When Can We Incrementally Prove Computations of Arbitrary Depth?*

Matteo Campanelli[1], Dario Fiore[2], and Mahak Pancholi[2]

[1] Offchain Labs
[2] IMDEA Software Institute

Abstract. Incrementally Verifiable Computation (IVC) allows one to prove the correctness of a computation of potentially unbounded length (or, *depth*) in an incremental way, while a computationally weak client can efficiently check its correctness in time sublinear in the computation's length. IVC are of practical relevance; yet, most existing IVC schemes are only provably secure for constant-depth computations. Arguing their security for computations of polynomial depth relies on heuristic assumptions, raising both theoretical and practical concerns. More generally, it remains unclear whether these schemes are genuinely insecure at superconstant depths or whether our current proof techniques are simply insufficient.

In this work, we delve into the security foundations of incremental proof systems, while at the same time looking for new approaches to prove security at superconstant depths. To this end, we study the relation between the *depth* of the target computation and IVC *security* as a question in its own right. Specifically, we ask:

- **Can we prove an IVC secure at depth $d = \omega(1)$ if it satisfies a "weak" security property at some "larger" depth D?** We uncover a surprising connection between *infinitely-often* soundness (guaranteed to hold only for *some* infinite set of parameters) and standard soundness in IVC: a scheme that is infinitely-often sound at D depth achieves *standard* soundness at some smaller depth $d = o(D)$.
- **How does a proof system's security degrade with depth?** More precisely, if a scheme loses negligible soundness beyond (e.g.) $O(1)$ depth—the current provable frontier of most practical schemes—can we at least achieve *noticeable* (but arbitrarily low) soundness for *some* superconstant depth? We show the answer is negative.
- **Depth boosting: If there exists an IVC scheme secure at depth d, does there exist one secure at a greater depth?** We show a simple and *black-box* boosting technique: given an IVC secure at depth d, we can construct one secure at depth $D = d^\rho$ for any function ρ such that d^ρ is polynomially bounded (ρ can even be superconstant). This allows us to systematically amplify security from modest depths to much greater ones (e.g., constant to polynomial depth, with only a logarithmic overhead).

Our results uncover a nuanced landscape of depth, efficiency, and security in IVC, and new strategies to prove security beyond $O(1)$ depth. They also apply to non-deterministic computations and other soundness notions, including **incremental functional commitments** (IFC), a streaming variant of functional commitments that we introduce.

* **A note on an earlier version of this work:** this preprint used to contain additional results related to the AGM—these were extended and moved to a separate write-up, namely [CFP25]—as well as different statements for current Theorems 8 and 9. This version also includes several new results on the relation between depth and security in IVC.

Table of Contents

When Can We Incrementally Prove Computations of Arbitrary Depth? 1
Matteo Campanelli, Dario Fiore, and Mahak Pancholi
1 Introduction 3
1.1 Our Results 4
1.2 Further Discussion and Other Future Directions 10
1.3 A Quick Overview of Our Techniques 10
1.4 Related Work 11
2 Preliminaries 14
2.1 Background on Deterministic IVC 14
3 Soundness From Infinitely-Often Soundness in IVC 16
4 On the Impossibility of Graceful Security Degradation 18
5 General Depth Boosting Results for IVC 21
5.1 A simple depth boosting approach for λ^{ε} depth bounds 21
5.2 A fully general depth boosting approach 22
6 Incremental Functional Commitments 26
A A Useful Fact On Sequences of Functions 33
B IVC for Non-Deterministic Computations 33
C Background on SNARKs 35
D Formal Treatment of Incremental Functional Commitments 36
D.1 Functions that can be described by a single building block 36
D.2 Incremental functional commitments 37
D.3 Our Construction 39
D.4 Security 41
E On the Induction-like Proof for IFC Security in the Main Text 45
F Extension of Main Results to Other Soundness Notions 46
F.1 Theorem statements 47
F.2 Why the proofs carry over 48

1 Introduction

Incrementally Verifiable Computation (IVC) [Val08] enables proving the correctness of computations of potentially *unbounded* length (or depth) in an *incremental* fashion. A verifier can check these proofs efficiently—in time sublinear in the length of the computation—making IVC a powerful tool for long-running or continuous processes. IVC can be seen as a class of succinct non-interactive arguments (SNARKs) specialized for incremental computations. Starting from an initial input z_0, the computation evolves through ℓ steps as $z_i = F(z_{i-1}, \omega_i)$ for $i = 1$ to ℓ, where each ω_i represents a non-deterministic input and F is the computation's step. The prover can generate proofs step by step, requiring only memory proportional to the computation's step and without needing to know "future" inputs. These properties make IVC a versatile primitive with several real-world applications, such as scalability in blockchains (e.g., Mina [BMRS20], zkSync [Mat23], Starkware [Sta18], among others), distributed and streaming computation [Mar23, LXZ+24], verifiable machine learning [PWZ+25], and more.

Why depth matters. A particularly appealing feature of IVC is that its functionality, in its strongest form, allows one to keep proving computation's steps *indefinitely*. While any real-world prover will always run in polynomial time, the number of steps (i.e., the polynomial) is neither fixed nor known in advance. This is crucial in applications like zkRollups where one uses IVC to prove the correctness of each new block posted on-chain—a task that goes on with the lifetime of the system and may potentially never end. Another example is the setting where we prove cryptographic primitives that are built from the iteration of a certain building block d times. In these settings, it is typical for security of the primitive to be guaranteed only when d is a superconstant function in λ. This includes *sequential functions*, a general class that includes Verifiable Delay Functions (see [BBBF18]). Another ubiquitous functionality that requires a superconstant number of steps is the Merkle–Damgård construction for hash functions, where one iteratively applies a compression function to process input blocks of arbitrary length [Mer89, Dam89].

The challenge of soundness beyond constant depth. Despite the advantages for arbitrary depth outlined above, most IVC constructions are provably secure only for computations of *constant* depth [BCCT13, BCTV14, BCMS20, BCL+21, KST22]. This limitation is most evident in the canonical IVC construction based on recursive composition of SNARKs[3]: at each step i, one proves that $z_i = F(z_{i-1}, \omega_i)$ and that a valid SNARK proof for z_{i-1} exists. In a nutshell, the reason behind the constant-depth barrier in provable security lies in the following. Current proof strategies rely on invoking the SNARK extractor in a nested fashion as many times as the depth of the computation, causing the reduction's runtime to grow exponentially with the depth. While this obstacle seems inherent if one aims to prove the knowledge soundness of this construction (since the security notion demands to extract the witness, in this case $\omega_1, \ldots, \omega_\ell$), there is no evidence of the need of such recursive extraction if one considers, say, IVC for *deterministic* computations, i.e., for proving $z_\ell = F^{(\ell)}(z_0) = F(F(\cdots F(z_0)))$. In this case, it is enough to prove soundness of the IVC construction, a notion that does not require to explicitly define an extractor. Yet, the constant-depth barrier remains because recursive extraction is simply the best-known strategy to prove security (this is also true of related and even more efficient blueprints [KST22, BCL+21]).

This raises the natural question: *may the constant-depth limitation of SNARK-based* IVC *be an artifact of known proof strategies rather than a fundamental barrier?* This state of affairs is clearly unsatisfactory given that *i)* a subset of SNARK-based IVC shows the most promising efficiency profiles but are not known to be secure beyond $O(1)$ depth; *ii)* several practical use-cases of IVC require proving chains of computations of superconstant length. In light of this, assuming heuristic security

[3] The same points hold for the case of other known constructions based on folding or accumulation [KST22, BCL+21]. In this section, we will often implicitly refer to such constructions as well whenever we write "SNARK-based" IVC or "canonical constructions" of IVC.

has become an increasingly common practice among both practitioners and academic researchers despite the lack of solid theoretical foundations[4]. Our work is trying to lay the groundwork towards (potentially) more constructions that are provably secure beyond $O(1)$ depth.

1.1 Our Results

In this work, we advance our understanding of the security foundations of IVC, with an eye toward finding new approaches to prove security at superconstant depth for constructions currently known to be secure only at $O(1)$ depth. We do so by analyzing three natural questions around the (non-extractable) security of IVC. Main differentiators between our approach and prior work are: *i)* identifying *depth as a structural security element* for IVC and the questions we ask around it; *(ii)* the focus on *black-box* results, valid for any (black-box) scheme Π and relying only on already assumed properties for Π (for example, our depth-boosting transformations in Section 5 can obtain soundness without having to require any extractable primitive, in contrast to prior transformations).

We stress that *our results hold across different notions of security*, namely they apply to the standard notions of deterministic and non-deterministic soundness, as well as to the security of Incremental Functional Commitments (IFC). The latter is a streaming variant of functional commitments that we introduce and is an additional contribution of this work (see next paragraphs in this section). IFC broadens the applicability of the standard soundness notions. For simplicity, in the main text we refer mainly to deterministic soundness and we formalize how our results extend in Appendix F.

1.1.1 Soundness beyond $O(1)$ depth from infinitely-often security.

Earlier in this introduction, we pointed out how, in principle, some of the canonical constructions of IVC *might* be secure beyond constant depth, but we may be lacking the right proof strategies. The next question is motivated by finding such proof strategies.

What are sufficient conditions to prove that an IVC *is sound for computations of superconstant depth?*

We uncover an intriguing connection between *infinitely-often* soundness[5] and standard soundness in IVC: a scheme infinitely-often sound at depth D satisfies *standard* soundness at some depth d (with $d = o(D)$). This connection may provide additional avenues to prove standard security of IVC indirectly; as we discuss below *infinitely-often* security—from standard cryptographic assumptions or even worst-case complexity-theoretic assumptions—is an active field of research, Also, to the best of our knowledge, IVC is the first cryptographic primitive where one finds this kind of *self-reducibility* from almost-everywhere soundness (the standard notion) to infinitely-often soundness. The following is an informal statement of our result:

Theorem (Informal statement of Corollary 1). *Let Π be an IVC scheme and $\mathsf{D} = \omega(1)$ be a depth bound. Let $E \subseteq \mathbb{N}$ be an infinite set of security parameters where Π achieves negligible soundness at depth bound D. Then there exists a depth bound $d = \omega(1)$ where Π achieves (standard) negligible soundness.*

[4] At this stage it is worth mentioning that there are alternative approaches to construct IVC for polynomial-depth computation by using *stronger* assumptions (and by modifying the construction) or employing other cryptographic primitives that are currently far from having practical realizations (see Section 1.4). One approach that is an exception in this sense is the one we refer to as the *tree-based* approach to IVC [BCCT13], to which we compare some of our results both in Section 1.4 and on page 8. See also beginning paragraphs in Section 1.1.

[5] We recall that, informally speaking, infinitely-often security is required to hold only for *some* infinite set of security parameters $\{\lambda_1, \lambda_2, \lambda_3, \dots\}$. This is contrast to the stronger and more common notion (*almost-everywhere* security), which requires security at *all but finitely many* security parameters $\{\lambda_0, \lambda_0 + 1, \lambda_0 + 2 \dots\}$.

Finding concrete secure depth bounds. One interesting property of our results on infinitely-often security is that they tell us not only that there exists a fully-secure depth bound d, but also *where* it lies. The sparsity of the infinite set E—how frequent the secure parameters are—plays a key role in determining for which bounds a scheme is sound. We stress that, as we discuss in the next paragraph, the type of sparsity we require is very mild and has been shown to hold, in certain settings, from standard assumptions. Our result on sparsity follows:

Theorem (Informal statement of Corollary 2). *Let $\Pi, \mathsf{D} = \omega(1)$, and E as in the previous theorem. If E is sub-exponentially sparse*[6] *then Π is sound at depth $d = O(\mathrm{polylog}\mathsf{D})$.*

Two remarks are in order. The first is that our general theorem statements in Section 3 already support even weaker sparsity conditions. Specifically, they show that exponential sparsity implies security at logarithmic depth d. Also, our results can be extended to support even weaker sparsity, albeit obtaining different and plausibly less natural secure depth bounds d (it is also unclear how interesting such sparsity requirements would be).

On the significance of our results. Our results on infinitely-often security have two goals: to obtain a more nuanced understanding of how the security of different depths "interact" with one another, and to provide new avenues to prove security at super-constant depths. In the context of proof strategies for IVC, our approach unlocks a methodology of this type: *in order to prove security at depth $d = \omega(1)$, show a weaker soundness notion at some appropriate "larger" depth D?*

The reason why this avenue may be promising is that the flavor of the "weaker" notion we look at, infinitely-often security (or, i.o.-security), constitutes an active topic of research and has been shown to hold from different types of assumptions in a variety of settings *including proof systems*. A recent work by Couteau et al. (CRYPTO 2023) has shown that sub-exponential CDH implies infinitely-often sound NIZKs with exactly the type of sub-exponential sparsity required by our result above [CJJQ23]. Potentially, their techniques could be ported to the IVC settings as tools to prove security beyond O(1) depth. Also, infinitely-often soundness may be useful to link (or separate) IVC security from other primitives. In particular, recent works have shown that infinitely-often security serves as a bridge between worst-case complexity assumptions and cryptography: both Liu and Pass [LP21](CRYPTO 2021) and Hirahara, Lu, and Oliveira [HLO24] (TCC 2024) showed that the existence of infinitely-often one-way functions is equivalent to closing seemingly minor gaps between different notions of average-case hardness for natural computational problems. In both cases, the infinitely-often variant is what enables clean, near-unconditional characterizations. Our hope is that some of the techniques from these works may be applicable to the realm of IVC soundness. We discuss additional implications of our results in the context of the Gentry–Wichs impossibility [GW11] in Section 1.2.

1.1.2 How does soundness degrade across (insecure) depths?

We introduce our next question through some gradual context. Assume that some of the canonical constructions we know (e.g., any of the ones in [KST22, BCL+21]) happened to be *genuinely insecure* for computations beyond depth $d_{\mathrm{safe}}(\lambda) = O(1)$. More generally, consider any IVC construction Π that is sound *only* for constant-depth computations. Under this assumption, the powers of PPT adversaries change substantially "between" d_{safe} and a fast-growing depth bound such as $D'(\lambda) = \lambda^3$—being able to obtain, when running in time $T = \mathsf{poly}(\lambda)$, only a negligible advantage at depth d_{safe}, but being able to obtain advantage at least $\frac{1}{p'(\lambda)}$ (for some polynomial $p'(\cdot)$ and infinitely many values of λ) in the soundness experiment for D'.

[6] Formally defined as follows: for all $\bar{\lambda} \in \mathbb{N}$, we have that $E \cap \left\{\bar{\lambda}, \bar{\lambda}+1, \ldots, 2^{\bar{\lambda}^T}\right\} \neq \varnothing$ for some $T \in (0,1)$. That is, sub-exponential sparsity states that security parameters in E are separated by a "jump" of at most sub-exponential size.

If the adversary got more powerful as we "moved away" from depth bound d_{safe}, it is reasonable to ask: would the adversary be *less* powerful at depth bounds that are "closer" to d_{safe}. As an example, consider the "intermediate" depth $D''(\lambda) = \sqrt{\lambda}$—will any adversary running in time T obtain a strictly worse advantage than $\frac{1}{p'(\lambda)}$ for depth D''? That is, technically speaking, will all such adversaries obtain only advantage (on any infinite set of parameters λ) at most $\frac{1}{p''(\lambda)}$ with $p'' = \omega(p')$[7]? And if yes, will this phenomenon continue indefinitely as depth bounds "get closer" to d_{safe} or will it plateau at some point?

Different answers to the last question would paint very different pictures of the dynamics between depth vs. security for the hypothetical IVC scheme Π we started from (see Fig. 1). In one scenario, intuitively, the "insecurity" of depth bounds keeps getting less and less bad as we get closer to d_{safe} (if that is the case, we say that Π shows a *graceful security degradation*). In the other case, in contrast, the transition has more of a "sharp cliff" behavior (or, more generally, it will look like a staircase with finitely many steps). The next question we investigate revolves around whether these two patterns are always both possible across IVC constructions that are insecure at some depth (or if only one of them can occur).

Further motivation. From a theoretical standpoint, the question above is more than an intriguing curiosity. In fact, technically, it reduces to a question about when and how a certain type of security—security with inverse polynomial (or *noticeable*) soundness error—is possible in IVC. And the existence of cryptographic arguments with inverse polynomial soundness—in settings where standard soundness is not easy (or possible) to obtain—is an active field of research [BIOW20, FS21, AFKR24, BHI+24]. At the same time, the possibility of graceful security degradation is also motivated by a practical angle. Lower security guarantees are in fact still useful sometimes in real-world protocols. For example, the soundness for proofs used in the Filecoin protocol [GN23] is only 2^{-10}; in general, less stringent security requirements are possible in settings with other incentive-based constraints on the adversaries. In such scenarios, a scheme Π with graceful security degradation may offer the possibility for both insecurity mitigations as well as tradeoffs to practitioners: they may obtain better security if they are willing to run the protocol at more modest depth bounds (even depth bounds that are technically "insecure").

Our result. Regrettably, we find not only that such mitigation may be impossible for *some* schemes insecure beyond $O(1)$ depth, but that it is *never* possible:

Theorem (Restatement of Theorem 2). *No* IVC *scheme that is secure only at constant depth can exhibit a graceful security degradation.*

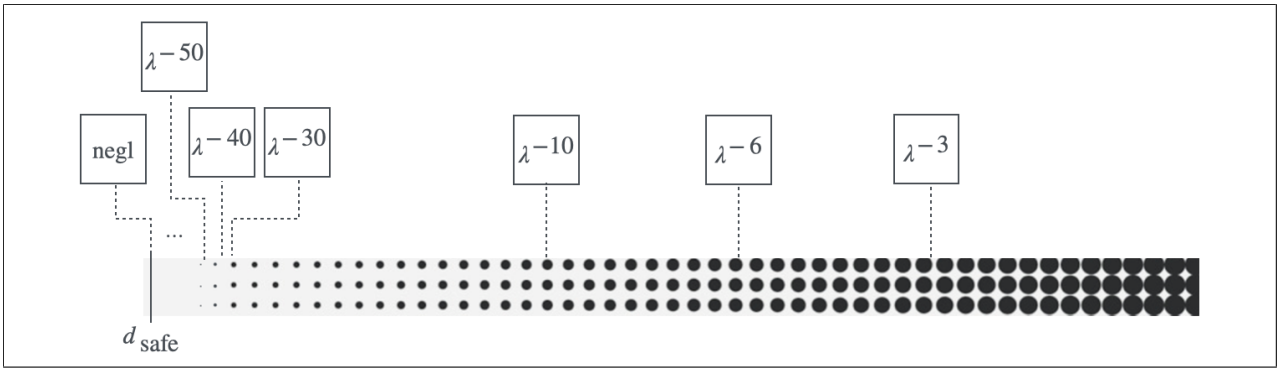
We also generalize our result to depths beyond constant: no proof scheme with a "thin" security boundary, i.e. at any depth $\omega(d)$ for some $d(\cdot)$, can exhibit graceful security degradation.

Is graceful security degradation ever possible then? If an IVC Π is insecure for some depth, there exists a type of "insecurity profile" that escapes our analysis and for which graceful security degradation is possible, at least in principle. Consider an hypothetical Π that is secure "around" $D = \log^2 \lambda$ (but not at D) in a specific pattern. In particular, assume Π is secure at all depth bounds in $\mathcal{S} := \bigcup_{\varepsilon \in [0,1)} O(\log^{1+\varepsilon}(\lambda))$ and is insecure instead anywhere else, i.e., at any depth bound in $\mathcal{U} := \mathsf{poly}(\lambda) \setminus \mathcal{S}$. In this case, our analysis does not seem to apply.

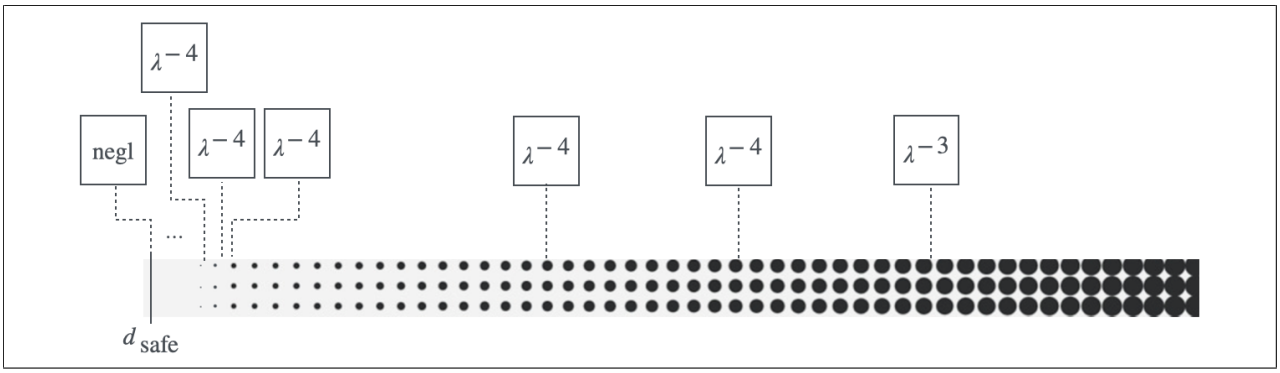
This suggests a few items for future work. Let us call an IVC *regular* (as in, having a "regular" safety profile) if any of these is true:

- there exists $d \in \mathsf{poly}(\lambda)$ such that the set of secure depth bounds $\mathcal{S}$ is such that $\mathcal{S} = O(d)$ or $\mathcal{S} = o(d)$.
- Π is secure at any polynomial depth bound.

[7] A polynomial of higher degree means a worse advantage for the adversary.



(a) An hypothetical scheme with graceful security degradation. Such schemes achieve arbitrarily good noticeable soundness for some $d' = \omega(d_{\text{safe}})$ "sufficiently close" to d_{safe}. We prove that no IVC that is insecure beyond constant depth can satisfy this property.



(b) An hypothetical scheme where security does not degrade gracefully: as we move "closer" to the depth bound d_{safe} the best advantage for a bounded adversary may get worse but will always be at least some inverse polynomial λ^{-k} (in the figure $k = 4$).

Fig. 1: Visual intuition behind *graceful security degradation*. Each point on the x-axis represents a depth bound; if $d'(\cdot)$ is on the right of $d(\cdot)$ then $d' = \omega(d)$. On the top part of the figures, we describe the best advantage of a bounded adversary against the scheme at the related depth bound. On the left, d_{safe}, denotes a depth bound for which the scheme is sound (negligible advantage).

Although we do not formalize explicitly in this work, our analysis extends immediately to the $o(d)$ case of regular IVCs above and so we know that *no regular IVC can exhibit graceful security degradation* (where this is true for IVCs that are secure everywhere as a sort of degenerate case).

All the IVC constructions we are aware of are plausibly regular. Non-regular IVCs seem to have a counterintuitive security profile. What is open is:

- Can non-regular IVCs actually exhibit graceful security degradation or can one prove it is impossible for them as well? If one found an impossibility it would settle the question of "graceful of degradation as a potential mitigation" as something that is not possible even for more exotic security profiles.
- Can one build non-contrived non-regular IVCs in the first place? That is, are these a type of IVC that we may find ourselves working with in the first place?

1.1.3 Can we use an IVC for constant-depth computation to build one for deeper computations? Assume there exists an IVC construction that is sound for computations of constant depth $d = O(1)$. Can we use it to construct IVC schemes that are sound for "larger" depth while

keeping minimal overhead in proof size, incrementality, proving time and memory? Our focus will be on results that introduce no assumption and that treat the IVC as a black-box (i.e., assuming nothing on how it works internally)

We identify two such *depth-boosting* approaches. One has less general applicability and stems from observations on the relation between security parameter and depth[8]. The other approach has essentially no requirements on the depth at which the starting scheme is sound, follows from simple algorithmic principles and provides different efficiency tradeoffs compared to the first result (which we will discuss later). We leave as future work to explore the practical relevance of our results (with the second one being the most promising candidate in this sense). Both our approaches, however, can be seen as conceptual evidence that *extractability may not be required* in order to obtain an IVC (with non-extractable soundness requirements) that achieves security for arbitrary polynomial depth. Thus, they stand in contrast with the canonical *tree-based* approach—proposed in [BCCT13] and used in recent works such as [NDC+24]—which bootstraps IVC sound at polynomial depth via *extractable* Proof-Carrying Data (PCD) sound up to constant depth.
We now state our two results:

Theorem (Informal statement of Theorem 5). *If there exists an* IVC *sound at depth bound* $d(\lambda) = \lambda^{\varepsilon}$, $\epsilon > 0$ *then there exists an* IVC Π' *that is secure at any polynomial depth. The scheme* Π' *has proof size, verification time and prover time as those of* Π *up to a* $O_\lambda(1)$ *multiplicative factor (see Section 2).*

One can show the result above by proving that a scheme Π such as the one above is *already* secure at depth D, *conditioned* on an appropriate choice of a (larger) security parameter. This result holds for depth bounds that are at least sublinear (or $\Omega(\lambda^{\varepsilon})$ with $\epsilon > 0$) thanks to the fact that, in this class of functions, we have that $\mathsf{D}(\lambda) = (d(\lambda))^c \implies \mathsf{D}(\lambda) = d(\lambda^c)$, which is true for sublinear/polynomial depths but not for every depth (consider, e.g., $d = \log \lambda$ and $\mathsf{D} = \log^2 \lambda$).

Our next result bypasses the limitation of the previous theorem.

Theorem (Informal statement of Theorem 6). *Let* Π *be an* IVC *sound at arbitrary depth bound* $d(\cdot)$ *, then there exists an* IVC Π' *that is secure at depth bound* $\mathsf{D}(\cdot) = \omega(d)$ *(arbitrarily chosen). Let* $\rho(\cdot) := \log_d \mathsf{D}$ *(potentially superconstant), then the scheme* Π' *has the following efficiency profile*[9]*:*

- *its proof size and verification time are those of* Π *up to a multiplicative factor* $O(\rho)$*;*
- *its prover has amortized proving time as that of* Π *up to a multiplicative factor* $O(\rho)$*.*

An implication of the second result is the following:

Theorem (Informal implication of Theorem 6).

- *If there exists an* IVC *secure at depth* $d = O(1)$ *then there exists a scheme secure at arbitrary polynomial depth.*
- *Moreover, if the original scheme has:*
  - *proof size and verification time* v*;*
  - *proving time* p*;*

  then the resulting scheme has:
  - *proof size and verification time* $O(v \log \lambda)$*;*
  - *amortized proving time* $O(p \log \lambda)$*.*

[8] A bit similar to the ones that allowed us to show our results for infinitely-often security although that setting required a more sophisticated analysis compared to our first depth-boosting result.

[9] The following summary of the efficiency is accurate up to minor nuances depending on the prover efficiency of Π or the size of the output of the function we are iterating on. See details in Section 5.2.

For an intuition on the construction. Consider a scheme Π secure up to depth d. To obtain one secure up to d^2, we follow the approach described in Fig. 2: we prove each step through Π, but every d steps we "discard" the work we did so far and we proceed by one single, more expensive step. The steps are spread out enough so that the final amortized work of this final prover is approximately that in the original scheme. The interspersing of proving big and small jumps is such that we never rely on a chain on more than d steps. This idea is easy to generalize to cases where $\mathsf{D} = d^c$ for an arbitrary $c \in \mathbb{N}$, but the same approach works for $\mathsf{D} = d^\rho$ with ρ superconstant (see also Remark 5). This is what allows us to show the implication in the last statement above, by picking $d = O(1)$ and $\rho = O(\log \lambda)$, we can obtain security at any depth $\mathsf{D} = \lambda^c$.

1.1.4 Incremental Functional Commitments (IFC).

We introduce a new notion of IVC of broad applicability: *Incremental Functional Commitments* (or IFC).

Motivation for IFC: incremental computations on committed data. The flavor of security we have in mind, as well as the settings in which it can be useful is better presented through an example (the following example in particular is from the field of *verifiable machine learning* [PWZ+25, ZFZS20, LXZ21, BFG+23, CFF+24, APKP24]). Consider a neural network NN, which we will assume to be public. We are interested in letting computationally weak clients verify the evaluation of a query q to the network. Such clients can hold a (succinct) commitment $\mathsf{cm}_{\mathsf{NN}}$ to the neural network and obtain a short proof of correct evaluation of their query.

There are several ways in which neural networks are natural examples of incremental computations. For instance, *recurrent* neural networks (RNN) repeatedly apply the same building block—the recurrent unit—at each step, where the unit takes both the current input and the previous hidden state to produce a new hidden state and output (see, e.g. [GBC16, Chapter 10]); this is also true of LLMs, where each layer applies the same architectural template with distinct parameters [VSP+17]. Incremental proofs can clearly be beneficial: the evaluator does not need to load into memory the whole trace of the evaluation to prove the NN's output, but can start proving while still computing the final results.

Another motivating examples is that of polynomial commitments [KZG10]. An IVC-friendly polynomial commitment would allow to prove an evaluation on a point t by accessing the polynomial coefficients in a streaming fashion. Since polynomial commitments are commonly a key building block in SNARKs [CHM+20, BFS20], this can in turn help build[10] SNARKs with streaming-friendly provers.

The notion of security we are interested in. What is interesting about the setting above is that the form of soundness sufficient for applications here is strictly weaker than knowledge soundness[11]. Intuitively, we are interested in what is known as *weak evaluation binding* in the functional commitments literature [CFT22]. That is—in the framework of the ML example above—all we are interested in is that a malicious prover should not able to provide a valid proof for a dishonest evaluation $y \neq \mathsf{NN}(\mathsf{q})$, a honestly generated commitment $\mathsf{cm}_{\mathsf{NN}}$ to NN and a query q. We will also show how we can support the stronger version of this form of security, called *evaluation binding*, where the adversary provides a malicious commitment and two accepting proofs for the same query and distinct outputs.

In our work we introduce a form of functional commitments with *incremental* proving (Incremental Functional Commitments); we show general techniques to construct them directly from the usual

[10] This can be particularly useful to unlock proving on weak devices. Despite their practical improvements, cryptographic proofs with short proofs still tend to require large memory, usually due to the prover having to load the whole trace of the computation in memory. We remark at the same time that *some* of the steps where polynomial commitments are used for SNARKs require them to be extractable. This is not the type of soundness requirement that we will propose in this section. At the same time, *computation commitments*, the ones encoding the structure of the circuits in a SNARK design, are usually required to offer only evaluation binding, which our constructions will satisfy.

[11] In the sense that the soundness notion will not require an extractor and any extractable construction will automatically satisfy this soundness notion.

templates for building IVC and PCD (this includes both SNARK recursion and folding/split accumulation [BCMS20, BCL+21]; our formal analysis explicitly covers the former, but it is immediately applicable to folding/accumulation without any major change).

Our construction. We show a general construction for IFC for any "chunkable" function from collision resistance and succinct arguments of knowledge. A function is chunkable if there exists a way to compute it in a streaming fashion. Both the commitment and the proving algorithm in our IFC are *incremental* (they need only the last commitment/proof and the new input) and thus offer the usual advantages of IVC/PCD-like approaches (small space requirement, streaming computation, etc). We are able to prove that our IFC satisfies evaluation binding even for *arbitrary depth* by relying on our depth-boosting results in Section 5.

Theorem (Informal). *Let* H_{cr} *be a collision resistant hash function and let* Π_{SNARK} *be a succinct non-interactive argument of knowledge. Then, there exists an incremental functional commitment scheme* Π *for chunkable functions (Definition 15 in Appendix D). If* Π_{SNARK} *has constant-sized proofs then* Π *satisfies evaluation binding for different depth bounds* D *and its succinctness is as follows:*

- *for* $\mathsf{D}(\cdot) = O(1)$*,* Π *has proofs of constant size;*
- *for* $\mathsf{D}(\cdot)$ *arbitrary polynomial,* Π *has proofs of logarithmic size (if the target function* F *has constant-sized output)*[12]*.*

1.2 Further Discussion and Other Future Directions

IVC for polylogarithmic depth from falsifiable assumptions and with only polynomial-hardness. Our results on infinitely-often soundness suggest an avenue to build IVCs for non-deterministic computations of *polylogarithmic* depth from *falsifiable* assumptions (see also remainder of this discussion on our extensions to other soundness notions). Such IVCs, in turn, may entail SNARGs for relations in NP with polylogarithmic witness size. The latter is a class of problems that is believed neither to be decidable in polynomial time nor to include NP-hard problems (unless the exponential-time hypothesis is false; see, for example, [IP01]). As a consequence, this class escapes the impossibility results from Gentry and Wichs [GW11] which rule out succinct non-interactive arguments for *hard* subset membership problems from the polynomial-hardness of falsifiable assumptions. Our results, therefore, provide for the first time an avenue to build SNARGs for this subset of NP from falsifiable assumptions and without relying on sub-exponential hardness or complexity leveraging.

New proof strategies from graceful security degradation. There exists an additional lens on our result on the impossibility of graceful security degradation. Beyond suggesting how to adjust our expectations, it may be useful to offer a potential approach to *proving that a scheme has negligible soundness beyond constant depth.* By the contrapositive of the theorem's statement, if we could show—towards a contradiction—an infinitely descending ladder of "insecurity levels" (all inverse polynomials) at slower and slower growing depth bounds (see for example Fig. 1a), then we can conclude that the scheme must be secure beyond constant depth.

1.3 A Quick Overview of Our Techniques

Here we quickly highlight the approach used in our first three results (the reader can find an extended technical overview for IFC in Section 6; the other sections are intended to be self-contained). These three results share a common methodology: they treat the depth bound as a first-class parameter and analyze how it interacts with the standard security definitions. The resulting arguments are

[12] We obtain this last result through our depth-boosting techniques in Section 5

relatively short—the main tools are reparameterization, monotonicity of advantage, and a recursive construction—but they are enabled by a lens on IVC security that, as far as we know, has not been adopted before. We believe the conclusions they reach would be difficult to anticipate without this viewpoint.

From infinitely-often to standard soundness. The idea is to reparameterize: when Π is i.o.-sound at depth D with respect to some infinite set E of good security parameters, we instantiate Π on the largest "good" parameter $\lambda_i \leq \lambda$ in E. Any adversary targeting depth $d'(\lambda)$ is then also a valid adversary at depth $D(\lambda_i)$, provided $d'(\lambda) \leq D(\lambda_i)$. What remains is a combinatorial analysis of the gaps in E: their sparsity translates into concrete depth bounds d' at which standard soundness holds. What is different from other settings with reparametrization is that we are exploiting a structural feature specific to IVC—namely, that the depth bound creates a *slack* between the parameters at which i.o.-security is given and the parameters at which standard security is needed. This is what allows the reduction to go through without any change to the scheme's algorithms, and it is why the connection between i.o.- and standard-soundness, which would be vacuous for generic primitives, becomes meaningful for IVC.

Impossibility of graceful degradation. Suppose for contradiction that security degrades gracefully: for every inverse-polynomial level λ^{-e}, some superconstant depth bound achieves it. This yields an infinite ladder of depths with improving security levels. The argument then proceeds in two steps.

First, a *monotonicity observation*: an attack valid at shallower depth remains valid at any deeper depth with the same advantage—this is immediate from the definitions but, combined with the ladder, it places strong constraints on any depth "in between." Second, a *diagonalization argument* (Theorem 3) produces a superconstant depth bound L that is eventually dominated by every rung of the ladder. Since L is superconstant, the scheme must be insecure at L; but monotonicity forces L's insecurity level to match or exceed every rung, contradicting the insecurity of L. We note that the modeling around graceful security degradation requires some care. For example, the parameterization of the insecurity level by the adversary's running time (Definition 8) is needed to prevent standard amplification from trivially collapsing the ladder. The argument generalizes to arbitrary base depths (not just $O(1)$) and to other soundness notions.

Depth boosting. Given Π sound up to depth d, we construct a scheme sound up to d^ρ by alternating "small" and "big" proof steps: the prover runs Π on individual steps but every d steps consolidates with a big-jump proof covering d applications of F at once (Figure 2). No chain exceeds d steps, yet d^2 steps are covered; iterating ρ times yields d^ρ.

The description is recursive, but—and this is the point—the unrolled construction consists of ρ independent proof chains of depth at most d. The security reduction is therefore *not* recursive: it is a union bound over ρ events, each amounting to a single invocation of Π's soundness. This distinction matters. In extraction-based approaches, each level of recursion incurs a polynomial blowup, making superconstant composition inherently problematic. Our construction sidesteps this entirely, which is what allows ρ to be superconstant (e.g., $\rho = O(\log \lambda)$ with $d = O(1)$, yielding polynomial depth). We view this as evidence that the constant-depth barrier in existing IVC may be partly an artifact of extraction-based proof strategies, not an inherent limitation of the constructions themselves.

1.4 Related Work

We are not aware of other works in literature discussing general relationships between depth and security of IVC schemes. Below, we offer an overview of the IVC literature and focus on comparison with our results that offer also constructions rather than only proof strategies (namely, those in Section 5). These works can be roughly separated among those that study *extractable* IVC vs those that provide constructions with non-extractable soundness.

Several works have addressed the problem of achieving IVC that is (knowledge) sound for computations of polynomial depth, using different approaches or models.

A generic tree-based approach. In [BCCT13], Bitansky et al. point out the issue of applying recursive extraction beyond polynomial depth. To address this issue they show that a constant-depth PCD[13] suffices to construct one for computations of polynomial depth by building a constant-depth tree of the computation steps, and applying recursion on each node of the tree. In contrast, our depth-boosting construction of Section 5 does not need to rely on extractable building blocks (extractable PCD or SNARKs), which may be of independent conceptual interest and may broaden the space for constructions of IVC secure at larger depths.

Straight-line extraction in an oracle model. Two works [CT10, CCG+23] prove knowledge-soundness of PCD for computations of arbitrary depth. They consider a construction of PCD based on the recursive composition of SNARKs and they overcome the issue of recursive extraction by assuming the SNARK to satisfy a stronger notion of knowledge soundness, that is the availability of a (black-box) straightline extractor. Straightline extraction however would be information-theoretically impossible for succinct proofs without extra assumptions. Therefore, they consider SNARKs working in a *relativized* setting working in oracle models, specifically the signed ROM [CT10] and the arithmetized ROM (AROM) [CCG+23]. This analysis has been made explicit, for any oracle model θ, in [CGSY24].

The recent work in [CFP25] combines AGM and oracle models through a general framework to show provably secure composition in incremental proofs and other primitives. A related and more restricted approach (assuming a very specific oracle related to the AGM) also appeared in [MMZ25]. In contrast to the aforementioned results, which crucially rely on (a specific flavor of) extraction to reason about security at arbitrary depths, our results also apply to non-extractable primitives. More generally, our angle is orthogonal to that in these works since our approach applies to any scheme proven secure (for a certain depth) under arbitrary assumptions.

Deterministic IVC from standard assumptions. Another line of work considers the problem of constructing IVC for arbitrarily long computations—and, for the most part, deterministic ones (see next paragraphs)—from standard falsifiable assumptions. Building upon a line of research that constructs SNARGs for P from such assumptions, these approaches leverage a particular tool: a special form of batch arguments (BARGs) known as rate-1 somewhere extractable BARGs (seBARGs), which is known to be realizable from standard assumptions. In [PP22, DGKV22] they build sound IVC for purely deterministic computations, while the recent work in [AG25] constructs a more expressive not strictly deterministic primitive (their security notion is called *strong soundness*[14]; they also define and obtain a form of zero-knowledge for their primitive). While this line of work offers a solution to the problem of constructing IVC that is secure for poly-depth computations, even from standard assumptions, it does so by using a cryptographic primitive, rate-1 seBARGs, whose currently known instantiations are unlikely to be practical. In contrast, our work investigates the security limits of IVC in general and can therefore shed light on the security of practical IVC schemes, whose security level for computations beyond constant depth is currently uncertain. Also, our constructions in Section 5 may provide different concrete efficiency tradeoffs.

Finally, the recent work in [DJJ+25] constructs IVC for NP from standard assumptions (which include LWE and iO); similar considerations about generality and practicality compared to our work apply here as well.

[13] Proof Carrying Data, a generalization of IVC [CT10].

[14] This notion can also be seen as providing something extremely close to IFC, although not directly and only after some modifications. For example, their notion does not include an explicit incremental commitment algorithm; instead, their Definition 3.1 has an incremental *update* algorithm—which produces an incremental *proof* and internally recomputes a digest—and a *digest* algorithm that does not directly support incremental hashing syntactically. However, one could define an incremental commitment algorithm by separately running step 6 in their update algorithm. See Construction 3.4 (notice that the inputs for step 6 are all independent of the RAM program).

Additional security notions in incremental proofs. In this paper we introduced a notion of evaluation binding for a form of incremental proofs on top of the more standard soundness (for both deterministic and non-deterministic IVCs) and extractability. We already mentioned the work in [AG25] which introduces a security notion dubbed *strong soundness*. The recent work in [ACDK25] studies non-malleability in the context of incremental proof systems and shows how the standard notion must be adapted for Proof-Carrying Data and Incrementally Verifiable Computation.

"Recursion-free" primitives close to IVC. The recent work of Wang, Papamanthou, Srinivasan, and Papadopoulos [WPSP24] proposes a different approach to building IVC without using recursive composition of SNARKs. The primitive they construct, called BIVC, achieves a functionality similar to IVC except for requiring the prover to produce incremental proofs with the help of an auxiliary data structure and for needing an a-priori bound on the length N of the computation. The approach of [WPSP24] builds upon ideas from [TFZ+22], which they improve by achieving a prover working with a data structure of size $\sqrt{N|F|}$ as opposed to $O(N)$. In relation to our work, the BIVC approach is an example of a method to achieve IVC avoiding recursive SNARKs and their security pitfalls, though the functionality it achieves does not satisfy the properties that are in the motivation of our work, that is IVC with unbounded incremental proving and short space.

Functional commitments. Related to our notion of IFC, we provide basic related work on functional commitments. The notion of functional commitment has been first formalized by Libert et al. [LRY16] as an extension of vector commitments [LY10, CF13, LM19, TAB+20, GRWZ20, Cam26]. While the first construction of Libert et al. supports the evaluation of *linear forms*, a series of recent works advanced the state of the art proposing schemes for key-value maps [AR20, LGG+22, CEO22], linear maps [LM19, CNR+22, BBC+25], polynomials [BFS20, CFT22, ABC+24, CHA24, CCC+25][15], and circuits [WW23, dCP23, BCFL23, CFK24, WW24]. In this work, we study a new property of functional commitments that is a streaming-friendly incremental proving algorithm. To the best of our knowledge, none of the existing constructions has this property. One FC scheme which has some flavor of incremental proving is [BCFL23] which builds an FC through *chainable* functional commitments (CFC). In particular, they use a CFC to prove the computation's correctness of each layer of an arithmetic circuit and the chainability property allows connecting these proofs. Their approach generalizes to sequential computations but inherently makes the proof size grow with the computation's depth. This is in contrast with the goal of our FC constructions where proofs do not grow with the depth.

Chunking and incremental hashing in IVC-like primitives. Here we discuss works with techniques resembling the general incremental hashing approach we adopt to build IFC. Mangrove [NDC+24] adopts incremental commitments (hash chain or Merkle tree) with IVC/PCD to prove circuit satisfiability in a streaming manner by breaking up the extended witness into chunks and using a "commit-and-fold" technique. Nebula [AS24] generalizes this notion to what they call *commitment-carrying-IVC*[16] and using a similar incremental commitment. From the notional point of view, we stress that these works have stronger security requirements than IFC, namely *extractability*. One of the main points of introducing IFC is exploring non-extractable (yet applicable in common settings) security guarantees. We also point out that another contrast between Mangrove and our work is that the "chain" construction in Mangrove (for $O(1)$ depth) requires *two streaming passes* on the input, while our respective IFC construction requires only *one pass*[17]. This advantage, for our IFC, comes at the cost of being restricted to *chunkable functions* (Definition 15), whereas Mangrove's

[15] Polynomial commitments had also already been proposed as a specialized form of functional commitments in [KZG10].
[16] Which, as in commit-carrying SNARKs [CFQ19] is a proof system whose output is both a proof and a commitment.
[17] The construction in Mangrove can be modified to use only one pass adopting the deterministic hash function from [CDvD+03] modeled as a random oracle. This approach, however, comes at the cost of heuristic security and a less efficient construction.

second pass allows it to support more expressive computations. Chunkable functions can thus be seen as a setting where we can obtain single-pass proof systems.

2 Preliminaries

We refer to a *depth bound* as a non-decreasing, positive function over the natural numbers. All the depth bounds we will consider are implicitly in $\mathrm{poly}(\lambda)$. Given two depth bounds f, g, we say that f eventually dominates g (denoted by $f \succeq g$ or $g \preceq f$) if there exists λ_0 such that for all $\lambda \geq \lambda_0$ $f(\lambda) \geq g(\lambda)$. We write $f = O_\lambda(g)$ if there exists $c \in \mathbb{N}$ such that $f = O(\lambda^c \cdot g)$. The notation $f = \tilde{O}(g)$ denotes quasilinear relations, that is $f = O(g \cdot \mathrm{polylog}(g))$. Occasionally we write $z \rightsquigarrow z'$ to denote the statement "z' is the output of z after iterating function F a certain number of times" with F being implicit from the context.

Adversarial model. Whenever we talk about *efficient* adversaries we consider non-uniform machines of polynomial size. That is, an efficient adversary $\mathcal{A}$ is a family of circuits $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ of size $\mathrm{poly}(\lambda)$.

2.1 Background on Deterministic IVC

For simplicity, this background section focuses on deterministic IVC. We provide security notions for IVC-like primitives for non-deterministic computations in Appendix B and Appendix D. Below $F : \{0,1\}^n \to \{0,1\}^n$ denotes a function computable in polynomial time; $|F|$ denotes the time required to compute F.

Definition 1 (Incrementally Verifiable Computation (for deterministic computations)). *An Incrementally Verifiable Computation (*IVC*) scheme is a tuple of PPT algorithms* $(\mathsf{G}, \mathsf{I}_{\mathsf{fun}}, , \mathsf{I}_{\mathsf{dpt}}, \mathsf{P}, \mathsf{V})$ *defined as follows:*

- $\mathsf{G}(1^\lambda, 1^N) \to \mathsf{pp}$: *Takes security parameter λ and N, a bound on $|F|$, and outputs public parameters* pp.
- $\mathsf{I}_{\mathsf{fun}}(\mathsf{pp}, F) \to (\mathsf{ipk}_{\mathsf{fun}}, \mathsf{ivk}_{\mathsf{fun}})$*: Takes as input public parameters* pp *and a function description F and outputs a proving key* $(\mathsf{ipk}_{\mathsf{fun}})$ *and a verification key* $(\mathsf{ivk}_{\mathsf{fun}})$ *for the function.*
- $\mathsf{I}_{\mathsf{dpt}}(\mathsf{pp}, 1^d) \to \mathsf{ivk}_{\mathsf{dpt}}$*: Takes as input public parameters* pp *and a depth d and preprocesses a depth verification key for d (we do not require a corresponding proving key).*
- $\mathsf{P}(\mathsf{ipk}_{\mathsf{fun}}, z_0, z_i, (z_{i-1}, \pi_{i-1})) \to \pi_i$*: Takes as input a proving key, messages z_0, z_i, incoming message z_{i-1}, proof π_{i-1}, and outputs a new proof string π_i for outgoing message z_i.*
- $\mathsf{V}\left(\mathsf{ivk} = (\mathsf{ivk}_{\mathsf{fun}}, \mathsf{ivk}_{\mathsf{dpt}}), z_0, z_{out}, \pi_{out}\right) \to \top/\bot$: *Takes as input verification keys, the final messages z_0, z_{out} and corresponding proof string π_{out} and outputs a decision $\top/\bot$.*

A secure IVC *satisfies the following properties.*

Correctness. For all functions F, efficient adversaries $\tilde{\mathsf{P}}$, and all $\lambda \in \mathbb{N}$, we have:
Base case:

$$\Pr\left[\mathsf{V}((\mathsf{ivk}_{\mathsf{fun}}, \bot), z_0, z_0, \bot) = \top \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, 1^{|F|}) \\ (\mathsf{ipk}_{\mathsf{fun}}, \mathsf{ivk}_{\mathsf{fun}}) \leftarrow \mathsf{I}_{\mathsf{fun}}(\mathsf{pp}, F) \\ z_0 \leftarrow \tilde{\mathsf{P}}(\mathsf{pp}, F) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Inductive case:

$$\Pr\left[\begin{array}{l} \Big(F(z_{i-1}) = z_i \\ \quad \wedge \ \mathsf{V}((\mathsf{ivk}_{\mathsf{fun}}, \mathsf{ivk}_{d-1}), z_0, z_{i-1}, \pi_{i-1}) = \top\Big) \\ \quad \implies \mathsf{V}((\mathsf{ivk}_{\mathsf{fun}}, \mathsf{ivk}_d), z_0, z_i, \pi_i) = \top \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, 1^{|F|}) \\ (\mathsf{ipk}_{\mathsf{fun}}, \mathsf{ivk}_{\mathsf{fun}}) \leftarrow \mathsf{I}_{\mathsf{fun}}(\mathsf{pp}, F) \\ (d, z_0, z_i, z_{i-1}, \pi_{i-1}) \leftarrow \tilde{\mathsf{P}}(\mathsf{pp}, F); \\ \mathsf{ivk}_{d-1} \leftarrow \mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp}, 1^{d-1}\right) \\ \mathsf{ivk}_d \leftarrow \mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp}, 1^{d}\right) \\ \pi_i \leftarrow \mathsf{P}(\mathsf{ipk}_{\mathsf{fun}}, z_0, z_i, (z_{i-1}, \pi_{i-1})) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Soundness. We say that the scheme is sound with depth bound $\mathsf{D}(\cdot)$ if for every function F, for every efficient adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\mathbf{Adv}^{(\mathsf{D})}_{\mathrm{IVC}\,[\mathcal{A}]}(\lambda) \leq \nu(\lambda)$$

where $\mathbf{Adv}^{(\mathsf{D})}_{\mathrm{IVC}\,[\mathcal{A}]}(\lambda) := \Pr\left[\mathsf{Exp}^{(\mathsf{D})}_{\mathrm{IVC}\,[\mathcal{A}]}(\lambda) = 1\right]$ and

$$\mathsf{Exp}^{(\mathsf{D})}_{\mathrm{IVC}\,[\mathcal{A}]}(\lambda) := \begin{cases} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, 1^{|F|}) \\ (\delta, F, z_0, z_\delta, \pi) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{ipk}_{\mathsf{fun}}, \mathsf{ivk}_{\mathsf{fun}}) \leftarrow \mathsf{I}_{\mathsf{fun}}(\mathsf{pp}, F) \\ \mathsf{ivk}_{\mathsf{dpt}} \leftarrow \mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp}, 1^{\delta}\right) \\ b := \Big(\mathsf{Verify}\left((\mathsf{ivk}_{\mathsf{fun}}, \mathsf{ivk}_{\mathsf{dpt}}), z_0, z_\delta, \pi\right) = \top \wedge \\ \qquad z_\delta \neq \underbrace{F(F(\ldots F(F(z_0))\ldots)}_{\delta \text{ times}} \wedge \delta \leq \mathsf{D}(\lambda)\Big) \\ \text{Return } b \end{cases}$$

Definition 2 (Linear and quasilinear proving time). *We say that an IVC has linear (resp. quasilinear) proving time if the running time of the incremental proving step is $O_\lambda(N)$ (resp. $\tilde{O}_\lambda(N)$) where $N := |F|$.*

Remark 1. In Section 3, we will refer to the (standard) notion of soundness above as almost-everywhere soundness (or a.e.-soundness) to distinguish it from infinitely-often soundness (or i.o.-soundness) which we formalize in Definition 4. Notice that a scheme that is a.e.-secure is also i.o.-secure.

Remark 2 (Abuse notation for IVC as algorithm tuples). In some of our results the two indexing algorithms of IVCs will play a minor role. To keep notation leaner, we will occasionally abuse notation writing IVCs as a tuple $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$(parameter generation, prover and verifier).

In some of our results we will show that a scheme Π is sound for a depth bound D conditioned to the security parameter being selected in a particular way (without any other changes to the scheme). The following definition captures this type of minor transformations on schemes.

Definition 3. *Let $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ be an IVC. We say that Π is sound for depth bound D if appropriately parameterized if there exists an efficient algorithm G' such that $(\mathsf{G}', \mathsf{P}, \mathsf{V})$ is a sound IVC for depth bound D.*

3 Soundness From Infinitely-Often Soundness in IVC

Here we show the connection between infinitely-often (i.o.) soundness and the standard (almost-everywhere, or a.e.) soundness of IVC at different depths.

Definition 4 (i.o.-security). *Let $E \subseteq \mathbb{N}$ be an infinite set, Π an IVC and $\mathsf{D}(\cdot)$ a depth bound. We say that Π satisfies i.o.-soundness (with respect to set E) for depth bound $\mathsf{D}(\cdot)$ if for every function F, for every efficient adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in E$,*

$$\mathbf{Adv}^{(\mathsf{D})}_{IVC[\mathcal{A}]}(\lambda) \leq \nu(\lambda)$$

where $\mathbf{Adv}^{(\mathsf{D})}_{IVC[\mathcal{A}]}(\lambda)$ is as in Definition 1.

Definition 5 (Constructible infinite sets and their density). *Let $E = \{\lambda_1 < \lambda_2 < \lambda_3 \dots\}$ be an infinite set of natural numbers. We say it is constructible if there exists a polynomial time TM M such that for all $\bar{\lambda} \in \mathbb{N}$, $M(1^{\bar{\lambda}}) = \max\{\lambda_i \in E : \lambda_i \leq \bar{\lambda}\}$. Let $f : \mathbb{N} \to \mathbb{N}$ be a (non-decreasing) function. We say that E is f-sparse if for all $\kappa \in \mathbb{N}$, we have that $E \cap \{\kappa, \kappa+1, \dots, f(\kappa)\} \neq \varnothing$.*

Lemma 1. *Let $E = \{\lambda_1 < \lambda_2 < \cdots\} \subseteq \mathbb{N}$ be an infinite $\left(2^{\kappa^T}\right)$-sparse set for some T with $0 < T \leq 1$, that is:*

$$\forall \kappa \in \mathbb{N} \qquad [\kappa,\, 2^{\kappa^T}] \cap E \neq \varnothing,$$

where $[a, b] := \{a, a+1, \dots, b\}$. Then for all $i \in \mathbb{N}$:

$$\lambda_i \;\geq\; \left(\log \lambda_{i+1}\right)^{1/T} - 1$$

Proof. We will prove the equivalent claim $\lambda_{i+1} \leq 2^{(\lambda_i+1)^T}$. First, apply the hypothesis on the sparsity of E with $\kappa = 1$. Since $[1, 2^{1^T}] = [1, 2]$ meets E, the minimum element satisfies $\lambda_1 \leq 2$.

Now fix $i \geq 1$. Consider the *first* interval $[\kappa, 2^{\kappa^T}]$ whose left endpoint lies strictly to the right of λ_i. This is obtained by taking

$$\kappa := \lambda_i + 1.$$

By the hypothesis there exists an element $e \in E$ with

$$\lambda_i + 1 \;\leq\; e \;\leq\; 2^{(\lambda_i+1)^T}. \qquad (*)$$

Since λ_{i+1} is, by definition, the least element of E that is strictly larger than λ_i, we have

$$\lambda_i < \lambda_{i+1} \leq e.$$

Combining this with $(*)$ yields

$$\lambda_{i+1} \leq e \leq 2^{(\lambda_i+1)^T},$$

and hence $\lambda_{i+1} \leq 2^{(\lambda_i+1)^T}$, as claimed. □

Theorem 1. *Let $E = \{\lambda_1 < \lambda_2 < \cdots\} \subseteq \mathbb{N}$ be a constructible $\left(2^{\kappa^T}\right)$-sparse set for some T with $0 < T \leq 1$. Let Π be an IVC that is i.o-sound with respect to E for depth bound $\mathsf{D}(\cdot)$. Let $d'(\cdot)$ be a depth bound. If for all $i \in \mathbb{N}$,*

$$d'\left(\lambda_{i+1} - 1\right) \leq \mathsf{D}\left(\lambda_i\right), \qquad (\blacktriangle)$$

then Π is (almost-everywhere) sound for depth bound d' if appropriately parameterized (Definition 3). The resulting proving time, verification time and proof size are like those originally in Π (up to constant factors).

Proof. We describe a scheme Π' that will match exactly Π with a minor modification at setup time. In particular, when invoked on security parameter λ, the setup of Π' will select an appropriate security parameter from E (the set of parameters that offer negligible soundness). The chosen parameter will be not greater than λ and potentially smaller, but still guaranteed to satisfy security as we will show. The setup G' of Π' works as follows:

$$\frac{\Pi'.\mathsf{G}'\left(1^{\lambda}, 1^{N}\right):}{\begin{array}{l}\bar{\lambda} \leftarrow M\left(1^{\lambda}\right) \\ \text{Return } \Pi.\mathsf{G}(1^{\bar{\lambda}}, 1^{N})\end{array}}$$

where M is as in Definition 5. The remaining algorithms of Π' directly invokes the respective algorithms of Π.

We now argue that $\Pi' = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ is almost-everywhere sound Definition 1 at d'. Consider an adversary $\mathcal{A}'$ against the soundness of Π' at depth bound d'. Notice that, by the way we picked $\bar{\lambda}$, we can conclude that we are actually running the setup algorithm on some $\lambda_i \in E$ (with $\lambda_i = \bar{\lambda}$) and such that $\lambda_i \leq \lambda$ (λ being the security parameter passed to the setup of Π').

We will now prove the following claim: let $\lambda \in \mathbb{N}$, let λ_i be the corresponding $\lambda_i \in E$ as described above, then

$$\mathbf{Adv}^{(d')}_{\mathrm{IVC}\,[\mathcal{A}']}(\lambda) \leq \mathbf{Adv}^{(\mathsf{D})}_{\mathrm{IVC}\,[\mathcal{A}']}(\lambda_i)\,.$$

This will allow us to conclude the (a.e.) soundness of Π' at d': by the i.o.-security w.r.t. set E of Π for depth bound D, we know that there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{(\mathsf{D})}_{\mathrm{IVC}\,[\mathcal{A}']}(\lambda_i) \leq \nu(\lambda_i).$$

Combining the last two inequalities with the fact that $\lambda_i \leq \lambda \implies \nu(\lambda_i) \leq \nu(\lambda)$, we can conclude that $\mathbf{Adv}^{(d')}_{\mathrm{IVC}\,[\mathcal{A}']}(\lambda) \leq \nu(\lambda)$.

Let us now prove the claim above. First observe that because of the way we picked $\bar{\lambda}$ and the definition of M, we have:

$$\bar{\lambda} = \lambda_i \leq \lambda \leq \lambda_{i+1} - 1.$$

Recall that in the soundness experiment, the adversary $\mathcal{A}'$ returns (among other items) a target statement of some depth $\delta' \leq d'(\lambda)$. By the hypothesis from (▲) we have:

$$d'(\lambda_{i+1}) \leq \mathsf{D}(\lambda_i)$$

which, together with the observation above, implies $\delta' \leq \mathsf{D}(\lambda_i)$. That is, the specific depth bound δ' targeted by $\mathcal{A}'$ is at most $\mathsf{D}(\lambda_i)$; therefore, the attack of $\mathcal{A}'$ at $\delta' \leq d'(\lambda_i)$ against depth bound d' is a valid attack against depth bound D *with the same security parameter* λ_i. Since $\lambda_i \in E$ it must be that the advantage of $\mathcal{A}'$ is bounded by $\nu(\lambda_i)$. This proves the claim and concludes the proof. □

Implications of Theorem 1

An implication of the last theorem is that if we are able to prove that a scheme Π is i.o.-secure with respect to a *very* sparse set of security parameters E (e.g., even exponentially sparse) for *some* superconstant depth bound, then it will be secure (in the standard sense) for some other superconstant depth bound. We stress that this is a very mild requirement—as a point of reference, we know that there exist NIZKs that are secure with respect to substantially less sparse sets of parameters (subexponentially sparse) from (subexponential) CDH alone [CJJQ23]. A strategy to show that Π is secure beyond constant depth is then to show that it satisfies this weaker notion of i.o-security and

then apply our results. Our next two corollaries formalize this observation; Corollary 2 provides the concrete depth bounds for which we obtain standard security (logarithmic for exponentially sparse sets, polylogarithmic for subexponentially sparse ones). Their proofs follow immediately from Theorem 1; for Corollary 1 we pick $d'(\lambda) := D(\lambda_i)$ for $\lambda \in [\lambda_i, \lambda_{i+1})$ which is non-decreasing, superconstant, and satisfies condition (▲) of Theorem 1.

Corollary 1. *If $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ is an i.o.-sound IVC at some $\omega(1)$ depth bound D then Π is sound at some $\omega(1)$ depth bound if appropriately parametrized.*

Corollary 2. *If $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ is an i.o.-sound IVC with respect to a (2^{κ^T})-sparse set E, with $0 < T \leq 1$ at depth bound D then Π is sound at depth bound $d = O(\log^a \mathsf{D})$ if appropriately parametrized, with $a := \lfloor \frac{1}{T} \rfloor \in \mathbb{N}$.*

4 On the Impossibility of Graceful Security Degradation

In this section we study the soundness degradation of IVC schemes that are secure only for computations of constant depth.

We start by introducing useful notation for presenting our results.

Definition 6 (Shortcut notation for sound/unsound depth bounds). *Let Π be an IVC scheme for deterministic computations (Definition 1). We say that a depth bound $d(\cdot)$ is "safe" for Π and write $\mathsf{safe}_\Pi(d)$ if Π is sound for depth bound d. If Π is not sound at depth bound d, we write $\neg\mathsf{safe}_\Pi(d)$. Often, whenever Π is implicit from the context, we will skip the subscript Π and simply write $\mathsf{safe}(d)$ (or $\neg\mathsf{safe}(d)$).*

The notation in Definition 6 can essentially be interpreted as follows: once fixed a particular scheme, if depth bound d is such that $\neg\mathsf{safe}(d)$, then there exists an efficient adversary, infinite set Λ and constant $e \in \mathbb{N}$ such that for all $\lambda \in \Lambda$

$$\mathbf{Adv}^{(d)}_{\mathrm{IVC}[\mathcal{A}]}(\lambda) \geq \lambda^{-e}.$$

If $\mathsf{safe}(d)$ is true then soundness holds for depth bound d and no such adversary $\mathcal{A}$, set Λ and constant e exists (the advantage of any adversary is negligible). We stress that here d is a function $d(\lambda)$ of the security parameter λ.

The following notion captures IVCs that are always unsound beyond a certain depth d (at which they are sound).

Definition 7. *Let d be a depth bound. An IVC scheme Π is said to be insecure beyond d if:*

- $\mathsf{safe}_\Pi(d)$, *and*
- *for any depth bound d' with $d' = \omega(d)$, $\neg\mathsf{safe}_\Pi(d')$.*

We now define the following useful notation. Its goal is to associate a constant $e^*_T(d)$ to each unsafe depth d that captures how badly security fails at that depth. Intuitively, it measures the inverse polynomial advantage the best adversary can obtain at that depth (expressed through its negated logarithm). For this definition to be robust, we will also parametrize it with the running time of the adversary (see also Remark 3). As an example of how to interpret e^*_T, consider time bound $\hat{T} := \lambda^2$ and depth bounds d, D with $d = o(D)$; then $e^*_{\hat{T}}(d) = 15$ and $e^*_{\hat{T}}(D) = 10$ means that the advantage of *any* adversary running in quadratic time are at most λ^{-15} (resp. λ^{-10}) for depth bounds d (resp. D).

Definition 8. *Let $T \in \text{poly}(\lambda), d \in \text{poly}(\lambda)$ such that $\neg\mathsf{safe}(d)$, we define:*

$$e^*_T(d) := \min \{e \in \mathbb{N} : \exists \text{ adversary } \mathcal{A} \text{ with } \mathsf{size}(\mathcal{A}) \leq T, \\ \text{infinite } \Lambda \subseteq \mathbb{N} : \forall \lambda \in \Lambda \ \mathbf{Adv}^{(d)}_{IVC[\mathcal{A}]}(\lambda) \geq \tfrac{1}{\lambda^e}\}$$

Note that such number is clearly well defined for unsafe depth bounds d.

Remark 3 (On the robustness of Definition 8). We parameterize $e^*_T(d)$ by the adversary's running time T. This makes both Definition 9 and Theorem 2 *stronger*: rather than showing impossibility for a single, unparameterized notion of insecurity level, we show that for *every* adversarial budget $T \in \text{poly}(\lambda)$, no infinite ladder of improving security levels exists. The parameterization is also necessary for the definition to be non-degenerate: without it, standard amplification via repetition (possible because verification is public and non-interactive) would "flatten" all insecurity levels, making graceful degradation trivially impossible for reasons unrelated to the structure of IVC. Our goal is to show that the impossibility is inherent to how IVC security interacts with depth—not an artifact of amplification. We stress that our result in Theorem 2 can also be proven for a hypothetical definition without the parameter referring to the running time.

Definition 9 (Graceful Security Degradation). *Let Π be an IVC scheme for deterministic computations (Definition 1) that is insecure at some depth bound $d' \in \text{poly}(\lambda)$. We say Π satisfies the graceful security degradation property if for every $T \in \text{poly}(\lambda), e \in \mathbb{N}$, there exists a depth bound $d^*(\cdot)$ such that $e^*_T(d^*) \geq e$. We say that Π satisfies the $\underline{T\text{-weak}}$ graceful security degradation property if, for time bound $T \in \text{poly}(\lambda)$, $\forall e \in \mathbb{N}$, there exists $d^*(\cdot)$ with $e^*_T(d^*) \geq e$.*

Remark 4 (Interpreting Definition 9). An intuition on how to interpret graceful security degradation is this: let Π be a scheme secure for some depth bound d but insecure for some other faster growing bound; then, for any desired *noticeable* soundness level λ^{-e}, there exists a depth bound "beyond" d (d is where we have negligible soundness) where we can at least have the guarantee that all adversaries will obtain, *at best*, λ^{-e} as an advantage.

We are now ready to state the main result of this section.

Theorem 2. *Let Π be an IVC scheme that is insecure beyond $O(1)$ depth. Then Π does not satisfy graceful security degradation.*

In our proof, we will actually obtain a stronger version of this result that rules out *weak* graceful security degradation for some T (notice that the regular variant implies the weak one for all bounds T). In order to prove this result we first give two useful lemmas. The first one, Lemma 2, intuitively states that graceful degradation would imply the existence of an endless ladder of increasingly secure depths: at every deeper level of the computation, attacks become progressively harder, the associated inverse polynomials become progressively smaller, and therefore the e^*_T values become progressively larger.

Lemma 2. *Let Π be an IVC scheme with T-weak graceful security degradation (Definition 9). Then there exists an infinite sequence of depth bounds $d_0, d_1, d_2, \ldots$, such that for each $i \in \mathbb{N}$:*

- $\neg\mathsf{safe}(d_i)$ *(i.e., $e^*_T(d_i) \in \mathbb{N}$);*
- $e^*_T(d_{i+1}) > e^*_T(d_i)$.

Proof. We can build this infinite sequence by the following process. Define e_0 as the exponent of the "best" (for the adversary) inverse polynomial that lower bounds the advantage of the adversary against

the scheme at *any* depth bound (of polynomial size). Let d_0 be the corresponding depth bound. That is:

$$d_0 := \underset{d \in \text{poly}(\lambda)}{\arg\min} \{e^*_T(d)\} \qquad e_0 := e^*_T(d_0)$$

Both d_0 and e_0 are guaranteed to exist because, by Definition 9, Π is not sound for some depth bound d' (and therefore $e^*_T(d') \in \mathbb{N}$ at that depth bound).

Then, for $i \geq 1$, define:

$$d_i := \underset{\substack{d \in \text{poly}(\lambda) \\ e^*_T(d) > e_{i-1}}}{\arg\min} \{e^*_T(d)\} \qquad e_i := e^*_T(d_i)$$

This sequence is well-defined since we can always find a depth bound d such that $e^*_T(d) > e_{i-1}$. If that were not the case we would be contradicting the assumption on graceful security degradation. □

The following intuitive lemma captures the fact that *insecurity does not get better*[18] *as we go deeper*. More precisely, it captures this fact: an attack that works for shallower depth automatically works for any deeper one with at least the same advantage.

Lemma 3. *Let Π be an IVC scheme and let L, R be depth bounds such that $L \preceq R$ and Π is not sound at both depth bounds, that is $\neg\mathsf{safe}(L)$ and $\neg\mathsf{safe}(R)$. Then for all bounds $T \in \text{poly}(\lambda)$, $e^*_T(L) \geq e^*_T(R)$.*

Proof. The main idea of the proof is that an attack for L will provide an attack for R with the same asymptotic advantage. Let $e := e^*_T(L)$. By Definition 8, there exists an efficient adversary $\mathcal{A}_L$ of size T and infinite set Λ_L such that for all $\lambda \in \Lambda_L, \mathbf{Adv}^{(L)}_{\text{IVC}[\mathcal{A}_L]}(\lambda) \geq \frac{1}{\lambda^e}$. Notice that at all the points $\lambda \in \Lambda_L$, a successful attack by this adversary will target depth $\delta_\lambda \leq L(\lambda)$. Because $L \preceq R$, eventually $\delta_\lambda \leq L(\lambda) \leq R(\lambda)$.

Consider the set $\Lambda_R := \Lambda_L \cap \{\lambda : L(\lambda) \leq R(\lambda)\}$. Since $L \preceq R$, the right hand side of this definition is cofinite (it includes all but finitely many natural numbers); therefore Λ_R is an infinite set. Now consider adversary $\mathcal{A}_R$ that simply runs $\mathcal{A}_L$. From the observations above we can conclude that for all $\lambda \in \Lambda_R, \mathbf{Adv}^{(R)}_{\text{IVC}[\mathcal{A}_R]}(\lambda) \geq \frac{1}{\lambda^e}$. Since there exists at least one adversary with this property for depth bound R we conclude that $e^*_T(R)$ is at most $e = e^*_T(L)$. □

In the proof of Theorem 2 we apply the following fact (proven in Appendix A).

Theorem 3. *Given any sequence of superconstant—i.e., $\omega(1)$—depth bounds $R_0, R_1, R_2, \ldots$, there always exists a depth bound $L = \omega(1)$ such that $\forall i$, $R_i \succeq L$.*

Proof of Theorem 2

Proof. Assume towards a contradiction that the security of Π degrades gracefully. If Π satisfies graceful security degradation then it satisfies its T-weak variant for all bounds $T \in \text{poly}(\lambda)$. Now consider an arbitrary bound T on the adversary size. By applying Lemma 2 there exists an infinite sequence of depth bounds $R_0, R_1, R_2, \ldots$ at which Π is unsound and such that $\lim_{i \to +\infty} e^*_T(R_i) = +\infty$ (this is simply an equivalent phrasing of the statement of the lemma). Since Π is secure at depth $O(1)$ it must be that all the depth bounds R_i in the sequence are superconstant.

By Theorem 3 in Appendix A, there always exists a function (a depth bound) L "in between" any $O(1)$ depth bound and any of the R_i-s. That is, $L = \omega(1)$ and for each i, $L \preceq R_i$. Because L is superconstant, by Definition 7, Π is not sound at L and therefore $e_L := e^*_T(L) \in \mathbb{N}$. By the hypothesis on the sequence $R_0, R_1, R_2, \ldots$, there must exists $i \geq 0$ such that $e^*_T(R_i) > e_L$. However, by invoking Lemma 3 we know that $e_L \geq e^*_T(R_i)$. Absurd. □

[18] Better for the protocol designer.

Beyond constant depth. As we mentioned in the introduction, it is possible to generalize our theorem to arbitrary depth. We first presented the statement specialized to constant depth for clarity and because it is plausibly the most relevant given the current landscape of the provable security of IVC. The proof of this more general version is exactly the same as the one of Theorem 2 with a minor tweak, which consists in invoking a more general version of the "sandwiching theorem" (Theorem 3): Theorem 4. The reason why the proof goes through is that we never used any special fact about constant depth—we only used the fact that it insecure for any depth bound $d' = \omega(d)$.

Theorem 4. *Let Π be an IVC scheme that is insecure beyond $O(d)$ depth for some depth bound d. Then Π does not satisfy graceful security degradation.*

5 General Depth Boosting Results for IVC

In this section we show to use an IVC scheme that is sound for a depth bound d in order to obtain an IVC that is sound at an asymptotically larger depth.

The first result formalizes the simple observation that by increasing the security parameter by a polynomial factor, the same IVC scheme achieves soundness for polynomially larger depth bounds. For this result to hold, however, one should start with a scheme sound at $d(\lambda) = \lambda^\epsilon$ for some $\epsilon > 0$. In the second result, we show how to overcome this limitation, showing how to turn an IVC sound for depth d into one that is sound at depth d^ρ, even for non-constant ρ.

5.1 A simple depth boosting approach for λ^ϵ depth bounds

Theorem 5. *Let Π be an IVC sound at depth bound $d = \lambda^\varepsilon$ with $\varepsilon > 0$, then for any $c \in \mathbb{N}$, Π is secure at $\mathsf{D} = \lambda^c$ if appropriately parameterized (Definition 3).*

Proof. Let $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ and let us construct another IVC scheme $\overline{\Pi} = \left(\overline{\mathsf{G}}, \mathsf{P}, \mathsf{V}\right)$ where the proving and verification algorithms are as in Π. Let $\mathsf{D} = \lambda^c$ for a constant $c \in \mathbb{N}$ and define $\rho = c/\varepsilon$ (also a constant). That is, $\rho = \log_d \mathsf{D}$. For simplicity we assume that ε and c are such that ρ is an integer (the proof can be adapted in a straightforward manner for the more general case).

Now consider this setup algorithm $\overline{\mathsf{G}}$:

$$\frac{\overline{\mathsf{G}}\left(1^{\bar{\lambda}}, 1^N\right):}{\lambda \leftarrow \bar{\lambda}^\rho}$$
$$\text{Return } \Pi.\mathsf{G}(1^\lambda, 1^N)$$

We claim that $\overline{\Pi} = \left(\overline{\mathsf{G}}, \mathsf{P}, \mathsf{V}\right)$ is sound for depth bound D. Let assume towards a contradiction that this is not the case. Then there exists efficient $\overline{\mathcal{A}}$, infinite set $\overline{\Lambda}$ and $e \in \mathbb{N}$ such that for all $\bar{\lambda} \in \overline{\Lambda}$:

$$\mathbf{Adv}^{(\mathsf{D})}_{\mathrm{IVC}\,[\overline{\mathcal{A}}]}\left(\bar{\lambda}\right) > \bar{\lambda}^{-e}.$$

Let us now build an adversary $\mathcal{A}$ against the soundness of scheme Π at depth d. When running in the soundness game of Π parametrized by λ, $\mathcal{A}$ simply forwards its security parameters to $\overline{\mathcal{A}}$ and returns its output. Notice that, if the security parameter λ is equal to $\bar{\lambda}^\rho$ for some $\bar{\lambda} \in \mathbb{N}$, then the public parameters passed as input to $\overline{\mathcal{A}}$ are distributed exactly as the ones $\overline{\mathcal{A}}$ would receive in the soundness experiment of $\overline{\Pi}$ parametrized by $\bar{\lambda}$. In other words, for such λ-s, $\mathcal{A}$ is emulating the soundness experiment of $\overline{\Pi}$ parametrized by $\bar{\lambda} = \sqrt[\rho]{\lambda}$.

We claim that adversary $\mathcal{A}$ obtains a noticeable advantage against Π for depth bound $d(\cdot)$ on the infinite set Λ defined as follows:

$$\Lambda := \{\bar{\lambda}^\rho : \bar{\lambda} \in \overline{\Lambda}\}$$

Notice that every $\lambda \in \Lambda$ is a perfect ρ-power and therefore our observation from above holds.

Recall that the IVC soundness experiment requires an adversary to output a target statement of the right depth. In particular the target statement output of $\mathcal{A}$ should be of depth $\delta(\lambda)$ at most $d(\lambda)$. Since $\mathcal{A}$ essentially returns the output of $\overline{\mathcal{A}}$, we need to make sure that the output of $\overline{\mathcal{A}}$ (when run in $\mathcal{A}$) satisfies this property. We can show that this is the case for all $\lambda \in \Lambda$ since, calling $\bar{\delta}(\cdot)$ the depth of the target statement output by $\overline{\mathcal{A}}$ as a function of its parameter, we have:

$$\delta(\lambda) = \bar{\delta}(\sqrt[\rho]{\lambda}) \leq \mathsf{D}(\sqrt[\rho]{\lambda}) = \left(d(\sqrt[\rho]{\lambda})\right)^{\rho} = \left(\lambda^{\frac{\varepsilon}{\rho}}\right)^{\rho} = \lambda^{\varepsilon} = d(\lambda).$$

We show that $\mathcal{A}$ obtains a noticeable advantage on every λ such that $\lambda = \bar{\lambda}^{\rho}$ for $\bar{\lambda} \in \overline{\Lambda}$:

$$\mathbf{Adv}^{(d)}_{\mathrm{IVC}\,[\mathcal{A}]}(\lambda) = \mathbf{Adv}^{(\mathsf{D})}_{\mathrm{IVC}\,[\overline{\mathcal{A}}]}\left(\bar{\lambda}\right) > \bar{\lambda}^{-e} = \lambda^{-e\rho}$$

where e is the exponent guaranteed by the hypothesis on $\overline{\mathcal{A}}$.

Since we have showed that there exists a constant $e' = e\rho$ such that $\mathcal{A}$ obtains advantage $\lambda^{-e'}$ on the infinite set of parameters Λ, we are contradicting the soundness of Π at depth bound d. Absurd. □

5.2 A fully general depth boosting approach

Given a scheme Π that is sound up to depth bound $d(\cdot)$, our goal is to construct a new scheme Π' that is sound up to a larger depth $D = d^{\rho}$, for some function ρ (possibly superconstant), while preserving efficiency within a polynomial bound. The only requirement on d and ρ is that $D = d^{\rho}$ remains polynomially bounded; for example, one may take $d = O(1)$ with $\rho = O(\log \lambda)$, or $d = O\left(\lambda^{\frac{1}{\log\log\lambda}}\right)$ with $\rho = O(\log\log\lambda)$, both obtaining security at polynomial D.

Our construction is best understood inductively. We define a family of schemes Π_j defined through a "boosting" construction which uses an IVC for "big jumps" and one for "small jumps":

$$\Pi_1 := \Pi, \quad \Pi_j := \mathrm{Boost}(\Pi_{\mathrm{jmp}} := \Pi_{j-1}, \Pi_{\mathrm{JMP}} := \Pi)$$

where each Π_j is sound up to depth d^j. The final scheme Π' will be Π_{ρ}. The base scheme Π_1 plays the role of the *small-jump* prover, and the original Π serves as the *big-jump* prover. Intuitively, Π_j combines the two: it uses Π_{j-1} to prove correctness for "short" segments of up to d^{j-1} steps, and Π to merge every d such segments into one larger verified segment of d^j steps.

To illustrate, consider the case $j = 2$ (see Fig. 2). The scheme Π_2 alternates between d *small jumps* (each applying F once and verified by Π) and one *big jump* (verifying $F^{(d)}$ using Π again). In this way, no proof ever involves a chain longer than d applications of F, yet the overall proof covers d^2 steps. Inductively repeating this composition, Π_3 extends security to depth d^3, and in general Π_{ρ} achieves soundness up to $D = d^{\rho}$.

Figure 3 formalizes this process. It specifies how the combined prover alternates between Π_{jmp} (for small jumps) and Π_{JMP} (for big jumps) to produce a single, consistent proof that remains valid across $D = d^{\rho}$ steps. The full theorem that follows quantifies the resulting bounds on proof size, verification time, and amortized proving time.

Remark 5. We stress that ρ is not required to be a constant for the security of the final scheme to hold. A common issue in security proofs in IVC is that we need to perform a number of reductions linear in the number of "recursive invocations" of the primitives. This can be problematic if the latter is not a constant *and* each reduction step produces a polynomial blowup (after a superconstant number of these, the final blowup is not a polynomial anymore). This is common in some extraction-based

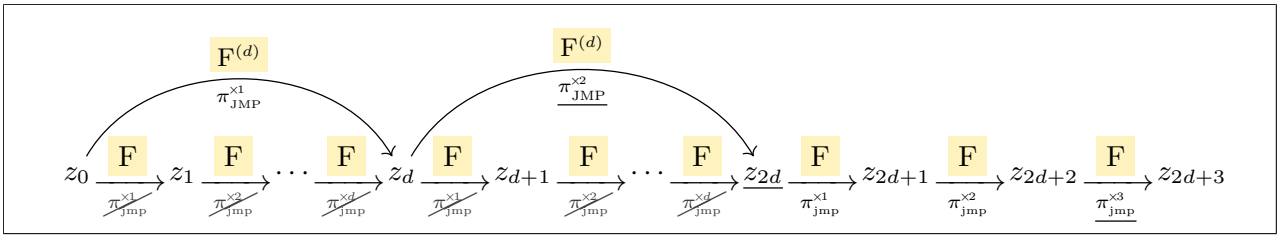


Fig. 2: Intuition about our "boosted" construction in Section 5.2 boosting a scheme secure up to depth d to one secure up to d^2. For concreteness, the figure represents the proof after $2d+3 \leq d^2$ steps. $F^{(d)}$ denotes the iteration of function F d times. A superscript $\times i$ on top of a proof denotes the fact that the is a proof obtained after i incremental proving steps. The final proof consists of the underlined items, that is $\left(\pi^{\times 2}_{\text{JMP}}, z_{2d}, \pi^{\times 3}_{\text{jmp}}\right)$. $\pi^{\times 2}_{\text{JMP}}$ certifies $z_0 \longrightarrow z_{2d}$; $\pi^{\times 3}_{\text{jmp}}$ certifies $z_{2d} \longrightarrow z_{2d+3}$. Crossed items denote those that become completely subsumed by a "big jump" (JMP) proof after this is produced.

proof systems for example. This problem never arises in our case. In fact, it would not arise even if we wanted to use it to prove a depth boosting for extractability rather than soundness alone. While, for sake of convenience and clarity, we describe our construction inductively, once "unrolled" it consists of ρ calls to the incremental prover of Π, each incrementing on a different "chain" of proofs, each of depth at most d. As a consequence, even the recursive flavor of our security analysis is not recursive at all under the hood: it could be stated alternatively as an application of the union bound on ρ events, with each event informally representing Π being broken (these ρ events intuitively correspond to the ρ different proofs that make up the final proof in the construction for Π_ρ).

Theorem 6. *Let Π be an IVC sound up to depth bound $d(\cdot)$. Let $\rho : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function such that $\mathsf{D} := d^\rho$ is polynomially bounded. Then there exists an IVC Π' sound up to depth bound D. When used to prove incrementally a function F, Π' has the following efficiency profile:*

- *(proof size)* $\mathsf{prf\text{-}size}[\Pi'] = O(\rho \cdot |\mathsf{out}_F| \cdot \mathsf{prf\text{-}size}[\Pi])$ *where* $|\mathsf{out}_F|$ *is the size in bits of the output of F.*
- *(verification time)* $\mathsf{T_V}[\Pi'] = O(\rho \cdot \mathsf{T_V}[\Pi])$.
- *(amortized proving time) The amortized proving time* $\mathsf{T}^{\mathsf{amort}}_{\mathsf{P}}[\Pi']$ *of Π' can be bound as follows:*
  - $\mathsf{T}^{\mathsf{amort}}_{\mathsf{P}}[\Pi'] = O(\rho \cdot \mathsf{T_P}[\Pi])$, *if Π has a linear time prover.*
  - $\mathsf{T}^{\mathsf{amort}}_{\mathsf{P}}[\Pi'] = \tilde{O}(\rho \cdot \mathsf{T_P}[\Pi] \cdot \text{polylog}(d))$ *if Π has a quasilinear time prover.*

Proof. **Completeness.** We can immediately observe by inspection that, if Π_{jmp} and Π_{JMP} are complete, then the resulting scheme is also complete. Proceeding by induction: for the base case $j = 2$, $\Pi_{\text{jmp}} = \Pi_{\text{JMP}} = \Pi$ so both building block schemes are complete and so is the resulting construction; for the case $j > 2$, $\Pi_{\text{JMP}} = \Pi$ is complete by the hypothesis of the theorem and $\Pi_{\text{jmp}} = \Pi_{j-1}$ is complete by the induction hypothesis and so is the resulting construction.

Soundness. We present a proof sketch. Let Π_j be the output of the j-th application of the construction in Fig. 3. Let Δ be a depth bound so that Π_{jmp} is sound up to Δ steps. We claim that Π_j is sound up to depth bound $\Delta' = d\Delta$. To prove the claim, consider an adversary $\mathcal{A}$ against the soundness of Π_j for depth Δ'. This adversary will output a target depth $\delta \leq d(\lambda)\Delta(\lambda)$, a claim $z_0 \rightsquigarrow z'$ (with $z' \neq F^{(\delta)}(z_0)$) and a proof $\pi = (z_{\text{JMP}}, \pi_{\text{jmp}}, \pi_{\text{JMP}})$. Let $\delta_{\text{JMP}}, \delta_{\text{jmp}} \in \mathbb{Z}$ such that $\delta = \delta_{\text{JMP}} \cdot \Delta + \delta_{\text{jmp}}$, with $0 \leq \delta_{\text{jmp}} < \Delta$. If π is an accepting proof then we have that both statements below hold:

Setup:

On input: λ (security parameter) and N (the size of F).

Run $\Pi_{\mathrm{JMP}}.\mathsf{G}\left(1^{\lambda}, 1^{N}\right)$ and $\Pi_{\mathrm{jmp}}.\mathsf{G}\left(1^{\lambda}, 1^{N}\right)$.

Indexing (function):

On input: public parameters $\mathsf{pp}_{\mathrm{JMP}}, \mathsf{pp}_{\mathrm{jmp}}$, function F.

Run $\Pi_{\mathrm{JMP}}.\mathsf{I}_{\mathsf{fun}}\left(\mathsf{pp}_{\mathrm{JMP}}, F^{(\Delta)}\right)$ and $\Pi_{\mathrm{jmp}}.\mathsf{I}_{\mathsf{fun}}\left(\mathsf{pp}_{\mathrm{jmp}}, F\right)$.

Indexing (depth):

On input: public parameters $\mathsf{pp}_{\mathrm{JMP}}, \mathsf{pp}_{\mathrm{jmp}}$, depth δ.

Let $\delta_{\mathrm{JMP}}, \delta_{\mathrm{jmp}} \in \mathbb{Z}$ such that $\delta = \delta_{\mathrm{JMP}} \cdot \Delta + \delta_{\mathrm{jmp}},\ (0 \leq \delta_{\mathrm{jmp}} < \Delta)$

Run $\Pi_{\mathrm{JMP}}.\mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp}_{\mathrm{JMP}}, 1^{\delta_{\mathrm{JMP}}}\right)$ and $\Pi_{\mathrm{jmp}}.\mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp}_{\mathrm{jmp}}, 1^{\delta_{\mathrm{jmp}}}\right)$.

Prover:

On input: proving keys, current index i, starting input z_0, previous input z_{i-1}, previous proof $\pi_{i-1} := \left(z_{\mathrm{JMP}}, \pi_{\mathrm{jmp}}, \pi_{\mathrm{JMP}}\right)$, next output z_i.

Let $i_{\mathrm{JMP}}, i_{\mathrm{jmp}} \in \mathbb{Z}$ such that $i = i_{\mathrm{JMP}} \cdot \Delta + i_{\mathrm{jmp}},\ (0 \leq i_{\mathrm{jmp}} < \Delta)$.

If $i_{\mathrm{jmp}} = 0$ and $i_{\mathrm{JMP}} > 0$ **then**

// π_{JMP} proved that $z_0 \rightsquigarrow z_{\mathrm{JMP}}$ in $i_{\mathrm{JMP}} - 1$ steps of $F^{(\Delta)}$; we now update it.

$\pi'_{\mathrm{jmp}} \leftarrow \perp$; $z'_{\mathrm{JMP}} \leftarrow z_i$

Run prover of Π_{JMP} on π_{JMP} for $\underbrace{z_0 \rightsquigarrow z_{\mathrm{JMP}}}_{i_{\mathrm{JMP}}-1 \text{ steps}}$ to show $\underbrace{z_0 \rightsquigarrow z'_{\mathrm{JMP}}}_{i_{\mathrm{JMP}} \text{ steps}}$

Let π'_{JMP} be the output of the previous step.

Else

// π_{jmp} proved that $z_{\mathrm{JMP}} \rightsquigarrow z_{i-1}$ in $i_{\mathrm{jmp}} - 1$ steps of F; we now update it.

$\pi'_{\mathrm{JMP}} \leftarrow \pi_{\mathrm{JMP}}$; $z'_{\mathrm{JMP}} \leftarrow z_{\mathrm{JMP}}$

Run prover of Π_{jmp} on π_{JMP} for $\underbrace{z_{\mathrm{JMP}} \rightsquigarrow z_{i-1}}_{i_{\mathrm{jmp}}-1 \text{ steps}}$ to show $\underbrace{z_{\mathrm{JMP}} \rightsquigarrow z_i}_{i_{\mathrm{jmp}} \text{ steps}}$

Let π'_{jmp} be the output of the previous step.

Output $\pi' := \left(z'_{\mathrm{JMP}}, \pi'_{\mathrm{jmp}}, \pi'_{\mathrm{JMP}}\right)$

Verifier:

On input: verification keys, proof $\pi := (z_{\mathrm{JMP}}, \pi_{\mathrm{jmp}}, \pi_{\mathrm{JMP}})$, statement (z_0, z_i).

Verify π_{JMP} for $z_0 \rightsquigarrow z_{\mathrm{JMP}}$ with Π_{JMP}. Verify π_{jmp} for $z_{\mathrm{JMP}} \rightsquigarrow z_i$ with Π_{jmp}.

Accept iff both proofs accept.

Fig. 3: Our general depth boosting construction (described inductively).

(1) π_{JMP} is accepted by $\Pi_{\text{JMP}}.\mathsf{V}$ for the claim "$z_0 \rightsquigarrow z_{\text{JMP}}$ in δ_{JMP} steps";

(2) π_{jmp} is accepted by $\Pi_{\text{jmp}}.\mathsf{V}$ for the claim "$z_{\text{JMP}} \rightsquigarrow z'$ in δ_{jmp} steps".

Since $z' \neq F^{(\delta)}(z_0)$, at least one among the claims in (1) and (2) above must be false. This means that in at least one of the two cases we can build an adversary $\tilde{\mathcal{A}}$ against the soundness game of either Π_{JMP} or Π_{jmp}. Such adversary would receive the public parameters for the scheme it's attacking, produce the remaining public parameters on its own and pass the concatenation to $\mathcal{A}$. It will then simply take the relevant portion of the output of $\mathcal{A}$ and forward it to its soundness challenger. If the advantage of $\mathcal{A}$ were negligible then so must be the advantage of $\tilde{\mathcal{A}}$ contradicting our hypothesis on the soundness of Π_{JMP} or Π_{jmp}.

We can apply the observations above to prove Π_j by induction that if Π_{jmp} is sound up to depth bound d^{j-1} and if Π_{JMP} is sound up to depth d[19] then Π_j is sound up to depth d^j. For the base case $j = 2$ we have that $\Pi_{\text{jmp}} = \Pi$ and therefore Π_{jmp} is sound up to depth $\Delta = d^{j-1} = d$. By applying the reduction above we can conclude that Π_2 is sound up to depth d^2. For the induction step, with $j > 2$, then by construction $\Pi_{\text{jmp}} = \Pi j - 1$ and by induction hypothesis Π_{j-1} is sound up to depth $\Delta = d^{j-1}$. We can apply our reduction again and conclude that Π_j is sound up to depth d^j.

Efficiency. Proof size and verification time are immediate by inspection and result from "unrolling" the invocations of $\Pi_{\text{jmp}} = \Pi_{j-1}$: each invocation adds to the final proof one intermediate output of the function F and one proof from the starting scheme Π. The final verifier will perform ρ invocations of the verifier from Π. The multiplicative factor $|\mathsf{out}_F|$ does not appear in the final verification time since the time for reading the output of F is already accounted by $\Pi.\mathsf{V}$ (because the output of F is always part of its inputs).

For proving time, let $T(N)$ be the time for the incremental prover of Π on a function of size N. We can write it as $T(N) = O(\text{poly}(\lambda) \cdot N \cdot f(N))$ where $f(N)$ is the constant 1 if Π has a linear proving time; f is polylogarithmic if Π has a quasilinear time prover. In the following, we write $g \approx h$ to denote $g = O(\text{poly}(\lambda) \cdot h)$. We can observe that the final prover in our construction in Fig. 3 for $j \geq 2$ (i.e., $\mathsf{D} = d^j$) works by running Π_{jmp} on F at every step, except for steps whose indices are (positive) multiples of d^{j-1} where it runs one "big" step of Π, that is it runs Π on $F^{(d^{j-1})}$. Therefore, its total proving time after d^j steps is:

$$T_j \approx d \cdot T_{\text{JMP}}(N') + d^j \cdot T_{\text{jmp}}^{(d^{j-1})}(N)$$

where $N' = d^{j-1}N$ (the size of $F^{(d^{j-1})}$), $T_{\text{JMP}}(N')$ is the time of $\Pi_{\text{JMP}} = \Pi$ on a function of size N', $T_{\text{jmp}}^{(d^{j-1})}(N)$ is the amortized prover time of Π_{jmp} running for d^{j-1} steps on a function of size N. Assume that there exists g such that $T_{\text{jmp}}^{(d^{j-1})}(N) \approx (j-1) \cdot N \cdot g(d, N)$, then after applying the assumption on the running time of Π, we can bound the amortized time of Π_j after d^j steps as:

$$T_j^{\text{amort}} \approx N\left(f(d^{j-1}N) + (j-1)g(d, N)\right)$$

and if, for all j, it holds that $f(d^{j-1}N) = O(g(d, N))$ then we can conclude $T_j^{\text{amort}} \approx j \cdot N \cdot g(d, N)$.

We are left with showing the implications of these observations for our two cases of interest. If Π has a linear prover then $f(N) = 1$ then consider the base case $j = 2$. We can pick $g(d, N) = 1$ and see that this provides a bound for the amortized proving time of $\Pi_{\text{jmp}} = \Pi$ after d steps (this is true by the assumption on Π having a linear prover). The observations above then take care immediately of the inductive case and we can conclude that $T_j^{\text{amort}} \approx j \cdot N$, which gives the desired bound.

If Π has a quasilinear prover then $f(N) = \text{polylog}(N)$. We can then pick $g(d, N) = \text{polylog}(d)$ and then reason analogously to how we did above and conclude that $T_j^{\text{amort}} \approx j \cdot N \cdot \text{polylog}(N) \cdot \text{polylog}(d)$, which gives the desired bound. □

[19] Notice that this is always the case by construction since Π_{JMP} is always Π.

6 Incremental Functional Commitments

In this section we describe how to build a non-extractable notion of incremental proofs with a non-deterministic flavor. The security notion we are specifically interested in is that of *evaluation binding*—in its weak and regular variant (described informally in Section 1.1)—for functional commitments whose proofs can be generated incrementally. Notice that this notion is more powerful than "existential soundness" of non-deterministic IVC: in contrast to it, evaluation binding gives the verifier the guarantee that the chain of witnesses not only exists but is somewhat encoded in a given commitment. Here we provide an overview of our approach, which we will exemplify through a specific example—*polynomial evaluation*. See also formal treatment in Appendix D .

A specific example: streaming polynomial commitments. In a *general* succinct functional commitment we are able to commit to a *datum* D and to verify, having the (short) commitment as input in place of D, claims of the form "$F(D) = y$", where F is a function. In a *polynomial* commitment, our datum D is a polynomial in some finite field[20] $\mathbb{F}$, that is D is of the form $g(X) := c_0 + c_1 X + \cdots + c_{\delta-1} X^{\delta-1}$ of degree $\delta - 1$. The functions that we want to prove are polynomial evaluations. That is, given a scalar t and a claimed output y we will prove that $F_t(g) = y$ where $F_t(g) := g(t)$. What we will build is an *evaluation binding* polynomial commitment where, for any t, we can compute an evaluation proof by working in $O_\lambda(1)$ memory and making a single pass over the polynomial g (that is the prover can access the polynomial in a streaming fashion). We do this through an IVC by describing F_t as a series of "chunks" that we prove through IVC. Our key observation is that this is possible without incurring any exponential security loss (which we would incur if we were requiring extractability out of the polynomial commitment).

Step 1: incremental evaluation. We start by describing how to chunk polynomial evaluation, that is, how to describe it as the evaluation of the same (small) building block. This is straightforward; we describe one natural approach in Fig. 4. Here we identify a "building block" function F_{cnk} that when invoked incrementally—starting from input $(0, 1)$—produces the output of $F_t(g) = g(t)$. More precisely, this incremental evaluation, computes an "augmented version" of F_t, which outputs a pair $(\mathsf{auxiliary\ info}, \mathsf{partial\ output})$. (In Fig. 4, we slightly abuse notation and refer to F_t as the evaluation in the dotted box, while technically the output of F_t corresponds only to the second element that is the output of incrementally invoking F_{cnk}) The building block function F_{cnk} intuitively involves multiplying the next monomial evaluation by respective next coefficient and summing the partial result; it also outputs monomial evaluations as partial results to compute them incrementally.

Step 2: incremental commitment opening & evaluation. Next, we describe how to build our commitment in an *incremental manner*. Let $\mathrm{H}_{\text{cr}}(\cdot)$ be a collision resistant hash function. In order to commit to g one computes the following "incremental" commitment:

$$\mathsf{cm}_g := \mathrm{H}_{\text{cr}}(c_{\delta-1} \mid\mid \mathrm{H}_{\text{cr}}(c_{\delta-2} \mid\mid \mathrm{H}_{\text{cr}}(\ldots \mid\mid \mathrm{H}_{\text{cr}}(c_0 || \perp))))$$

More precisely, the commitment algorithm computes: $h_0 := \perp$ and for all $i > 0$ $h_i := \mathrm{H}_{\text{cr}}(c_i || h_{i-1})$. The prover stores as auxiliary information the tuple of coefficients $(c_i)_i$.

We now generically extend our F_{cnk} in Fig. 4 to support a *commit-and-prove*[21] flavor in order to also include the incremental hashing above in its execution:

$$\mathsf{cp}[F_{\text{cnk}}]\Big(\overbrace{(h_{\text{rst}}, z_i)}^{\tilde{z}_i}; \overbrace{w_i}^{\tilde{w}_i}\Big) := \overbrace{(h_i, F_{\text{cnk}}(z_i, w_i))}^{\tilde{z}_{i+1}} \text{ where } h_i = \mathrm{H}_{\text{cr}}(w_i || h_{\text{rst}})$$

[20] We stress that our approach works even if we are interested in polynomial evaluation in an arbitrary ring where each element can be represented in it in $O_\lambda(1)$

[21] We use this phrase (loosely) in the sense of [CFQ19]. That is, we refer to a functionality that includes both opening a commitment and performing some other operation; this functionality is the one targeted for proving/verifying, hence the name.

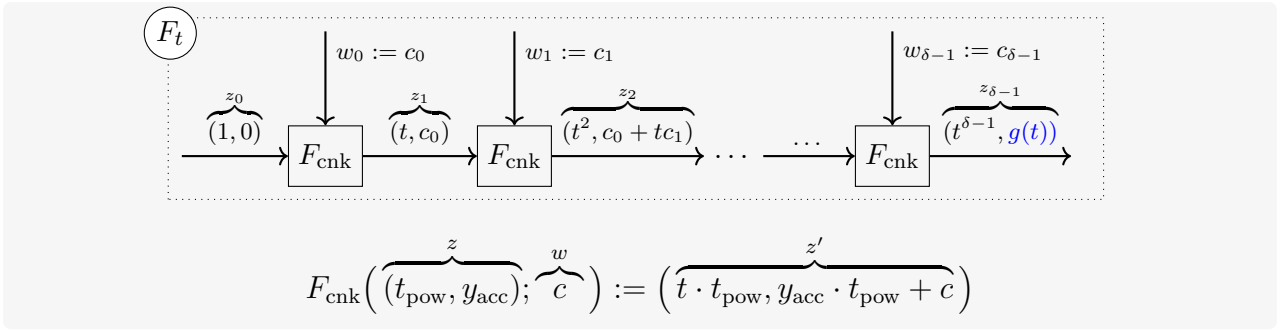


Fig. 4: "Chunk" function F_{cnk} for the evaluation of a polynomial $g(X) := \sum_i c_i X^i$ on point t (above t is implicitly embedded in the definition of F_{cnk}). The "whole" polynomial evaluation F_t is obtained through sequence of invocations of F_{cnk}.

Step 3: Knowledge-sound recursive proofs yield evaluation binding. Now that we have these building blocks (commitment and basic functionality), we describe how to actually *prove* an opening. The idea is to leverage an extractable proof system building block. For simplicity we describe (a simplified version of) the standard construction from recursive SNARKs. That is, we assume that we have a knowledge sound succinct proof system through which we are going to prove the following "chunk" recursive circuit:

$[C_{\text{rec}}]((\tilde{z}_0, \tilde{z}_{\text{out}}), (\tilde{z}_{\text{in}}, \tilde{w}, \pi_{\text{in}}))$
1. Check that $\mathsf{cp}\big[F_{\text{cnk}}\big](\tilde{z}_{\text{in}}, \tilde{w}) = \tilde{z}_{\text{out}}$
2. If $\mathsf{isSorceInput}(\tilde{z}_{\text{in}}) = 1$: Check that $\tilde{z}_{\text{in}} = \tilde{z}_0$
3. Else: Check that $\mathcal{V}((\tilde{z}_0, \tilde{z}_{\text{in}}), \pi_{\text{in}}) = 1$

Above we generically denote by $\mathsf{isSorceInput}$ an abstract check to test whether we are at a base case (see Sect. D.3 for more details). For the case of polynomial evaluation we will invoke the prover of an extractable proof system on incrementally generated inputs starting from:

$$\tilde{z}_0 := (h_0 := \bot, (1,0)) \qquad \tilde{w}_0 := (\bot, c_0) \tag{1}$$

where $(1,0)$ is the initial input z_0 as in Fig. 4 and h_0 is as defined above. On the first step, we also have $\tilde{z}_{\text{in}} = \tilde{z}_0$ and $\pi_{\text{in}} = \bot$ in order to satisfy the base case of C_{rec} above. The verifier will check the final verification using $\tilde{z}_{\text{out}} = (\mathsf{cm}_g, (t_{\text{pow}}, y))$ where in the honest case $y = g(t)$ and $t_{\text{pow}} = t^{\delta-1}$. This last element, t_{pow}, is provided as auxiliary information to the verifier and *is provided as part of the proof*. The other elements that are part of the check—cm_g and y—are already available to the verifier as input commitment and claimed output respectively.

How do we prove evaluation binding? For simplicity, in this overview, we will just show *weak* evaluation binding. We will proceed by showing that, if cm_g was computed honestly, we can maintain an invariant that is slightly stronger than weak evaluation binding, which it implies. Informally stated, this invariant is that for any polynomial degree $\delta - 1$ with $\delta \geq 1$:

($\star$) *If an efficient adversary provides $g \in \mathbb{F}_{<\delta}[X]$, scalar $t \in \mathbb{F}$, claimed output y^{adv} and an accepting proof $\left(t_{pow}^{(adv)}, \pi_\delta\right)$, with overwhelming probability we have $y^{adv} = g(t)$ and $t_{pow}^{(adv)} = t^{\delta-1}$.*

In order to prove that this invariant holds, we will assume—*towards a contradiction*—that there exists some polynomial degree bound[22] $\delta \geq 1$ for which the invariant if false. That is, for such δ,

[22] In our formal treatment in Appendix D generalize our proof to move from the specific case of degree of polynomials to "depth" (length) of the arbitrary committed input.

some efficient adversary is able with non-negligible probability to provide a polynomial $g \in \mathbb{F}_{<\delta}[X]$, accepting proof and claims such that $y^{\text{adv}} \neq g(t)$ or $t_{\text{pow}}^{(\text{adv})} \neq t^{\delta-1}$ (where the notation is the same as the one applied above).

If such δ exists, what is its minimum value? In particular, might the invariant be false for $\delta = 1$? Clearly this cannot be the case. In fact, for a polynomial of degree zero—i.e., $g(\cdot) = c_0$—we must be able to extract with overwhelming probability $\tilde{w}_0$ which will include the coefficient c_0. The latter should match the provided hash as well as the claimed evaluation y^{adv} (see Eq. (1)).

If the invariant is broken, it must necessarily be for some degree bound $\delta > 1$. Again, let us consider the *minimum* such δ. Informally, this means that for $\hat{\delta} := \delta - 1$, no efficient adversary can break invariant ($\star$) on a polynomial of degree lower than $\hat{\delta}$ (and the provided $t_{\text{pow}}^{(\text{adv})}$ is correct, i.e., $t_{\text{pow}}^{(\text{adv})} = t^{\delta-1}$). That is, no efficient adversary will be able to provide—except with negligible probability—$\hat{g} \in \mathbb{F}_{<\hat{\delta}}[X]$, $t \in \mathbb{F}, \hat{y}^{(\text{adv})} \neq \hat{g}(t)$ and proof $\pi_{\hat{\delta}}$ such that the proof accepts for the honestly generated commitment $\mathsf{cm}_{\hat{g}}$, point t and output $\hat{y}^{(\text{adv})}$.

By the assumption we made on δ there exists an efficient adversary for the weak evaluation binding game that wins the game on a polynomial in $\mathbb{F}_{<\delta}[X]$ with non-negligible probability. That is, the adversary is able to provide $g \in \mathbb{F}_{<\delta}[X]$, $t \in \mathbb{F}, y^{\text{adv}} \neq g(t)$ and a corresponding accepting proof $(t_{\text{pow}}^{(\text{adv})}, \pi_\delta)$. Observe that we can write the polynomial g as $g(X) := c^{\delta-1}X^{\delta-1} + \hat{g}(X)$ with $\hat{g}(X) := \sum_{i \leq \hat{\delta}} c_i X^i$. We stress this polynomial is in $\mathbb{F}_{<\hat{\delta}}[X]$ with $\hat{\delta} = \delta - 1$. Once we receive the proof (composed of a claimed $t_{\text{pow}}^{(\text{adv})}$ and an extractable non-interactive argument), we are able to extract:

$$h_{\text{rst}}^{(\text{adv})}, c_{\delta-1}^{(\text{adv})}, \hat{y}^{(\text{adv})}, \hat{t}_{\text{pow}}^{(\text{adv})}, \pi_{\hat{\delta}}, \text{ s.t. } \begin{cases} \mathsf{cm}_g = \mathrm{H}_{\text{cr}}(c_{\delta-1} || h_{\text{rst}}^{(\text{adv})}) \\ y^{\text{adv}} = c_{\delta-1}^{(\text{adv})} \cdot t_{\text{pow}}^{(\text{adv})} + \hat{y}^{(\text{adv})} \\ t_{\text{pow}}^{(\text{adv})} = t \cdot \hat{t}_{\text{pow}}^{(\text{adv})} \\ \pi_{\hat{\delta}} \text{ “valid proof on prev. step”} \end{cases} \tag{2}$$

We can now observe that, except with negligible probability, it must be the case that $h_{\text{rst}}^{(\text{adv})}, c_{\delta-1}^{(\text{adv})}$ are both equal to their honest counterpart (otherwise we either broke collision resistance of the hash function Eq. (2) or the extractability of the argument). Now, it is the case that either $t_{\text{pow}}^{(\text{adv})}$ is equal to the honest value $t^{\delta-1}$ or it is not. Let us first assume it is (we will later remark how to take care of the other case). Gathering all the observations so far we have that:

$$y^{\text{adv}} = c_{\delta-1}t^{\delta-1} + \hat{y}^{(\text{adv})} \text{ and } \mathsf{cm}_{\hat{g}} = h_{\text{rst}}^{(\text{adv})}$$

Since $y^{\text{adv}} \neq g(t)$, by assumption, it must be the case that $\hat{y}^{(\text{adv})} \neq \hat{g}(t)$. But now we suddenly have a way to break the invariant for $\hat{\delta} < \delta$, through the extraction step above (invoked *only once*). We can in fact wrap these values to have an adversary $\hat{\mathcal{A}}$ against weak evaluation binding for a polynomial of lower degree. This adversary would output the following:

- target polynomial $\hat{g}$;
- target evaluation point t;
- false claim $\hat{y}^{(\text{adv})}$;
- proof $(\hat{t}_{\text{pow}}^{(\text{adv})}, \pi_{\hat{\delta}})$.

Since by Eq. (2) the proof accepts (with high probability) we have just built a successful adversary against weak evaluation binding for a degree bound strictly lower than δ. This contradicts our initial assumption on δ being the minimum such bound. Absurd.

In order to account for the case where $t_{\text{pow}}^{(\text{adv})}$ is dishonest, i.e., $t_{\text{pow}}^{(\text{adv})} \neq t^{\delta-1}$, we observe that, by Eq. (2), this implies that $\hat{t}_{\text{pow}}^{(\text{adv})} \neq t^{\hat{\delta}}$. We can now, analogously to how we did above, reduce to a

different adversary that breaks our invariant for a smaller degree bound by specifically contradicting the part regarding the honesty of $t_{\mathrm{pow}}^{(\mathrm{adv})}$.

The approach above works only for constant δ (see discussion in Appendix E). In order to lift the soundness of an IFC beyond constant depth, however, we are able to invoke our result in Section 5.2.

Generalizing our approach beyond polynomial evaluation. The template above can be generalized to any functions that are nicely "chunkable", that is where we can compute them with some form of streaming access to the data. In Appendix D we generalize our construction in this sense and also slightly modify it in order to satisfy a stronger form of security (evaluation binding). While we choose to present our results for "chains" (an IVC-flavor of functional commitments), they can be adapted easily to settings where the computation is an arbitrary tree (a PCD-flavor of functional commitments). We leave formalizing the latter as future work.

Acknowledgments

We thank Giacomo Fenzi and Nick Spooner for helpful criticism on a different proof strategy we used for IFC in an earlier version of this paper. Matteo Campanelli would like to thank Mathias Hall-Andersen for a ton of useful inputs, as well as Rosario Gennaro for insightful discussions. Thanks to Wilson Nguyen and Babis Papamanthou for pointing out useful related work.

References

ABC+24. Giuseppe Ateniese, Foteini Baldimtsi, Matteo Campanelli, Danilo Francati, and Ioanna Karantaidou. Advancing scalability in decentralized storage: A novel approach to proof-of-replication via polynomial evaluation. pages 3–39, 2024.

ACDK25. Behzad Abdolmaleki, Matteo Campanelli, Quang Dao, and Hamidreza Khoshakhlagh. On the simulation-extractability of proof-carrying data. Cryptology ePrint Archive, Paper 2025/2037, 2025.

AFKR24. Noor Athamnah, Eden Florentz-Konopnicki, and Ron D. Rothblum. Rate-1 zero-knowledge proofs from one-way functions. pages 319–350, 2024.

AG25. Abtin Afshar and Rishab Goyal. Verifiable streaming computation and step-by-step zero-knowledge. Cryptology ePrint Archive, Paper 2025/251, 2025.

APKP24. Kasra Abbaszadeh, Christodoulos Pappas, Jonathan Katz, and Dimitrios Papadopoulos. Zero-knowledge proofs of training for deep neural networks. pages 4316–4330, 2024.

AR20. Shashank Agrawal and Srinivasan Raghuraman. KVaC: Key-Value Commitments for blockchains and beyond. pages 839–869, 2020.

AS24. Arasu Arun and Srinath Setty. Nebula: Efficient read-write memory and switchboard circuits for folding schemes. Cryptology ePrint Archive, Report 2024/1605, 2024.

BBBF18. Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. pages 757–788, 2018.

BBC+25. Vincenzo Botta, Simone Bottoni, Matteo Campanelli, Emanuele Ragnoli, and Alberto Trombetta. qedb: Expressive and modular verifiable databases (without SNARKs). Cryptology ePrint Archive, Paper 2025/1408, 2025.

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

BCFL23. David Balbás, Dario Catalano, Dario Fiore, and Russell W. F. Lai. Chainable functional commitments for unbounded-depth circuits. pages 363–393, 2023.

BCL+21. Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. pages 681–710, 2021.

BCMS20. Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive proof composition from accumulation schemes. pages 1–18, 2020.

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

BFG+23. David Balbás, Dario Fiore, María Isabel González Vasco, Damien Robissout, and Claudio Soriente. Modular sumcheck proofs with applications to machine learning and image processing. pages 1437–1451, 2023.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. pages 677–706, 2020.

BHI+24. Nir Bitansky, Prahladh Harsha, Yuval Ishai, Ron D. Rothblum, and David J. Wu. Dot-product proofs and their applications. pages 806–825, 2024.

BIOW20. Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. pages 776–806, 2020.

BMRS20. Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Mina: Decentralized cryptocurrency at scale, 2020. https://eprint.iacr.org/2020/352.

Cam26. Matteo Campanelli. Practical subvector commitments with optimal opening complexity. Cryptology ePrint Archive, Paper 2026/118, 2026.

CCC+25. Matteo Campanelli, Mario Carrillo, Ignacio Cascudo, Dario Fiore, Danilo Francati, and Rosario Gennaro. On the power of polynomial preprocessing: Proving computations in sublinear time, and more. Cryptology ePrint Archive, Report 2025/238, 2025.

CCG+23. Megan Chen, Alessandro Chiesa, Tom Gur, Jack O'Connor, and Nicholas Spooner. Proof-carrying data from arithmetized random oracles. pages 379–404, 2023.

CDvD+03. Dwaine E. Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh. Incremental multiset hash functions and their application to memory integrity checking. pages 188–207, 2003.

CEO22. Matteo Campanelli, Felix Engelmann, and Claudio Orlandi. Zero-knowledge for homomorphic key-value commitments with applications to privacy-preserving ledgers. pages 761–784, 2022.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. pages 55–72, 2013.

CFF+24. Matteo Campanelli, Antonio Faonio, Dario Fiore, Tianyu Li, and Helger Lipmaa. Lookup arguments: Improvements, extensions and applications to zero-knowledge decision trees. pages 337–369, 2024.

CFK24. Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. pages 132–167, 2024.

CFP25. Matteo Campanelli, Dario Fiore, and Mahak Pancholi. On composing AGM-secure functionalities with cryptographic proofs: Applications to unbounded-depth IVC and more. Cryptology ePrint Archive, Paper 2025/2086, 2025.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. pages 2075–2092, 2019.

CFT22. Dario Catalano, Dario Fiore, and Ida Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. pages 159–188, 2022.

CGSY24. Alessandro Chiesa, Ziyi Guan, Shahar Samocha, and Eylon Yogev. Security bounds for proof-carrying data from straightline extractors. pages 464–496, 2024.

CHA24. Matteo Campanelli and Mathias Hall-Andersen. Fully succinct arguments over the integers from first principles. Cryptology ePrint Archive, Report 2024/1548, 2024.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. pages 738–768, 2020.

CJJQ23. Geoffroy Couteau, Abhishek Jain, Zhengzhong Jin, and Willy Quach. A note on non-interactive zero-knowledge from CDH. pages 731–764, 2023.

CNR+22. Matteo Campanelli, Anca Nitulescu, Carla Ràfols, Alexandros Zacharakis, and Arantxa Zapico. Linear-map vector commitments and their practical applications. pages 189–219, 2022.

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

Dam89. Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

dCP23. Leo de Castro and Chris Peikert. Functional commitments for all functions, with transparent setup and from SIS. pages 287–320, 2023.

DGKV22. Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. pages 1057–1068, 2022.

DJJ+25. Pratish Datta, Abhishek Jain, Zhengzhong Jin, Alexis Korb, Surya Mathialagan, and Amit Sahai. Incrementally verifiable computation for NP from standard assumptions. In *Annual International Cryptology Conference*, pages 611–642. Springer, 2025.

FS21. Nils Fleischhacker and Mark Simkin. On publicly-accountable zero-knowledge and small shuffle arguments. pages 618–648, 2021.

GBC16. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

GN23. Irene Giacomelli and Luca Nizzardo. Filecoin proof of useful space – Technical report, 2023. https://research.protocol.ai/publications/filecoin-proof-of-useful-space/giacomelli2023.pdf.

GRWZ20. Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. pages 2007–2023, 2020.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. pages 99–108, 2011.

HLO24. Shuichi Hirahara, Zhenjian Lu, and Igor C. Oliveira. One-way functions and pKt complexity. pages 253–286, 2024.

IP01. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

KST22. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. pages 359–388, 2022.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. pages 177–194, 2010.

LGG+22. Derek Leung, Yossi Gilad, Sergey Gorbunov, Leonid Reyzin, and Nickolai Zeldovich. Aardvark: An asynchronous authenticated dictionary with applications to account-based cryptocurrencies. pages 4237–4254, 2022.

LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. pages 530–560, 2019.

LP21. Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on $\mathsf{EXP} \neq \mathsf{BPP}$. pages 11–40, 2021.

LRY16. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. pages 30:1–30:14, 2016.

LXZ21. Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. pages 2968–2985, 2021.

LXZ+24. Tianyi Liu, Tiancheng Xie, Jiaheng Zhang, Dawn Song, and Yupeng Zhang. Pianist: Scalable zkRollups via Fully Distributed Zero-Knowledge Proofs . In *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. pages 499–517, 2010.

Mar23. Daniel Marin. Nexus 1.0: Enabling verifiable computation, 2023. https://whitepaper.nexus.xyz/.

Mat23. Matter Labs. ZKsync Era: A ZK rollup for scaling Ethereum, 2023. https://docs.zksync.io/.

Mer89. Ralph C Merkle. One way hash functions and des. In *Conference on the Theory and Application of Cryptology*, pages 428–446. Springer, 1989.

MMZ25. Mary Maller, Nicolas Mohnblatt, and Arantxa Zapico. IVC in the open-and-sign random oracle model. Cryptology ePrint Archive, Paper 2025/1663, 2025.

NDC+24. Wilson D. Nguyen, Trisha Datta, Binyi Chen, Nirvan Tyagi, and Dan Boneh. Mangrove: A scalable framework for folding-based SNARKs. pages 308–344, 2024.

PP22. Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. pages 1045–1056, 2022.

PWZ+25. Zhizhi Peng, Taotao Wang, Chonghe Zhao, Guofu Liao, Zibin Lin, Yifeng Liu, Bin Cao, Long Shi, Qing Yang, and Shengli Zhang. A survey of zero-knowledge proof based verifiable machine learning. *arXiv preprint arXiv:2502.18535*, 2025.

Sta18. StarkWare Industries. Starknet: A permissionless decentralized ZK-Rollup, 2018. https://starkware.co/.

TAB+20. Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. pages 45–64, 2020.

TFZ+22. Nirvan Tyagi, Ben Fisch, Andrew Zitek, Joseph Bonneau, and Stefano Tessaro. VeRSA: Verifiable registries with efficient client audits from RSA authenticated dictionaries. pages 2793–2807, 2022.

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

VSP+17. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

WPSP24. Weijie Wang, Charalampos Papamanthou, Shravan Srinivasan, and Dimitrios Papadopoulos. Dynamic zk-SNARKs. Cryptology ePrint Archive, Report 2024/1566, 2024.

WW23. Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. pages 385–416, 2023.

WW24. Hoeteck Wee and David J. Wu. Succinct functional commitments for circuits from k-Lin. pages 280–310, 2024.

ZFZS20. Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. pages 2039–2053, 2020.

A A Useful Fact On Sequences of Functions

Before stating the next theorem, we remind the reader that we require depth bounds to have positive values and to be non-decreasing.

Theorem (Restatement of Theorem 3). *Given any sequence of superconstant—i.e., $\omega(1)$—depth bounds $R_0, R_1, R_2, \ldots$, there always exists a superconstant depth bound L such that for all i, $R_i \succeq L$.*

Proof. We will employ a diagonalizing approach. For each integer $k \geq 1$ and each $i \leq k$, since $R_i \in \omega(1)$, there exists an $N_i(k)$ such that

$$n \geq N_i(k) \quad \Rightarrow \quad R_i(n) \geq k.$$

Define an increasing sequence

$$n_0 := 0, \qquad n_k := 1 + \max\{n_{k-1},\, N_0(k), N_1(k), \ldots, N_k(k)\}.$$

Now define a non-decreasing step function $L : \mathbb{N} \to \mathbb{N}$ by

$$L(n) = k \quad \text{for } n_k \leq n < n_{k+1}.$$

Then L is unbounded, hence $L \in \omega(1)$.

Fix any i. For any $k \geq i$ and any $n \in [n_k, n_{k+1})$, we have $n \geq N_i(k)$ by definition of n_k, so

$$R_i(n) \geq k = L(n).$$

Therefore, for all $n \geq n_i$ (once $k \geq i$), we have $R_i(n) \geq L(n)$. Thus each R_i eventually dominates L. □

It is possible to generalize the theorem above to any (not necessarily constant) depth bound d.

Theorem 7. *Let d be a depth bound. Given any sequence of $\omega(d)$ depth bounds $R_0, R_1, R_2, \ldots$, there always exists a depth bound $L = \omega(d)$ such that for all i, $R_i \succeq L$.*

Proof. The proof is like the one of Theorem 3 with only the following modification: every fixed threshold k is replaced by the scaled threshold $k\,d(n)$. In particular, wherever the original argument used that $R_i(n) \geq k$ for sufficiently large n, we now use that $R_i(n) \geq k\,d(n)$; and wherever $L(n)$ was defined as the step function $L(n) = k$ on the k-th block, we instead define $L(n) = k\,d(n)$. All subsequent steps of the proof proceed identically. □

B IVC for Non-Deterministic Computations

Here we provide definitions of completeness and security (soundness and extractability) for IVC for non-deterministic computations. In this section we always consider a function $F : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ for an implicit and fixed domain $\mathcal{D}$. In our definitions we will apply the notation in Definition 14. IVC for non-deterministic computations have the same syntax has Definition 1 with the only difference that the proving algorithm will take as additional inputs local witnesses w as illustrated in the following definition.

Definition 10 (Completeness for non-deterministic IVC). *For all functions F, adversaries $\tilde{\mathsf{P}}$, and all $\lambda \in \mathbb{N}$, we have:*

Base case:

$$\Pr\left[\mathsf{V}((\mathsf{ivk}_{\mathsf{fun}},\bot),z_0,z_0,\bot)=\top \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathsf{G}(1^\lambda,1^{|F|})\\(\mathsf{ipk}_{\mathsf{fun}},\mathsf{ivk}_{\mathsf{fun}})\leftarrow\mathsf{I}_{\mathsf{fun}}(\mathsf{pp},F)\\z_0\leftarrow\tilde{\mathsf{P}}(\mathsf{pp},F)\end{array}\right]\geq 1-\mathsf{negl}(\lambda).$$

Inductive case:

$$\Pr\left[\begin{array}{l}\quad F(z_{i-1},w_{i-1})=z_i\\\wedge\ \mathsf{V}((\mathsf{ivk}_{\mathsf{fun}},\mathsf{ivk}_{d-1}),z_0,z_{i-1},\pi_{i-1})=\top\\\implies \mathsf{V}((\mathsf{ivk}_{\mathsf{fun}},\mathsf{ivk}_d),z_0,z_i,\pi_i)=\top\end{array}\;\middle|\;\begin{array}{l}\mathsf{pp}\leftarrow\mathsf{G}(1^\lambda,1^{|F|})\\(\mathsf{ipk}_{\mathsf{fun}},\mathsf{ivk}_{\mathsf{fun}})\leftarrow\mathsf{I}_{\mathsf{fun}}(\mathsf{pp},F)\\(d,z_0,z_i,w_{i-1},z_{i-1},\pi_{i-1})\leftarrow\tilde{\mathsf{P}}(\mathsf{pp},F);\\\mathsf{ivk}_{d-1}\leftarrow\mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp},1^{d-1}\right)\\\mathsf{ivk}_d\leftarrow\mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp},1^{d}\right)\\\pi_i\leftarrow\mathsf{P}(\mathsf{ipk}_{\mathsf{fun}},z_0,z_i,w_{i-1},(z_{i-1},\pi_{i-1}))\end{array}\right]\geq 1-\mathsf{negl}(\lambda).$$

Definition 11 ("Existential" (standard) soundness for non-deterministic IVC).
We say that a non-deterministic IVC is sound with depth bound $\mathsf{D}(\cdot)$ if for every function F, for every efficient adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda\in\mathbb{N}$,

$$\mathbf{Adv}^{(\mathsf{D})}_{\exists\text{-}IVC[\mathcal{A}]}(\lambda)\leq\nu(\lambda)$$

where $\mathbf{Adv}^{(\mathsf{D})}_{\exists\text{-}IVC[\mathcal{A}]}(\lambda):=\Pr\left[\mathsf{Exp}^{(\mathsf{D})}_{\exists\text{-}IVC[\mathcal{A}]}(\lambda)=1\right]$ *and*

$$\mathsf{Exp}^{(\mathsf{D})}_{\exists\text{-}IVC[\mathcal{A}]}(\lambda):=\begin{cases}\mathsf{pp}\leftarrow\mathsf{G}(1^\lambda,1^{|F|})\\(\delta,F,z_0,z_\delta,\pi)\leftarrow\mathcal{A}(\mathsf{pp})\\(\mathsf{ipk}_{\mathsf{fun}},\mathsf{ivk}_{\mathsf{fun}})\leftarrow\mathsf{I}_{\mathsf{fun}}(\mathsf{pp},F)\\\mathsf{ivk}_{\mathsf{dpt}}\leftarrow\mathsf{I}_{\mathsf{dpt}}\left(\mathsf{pp},1^{\delta}\right)\\b:=\Big(\mathsf{Verify}\left((\mathsf{ivk}_{\mathsf{fun}},\mathsf{ivk}_{\mathsf{dpt}}),z_0,z_\delta,\pi\right)=\top\ \wedge\\\qquad\nexists w_0,\ldots,w_{\delta-1}\ s.t.\ z_0\xrightarrow[w_0]{F}z_1\xrightarrow[w_1]{F}\cdots\xrightarrow[w_{\delta-2}]{F}z_{\delta-1}\xrightarrow[w_{\delta-1}]{F}z_\delta\ \wedge\\\qquad\delta\leq\mathsf{D}(\lambda)\Big)\\\textit{Return } b\end{cases}$$

Definition 12 (Knowledge soundness for non-deterministic IVC). *We say that a non-deterministic IVC is knowledge sound with depth bound $\mathsf{D}(\cdot)$ if for every function F, for every efficient adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}$ and a negligible function $\nu(\cdot)$ such that for all $\lambda\in\mathbb{N}$,*

$$\mathbf{Adv}^{(\mathsf{D})}_{\nexists\text{-}IVC[\mathcal{A},\mathcal{E}]}(\lambda)\leq\nu(\lambda)$$

where $\mathbf{Adv}^{(\mathsf{D})}_{\nexists\text{-}IVC[\mathcal{A},\mathcal{E}]}(\lambda):=\Pr\left[\mathsf{Exp}^{(\mathsf{D})}_{\nexists\text{-}IVC[\mathcal{A},\mathcal{E}]}(\lambda)=1\right]$ *and*

$$\mathsf{Exp}^{(\mathsf{D})}_{\not\vdash\text{-}IVC[\mathcal{A},\mathcal{E}]}(\lambda) := \begin{cases} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda, 1^{|F|}) \\ (\delta, F, z_0, z_\delta, \pi) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{ipk_{fun}}, \mathsf{ivk_{fun}}) \leftarrow \mathsf{I_{fun}}(\mathsf{pp}, F) \\ \mathsf{ivk_{dpt}} \leftarrow \mathsf{I_{dpt}}\left(\mathsf{pp}, 1^\delta\right) \\ (w_0, \ldots, w_{\delta-1}) \leftarrow \mathcal{E}(\mathsf{pp}) \\ b := \Big(\mathsf{Verify}((\mathsf{ivk_{fun}}, \mathsf{ivk_{dpt}}), z_0, z_\delta, \pi) = \top \ \wedge \ \delta \leq \mathsf{D}(\lambda) \ \wedge \\ \qquad\qquad z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{\delta-2}]{F} z_{\delta-1} \xrightarrow[w_{\delta-1}]{F} z'_\delta \neq z_\delta\Big) \\ \textit{Return } b \end{cases}$$

Remark 6. Notice that the predicate that defines b above is informally equivalent to stating "although the proof provided by the adversary is accepting and the depth is within the bound, we cannot successfully extract a sequence of witnesses".

C Background on SNARKs

Indexed Relation. An indexed relation $\mathcal{R}$ is a set of tuples of the form $(\mathsf{i}, \mathsf{x}, \mathsf{w})$, where i denotes the index, x the instance, and w the witness. Typically, i describes an arithmetic circuit, x is a public wire assignments, w denotes the rest of the private inputs, and $(\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$ iff the circuit determined by i outputs 1 on inputs (x, w).

Definition 13 (SNARK). *A succinct non-interactive argument of knowledge in the preprocessing model for an indexed relation $\mathcal{R}$ is a tuple of PPT algorithms $(\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V})$, defined as:*

- $\mathcal{G}(1^\lambda) \to \mathsf{pp}$*: Takes security parameter λ and outputs public parameters* pp*.*
- $\mathcal{I}(\mathsf{i}, \mathsf{pp}) \to (\mathsf{ipk}, \mathsf{ivk})$*: Takes as input an index* i *and public parameters* pp *and deterministically outputs an index-specific proving key* (ipk) *and a verification key* (ivk)*.*
- $\mathcal{P}(\mathsf{ipk}, \mathsf{x}, \mathsf{w}) \to \pi$*: Takes as input a proving key* ipk*, an instance* x*, and a corresponding witness* w*, and outputs a proof string π.*
- $\mathcal{V}(\mathsf{ivk}, \mathsf{x}, \pi) \to \top/\bot$*: Takes as input a verification key* ivk*, an instance* x*, and a proof string π, and outputs a decision $\top/\bot$.*

A SNARK should satisfy the following properties:

Completeness. *For all efficient adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c} (\mathsf{i}, \mathsf{x}, \mathsf{w}) \in \mathcal{R} \\ \implies \mathcal{V}(\mathsf{ivk}, \mathsf{x}, \pi) = \top \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda); \\ (\mathsf{i}, \mathsf{x}, \mathsf{w}) \leftarrow \mathcal{A}(\mathsf{pp}); \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{i}, \mathsf{pp}); \\ \pi \leftarrow \mathcal{P}(\mathsf{ipk}, \mathsf{x}, \mathsf{w}) \end{array}\right] = 1.$$

Knowledge Soundness. *For all efficient adversaries $\mathcal{A}$, there exists an efficient extractor $\mathcal{E}$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c} \mathcal{V}(\mathsf{ivk}, \mathsf{x}, \pi) = \top \\ \wedge (\mathsf{i}, \mathsf{x}, \mathsf{w}) \notin \mathcal{R} \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda); \\ (\mathsf{i}, \mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathsf{pp}); \\ (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}(\mathsf{i}, \mathsf{pp}); \\ \mathsf{w} \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp}, \mathsf{i}, \mathsf{x}, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Succinctness. *The running time of the verifier is bounded by a* $\mathrm{poly}(\lambda + |\mathsf{x}|)$ *and is independent of the size of the index* i*.*

Remark 7. For sake of simplicity, we consider a strong notion of succinctness. However, one can consider a a trade-off between function size and the verifier circuit size to be able to construct IVC*. The main reason is that the size of the recursive circuits we write for each iteration of computation must not grow with depth (see [BCL+21] for example).*

D Formal Treatment of Incremental Functional Commitments

This section concerns our notion of IFC, or Incremental Functional Commitment. We first describe appropriate preliminaries, then describe our model, and finally we develop and analyze our construction. Given the relatively heavy notation used in this section, we provide a table summarizing some of our conventions in Table 1.

D.1 Functions that can be described by a single building block

It is common in the IVC setting to think of the computation being proven from a *bottom-up* perspective. A building block function F (a "low" level building block) can be repeated multiple times; the final computation *emerges* as the repetition of F. In this section we will instead proceed in a *top-down* fashion since it will be handy to be able to talk about a *predetermined*, *higher level* computation $F^\star$ that we are interested in obtaining through incremental steps (e.g., polynomial evaluation). We can then think of the "*lower level*" building block F—whose iteration will result in actually computing $F^\star$—as a function of $F^\star$.

In this section we assume all functions to be such that all the items in the tuples taken as input or output belong to an implicit domain $\mathcal{D}$ of bounded size. This implies in particular, that all the "chunks", auxiliary information and other partial outputs all belong to this domain.

Definition 14 (Incremental Execution). *Let $F^\star : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ be a function over domain $\mathcal{D}$. We denote the incremental execution of $F^\star$ over initial input z_0 and local inputs $w_0, \ldots, w_{d-1}$ like so:*

$$z_0 \xrightarrow[w_0]{F} z_1 \xrightarrow[w_1]{F} \cdots \xrightarrow[w_{d-2}]{F} z_{d-1} \xrightarrow[w_{d-1}]{F} z_d$$

That is, above, $z_i := F^\star(z_{i-1}, w_{i-1})$ for each $i \in [d]$. Often we will use the graphical notation above for the special case of an incremental execution of length 1.

Informally we say that a function $F^\star$ is *chunkable* if it can be evaluated by incrementally evaluating *chunks* (through a known "chunk" function F) starting from a known base case.

Definition 15 (Chunkable functions). *Let $F^\star : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ be a function over domain $\mathcal{D}$. We say $F^\star$ chunkable if there exist efficiently computable maps* chnk *and* base *such that $F = \mathsf{chnk}(F^\star)$ and $(\mathsf{aux}_0, y_0) = \mathsf{base}(F^\star)$ are such that for all data $D = (\mathsf{chunk}_{D,0}, \ldots, \mathsf{chunk}_{D,d-1})$ we have that:*

$$(\mathsf{aux}_0, y_0) \xrightarrow[\mathsf{chunk}_{D,0}]{F} (\mathsf{aux}_1, y_1) \xrightarrow[\mathsf{chunk}_{D,1}]{F} \cdots \xrightarrow[\mathsf{chunk}_{D,d-2}]{F} (\mathsf{aux}_{d-1}, y_{d-1}) \xrightarrow[\mathsf{chunk}_{D,d-1}]{F} (\mathsf{aux}_d, y_d)$$

with $y_D = F^\star(D)$.

Examples of chunkable functions:

- For an intuition on a chunkable functions and the corresponding maps, consider the example on **polynomial evaluation** in the technical overview. There, given $F^\star = F_t$ for some t, the corresponding chunk function is the one in Fig. 4 and the base case is $(0, 1)$, incidentally this is the same regardless of the scalar t.

- Another example setting can be found from the domain of **hash functions** such as SHA-2: the function $F^\star(m)$ computes the SHA-2 digest of a message m by iterating d times a round function. The round function corresponds to F. The data D contains the chunks of the message and appropriate round constants which are the same for every message.

The goal of the auxiliary information in the definition above is to provide some incremental additional information which may be useful for the evaluation. We stress that this auxiliary information is not the same as the one output by the commitment algorithm later in Definition 17: the former has to do with the specific evaluation we are computing, whereas the latter refers to the commitment datum regardless of the function we will evaluate on it later.

The following definition, which we will use to prove evaluation binding of our construction, essentially requires a class of chunkable functions to have an incremental auxiliary input to depend only by the base case and by the number of iterations. This is a very mild requirement and is arguably satisfied by all natural incremental computations we may care about in practice. In particular it holds for polynomial evaluation from the way we compute t_{pow}.

Definition 16 (Chunk-independent auxiliary input). *Let $F^\star : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$. We say it has chunk-independent auxiliary input if there exists a function $F_{\mathsf{aux}} : \mathcal{D} \to \mathcal{D}$ such that for all data D and aux_i as in Definition 15 we have that $\mathsf{aux}_i = F_{\mathsf{aux}}(\mathsf{aux}_{i-1})$ where $(\mathsf{aux}_0, \cdot) = \mathsf{base}(F^\star)$.*

D.2 Incremental functional commitments

In a *standard* functional commitment we are able to commit to a datum D and then, on input a function $F^\star$ we can prove a correct output of $F^\star(D)$. The latter can be verified given the commitment and a short proof[23]. Below we describe a variant of functional commitments where both the commitment algorithm and the proving algorithm can work *incrementally*, that is by taking as input a "chunk" of the datum at a time.

Definition 17 (Incremental Functional Commitment (IFC)). *An incremental functional commitment consists of the following algorithms:*

- $\mathsf{G}(1^\lambda) \to \mathsf{pp}$*: outputs public parameters.*
- $\mathsf{IncCommit}\left(\mathsf{pp}, \mathsf{cm}, \mathsf{chunk}_D\right) \to \left(\mathsf{cm}', \mathsf{aux}_{\mathsf{chunk}}^{(com)}\right)$*: produces an incremental commitment from a data chunk and a previous partial commitment; it also outputs some short auxiliary information for the prover to store (asymptotically the same size as chunk_D; we will assume wlog that chunk_D is contained in this auxiliary information).*
- $\mathsf{IncProveEval}\left(\mathsf{pp}, F, \left(\hat{\mathsf{aux}}, \hat{y}\right), \pi, \mathsf{aux}_{\mathsf{chunk}}^{(com)}\right) \to \pi'$*: it proves a step of an evaluation given a chunk function, the incremental output for that step, the corresponding proof up to that point, and auxiliary information for the chunk on which to perform the next evaluation step.*
- $\mathsf{Verify}\left(\mathsf{pp}, F^\star, \mathsf{cm}^\star, y^\star, \pi^\star\right) \to \bot/\top$*: Given a chunkable function $F^\star$, a commitment to a datum, a claimed output, and a proof for it outputs whether it accepts or rejects the proof.*

Additionally, we require an IFC to have the following algorithm depthAtMost*:*

- $\mathsf{depthAtMost}(\mathsf{pp}, \mathsf{cm}, d \in \mathbb{N})$*: on input public parameter, a commitment and a depth bound d, it returns whether the commitment* cm *refers to an opening of size at most d (which entails that any computation on it will be of depth at most d). We discuss the technical reasons for having this algorithm in Remark 11.*

[23] The final succinctness of the verifier, naturally, depends on whether the representation of $F^\star(D)$ is short as well. This is not uncommon—for example, in polynomial commitments $D := f(X)$ and the function $F^\star$ is described by the evaluation point alone.

Remark 8 (On the asymmetry between prover's and verifier's inputs). In Definition 17 we let the prover take the chunk function whereas the verifier takes the resulting aggregate function. This is a purely stylistic choice (both could have taken either the chunk or the aggregate function) trying to mirror the different roles of the two algorithms: the prover acts in a purely incremental fashion—a new chunk *in*, a new proof *out*—whereas the verifier is concerned uniquely with the final result—the output of the aggregate function—regardless of how it was obtained.

Remark 9 (On indexing algorithms in IFC). In contrast to the notion in Definition 1, we did not include indexing algorithms. Recall that there are two such algorithms in Definition 1, one is for indexing the function F and one is for indexing the depth (and useful only to the verifier). Both could easily be added to our syntax (and instantiated). We kept them out in order to let this new definition be as simple as possible.

Completeness. The following definition captures the honest execution in an IFC. Its intuition is the natural one: for any sequence of data chunks, if both the commitment and proof are generated honestly (through an incremental process), then the proof will verify correctly when checked against the honest result. Formally, we say that a tuple of algorithms as in Definition 17 is complete w.r.t. a class of chunkable functions $\mathcal{C}$ and depth bound $\mathsf{D}(\cdot)$ if, for every $F^\star \in \mathcal{C}$, $\lambda \in \mathbb{N}, d \in \{1, \ldots, \mathsf{D}(\lambda)\}$, $D = (\mathsf{chunk}_{D,0}, \ldots, \mathsf{chunk}_{D,d-1})$, the following probability is overwhelming in λ:

$$\Pr\left[\mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}_d, y_d, \pi_d) = \top \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda);\ F := \mathsf{chnk}(F^\star);\ (\mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk},0}, \mathsf{cm}_0) := (\bot, \bot) \\ (\mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk},i}, \mathsf{cm}_i) \leftarrow \mathsf{IncCommit}\left(\mathsf{pp}, \mathsf{cm}_{i-1}, \mathsf{chunk}_{D,i}\right), i \in [d] \\ (\mathsf{aux}_0, y_0) \leftarrow \mathsf{base}\left(F^\star\right);\ \pi_0 \leftarrow \bot \\ \text{For } i \in [d]: \\ \pi_i \leftarrow \mathsf{IncProveEval}\left(\mathsf{pp}, F_{\mathsf{chunk}}, \left(\mathsf{aux}_{i-1}, y_{i-1}\right), \pi_{i-1}, \mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk},i}\right) \\ (\mathsf{aux}_{i-1}, y_{i-1}) \xrightarrow[\mathsf{chunk}_{D,i}]{\mathrm{F}} (\mathsf{aux}_i, y_i) \end{array}\right]$$

For completeness, we also require $\mathsf{depthAtMost}$ to be *admissible* in the following sense: with overwhelming probability for honestly generated parameters pp, any $d > 0$, adversarially provided $\mathsf{cm}, \mathsf{chunk}_D$, if $\mathsf{depthAtMost}(\mathsf{pp}, \mathsf{cm}', d) = \top$ where $(\mathsf{cm}', \ldots) \leftarrow \mathsf{IncCommit}\left(\mathsf{pp}, \mathsf{cm}, \mathsf{chunk}_D\right)$ then $\mathsf{depthAtMost}(\mathsf{pp}, \mathsf{cm}, d-1) = \top$[24].

Weak evaluation binding. We say that a tuple of algorithms as in Definition 17 satisfies weak evaluation binding w.r.t. a class of functions $\mathcal{C}$ and depth bound $\mathsf{D}(\cdot)$ if, for every efficient adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$, it holds:

$$\Pr\left[\begin{array}{l} F^\star \in \mathcal{C} \ \wedge\ y \neq F^\star(D) \ \wedge\ d \leq \mathsf{D}(\lambda) \ \wedge \\ \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y, \pi) = \top \ \wedge \end{array} \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda); \\ \left(F^\star, D := (d, \mathsf{chunk}_{D,i-1})_{i\in[d]}, y, \pi\right) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{cm} = \mathsf{Commit}^\star\left(\mathsf{pp}, D\right) \end{array}\right] \leq \nu(\lambda)$$

where above $\mathsf{Commit}^\star\left(\mathsf{pp}, D\right)$ is a shortcut notation for the honest incremental commitment generation from D, i.e., it computes the following: $(\mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk},i}, \mathsf{cm}_i) \leftarrow \mathsf{IncCommit}\left(\mathsf{pp}, \mathsf{cm}_{i-1}, \mathsf{chunk}_{D,i}\right)$ for $i \in [d]$, with $(\mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk},0}, \mathsf{cm}_0) := (\bot, \bot)$; it then outputs cm_d.

[24] Another way of seeing it is that we formally require the pair $(\mathsf{G}, \mathcal{X})$ where $\mathcal{X} := \left\{\mathsf{depthAtMost}(\mathsf{pp}, \cdot, \cdot)\right\}_{\mathsf{pp}}$ to be an *incremental computation* as defined in [CFP25, Definition 3].

Evaluation binding. We say that a tuple of algorithms as in Definition 17 satisfies evaluation binding w.r.t. a class of functions $\mathcal{C}$ and a depth bound $\mathsf{D}(\cdot)$ if, for every efficient adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds:

$$\Pr\left[\begin{array}{l} F^\star \in \mathcal{C} \ \wedge \ y \neq y' \ \wedge \ d \leq \mathsf{D}(\lambda) \ \wedge \\ \mathsf{depthAtMost}\,(\mathsf{pp}, \mathsf{cm}, d) = \top \ \wedge \\ \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y, \pi) = \top \ \wedge \\ \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y', \pi') = \top \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda); \\ (d, F^\star, \mathsf{cm}, y, y', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \nu(\lambda)$$

Remark 10 (Weak evaluation binding vs evaluation binding). Weak evaluation binding is defined similarly to evaluation binding, except that the adversary provides explicitly a datum D, a claimed output $y \neq F^\star(D)$ and a single proof π; the commitment cm is generated honestly from D. Later in this section we only prove evaluation binding both because[25] its proof is slightly simpler and mainly captured by the technical overview, but also because evaluation binding implies its weak counterpart. To see why, let us reduce to the security of evaluation binding: assume you have a successful attacker against weak evaluation binding then we can build an evaluation binding adversary—returning a false output $\tilde{y}$ together with a proof $\tilde{\pi}$ as well as the data D—by producing a proof π for the honest output $y = f(x)$; we can now return both proofs and outputs in the evaluation binding game. Notice that our definition of weak evaluation binding for IFC differs substantially from the standard ones (see e.g., [BCFL23]) in that we have the additional requirement on the depth. The reduction we just showed is such that the commitment will satisfy that property (because of the requirement from correctness).

Remark 11 (On the role of $\mathsf{depthAtMost}$*).* We require the predicate $\mathsf{depthAtMost}$ to be part of an IFC in the evaluation binding case, but not in the weak evaluation binding setting. This is for technical reasons, which we now explain. As a conscious design choice, we strived not to have the depth of the computation—i.e., the number of chunks in the committed data—explicitly appear in the view of the verifier. This was done because it seemed unnecessary: there are natural settings where the verifier is not particularly concerned with the exact depth a computation has, but just whether the committed datum (whatever its size) yields output y when evaluated on $F^\star$. For example, the incremental commitment might capture the history of a blockchain; a client may hold such a (plausibly trusted) commitment and just be concerned on how many transactions of a certain type, say, appear in such history; the length of the blockchain history may be irrelevant for such an evaluation. In the evaluation binding game, if the depth does not appear in the view of the verifier explicitly, we nonetheless need to take additional care. In that setting we intuitively need to explicitly enforce the commitment to be somewhat bound to the depth of the computation. This is because the challenger does see the opening of the commitment, and the adversary may thus be giving us a malformed commitment that could, for example, correspond to a "bottomless" opening. Introducing the predicate $\mathsf{depthAtMost}$ in our design is a way of ruling out such cases.

D.3 Our Construction

Our construction for IFC is described in Fig. 5. It follows the template we described in the technical overview; it also generalizes it. We point out that the evaluation binding case requires some additional complexities in the construction, in particular the depth counter" commitment $\mathsf{cm}_{\mathrm{dpt}}$ (see also Remark 11).

[25] At the same time we provide an explicit theorem statement (Theorem 8) to show the nuances of the fewer requirements for the weak evaluation binding case.

- $\mathsf{G}(1^\lambda)$:
  1. Sample H_{cr}
  2. Run $\mathsf{pp}_{\Pi} \leftarrow \mathcal{G}(1^\lambda)$ for the SNARK parameters.
  3. Construct index for circuit $C := C^{\lambda}_{\mathcal{V}}$.
  4. Run $(\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathcal{I}^{\theta}(C, \mathsf{pp}_{\Pi})$.
  5. Output $(\mathsf{pp}_{\Pi}, \mathsf{ipk}, \mathsf{ivk}, \mathrm{H}_{\mathrm{cr}})$.
- $\mathsf{IncCommit}\left(\mathsf{pp}, \mathsf{cm} = \left(\mathsf{cm}_D, \mathsf{cm}_{\mathrm{dpt}}\right), \mathsf{chunk}_D\right)$:
  1. Compute $\mathsf{cm}'_D \leftarrow \mathrm{H}_{\mathrm{cr}}\left(\mathsf{chunk}_D || \mathsf{cm}_D\right)$.
  2. Compute $\mathsf{cm}'_{\mathrm{dpt}} \leftarrow \mathrm{H}_{\mathrm{cr}}(\text{“one”} || \mathsf{cm}_{\mathrm{dpt}})$.
  3. Output $\left(\mathsf{cm}' := \left(\mathsf{cm}'_D, \mathsf{cm}'_{\mathrm{dpt}}\right), \mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk}} := \left(\mathsf{chunk}_D, \mathsf{cm}\right)\right)$.
- $\mathsf{IncProveEval}\left(\mathsf{pp}, F, \left(\hat{\mathsf{aux}}, \hat{y}\right), \hat{\pi}, \mathsf{aux}^{(\mathrm{com})}_{\mathsf{chunk}} := \left(\mathsf{chunk}_D, \hat{\mathsf{cm}}\right)\right)$:
  1. Define $z_{in} := \left(\hat{\mathsf{aux}}, \hat{y}, \hat{\mathsf{cm}}\right)$, $w_{loc} := \mathsf{chunk}_D$.
  2. Retrieve $\hat{\pi}_{\Pi}$ from $\hat{\pi}$ and define $\pi_{in} := \hat{\pi}_{\Pi}$.
  3. Compute $z_{out} := \mathsf{cp}\left[F\right](z_{in}, w_{loc})$.
  4. Run $\pi \leftarrow \mathcal{P}(\mathsf{ipk}, F, (\mathsf{ivk}, z_0, z_{out}), (w_{loc}, z_{in}, \pi_{in}))$.
  5. Parse z_{out} as $(\mathsf{aux}, y, \mathsf{cm})$
  6. Output (aux, π).
- $\mathsf{Verify}\left(\mathsf{pp}, F^\star, \mathsf{cm}^\star, y^\star, \pi^\star := (\mathsf{aux}, \pi_{\Pi})\right)$:
  1. Retrieve ivk from pp
  2. Define $F := \mathsf{chnk}(F^\star)$, $(\mathsf{aux}_0, z_0) := \mathsf{base}(F^\star)$, $\bar{z}_0 := (\mathsf{aux}_0, z_0, (\bot, \bot))$
  3. Define $z_{out} := (\mathsf{aux}, y^\star, \mathsf{cm}^\star)$
  4. Run $b \leftarrow \mathcal{V}(\mathsf{pp}, F, (\mathsf{ivk}, \bar{z}_0, z_{out}), \pi^\star)$.
- $\mathsf{depthAtMost}\left(\mathsf{pp}, \mathsf{cm} = \left(\mathsf{cm}_D, \mathsf{cm}_{\mathrm{dpt}}\right), d\right)$: If $\mathsf{cm}_{\mathrm{dpt}} = \bot$ output $\top$; else for $i = 1 \ldots, d$, compute $h_i \leftarrow \mathrm{H}_{\mathrm{cr}}(\text{“one”} || h_{i-1})$ (with $h_0 = \bot$) and return $\top$ if $\mathsf{cm}_{\mathrm{dpt}} = h_i$ for any $i \leq d$; otherwise return $\bot$.

Fig. 5: Our general construction for IFC (parts in blue are not required for weak evaluation binding). An empty commitment is initialized as $(\bot, \bot)$. See also Fig. 6 and Fig. 7.

Let $\mathcal{C}$ be a class of chunkable functions. Let $F^\star \in \mathcal{C}$ for a specified λ and let $F := \mathsf{chnk}(F^\star)$. We define a commit-and-prove chunk function $\mathsf{cp}\big[F\big]$ in Fig. 6. Let $\Pi = (\mathcal{G}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ be a SNARK for circuit-satisfiability. The circuit in Fig. 7 defines the recursive computation for incremental proving.

$\mathsf{cp}\big[F\big](\mathsf{pp}, z_{in}, w_{loc})$:

1. Parse z_{in} as $(\mathsf{aux}_{in}, y_{in}, \mathsf{cm}_{in})$, and w_{loc} as $\mathsf{chunk}_{D,loc}$.
2. $(\mathsf{aux}_{out}, y_{out}) := F(\mathsf{aux}_{in}, y_{in}, \mathsf{chunk}_{D,loc})$
3. $(\mathsf{cm}_{out}, \cdot) \leftarrow \mathsf{IncCommit}(\mathsf{pp}, \mathsf{cm}_{in}, \mathsf{chunk}_{D,loc})$
4. Output $(\mathsf{aux}_{out}, y_{out}, \mathsf{cm}_{out})$

Fig. 6: Commit-and-prove version of the chunk function F.

$C^\lambda_{\mathcal{V}}(\mathsf{pp}, \mathsf{cp}\big[F\big], (\mathsf{ivk}, z_0, z_{out}), (w_{loc}, z_{in}, \pi_{in}))\{$

1. Check that $\mathsf{cp}\big[F\big](z_{in}, w_{loc}) = z_{out}$.
2. Parse z_{in} as $(\cdot, \cdot, \mathsf{cm}_{in})$.
3. If $\mathsf{cm}_{in} = (\bot, \bot)$: Check that $z_{in} = z_0$.
4. Else: Check that $\mathcal{V}(\mathsf{ivk}, (\mathsf{ivk}, z_0, z_{in}), \pi_{in}) = 1$.

$\}$

Fig. 7: Recursive circuit for IFC proving/verification. Above $F^\star \in \mathcal{C}_\lambda$, $F = \mathsf{chnk}(F^\star)$ and $(\mathsf{aux}_0, y_0) = \mathsf{base}(F^\star)$. Blue parts as in Fig. 5.

Remark 12 (Efficiency). For chunk functions that can be represented in constant size, the proof size and verification time of our construction will be constant (e.g., this is the case for polynomial evaluation where the function is encoded by the scalar t on which we are performing the evaluation). In general, these metrics will grow with the size of $|F|$, depending on the application (recall that $F^\star$ is passed to the verifier only for stylistic purposes but the verifier can take as input only the chunk function F). The settings we have in mind are such that the chunk function is small compared to the committed date (e.g., the chunk function could represent the input to an LLM).

D.4 Security

Theorem 8. *Let* H_{cr} *be a collision resistant hash function and let* $\mathsf{SNARK} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *be a succinct non-interactive argument of knowledge for an indexed relation* $\mathcal{R}_{\mathsf{csat}}$*. Then, the construction* IFC *in Fig. 5 (without the parts in blue) is an incremental functional commitment satisfying completeness and weak evaluation binding for any class of chunkable functions* $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ *and for any depth bound* $\mathsf{D}(\cdot) = \mathsf{D} \in \mathbb{N}$.

The proof of the theorem above (Theorem 8) follows with very minor variations from the one for the following theorem for evaluation binding.

Theorem 9. *Let* H_{cr} *be a collision resistant hash function and let* $\mathsf{SNARK} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *be a succinct non-interactive argument of knowledge for an indexed relation* $\mathcal{R}_{\mathsf{csat}}$*. Then, the construction* IFC *in*

Fig. 5 is an incremental functional commitment satisfying completeness and evaluation binding for any class of chunkable functions $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ with chunk-independent auxiliary input (Definition 16) and for any depth bound $\mathsf{D}(\cdot) = \mathsf{D} \in \mathbb{N}$.

Proof. Completeness follows immediately by inspection. In particular, notice also that $\mathsf{depthAtMost}(\mathsf{pp}, \bot, 0) = \top$ and that the completeness requirement on the depth being "strictly increasing" is satisfied by our construction.

We now prove evaluation binding. We will specifically prove a slightly stronger claim—Claim (†)—which expresses a variant of the event in the evaluation binding game (which we changed by expressing it as a form of implication and by adding the additional requirement on aux and aux'[26]).

Before proceeding to state our claim, we will define the following: let $\mathsf{D}(\cdot) = \mathsf{D} \in \mathbb{N}$, $\mathcal{A}$ an efficient adversary and $d, \lambda \in \mathbb{N}$. We define the advantage of $\mathcal{A}$ for depth bound D, target depth d with security parameter λ as follows:

$$\mathsf{Adv}_{d,\mathcal{A}}(\lambda) := 1 - \Pr\left[\begin{array}{l} (F^\star \in \mathcal{C}_\lambda \ \wedge \ d \le \mathsf{D} \ \wedge \\ \mathsf{depthAtMost}(\mathsf{pp}, \mathsf{cm}, d) = \top \ \wedge \\ \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y, \pi) = \top \ \wedge \\ \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y', \pi') = \top) \\ \qquad \xrightarrow{\text{implies}} \\ (y = y' \ \wedge \\ \quad \mathsf{aux} = \mathsf{aux}' = \\ \qquad \underbrace{F_{\mathsf{aux}}(F_{\mathsf{aux}}(\ldots(F_{\mathsf{aux}}(\mathsf{aux}_0))\ldots))}_{d \text{ times}}) \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}(1^\lambda); \\ (F^\star, \mathsf{cm}, y, y', \\ \quad \underbrace{(\mathsf{aux}, \pi_{\mathsf{SNARK}})}_{\pi}, \underbrace{(\mathsf{aux}', \pi'_{\mathsf{SNARK}})}_{\pi'}) \leftarrow \mathcal{A}(1^d, \mathsf{pp}) \end{array}\right]$$

where F_{aux} and aux_0 are as in Definitions 15 and 16.

Claim (†). For any depth bound $\mathsf{D}(\cdot) = \mathsf{D} \in \mathbb{N}$, for any efficient adversary $\mathcal{A}$ and $d \in \mathbb{N}_{>0}$, there exists a negligible function $\nu(\cdot)$ such that for every $\lambda \in \mathbb{N}$:

$$\mathsf{Adv}_{d,\mathcal{A}}(\lambda) \le \nu(\lambda)$$

The claim above is sufficient to argue security of IFC[27]. In order to prove it we proceed by contradiction. That is, assume there exists a depth bound $\mathsf{D} \in \mathbb{N}$ such that Claim (†) is false, or, equivalently, that the following set is non-empty:

$$S_{\text{adv}} := \left\{\text{efficient } \mathcal{A} : \exists d \in \mathbb{N}_{>0}, \exists e > 0 \text{ s.t. for infinitely many } \lambda, \mathsf{Adv}_{d,\mathcal{A}}(\lambda) \ge \lambda^{-e}\right\}$$

where above the condition on e is capturing the fact that the advantage of $\mathcal{A}$ is non-negligible.
Let us now describe a related family of sets of adversaries. For $d' \ge 1$ we define:

$$S^{(d')}_{\text{adv}} := \left\{\text{efficient } \mathcal{A} : \exists e > 0 \text{ s.t. for infinitely many } \lambda, \mathsf{Adv}_{d',\mathcal{A}}(\lambda) \ge \lambda^{-e}\right\}$$

[26] This last condition is not necessary in the weak evaluation binding proof.

[27] Assume the claim is true but an IFC is insecure. Let $\mathcal{A}$ be a successful adversary against IFC security. Build $\mathcal{A}'$ against the claim's experiment as follows: on input 1^d and 1^λ, run $\mathcal{A}(1^\lambda)$ and abort if the δ output by $\mathcal{A}$ is different from d; otherwise return $\mathcal{A}$'s output. We can conclude that if the advantage of $\mathcal{A}$ is non-negligible then the advantage of $\mathcal{A}'$ is non-negligible by observing that the winning condition in the claim is weaker has weaker requirements than the one in IFC security and that D is at most polynomial (which appropriately bounds the loss from sampling d).

Clearly, $S_{\text{adv}} = \bigcup_{d'=1}^{\infty} S_{\text{adv}}^{(d')}$. By the assumption we made towards a contradiction $S_{\text{adv}} \neq \varnothing$, and therefore there exists $d' \geq 1$ such that $S_{\text{adv}}^{(d')} \neq \varnothing$. Let us denote by d^* the minimum depth for which this is the case, that is:

$$d^* := \min \left\{ d' \geq 1 : S_{\text{adv}}^{(d')} \neq \varnothing \right\}$$

The remainder of this proof proceeds in two parts. We first show that if such d^* exists then it cannot equal 1. We then show that the existence of a $d^* > 1$ leads to a contradiction.

- *Proving that* $d^* \neq 1$: We now directly prove that $S_{\text{adv}}^{(1)} = \varnothing$. For an efficient adversary $\mathcal{A}$, let us consider the event in the definition of $\mathsf{Adv}_{1,\mathcal{A}}(\lambda)$ assuming the left hand side of the implication in that event. That is, assume the output of $\mathcal{A}$ is such that

$$F^\star \in \mathcal{C}_\lambda \wedge \mathsf{depthAtMost}\left(\mathsf{pp}, \mathsf{cm}, d^*\right) = \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y, \pi) = \mathsf{Verify}(\mathsf{pp}, F^\star, \mathsf{cm}, y', \pi') = \top \quad (3)$$

  for $d^* = 1$. We can then show that $\mathcal{A} \notin S_{\text{adv}}^{(1)}$ by invoking collision resistance and the extractability of the two SNARK proofs π_{SNARK} and π'_{SNARK}. More specifically, consider $\mathsf{cm} = (\mathsf{cm}_D, \mathsf{cm}_{\text{dpt}})$. We are able to build a new KSND adversary that outputs $(\mathsf{cm}, y, \mathsf{aux}, \pi_{\mathsf{SNARK}})$ from which we can extract the tuple $(\hat{\mathsf{cm}}_D, \hat{\mathsf{cm}}_{\text{dpt}}, \hat{\mathsf{aux}}, \hat{y}, \mathsf{chunk}_D)$ such that—except with negligible probability—the following holds:

$$\begin{aligned} \hat{\mathsf{aux}} &= \mathsf{aux}_0, & \hat{\mathsf{cm}}_D &= \hat{\mathsf{cm}}_{\text{dpt}} = \bot, \\ \mathsf{cm}_D &= \mathrm{H}_{\text{cr}}(\mathsf{chunk}_D || \hat{\mathsf{cm}}_D), & \mathsf{cm}_{\text{dpt}} &= \mathrm{H}_{\text{cr}}(\text{“one”} || \hat{\mathsf{cm}}_{\text{dpt}}), \\ (\hat{\mathsf{aux}}, \hat{y}) &\xrightarrow[\mathsf{chunk}_D]{\mathrm{F}} (\mathsf{aux}, y), & \hat{y} &= y_0 \end{aligned}$$

  Above, we can observe $\hat{\mathsf{cm}}_D = \hat{\mathsf{cm}}_{\text{dpt}} = \bot$ by inspecting the base case of our recursive verification (in particular the definition of $\hat{z}_0$ at line 2 in Fig. 5 and line 3 in Fig. 7) and then invoking collision resistance.

  Similarly, we can extract appropriate preimages from the second proof π'. In particular we can extract chunk'_D and since $\mathsf{cm}_D = \mathrm{H}_{\text{cr}}(\mathsf{chunk}_D || \bot) = \mathrm{H}_{\text{cr}}(\mathsf{chunk}'_D || \bot)$ we can conclude $\mathsf{chunk}_D = \mathsf{chunk}'_D$. This in turn implies $y = y'$ and $\mathsf{aux} = \mathsf{aux}' = F_{\mathsf{aux}}(\mathsf{aux}_0)$—the right hand side of the implication in the event in $\mathsf{Adv}_{1,\mathcal{A}}(\lambda)$—holds with overwhelming probability. This means that $\mathsf{Adv}_{1,\mathcal{A}}(\lambda)$ is negligible and, equivalently, that $\mathcal{A} \notin S_{\text{adv}}^{(1)}$.

- *Reaching a contradiction from the existence of* $d^* > 1$. By definition of d^* we know that $S_{\text{adv}}^{(d^*-1)} = \varnothing$. By our assumption, in contrast, $S_{\text{adv}}^{(d^*)}$ is not. Now consider an adversary $\mathcal{A}^* \in S_{\text{adv}}^{(d^*)}$. We will show a contradiction by showing an adversary $\hat{\mathcal{A}} \in S_{\text{adv}}^{(d^*-1)}$, thus violating the assumption on $S_{\text{adv}}^{(d^*)}$ being empty. This new adversary $\hat{\mathcal{A}}$ works as follows:
  - Run $\mathcal{A}^*$, which outputs $\left(F^\star, \mathsf{cm} = (\mathsf{cm}_D, \mathsf{cm}_{\text{dpt}}), y, y', \pi := (\mathsf{aux}, \pi_{\mathsf{SNARK}}), \pi' := (\mathsf{aux}', \pi'_{\mathsf{SNARK}})\right)$.
  - Extract[28] from π_{SNARK} the tuple $(\hat{\mathsf{cm}}_D, \hat{\mathsf{cm}}_{\text{dpt}}, \hat{\mathsf{aux}}, \hat{y}, \mathsf{chunk}_D, \hat{\pi}_{\mathsf{SNARK}})$.
  - Extract from π'_{SNARK} the tuple $(\hat{\mathsf{cm}}'_D, \hat{\mathsf{cm}}'_{\text{dpt}}, \hat{\mathsf{aux}}', \hat{y}', \mathsf{chunk}'_D, \hat{\pi}'_{\mathsf{SNARK}})$.
  - Output $\left(F^\star, \hat{\mathsf{cm}} = (\hat{\mathsf{cm}}_D, \hat{\mathsf{cm}}_{\text{dpt}}), \hat{y}, \hat{y}', \hat{\pi} := (\hat{\mathsf{aux}}, \hat{\pi}_{\mathsf{SNARK}}), \hat{\pi}' := (\hat{\mathsf{aux}}', \hat{\pi}'_{\mathsf{SNARK}})\right)$.

[28] This extraction step is analogous to how we extracted in the proof showing $d^* \neq 1$ and is a straightforward invocation of knowledge soundness: we define an adversary that runs $\mathcal{A}^*$ and outputs π_{SNARK} and a target public input which includes cm and $F^\star$; by extractability there exists an extractor for this last adversary. This is the machine invoked at this step. We also refer the reader to the remainder of this proof and to Fig. 8.

We claim that the advantage of $\mathcal{A}^*$ is related to that of $\hat{\mathcal{A}}$ as follows. For any $\lambda \in \mathbb{N}$:

$$\mathsf{Adv}_{d^*,\mathcal{A}^*}(\lambda) \leq \mathsf{Adv}_{d^*-1,\hat{\mathcal{A}}}(\lambda) + \nu^*(\lambda) \tag{4}$$

where ν^* is a negligible function (on which we will expand later). We show why Eq. (4) holds towards the end of this proof; meanwhile, the reader can intuitively think of this being true because the $\hat{\mathcal{A}}$ is able to be successful (in the game in the definition of $\mathsf{Adv}_{d^*-1,\hat{\mathcal{A}}}(\lambda)$) as frequently as $\mathcal{A}^*$ is (in its respective game) as long as some rare events—whose probability is bounded by ν^*—do not occur.

We now show the implication of Eq. (4) holding. Note that since we assumed that $\mathcal{A}^* \in S^{(d^*)}_{\text{adv}}$ then its advantage $\mathsf{Adv}_{d^*,\mathcal{A}^*}(\lambda)$ is greater than the inverse polynomial $\frac{1}{\lambda^e}$ for infinitely many values of λ for some constant $e > 0$. It follows that for infinitely many values of λ:

$$\begin{aligned}
\mathsf{Adv}_{d^*-1,\hat{\mathcal{A}}}(\lambda) &\geq \mathsf{Adv}_{d^*,\mathcal{A}^*}(\lambda) - \nu^*(\lambda) && (5)\\
&\geq \frac{1}{\lambda^e} - \nu^*(\lambda) && (6)\\
&> \frac{1}{\lambda^e} - \frac{1}{\lambda^{e+1}} && (7)\\
&\geq \frac{1}{\lambda^{e'}} && (8)
\end{aligned}$$

where above e' is an arbitrary value such that $e' > e$ and we applied these observations: Eq. (5) follows directly by a simple rearrangement of the terms in Eq. (4); Eq. (6) follows by the definition of $S^{(d^*)}_{\text{adv}}$ and from the observations above; Eq. (7) follows by the fact that $\nu^*(\lambda)$ is a negligible quantity and is eventually smaller than any inverse polynomial (in this case $\frac{1}{\lambda^{e+1}}$); Eq. (8) follows by the fact that, for $e' > e$, eventually $\frac{1}{\lambda^{e'}} \leq \frac{1}{\lambda^e} - \frac{1}{\lambda^{e+1}}$.

We just showed that there exists $e' > 0$ such that, for infinite values of λ, $\mathsf{Adv}_{d^*-1,\hat{\mathcal{A}}}(\lambda) > \frac{1}{\lambda^{e'}}$. We can thus apply the definition of $S^{(d^*-1)}_{\text{adv}}$ and conclude that $\hat{\mathcal{A}}$ is an element of $S^{(d^*-1)}_{\text{adv}}$, a set that was empty because of our assumption. Absurd.

We now expand on why the bound in Eq. (4) holds. This is a straightforward application of the union bound and the observation that, if $\hat{\mathcal{A}}$ is not successful, then we should be able to break knowledge soundness of the underlying SNARK, collision resistance of the hash function or admissibility of $\mathsf{depthAtMost}$. That is, we can express function $\nu^*(\lambda)$ in Eq. (4) as the sum of negligible functions $\nu_{\text{ksnd}}(\lambda) + \nu_{\text{crhf}}(\lambda) + \nu_{\text{adm}}(\lambda)$. We now provide additional details on these reductions, as well as how they relate to $\hat{\mathcal{A}}$ being successful.

Since later we want to argue about the relationship between $\mathsf{Adv}_{d^*-1,\hat{\mathcal{A}}}(\lambda)$ and $\mathsf{Adv}_{d^*,\mathcal{A}^*}(\lambda)$, we consider the outputs of $\mathcal{A}^*$ that are successful (see definition of Adv): that is, they satisfy Eq. (3) *but* $y \neq y'$ or one among aux and aux' does not equal the iterated evaluation of F_{aux} starting from aux_0.

By Eq. (3) the proofs output of $\mathcal{A}^*$ verify and we can invoke the KSND property of SNARK. We can then deduce properties of the elements that $\hat{\mathcal{A}}$ extracted from proofs π and π'. In Fig. 8 we list these elements together with the relations that hold between them and the rest of the inputs. Such relations hold unless extraction fails (captured by the negligible function ν_{ksnd} in the definition above).

Then, by reducing to collision resistance, we can conclude that, except with negligible probability ν_{crhf}, several of the elements in Fig. 8 must be equal to each other. In particular, by putting together a) and a') in Fig. 8 we can conclude that $\mathsf{chunk}_D = \mathsf{chunk}'_D$ and $\hat{\mathsf{cm}}_D = \hat{\mathsf{cm}}'_D$. By the same token, we can argue that $\hat{\mathsf{cm}}_{\text{dpt}} = \hat{\mathsf{cm}}'_{\text{dpt}}$ with overwhelming probability.

We can now start arguing that the relations among extracted elements we observed in Fig. 8 have direct consequences for the advantage $\mathsf{Adv}_{d^*-1,\hat{\mathcal{A}}}(\lambda)$ of $\hat{\mathcal{A}}$. For that, we start observing that, except with negligible probability ν_{adm}, it must be the case that $\mathsf{depthAtMost}(\mathsf{pp}, \hat{\mathsf{cm}}, d^* - 1) = \top$. This follows by the admissibility property of $\mathsf{depthAtMost}$ (see completeness definition). This is because cm is such that $\mathsf{depthAtMost}(\mathsf{pp}, \mathsf{cm}, d^*) = \top$ and is incrementally derived from $\hat{\mathsf{cm}}$—i.e., $\mathsf{cm} = \mathsf{IncCommit}(\mathsf{pp}, \hat{\mathsf{cm}}, \mathsf{chunk}_D)$ (recall also $\mathsf{chunk}_D = \mathsf{chunk}'_D$). Therefore, by admissibility of $\mathsf{depthAtMost}$, it must hold, except with negligible probability, that $\mathsf{depthAtMost}(\mathsf{pp}, \hat{\mathsf{cm}}, d^* - 1) = \top$.

By the assumption on the proofs (see d) and d') in Fig. 8) we can conclude that the proofs will be valid against commitment $\hat{\mathsf{cm}}$. This, together with our observation on $\mathsf{depthAtMost}$ above, show how our assumptions so far imply that the output of $\hat{\mathcal{A}}$ satisfy Eq. (3) (for $d^* - 1$).

To finally show $\hat{\mathcal{A}}$ is successful, we are left with arguing that $\hat{y} \neq \hat{y}'$ or one among $\hat{\mathsf{aux}}, \hat{\mathsf{aux}}'$ is incorrect. This follows by our assumption on the output of $\mathcal{A}^*$ being successful and by Fig. 8. In fact, assume by contradiction that $\hat{\mathsf{aux}} = \hat{\mathsf{aux}}' = \underbrace{F_{\mathsf{aux}}(F_{\mathsf{aux}}(\dots(F_{\mathsf{aux}}(\mathsf{aux}_0))\dots)}_{d^*-1 \text{ times}}$ and that $\hat{y} = \hat{y}'$.

From c) ad c') in Fig. 8 we know that the (at least partly *incorrect*) $y, y', \mathsf{aux}, \mathsf{aux}'$ are *correctly* computed through F and from the same value chunk_D as local input. Then, it could not clearly be the case that $y \neq y'$ or $\mathsf{aux} \neq \mathsf{aux}'$. It must be the case that $\mathsf{aux} = \mathsf{aux}'$ and that these two are different from the correct evaluation of F_{aux}, but this contradicts Definition 16 since the first output of F must be equal to the output of F_{aux}. We can thus conclude that Eq. (4) holds and that the advantage of $\hat{\mathcal{A}}$ is negligible close to that of $\mathcal{A}^*$.

Elements extracted through π_{SNARK}: $(\hat{\mathsf{cm}}_D, \hat{\mathsf{cm}}_{\text{dpt}}, \hat{\mathsf{aux}}, \hat{y}, \mathsf{chunk}_D, \hat{\pi}_{\mathsf{SNARK}})$

a) $\mathsf{cm}_D = \mathrm{H}_{\text{cr}}(\mathsf{chunk}_D || \hat{\mathsf{cm}}_D)$,

b) $\mathsf{cm}_{\text{dpt}} = \mathrm{H}_{\text{cr}}(\text{“one”} || \hat{\mathsf{cm}}_{\text{dpt}})$,

c) $(\hat{\mathsf{aux}}, \hat{y}) \xrightarrow[\mathsf{chunk}_D]{\mathrm{F}} (\mathsf{aux}, y)$

d) proof $\hat{\pi}_{\mathsf{SNARK}}$ verifies on respective input for previous step

Elements extracted through π'_{SNARK}: $(\hat{\mathsf{cm}}'_D, \hat{\mathsf{cm}}'_{\text{dpt}}, \hat{\mathsf{aux}}', \hat{y}', \mathsf{chunk}'_D, \hat{\pi}'_{\mathsf{SNARK}})$

a') $\mathsf{cm}_D = \mathrm{H}_{\text{cr}}(\mathsf{chunk}'_D || \hat{\mathsf{cm}}'_D)$,

b') $\mathsf{cm}_{\text{dpt}} = \mathrm{H}_{\text{cr}}(\text{“one”} || \hat{\mathsf{cm}}'_{\text{dpt}})$,

c') $(\hat{\mathsf{aux}}', \hat{y}') \xrightarrow[\mathsf{chunk}'_D]{\mathrm{F}} (\mathsf{aux}', y')$

d) proof $\hat{\pi}'_{\mathsf{SNARK}}$ verifies on respective input for previous step

Fig. 8: Elements extracted by $\hat{\mathcal{A}}$ and the relation among them. Part of the proof of Theorem 9.

We showed that negating Claim (†) leads to a contradiction. We are left with observing that Claim (†) directly imply evaluation binding. This concludes the proof. □

E On the Induction-like Proof for IFC Security in the Main Text

The approach outline at the end of Section 6 works only for constant δ. The reason is that it can be seen as a proof by induction with potentially a too large adversarial blow-up. Proof by induction

Notation	Description/Intuition
$D = (\mathsf{chunk}_{D,0}, \ldots, \mathsf{chunk}_{D,d-1})$	Data split into d chunks. D is the opening of the (incrementally generated) commitment in IFC.
$F^\star$	A specific function (from the chunkable class $\mathcal{C}$). Example: polynomial evaluation.
$y = F^\star(D)$	Example syntax for invoking F. For instance, y can be the polynomial evaluation output and D, the data, a polynomial whose evaluation we want to prove.
$F = \mathsf{chnk}(F^\star)$	The "chunk" function corresponding to $F^\star$. Example: the partial evaluation function in Fig. 4.
$(\mathsf{aux}_{i-1}, y_{i-1}) \xrightarrow[\mathsf{chunk}_D]{\mathsf{F}} (\mathsf{aux}_i, y_i)$	Example notation for F. Equivalent to $(\mathsf{aux}_i, y_i) = F((\mathsf{aux}_{i-1}, y_{i-1}); \mathsf{chunk}_D)$.
$\mathsf{base}(F^\star)$	Returns the initial input (base case) (aux_0, y_0) for function $F^\star$.
F_{aux}	Expresses the computation of the auxiliary information aux above as a function of previous auxiliary information alone (see Definition 16). Defined for technical reasons and not required for the weak evaluation binding case.
$\mathsf{cp}\left[F\right]$	Commit-and-prove version of chunk function F, which it augments with digest-related computations.
$\mathsf{aux}_{\mathsf{chunk}}^{(\mathrm{com})}$	Additional auxiliary information required stored by prover in order to generate IFC proofs.
cm_D	Incremental digest for data.
$\mathsf{cm}_{\mathrm{dpt}}$	Incremental digest for depth counter.
$\mathsf{depthAtMost}(\mathsf{pp}, \mathsf{cm}, d)$	A predicate required as part of IFC for technical reasons. Intuitively, it is true for a commitment cm that is the digest of data with at most d chunks. This is not required for the weak evaluation binding case. See also completeness definition of IFC and Remark 11.

Table 1: Summary of notation for Appendix D.

works by explicitly showing the base cases to be true, and then showing the inductive case that for every $\delta - 1$, *if* the statement holds for $\delta - 1$, then it holds for δ. Retracing our steps above, we can argue that assuming the invariant holds at $\delta - 1$, then it holds for δ: given a PPT adversary $\mathcal{A}$ that breaks the invariant at δ, we can produce another adversary that breaks the invariant at $\delta - 1$ *and* such that $|\mathcal{A}'| = \mathrm{poly}(|\mathcal{A}|)$, that is the size of $\mathcal{A}'$ is polynomially related to that of $\mathcal{A}$. If the invariant holds at $\delta - 1$, i.e. there is no PPT adversary breaking this invariant, then there cannot be such a $\mathcal{A}'$ constructed above. However, we must now argue the *if* clause actually holds true to be able to break the assumption. To do so, we could claim that $\mathcal{A}'$ in turn yields an adversary $\mathcal{A}''$ (for $\delta - 2$), and this an adversary $\mathcal{A}'''$, ... till we reach an *if* clause we know is true. In the above, this happens to be the base case, since we can show an explicit adversary breaking some cryptographic property. Unfortunately, this chain of machines has a polynomial blowup at every step and if we continue all the way to the base case, the chain will not terminate with a polynomial size adversary: we get $|\mathcal{A}| = \exp(|\mathcal{A}_0|)$, where $\mathcal{A}_0$ is the adversary against invariant in the base case. In order to lift the soundness of our scheme beyond constant depth, we invoke the result in Section 5.2.

F Extension of Main Results to Other Soundness Notions

In the main body, Theorems 1 and 4 to 6 were stated and proved for the deterministic soundness of IVC (Definition 1). As discussed in Section 1.2, all four results extend to both *existential soundness* of non-deterministic IVC (Definition 11) and *evaluation binding* of Incremental Functional Commitments (as defined in Appendix D). In this appendix we state the corresponding theorem variants and explain why the proofs carry over.

Setup and notation. We write Σ to denote either a non-deterministic IVC or an IFC, equipped with a security notion $\mathfrak{S}$ that is either existential soundness or evaluation binding, respectively. For an efficient adversary $\mathcal{A}$ and depth bound D, we write

$$\mathbf{Adv}^{(D)}_{\mathfrak{S}[\mathcal{A}]}(\lambda)$$

for the corresponding advantage: concretely, $\mathbf{Adv}^{(D)}_{\exists\mathrm{HVC}\,[\mathcal{A}]}(\lambda)$ (as in Definition 11) or the evaluation binding advantage (as in Appendix D). In both cases, the depth bound D enters the security experiment exclusively through the constraint "$\delta \le D(\lambda)$" on the adversary's output.

We now generalize the definitions from Section 4 to our setting. The safe/unsafe notation (Definition 6), the notion of being *insecure beyond* a depth bound (Definition 7), the insecurity level e^*_T (Definition 8), and graceful security degradation (Definition 9) are defined identically for $\mathfrak{S}$, simply by replacing every occurrence of $\mathbf{Adv}^{(d)}_{\mathrm{IVC}\,[\mathcal{A}]}(\lambda)$ with $\mathbf{Adv}^{(d)}_{\mathfrak{S}[\mathcal{A}]}(\lambda)$. For instance, the generalized insecurity level is:

Definition 18 (Insecurity level for $\mathfrak{S}$). *Let $T \in \mathrm{poly}(\lambda)$, $d \in \mathrm{poly}(\lambda)$ such that $\neg\mathsf{safe}_{\mathfrak{S}}(d)$. We define:*

$$e^*_{T,\mathfrak{S}}(d) := \min\{e \in \mathbb{N} : \exists \text{ adversary } \mathcal{A} \text{ with } \mathsf{size}(\mathcal{A}) \le T, \\ \text{infinite } \Lambda \subseteq \mathbb{N} : \forall \lambda \in \Lambda\ \mathbf{Adv}^{(d)}_{\mathfrak{S}[\mathcal{A}]}(\lambda) \ge \tfrac{1}{\lambda^e}\}$$

Graceful security degradation (Definition 9) generalizes analogously: a scheme satisfies it (w.r.t. $\mathfrak{S}$) if for every $T \in \mathrm{poly}(\lambda)$ and every $e \in \mathbb{N}$, there exists a depth bound $d^*(\cdot)$ with $e^*_{T,\mathfrak{S}}(d^*) \ge e$. Since the notion $\mathfrak{S}$ is always clear from context in the statements below, we will simply write $e^*_T(d)$ and $\mathsf{safe}(d)$.

F.1 Theorem statements

Each of the following four theorems simultaneously captures the extension to both existential soundness and evaluation binding.

Theorem 10 (Analog of Theorem 1). *Let $E = \{\lambda_1 < \lambda_2 < \cdots\} \subseteq \mathbb{N}$ be a constructible (2^{κ^T})-sparse set for some T with $0 < T \le 1$. Let Σ be a non-deterministic IVC (resp. IFC) that is i.o.-existentially-sound (resp. i.o.-evaluation-binding) with respect to E for depth bound $\mathsf{D}(\cdot)$. Let $d'(\cdot)$ be a depth bound satisfying, for all $i \in \mathbb{N}$,*

$$d'\left(\lambda_{i+1} - 1\right) \le \mathsf{D}\left(\lambda_i\right). \qquad (\blacktriangle)$$

Then Σ is (almost-everywhere) existentially sound (resp. evaluation binding) for depth bound d' if appropriately parameterized (Definition 3). The resulting proving time, verification time and proof size are like those originally in Σ (up to constant factors).

Analogs of Corollary 1 and Corollary 2 follow immediately from Theorem 10.

Theorem 11 (Analog of Theorem 4). *Let d be a depth bound and let Σ be a non-deterministic IVC (resp. IFC) that is insecure (in the existential soundness, resp. evaluation binding, sense) beyond $O(d)$ depth. Then Σ does not satisfy graceful security degradation (Definition 9, instantiated with the corresponding advantage as in Definition 18).*

Theorem 12 (Analog of Theorem 5). *Let Σ be a non-deterministic IVC (resp. IFC) satisfying existential soundness (resp. evaluation binding) at depth bound $d = \lambda^\varepsilon$ with $\varepsilon > 0$. Then for any $c \in \mathbb{N}$, Σ is existentially sound (resp. evaluation binding) at $\mathsf{D} = \lambda^c$ if appropriately parameterized (Definition 3).*

Theorem 13 (Analog of Theorem 6). *Let Σ be a non-deterministic IVC (resp. IFC) satisfying existential soundness (resp. evaluation binding) up to depth bound $d(\cdot)$. Let $\rho\colon \mathbb{N} \to \mathbb{N}$ be a non-decreasing function such that $\mathsf{D} := d^{\rho}$ is polynomially bounded. Then there exists a non-deterministic IVC (resp. IFC) Σ' satisfying existential soundness (resp. evaluation binding) up to depth bound D. When used to prove incrementally a function F, Σ' has the following efficiency profile:*

- *(proof size)* $\mathsf{prf\text{-}size}[\Sigma'] = O(\rho \cdot |\mathsf{out}_F| \cdot \mathsf{prf\text{-}size}[\Sigma])$ *where* $|\mathsf{out}_F|$ *is the size in bits of the output of* F.
- *(verification time)* $\mathsf{T_V}[\Sigma'] = O(\rho \cdot \mathsf{T_V}[\Sigma])$.
- *(amortized proving time) The amortized proving time* $\mathsf{T_P^{amort}}[\Sigma']$ *of* Σ' *can be bound as follows:*
  - $\mathsf{T_P^{amort}}[\Sigma'] = O(\rho \cdot \mathsf{T_P}[\Sigma])$, *if* Σ *has a linear time prover.*
  - $\mathsf{T_P^{amort}}[\Sigma'] = \tilde{O}(\rho \cdot \mathsf{T_P}[\Sigma] \cdot \mathrm{polylog}(d))$ *if* Σ *has a quasilinear time prover.*

Combining Theorem 13 with Theorem 9 (which provides evaluation binding at constant depth) yields an IFC satisfying evaluation binding at any polynomial depth with proof size and verification time $O(\log \lambda)$.

F.2 Why the proofs carry over

We now explain why the proofs of the original theorems extend to both existential soundness and evaluation binding with no substantive modification.

The key abstract property Inspecting the proofs of Theorems 1, 2, 4 and 5 reveals that they interact with the security notion exclusively through the advantage function $\mathbf{Adv}^{(D)}_{\mathfrak{S}[\mathcal{A}]}(\lambda)$, and they rely on a single structural property of this function that we call *depth monotonicity*:

If $d \preceq D$ and the adversary $\mathcal{A}$ outputs a target depth $\delta \le d(\lambda)$, then $\mathbf{Adv}^{(d)}_{\mathfrak{S}[\mathcal{A}]}(\lambda) \le \mathbf{Adv}^{(D)}_{\mathfrak{S}[\mathcal{A}]}(\lambda)$.

Depth monotonicity holds because, in all three security experiments (deterministic soundness, existential soundness, evaluation binding), the depth bound D appears only through the constraint "$\delta \le D(\lambda)$" on the adversary's output. Relaxing this constraint from d to a larger D can only enlarge the set of winning adversarial outputs, hence the advantage cannot decrease. We stress that nothing about the specific predicate—whether it checks $z_\delta \neq F^{(\delta)}(z_0)$, or the non-existence of witnesses, or the existence of two distinct accepted evaluations—is used.

Theorem 1 (i.o. to a.e. soundness). The core step in the proof is the inequality $\mathbf{Adv}^{(d')}_{\mathfrak{S}[\mathcal{A}]}(\lambda) \le \mathbf{Adv}^{(\mathsf{D})}_{\mathfrak{S}[\mathcal{A}]}(\lambda_i)$, which holds because the adversary's target depth satisfies $\delta \le d'(\lambda) \le d'(\lambda_{i+1}-1) \le \mathsf{D}(\lambda_i)$ by condition (▲). This is precisely depth monotonicity. The subsequent step—$\nu(\lambda_i) \le \nu(\lambda)$ from $\lambda_i \le \lambda$—is purely about negligible functions. Both steps are independent of the security notion, so the proof applies verbatim.

Theorem 4 (no graceful degradation). The proof uses two lemmas, both relying only on depth monotonicity:

- **Lemma 3:** If $L \preceq R$ and both are unsafe, then $e^*_T(L) \ge e^*_T(R)$. The proof constructs $\mathcal{A}_R$ that runs $\mathcal{A}_L$; since $\delta \le L(\lambda) \le R(\lambda)$, the same output is a valid attack at depth R. This is depth monotonicity.
- **Lemma 2:** This lemma reasons only about the integers $e^*_T(d_i)$ and their ordering; it uses no property of the underlying experiment.

The final contradiction via the sandwiching theorem (Theorems 3 and 7) never inspects the soundness predicate, so the proof is identical for existential soundness and evaluation binding.

Theorem 5 (simple depth boosting). The proof builds a reduction $\bar{\mathcal{A}}$ that forwards parameters; the key step shows the target depth satisfies $\delta(\lambda) \leq d(\lambda)$ via the identity $\mathsf{D}(\sqrt[\rho]{\lambda}) = d(\sqrt[\rho]{\lambda})^\rho = \lambda^\varepsilon = d(\lambda)$. This is a purely algebraic fact about depth bound functions. The advantage is then related by $\mathbf{Adv}^{(d)}_{\mathfrak{G}[\bar{\mathcal{A}}]}(\lambda) = \mathbf{Adv}^{(\mathsf{D})}_{\mathfrak{G}[\mathcal{A}]}(\bar{\lambda}) > \bar{\lambda}^{-e} = \lambda^{-e/\rho}$, a chain that uses only that the two experiments are identical (same parameters forwarded, same output returned) and depth monotonicity.

Theorem 6 (general depth boosting). This is the only constructive result: it builds a new scheme Σ' via the construction in Fig. 3 and requires a soundness reduction in addition to the abstract manipulation of advantages. Concretely, if the adversary provides a proof $\pi = (z_{\text{JMP}}, \pi_{\text{jmp}}, \pi_{\text{JMP}})$ for a false claim "$z_0 \rightsquigarrow z'$ in δ steps", then at least one of the following is false: (1) the big-jump claim verified by Π_{JMP}, or (2) the small-jump claim verified by Π_{jmp}. This decomposition holds for all three security notions:

- **Deterministic soundness:** if $z' \neq F^{(\delta)}(z_0)$ then either the big-jump or small-jump output is incorrect, which is immediate.
- **Existential soundness:** if no sequence of witnesses $(w_0, \ldots, w_{\delta-1})$ connects z_0 to z' via F, then a valid witness sequence cannot exist for *both* segments—otherwise their concatenation would witness the whole computation.
- **Evaluation binding:** if the adversary provides two proofs for the same commitment and function yielding distinct outputs $y \neq y'$, the disagreement must appear in at least one segment. Specifically, if the intermediate values y_{JMP} and y'_{JMP} extracted from the two big-jump proofs agree, then the two small-jump proofs yield distinct outputs from the same starting point; if they disagree, the big-jump proofs already break evaluation binding. The commitment structure decomposes consistently because the construction interleaves proofs at different scales without modifying the commitment scheme itself.

In all cases, one builds $\tilde{\mathcal{A}}$ against Π_{JMP} or Π_{jmp} by forwarding the relevant portion of $\mathcal{A}$'s output. Non-negligible advantage of $\mathcal{A}$ implies non-negligible advantage of $\tilde{\mathcal{A}}$, contradicting the assumed security. The inductive argument over $j = 1, \ldots, \rho$ and the efficiency analysis (proof size, verification time, amortized proving time) are identical to those in the original proof and do not depend on the specific predicate.

Addition minor changes to depth-boosting outside the deterministic setting. When applying our construction in Fig. 3 beyond the deterministic setting, the prover must account for the additional inputs consumed at each step. In the deterministic case, the big-jump prover can execute $F^{(d)}$ given only the starting state z_{JMP}, since all intermediate states are uniquely determined. This is no longer the case for non-deterministic IVC and IFC.

For **non-deterministic IVC**, each step computes $z_i = F(z_{i-1}, w_{i-1})$, where w_{i-1} is a local witness. The big-jump step function $F^{(d)}$ therefore requires as non-deterministic input the entire tuple $(w_0, \ldots, w_{d-1})$ of witnesses consumed during the d small-jump steps it subsumes. Accordingly, the prover must buffer the d intermediate witnesses encountered between consecutive big jumps and supply them when invoking the big-jump prover Π_{JMP} on $F^{(d)}$.

For **IFC**, the situation is analogous. Each incremental proving step (Definition 17) consumes a data chunk chunk_D and the auxiliary commitment information $\mathsf{aux}^{(\text{com})}_{\mathsf{chunk}}$ produced by $\mathsf{IncCommit}$. When performing a big jump that subsumes d small-jump steps, the prover must re-execute d consecutive invocations of $\mathsf{IncProveEval}$; this requires the d data chunks $(\mathsf{chunk}_{D,0}, \ldots, \mathsf{chunk}_{D,d-1})$ and the corresponding auxiliary information $(\mathsf{aux}^{(\text{com})}_{\mathsf{chunk},0}, \ldots, \mathsf{aux}^{(\text{com})}_{\mathsf{chunk},d-1})$. As in the non-deterministic case, the prover buffers these intermediate inputs during the d small-jump steps and passes them to Π_{JMP} when the big jump is executed.

In both cases, the buffering does not affect proof size, verification time, or the amortized proving time bounds stated in Theorem 13. The verification logic in Fig. 3 require no changes.