and a union bound that

$$\Pr[\mathsf{XTS}_{0,1}] \geq 1 - \Pr[\overline{\mathsf{VER}_{0,1,0}}] - \Pr[\overline{\mathsf{XTF}_{0,1}}] \ .$$

By Lemma 12 with $b = 1$ and Lemma 7 to switch from $a = 1$ to $a = 0$ we have that

$$\Pr[\overline{\mathsf{XTF}_{0,1}}] < \mathsf{negl}(\lambda) \ .$$

By Lemma 14 and Lemma 7

$$\Pr[\mathsf{VER}_{0,0,1}] \geq 1 - \mathsf{negl}(\lambda) \ .$$

By Lemma 14 we have that

$$\Pr[\mathsf{VER}_{0,1,0}] \geq \Pr[\mathsf{VER}_{0,0,1}] - \Pr[\overline{\mathsf{QFS}_{0,0,0}}] - \Pr[\overline{\mathsf{NSF}_{0,0}}] \ .$$

Combining the last two inequalities we get that

$$\Pr[\overline{\mathsf{VER}_{0,1,0}}] \leq \mathsf{negl}(\lambda) + \Pr[\overline{\mathsf{QFS}_{0,0,0}}] + \Pr[\overline{\mathsf{NSF}_{0,0}}] \ .$$

From Lemma 15 we have that there exists a polynomial $p_{\text{STRUC}} = {}^1\!/\!_{\lambda^{O(1)}}$ such that

$$\Pr[\overline{\mathsf{NSF}_{0,0}}] < 1 - p_{\text{STRUC}} \ .$$

By Lemma 11 we have that for all polynomials q and p_{QFS} we can set ℓ and p such that

$$\Pr[\overline{\mathsf{QFS}_{0,0,1}}] < p_{\text{QFS}} \ .$$

Using Lemma 9 we get that

$$\Pr[\overline{\mathsf{QFS}_{0,0,0}}] < p_{\text{QFS}} \ .$$

If we set

$$p_{\text{QFS}} = p_{\text{STRUC}}/2$$

then we have that

$$\Pr[\overline{\mathsf{VER}_{0,1,0}}] \leq \mathsf{negl}(\lambda) + 1 - p_{\text{STRUC}}/2 \ .$$

Putting all the above together we have that

$$\Pr[\mathsf{XTS}_{0,1}] \geq p_{\text{STRUC}}/2 - \mathsf{negl}(\lambda) \ \in {}^1\!/\!_{\lambda^{O(1)}}.$$

□

Lemma 17. *There exists a polynomial $q \in \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$ such that it is not possible to set ℓ to a polynomial and set p such that*

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

Proof. The intuition behind the proof is that $\vec{w}^{\text{STR}}$ is long and uniformly random and used_{p+q} is short, and therefore used_{p+q} cannot encode all of $\vec{w}^{\text{STR}}$. However, since there are no oracle queries to $\mathcal{O}$ during the stretch, the only information that the extractor can get about $\vec{w}^{\text{STR}}$ is used_{p+q}. Therefore extraction must fail. We now formalise the proof.

We first formalise the fact that we can encapsulate the use of $\widetilde{\mathcal{W}}$ such that the only information passed on is used_{p+q}.

The oracle $\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}(\cdot)$ has the following values hard-coded. A hash function H, the number of steps q, a random tape $\rho^{\text{STR}} = (\rho_p, \ldots, \rho_{p+q-1})$ long enough to provide $\mathbb{P}$ with randomness q times, a uniformly random oracle $\widetilde{\mathcal{O}} : \{0,1\}^\lambda \to \{0,1\}^\lambda$, and a uniformly random oracle $\widetilde{\mathcal{W}} : \{0,1\}^* \to (\{0,1\}^\lambda)^q$.

1. The input to the oracle is of the form (T, M_{p-1}, π_{p-1}) (else ignore).
2. Parse T as $((\mathfrak{q}_1, y_1), \ldots, (\mathfrak{q}_h, y_h))$.
3. Let $\mathcal{O}^{\text{STR}}$ be the oracle which on $\mathfrak{q}$ returns y_i if $(\mathfrak{q}, y_i) \in T$ and otherwise returns $\widetilde{\mathcal{O}}(\mathfrak{q})$.
4. Define the *stretch witnesses* $\vec{w}^{\text{STR}} = (w_p, \ldots, w_{p+q-1}) = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})$.
5. For $i = p, \ldots, p+q-1$ compute $M_i = \mathsf{H}(M_{i-1}, w_i)$ and
$$\pi_i = \mathbb{P}^{\mathcal{O}^{\text{STR}}}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i) .$$
6. Let $\mathsf{used}_{p+q} = \mathsf{used}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i)$
7. Return $(M_{p+q-1}, \pi_{p+q-1}, \mathsf{used}_{p+q})$.

We then describe a class of pruned adversaries. We take $\mathcal{A}^{\mathcal{O}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}},\widetilde{\mathcal{W}},b}$ as run by $\textsc{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$ and specialise to the case $a = 0$ and $b = 1$. At the same time we factor out the production of the stretch witnesses using the above oracle. The oracle will be accessed in a blackbox manner and will be called $\mathcal{O}_2$.

The adversary $\mathcal{A}^{\mathcal{O},\mathcal{O}_2}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}}$ 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* $\vec{w}^{\text{POST}} = (w_{p+q},\ldots,w_\ell)$, random tape ρ^{PRE}, ρ^{POST} long enough to provide $\mathbb{P}$ with randomness p respectively $\ell - p - q$ times. 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.
5. For the i'th query $\mathfrak{q}_i \in \mathcal{P}^{\text{PRE}}_{\downarrow}$ to $\mathcal{O}$ let $y_i = \mathcal{O}(\mathfrak{q}_i)$ be the reply given by $\mathcal{O}$ to the query $\mathfrak{q}_i$ above. Define the *query tag* $T = ((\mathfrak{q}_1, y_1),\ldots,(\mathfrak{q}_h, y_h))$.
6. Let $(M_{p+q-1}, \pi_{p+q-1}, \mathsf{used}_{p+q}) = \mathcal{O}_2(T, M_{p-1}, \pi_{p-1})$.
7. Let $\mathcal{O}^{\text{POST}}$ be the following oracle.
$$\mathcal{O}^{\text{POST}}(\mathfrak{q}) = \begin{cases} \widetilde{\mathcal{O}}(\mathfrak{q}) & \text{if } \mathsf{used}_{p+q}(\mathfrak{q}) = \top \wedge \mathfrak{q} \notin \mathcal{P}^{\text{PRE}}_{\downarrow} \\ \mathcal{O}(\mathfrak{q}) & \text{otherwise} \end{cases}$$
8. For $i = p+q,\ldots,\ell$ compute $M_i = \mathsf{H}(M_{i-1}, w_i)$ and
$$\pi_i = \mathbb{P}^{\mathcal{O}^{\text{POST}}_b}(\mathsf{H}, M_{i-1}, \pi_{i-1}, w_i; \rho_i) \ .$$
9. Output $(\mathsf{H}, M_\ell, \pi_\ell)$.

Consider now the following pruned experiment where we specialise to $a = 0$ and $b = 1$.

1. $(\mathsf{H}, M_\ell, \pi_\ell) \leftarrow \mathcal{A}^{\mathcal{O},\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}}$.
2. Let $\vec{v} = \mathbb{E}^{\mathcal{A}^{(\cdot),\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}},\mathcal{O}}$.
3. Let $\vec{v}^{\text{PRE}} \| \vec{v}^{\text{STR}} \| \vec{v}^{\text{POST}} = \vec{v}$, where $|\vec{v}^{\text{PRE}}| = p - 1$ and $|\vec{v}^{\text{STR}}| = q$.
4. Let XTS be the *stretch extraction* event that $\vec{v}^{\text{STR}} = \vec{w}^{\text{STR}}$.

Recall that when $b = 1$ then $\text{ExtExp}^{\mathcal{O}}_{\ell,(p,q),a,b}$ uses the oracle $\mathcal{O}^{\text{POST}}_1$ defined via used_{p+q} as does the pruned oracle. In general it is easy to verify that by construction

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

and

$$\mathcal{A}^{(\cdot)} = \mathcal{A}^{(\cdot)}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho,\widetilde{\mathcal{O}}_0,\widetilde{\mathcal{W}}_0,1}$$

have the same blackbox behaviour. Therefore

$$\Pr[\mathsf{XTS}] = \Pr[\mathsf{XTS}_{0,1}]$$

by definition of XTS and $\mathsf{XTS}_{0,1}$ and the fact that extraction is blackbox. It is therefore sufficient to prove that XTS happens with negligible probability.

Let $\mathcal{O}_2 = \mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}$ below. We can construct an algorithm $\widetilde{\mathbb{E}}$ with

$$\mathsf{H}, \ell, (p,q), \vec{w}^{\text{PRE}}, \vec{w}^{\text{POST}}, \rho^{\text{PRE}}, \rho^{\text{POST}}, \widetilde{\mathcal{O}}$$

hardcoded such that

$$\widetilde{\mathbb{E}}^{\mathcal{O},\mathcal{O}_2}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}} = \mathbb{E}^{\mathcal{A}^{(\cdot),\mathcal{O}_2}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}},\mathcal{O}} \ .$$

We can simply let $\widetilde{\mathbb{E}}^{\mathcal{O},\mathcal{O}_2}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}}$ construct

$$\mathcal{A}^{(\cdot)} = \mathcal{A}^{(\cdot),\mathcal{O}_2}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}}$$

using its own oracle access to $\mathcal{O}_2$. Then it outputs whatever is returned by $\mathbb{E}^{\mathcal{A}^{(\cdot)},\mathcal{O}}$. Note that $\vec{w}^{\text{STR}} = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})$. It follows that

$$\Pr\left[\widetilde{\mathbb{E}}^{\mathcal{O},\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}}_{\mathsf{H},\ell,(p,q),\vec{w}^{\text{PRE}},\vec{w}^{\text{POST}},\rho^{\text{PRE}},\rho^{\text{POST}},\widetilde{\mathcal{O}}} = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})\right] = \Pr[\mathsf{XTS}] \ .$$

Notice now that $\widetilde{\mathbb{E}}$ does not have oracle access to $\widetilde{\mathcal{W}}$. The only way for $\widetilde{\mathbb{E}}$ to gain information on $\widetilde{\mathcal{W}}$ is by its access to $\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}$. Note that when $\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}$ is queried on (T, M_{p-1}, π_{p-1}) then $\vec{w}^{\text{STR}} = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})$. This means that the witnesses used on different inputs to the oracle are uniformly random and independent. Now note that used_{p+q} is deterministically given once (T, M_{p-1}, π_{p-1}) is given and $\vec{w}^{\text{STR}}$ are given. So each query of $\mathcal{O}_{\mathsf{H},q,\rho^{\text{STR}},\widetilde{\mathcal{O}},\widetilde{\mathcal{W}}}$ on (T, M_{p-1}, π_{p-1}) outputs the same $(M_{p+q-1}, \pi_{p+q-1}, \mathsf{used}_{p+q})$. Therefore the only information $\widetilde{\mathbb{E}}$ has on $\vec{w}^{\text{STR}} = \widetilde{\mathcal{W}}(T, M_{p-1}, \pi_{p-1})$ is the single value $(M_{p+q-1}, \pi_{p+q-1}, \mathsf{used}_{p+q})$.

Now notice that $\mathsf{used}_{p+q} = \mathsf{used}(\mathsf{H}, M_{p+q-1}, \pi_{p+q-1}, w_{p+1}, \rho_{p+q})$, so we can compute used_{p+q} from $\mathsf{used}, \mathsf{H}, M_{p+q-1}, \pi_{p+q-1}, w_{p+q}, \rho_{p+q}$. The value $\vec{w}^{\text{STR}}$ was pick uniformly at random and independent from used, H, ρ_{p+q} and w_{p+q}. So at most the values M_{p+q-1} and π_{p+q-1} contain information on $\vec{w}^{\text{STR}}$. We have that $|M_{p+q-1}| = \mathsf{poly}(|\mathcal{R}|, \lambda)$ and $|\pi_{p+q-1}| = \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell)$, so

$$|(M_{p+q-1}, \pi_{p+q-1})| = \mathsf{poly}(|\mathcal{R}|, \lambda, \log \ell) \ .$$

We have that

$$|\vec{w}^{\text{STR}}| = q\lambda \ .$$

We can therefore by Lemma 1 set q such that

$$|\vec{w}^{\text{STR}}| > |(M_{p+q-1}, \pi_{p+q-1})| + \lambda \ .$$

From this it is easy to show that the probability of guessing the uniformly random $\vec{w}^{\text{STR}}$ from (M_{p+q-1}, π_{p+q-1}) is negligible, even given unbounded running time. This concludes the proof. ☐

Theorem 2. *If there exist collision resistant hash functions then there does not exist non-deterministic IVC for the random oracle model (Definition 3) with blackbox knowledge soundness (Definition 6) which is $1/\lambda^{O(1)}$-SOQ (Definition 16).*

Proof. Under the premises of the theorem we can prove both Lemma 17 and Lemma 16, and these two lemmas are in contradiction. ☐

References

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

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

[BCL+20] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. Cryptology ePrint Archive, Report 2020/1618, 2020. `https://eprint.iacr.org/2020/1618`.

[BCMS20a] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. Cryptology ePrint Archive, Report 2020/499, 2020. `https://eprint.iacr.org/2020/499`.

[BCMS20b] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive proof composition from accumulation schemes. pages 1–18, 2020.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. pages 31–60, 2016.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. pages 276–294, 2014.

[CL20] Alessandro Chiesa and Siqi Liu. On the impossibility of probabilistic proofs in relativized worlds. pages 57:1–57:30, 2020.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. pages 769–793, 2020.

[CT10] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. pages 310–331, 2010.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 99108, New York, NY, USA, 2011. Association for Computing Machinery.

[Mic94] S. Micali. Cs proofs. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 436–453, Los Alamitos, CA, USA, nov 1994. IEEE Computer Society.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. pages 1–18, 2008.