On the Subtleties of Non-Malleability in IVC: Notions, Attacks and Constructions from Falsifiable Assumptions

Matteo Campanelli[1], Suvradip Chakraborty[2], and Chaya Ganesh[3]

[1] Offchain Labs & University of Tartu, Estonia
binarywhalesinternaryseas@gmail.com
[2] Visa Research
suvradip1111@gmail.com
[3] Indian Institute of Science
chaya@iisc.ac.in

Abstract. We investigate *non-malleable* Incrementally Verifiable Computation (IVC), a subject that has received limited attention. We focus on whether and how simulation-sound IVC can be obtained from falsifiable assumptions. We provide additional evidence of the unique aspects of non-malleable incremental proofs, identifying definitional challenges in formalizing simulation-sound IVC, and showing how existing IVC designs from falsifiable assumptions cannot exhibit simulation-soundness "for free" (without modifications).

In addition to a new family of definitions of simulation-sound IVC, our main results are the following.

- *A general negative result: common Valiant-style proofs are malleable.* We isolate and formalize a feature shared by the dominant Valiant-style IVC designs (Valiant, TCC 2008) related to the ability to merge proofs together. We show that, assuming one-way functions, natural zero-knowledge versions of Valiant's design are malleable. As an implication of our framework of attacks, we show that the only IVC for **NP** from falsifiable assumptions (Datta et al., CRYPTO 2025) cannot satisfy any form of simulation-soundness; this is the case even when instantiating this scheme from non-malleable building blocks.
- *A positive result: first IVC from falsifiable assumptions with non-malleability features.* We construct an IVC satisfying a non-trivial form of simulation-soundness under subexponential $i\mathcal{O}$ and either subexponential LWE or subexponential bilinear-map assumptions.

Table of Contents

1 Introduction **4**
1.1 Our contributions . . . 5

2 Technical Overview **7**
2.1 Modeling simulation-soundness in IVC . . . 7
2.2 Attacks on Valiant-style proofs . . . 7
2.3 (Weak) simulation-soundness via context chaining . . . 9
2.4 Proof overview of weak simulation soundness . . . 11
2.5 Towards SIM-SND . . . 14

3 Preliminaries **14**
3.1 Non-interactive zero-knowledge proof (NIZK) and signatures of knowledge . . . 14
3.2 Succinct non-interactive argument (SNARG) . . . 15
3.3 Public-key encryption . . . 16
3.4 Batch arguments (BARGs) . . . 16

4 Defining Simulation-Soundness in IVC **18**
4.1 Background on IVC . . . 18
4.2 Auxiliary notions: promise leaps and compliance . . . 19
4.3 Simulation-soundness notions for IVC . . . 19

5 A Framework for Mauling Attacks in IVC **20**
5.1 Mergeable IVCs . . . 20
5.2 Hard reachability problems . . . 21
5.3 Structured hard reachability problems from one-way functions. . . . 22
5.4 Sufficient conditions for attacks against newrt-SIM-SND . . . 23
5.5 Sufficient conditions for attacks against w-SIM-SND . . . 23

6 Context-chained IVC Construction **24**
6.1 Completeness . . . 25
6.2 Efficiency . . . 28
6.3 Zero knowledge . . . 29
6.4 Weak simulation soundness . . . 31

References **33**

A Additional Related Work **35**

B Examples of (Non) Compliance **35**

C The IVC of [DJJ+25] and Concrete Malleability Attacks **35**
C.1 Background: the IVC construction $\Pi^{\mathrm{IVC}}_{[\mathrm{DJJ+25}]}$ . . . 36
C.2 IVC of [DJJ+25] is not SIM-SND . . . 38
C.3 IVC of [DJJ+25] is not newrt-SIM-SND . . . 39

D newrt-SIM-SND IVC Construction **41**
D.1 Languages and relations for the construction . . . 41
D.2 Root binding IVC construction . . . 42
D.3 Root binding IVC satisfies newrt-SIM-SND . . . 43

E w-SIM-SND Proof of Context-chained IVC **50**
E.1 Unexplainable encryption . . . 50
E.2 Weak simulation soundness . . . 52

F Additional Formal Treatment of Mergeability **65**
F.1 ZK simulators and hard reachability problems . . . 65
F.2 Proof of Lemma 5.7 . . . 66
F.3 Proof of Lemma 5.8 . . . 66
F.4 Proof of Theorem 5.9 . . . 66
F.5 Proof of Theorem 5.10 . . . 67
F.6 Proof of Theorem 5.11 . . . 67
F.7 Proof of Theorem 5.13 . . . 69
F.8 Proof of Theorem 5.14 . . . 69

G Valiant-Style Proofs in the Zero-Knowledge Setting **70**

H Ordering of the Simulation-Soundness Notions **71**

I Example Applications **71**
I.1 Proofs of authorized image manipulation succinct in the number of edits . . . 71
I.2 Zero-knowledge (non-malleable) shuffles with sublinear proof-size . . . 73

J Simulation-Extractability for PCD [ACDK26] **74**
J.1 Proof-carrying data . . . 74
J.2 Simulation-extractability for PCD . . . 75

1 Introduction

Incrementally Verifiable Computation (IVC) [Val08] allows a prover to certify a long computation *one step at a time*, updating a succinct proof as the computation progresses. At each step, a step circuit C transforms the current state x_{i-1} into the next state $x_i = C(x_{i-1}, w_i)$, using a witness w_i supplied by the prover. We write $x_0 \overset{t}{\rightsquigarrow} x_t$ to mean that some sequence of such steps takes x_0 to x_t in exactly t steps. An IVC proof certifies this reachability claim and can be extended with each new step, while both its size and the work required to update it remain sublinear in the overall computation depth. Since its introduction, IVC has found numerous applications, ranging from succinct blockchains [BMRS20] to verifiable delay functions [BBBF18]. The reader can find further examples of application settings in Appendix I and in [AG25, §6].

Only recently, Datta et al. [DJJ+25] gave the first construction of IVC for all of **NP**[1] from falsifiable assumptions—roughly speaking, those that can be modeled as an efficient game between a challenger and an adversary [Nao03]. Their construction combines somewhere-extractable batch arguments (BARGs) with *non-adaptive* SNARGs, (non-succinct) NIZKs, and public-key encryption. The construction in [DJJ+25] is an adaptation of the **Valiant-style design** based on "mergeable" proofs [Val08, PP22], the dominant blueprint for incremental proofs from falsifiable assumptions. Before the work of Datta et al. [DJJ+25], provably secure IVCs from *falsifiable* assumptions have long been elusive and have have relied on idealized models or on heuristic or otherwise strong assumptions [CT10, BCL+21, KST22, CFP25b, CFP25a, MMZ25].

Towards stronger security guarantees: non-malleability. The properties usually demanded of a proof system—(knowledge) soundness and zero knowledge—are a *bare minimum*: they concern a proof produced and verified "in isolation". Once a proof system is deployed inside a larger protocol, an adversary may instead obtain simulated or honestly generated proofs and attempt to *maul*, recombine, or adapt them into new accepting proofs for statements whose witnesses it does not know. Resilience to such attacks is captured by *non-malleability*: an accepted proof cannot be tinkered with into a new one for a different statement without knowing a witness. It is formalized as *simulation soundness* [Sah99]—which requires that soundness hold even against an adversary granted access to a proof-simulation oracle—and, in the knowledge-soundness regime, as *simulation extractability* [DDO+01, GMY03]. Such guarantees are what make it safe to compose proofs within larger protocols, and what rule out elementary attacks on transaction systems such as double spending; these attacks are far from hypothetical, with hundreds of thousands of Bitcoins reportedly involved in malleability incidents [DW14]. The concern is especially acute for IVC, whose most prominent applications—verifiable virtual-machine execution [AST24] and other blockchain settings [BMRS20]—are exactly the deployments in which an adversary realistically has access to a proof oracle. See also Appendix I where we discuss additional settings that can naturally benefit from non-malleability.

The state of the art and its limitations. Despite its importance, non-malleability in the incremental setting remains understudied. The work of Abdolmaleki et al. [ACDK26] only recently initiated this study for Proof-Carrying Data (PCD, of which IVC is a special case) [CT10] and pointed out inherent challenges in obtaining a workable notion of non-malleability. They show that, because of the quirky nature of the incremental setting, the traditional notion of simulation extractability does not directly transfer to incremental settings. Intuitively, this is because incremental proofs carry an unavoidable degree of malleability, which a good notion should account for.[2] To accommodate this, they propose a relaxed definition in which recursive extraction is permitted to *stop* at simulated sub-proofs. Under this new definition, they study two kinds of PCD constructions: i) those obtained by recursing over a simulation-extractable SNARK; ii) those obtained by recursive over accumulation/folding schemes—a "lightweight" kind of proof systems useful as a building block for PCD [BCL+21, KST22]—combined with a simulation-extractable argument. They show that the

[1] IVC for **NP** allows each computation step to use a local (potentially private) witness at each step, thereby supporting non-deterministic computations.

[2] In particular, an adversary may legitimately query the simulator for a proof on an intermediate state for which it does not know the witness (or for which a witness does not exist). It could then *honestly* extend the computation to some *truly reachable* state. Thus its final—and genuinely fresh—proof embeds a simulator-issued sub-proof for which no extractor can be expected to produce a witness.

resulting PCD immediately inherit simulation extractability from that of their building blocks and that these construction families are non-malleable essentially "for free"[3].

Limitations. Despite the important initial observations in [ACDK26], the work of Abdolmaleki et al. still leaves us with an unsatisfactory understanding around (non-)malleability in incremental proofs. In particular, it leaves open:

- **Valiant-style constructions:** this popular blueprint has been applied in many major constructions in IVC [Val08, PP22, NDC+24, AG25]. It also underlies the only secure construction for NP from falsifiable assumptions [DJJ+25]; understanding whether and how the latter can be made non-malleable is an important open question. We note that the techniques used for the analysis in [ACDK26] do not easily apply to study this family of designs for two reasons: they (these techniques) rely heavily on the extractability of some succinct building block (folding or SNARKs); the Valiant-style construction template differs fundamentally from those considered in [ACDK26], as we discuss later in this work.
- **Simulation-soundness:** Simulation-extractability, the notion that appears in [ACDK26] for PCDs, cannot be used to study schemes supporting non-deterministic computations and that do not target knowledge soundness. Schemes of this kind include both the already mentioned [DJJ+25] and the more recent, albeit quite more limited, construction in [JMW26] (see also Appendix A), both from falsifiable assumptions. In order to investigate these constructions one first needs to model simulation-soundness in the IVC setting. To do that, one would need to solve similar challenges as those encountered by Abdolmaleki et al. but at the same time do some non-trivial new modeling work. In fact, the definition in [ACDK26] cannot be easily adapted as the aspects in it that are specific to the incremental setting are strongly coupled to the presence of an extractor, unavailable in simulation-soundness (we provide more details in Appendix J).

1.1 Our contributions

We advance the state of the art on non-malleability in incremental proof systems and propose, for the first time, results applicable to IVC from falsifiable assumptions. Our work confirms, in a new setting (simulation-soundness), some of the insights from [ACDK26] on the idiosyncratic features of non-malleability in incremental proofs. At the same time, it offers new ones, by i) showing, through our framework, mauling attacks whose nature is strongly tied to the IVC domain, and by ii) identifying relaxations of SIM-SND with no equivalent in the standard NIZK setting (our notion of newrt-SIM-SND, discussed below).

New notions of simulation-soundness for IVC. We first propose new notions of simulation soundness with security guarantees of different strengths (see Fig. 1): SIM-SND , w-SIM-SND , newrt-SIM-SND. At one extreme, SIM-SND is the strongest analogue of standard simulation soundness in NIZKs. At the other extreme, newrt-SIM-SND is a specifically incremental notion with no direct analogue in the traditional setting: it only requires security for targets whose initial state x_0 is fresh, meaning that the simulator was never queried on any statement of the form $x_0 \rightsquigarrow _$. The intermediate notion w-SIM-SND mirrors weak simulation soundness for standard NIZKs [FKMV12] (which, intuitively, permit rerandomizing observed proofs without a witness). We discuss how these can be meaningful in different application settings in Appendix I. We also show that each of this notion is strictly stronger than the next. We discuss how to formally model simulation-soundness for IVC in Section 2.

Mauling attacks against Valiant-style IVCs. We then introduce a general framework, centered around a notion of *"mergeable"* proofs in the zero-knowledge setting, to study mauling attacks in IVC. We use this to show how Valiant-style are susceptible to attacks that do not apply to the blueprints studied in [ACDK26]. We provide an overview of our attack framework in the technical overview.

The IVC in Datta et al. [DJJ+25] is malleable (and cannot be secured "for free"). We use our framework to study the most expressive IVC construction in the literature from falsifiable assumption, that of Datta et al. [DJJ+25]; we prove that this construction is malleable and cannot satisfy even the most relaxed of our simulation-soundness notions, newrt-SIM-SND.

[3] Jumping ahead to some of our results, we will show that this is not always the case and that other popular designs do not have this property.

Notion	Has analogue in NIZKs?	General Intuition
SIM-SND	✔ (standard SIM-SND)	Captures the broadest set of attacks
w-SIM-SND	✔ (weak SIM-SND)	Not attacks: proofs for $x \overset{t}{\rightsquigarrow} y$ if $\mathcal{A}$ queried $x \overset{t}{\rightsquigarrow} y$
newrt-SIM-SND	✗	Not attacks: proofs for $x \overset{t}{\rightsquigarrow} y$ if $\mathcal{A}$ queried $x \overset{\hat{t}}{\rightsquigarrow} _$

Fig. 1: Our notions of simulation-soundness and parallels with the NIZK setting (we stress that their respective model ultimately differ). Notions in lower rows place more restrictions on an adversary. Notation $x \overset{\hat{t}}{\rightsquigarrow} _$ means $x \overset{\hat{t}}{\rightsquigarrow} \hat{y}$ for arbitrary $\hat{y}, \hat{t}$.

Our attack patterns apply to any instantiation of the construction that preserves zero knowledge and a specific form of mergeability, regardless of any additional security properties of its building blocks. Thus, strengthening those building blocks—even requiring them to satisfy non-malleability—cannot by itself yield a simulation-sound IVC. Obtaining such a guarantee requires modifying the construction so that it no longer supports the form of mergeability exploited by our attack. Interestingly, in this sense, non-malleability cannot be obtained "for free" by merely strengthening the underlying primitives, showing a different landscape from the one in the incremental constructions analyzed in [ACDK26][4].

A w-SIM-SND IVC from falsifiable assumptions. We construct a new IVC scheme, building on the framework of Datta et al. [DJJ+25], that satisfies weak simulation soundness. At a high level, the IVC of Datta et al. certifies each one-step transition between two encrypted states using a (succinct) NIZK, which we call a level-0 proof, and then recursively merges pairs of adjacent proofs, level by level, using rate-1 somewhere-extractable batch arguments (BARGs). Here, a BARG is a SNARG for a batch of **NP** statements $(x_1, \ldots, x_k)$, asserting the existence of witnesses $(w_1, \ldots, w_k)$ such that $R(x_i, w_i) = 1$ for every i; rate 1 means that the proof has size $|w| + \mathsf{poly}(\lambda, \log k)$. Repeating this merging process yields a succinct proof for an arbitrarily long computation.

The difficulty is that this recursive structure does not, by itself, record *where* a sub-proof occurs in the computation. A proof obtained from the simulator can therefore potentially be transplanted to a different location while leaving the local BARG statement unchanged. Our main structural idea is a *context chain* that binds every recursive proof to the computation prefix preceding it. Concretely, each block carries a short hash value summarizing its left context, and this value is included in the recursive BARG instances. Thus, moving a block to a different position changes the statement being proved.

We then ask what the minimal requirements are on each of the cryptographic primitives involved. We replace the level-0 NIZK by a simulation-extractable NIZK *signature of knowledge* (NIZK-SoK), using the context-chain value as its message. The higher levels continue to use ordinary rate-1 somewhere-extractable BARGs: once the context is made part of the recursive instance, somewhere extraction is sufficient to propagate this binding down the proof tree. In particular, we require no form of non-malleability or simulation-extractability from the BARG. Beyond these, the construction uses collision-resistant hashing and a perfectly correct CPA-secure encryption scheme satisfying *unexplainability*: even given the secret key, an adversary cannot recover valid encryption randomness for a challenge ciphertext generated with hidden randomness. All of these ingredients remain within the regime of falsifiable assumptions.

Theorem (Informal). *Assume subexponentially secure versions of the following primitives, with perfect completeness for the proof systems a simulation-extractable NIZK-SoK with adaptive multi-theorem zero knowledge, rate-1 somewhere-extractable BARGs, non-adaptive SNARGs for NP, collision-resistant hash functions, and perfectly correct, CPA-secure, unexplainable public-key encryption. Then, for every polynomial-size step circuit C over n-bit states and every depth bound $1 \leq T \leq 2^\lambda$, there exists a perfectly complete, zero-knowledge IVC scheme that satisfies w-SIM-SND and has proof size $\mathsf{poly}(\lambda, n, \log T)$. Its CRS size and setup, update, and verification times are $\mathsf{poly}(\lambda, \log T, |C|)$.*

[4] These constructions are based on recursion over succinct extractable building blocks and do not support merging proofs without having a witness.

Corollary (Informal). *Under subexponential $i\mathcal{O}$ and either subexponential LWE or the subexponential bilinear-map assumptions, there exists a perfectly complete, zero-knowledge IVC for* **NP** *with proof size* $\mathrm{poly}(\lambda, n, \log T)$ *that satisfies* w-SIM-SND. *The formal instantiation is given in Corollary 6.8.*

2 Technical Overview

2.1 Modeling simulation-soundness in IVC

We first recall the general principles behind the definition of *simulation-soundness* in its standard NIZK formulation [Sah99]. Under this notion, an adversary $\mathcal{A}$ is given access to a *proof simulation oracle* that returns accepting proofs for arbitrary (possibly false) statements and should be unable to produce a *fresh* accepting proof on any statement outside the target language. In the incremental setting, the target language is the reachability language $\mathcal{L}_{\mathsf{IVC}}$ and the simulator (as in the zero-knowledge modeling in [DJJ+25]) receives queries on reachability statements of the form "$x_0 \overset{t}{\rightsquigarrow} x_t$". Notice that, as mentioned earlier, if we just applied the standard formulation to the IVC setting we would obtain an unachievable notion—here is why. Consider a step circuit C, states x_0, x_t and a witness w such that both x_t and x_{t+1} are unreachable from x_0 (with respect to C) with $x_{t+1} := C(x_t, w)$. Now given any simulator-supplied proof for the statement "$x_0 \overset{t}{\rightsquigarrow} x_t$", the adversary can apply the honest IVC prover for one additional step using witness w to obtain a fresh accepting proof for the target statement "$x_0 \overset{t+1}{\rightsquigarrow} x_{t+1}$", which still lies outside $\mathcal{L}_{\mathsf{IVC}}$(!). We provide a concrete example for this scenario in Remark B.1. In light of this subtle inherent "malleability" in IVC, we need a new notion that is tailored to it.

Towards the goal of ruling out "non-attacks" like the one above, our approach is the following. For an attack to be valid, the target statement should remain false even in the *hypothetical universe* where the statements queried to the simulator are actually true. If this is the case, we say that the target statement is "non-compliant"; this will be a necessary condition for the adversary to win. In order to model this hypothetical universe, we define a notion of "promise leaps". These capture reachability statements (leaps) that are supposedly true in the thought-experiment universe. Concretely, a statement becomes "promised" if the simulator has been queried on it. With this central modeling choice under our belt, formalizing the notion of "compliance" is relatively straightforward. We are then able to capture the standard strong notion of simulation-soundness in NIZK [Sah99]. Besides the standard strong notion we also (see Fig. 1):

- capture the relaxation of weak simulation-soundness [FKMV12] (which intuitively permits, e.g., rerandomizations of previously seen proofs), by additionally disregarding targets whose statement was queried to the simulator;
- observe the possibility of a further natural relaxation, with no counterpart in non-incremental settings, which restricts the adversary to target statements whose "roots" (the starting point x_0) it has never queried the simulator on. This new notion, which we dub new-root simulation soundness (newrt-SIM-SND), is applicable to natural settings (see Appendix I).

2.2 Attacks on Valiant-style proofs

Valiant proposes to obtain IVC from *proof merging*, intuitively described as follows. Assume that we are given a proof π_L for the statement $x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{mid}}$ and a proof π_R for the statement $x_{\mathsf{mid}} \overset{d_R}{\rightsquigarrow} x_{\mathsf{fin}}$. To be able to merge proofs means being able to efficiently produce a new proof π^* for the statement $x_{\mathsf{init}} \overset{d_L+d_R}{\rightsquigarrow} x_{\mathsf{fin}}$ in time independent of d_L and d_R and *without access to the underlying witnesses* used to generate π_L and π_R (see also Fig. 2).

We will now argue why this design pattern is prone to malleability attacks by showing an adversary for the weakest of our notion, newrt-SIM-SND. For the time being we will be slightly imprecise and ignore some technical aspects, which we will discuss in later paragraphs. Consider the *false* statement $s^* := x_{\mathsf{init}} \overset{d}{\rightsquigarrow} x_{\mathsf{fin}}$. Our adversary $\mathcal{A}$ will query the simulator once and produce a valid proof for s^*. In order to do that, it can pick an intermediate string x_{mid} and such that for some $d_L, d_R \geq 1$ with $d = d_L + d_R$, we have:

$$x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{mid}} \quad \text{and} \quad x_{\mathsf{mid}} \overset{d_R}{\not\rightsquigarrow} x_{\mathsf{fin}}.$$

Call the left statement s_L and the right statement s_R. The adversary then i) queries the simulator on s_R obtaining a proof π_R; ii) produces an honest proof π_L for the statement s_L; iii) merges the two proofs obtaining a proof π^* for s^*. In order for this adversary to be breaking newrt-SIM-SND we need to guarantee two things: that it has never queried the simulator on x_{init}, the "root" of the target statement s^*, and that the verifier accepts π^* as valid for s^*. We can easily see that, in order to obtain the above, it is sufficient to show that $\mathcal{A}$ can pick inputs $x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}$ satisfying the truth/falsity requirements on s_L, s_R, s^* above and such that: a) $x_{\mathsf{mid}} \neq x_{\mathsf{init}}$ (which guarantees the condition on simulator queries); b) it is easy to compute witnesses for $x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{mid}}$ (in order to be able to produce honest π_L); c) the simulator should produce a valid proof π_R for s_R (so that by merging π_L and π_R, both valid proofs, we have the guarantee that the resulting π^* is also valid).

The approach we use to obtain c) is to sample, for a given d_R, $x_{\mathsf{mid}}, x_{\mathsf{fin}}$ so that: $s_R = x_{\mathsf{mid}} \overset{d_R}{\rightsquigarrow} x_{\mathsf{fin}}$ is false *and* it is hard to distinguish s_R from a true reachability statement. A simulated proof for such s_R would then need to verify with overwhelming probability (since otherwise it would yield a distinguisher for true vs. false statements s_R). We formalize this form of *hard reachability problems* and show that they are implied by one-way functions and that they can be made to have a structure guaranteeing a) and b).

On Valiant-style proofs and zero knowledge. One technical aspect that we ignored so far is that, in the zero-knowledge setting, mergeability in Valiant-style proofs does not work as cleanly as we assumed in our description above. Merging, as described above, requires each proof to expose the endpoints of its segment, so that adjacent proofs can be matched. In the zero-knowledge setting, however, these endpoints are intermediate states of the computation and should remain hidden. This is often solved by employing an encoding of the endpoints—usually as ciphertexts—which allows the recursive construction to refer to them and check that adjacent segments meet, without exposing their contents. In works considering some form of zero knowledge in IVC from falsifiable assumptions ([DJJ+25, AG25]) the proof-merging blueprint is adapted in the following natural way[5]: a proof π for $x_{\mathsf{init}} \overset{d}{\rightsquigarrow} x_{\mathsf{fin}}$ consists of a tuple $\pi = (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, r_{\mathsf{init}}, r_{\mathsf{fin}}, \tilde{\pi})$ so that $\mathsf{ct}_\chi = \mathsf{PKE.Enc}(\mathsf{pk}, x_\chi; r_\chi)$ for $\chi \in \{\mathsf{strt}, \mathsf{fin}\}$ where PKE is a public-key encryption scheme and pk is a public key included in the scheme parameters. Here the capabilities of proof merging have additional constraints: we can merge proofs π_L and π_R at mid-point x_{mid} only if the respective ciphertexts $\mathsf{ct}_{\mathsf{mid}}$ in the two proofs match (as well as the corresponding randomness). That is there exists an algorithm Merge that on inputs $x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}$, proofs $\pi_L = (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, r_{\mathsf{init}}, r_{\mathsf{mid}}, \tilde{\pi}_L)$ and $\pi_R = (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, r_{\mathsf{mid}}, r_{\mathsf{fin}}, \tilde{\pi}_R)$, outputs $\pi^* = (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, r_{\mathsf{init}}, r_{\mathsf{fin}}, \tilde{\pi}^*)$ where $\tilde{\pi}^*$ is the output of some additional "inner" merging algorithm which does not require the ciphertext randomnesses.

Despite the additional constraints, this type of mergeability still leads to the kind of attack we outlined earlier. In fact, all we need in our attack is a form of "post-hoc" mergeability "for fresh proofs from the left": *after* receiving simulated proof π_R, the adversary simply needs to *produce a left proof* π_L that will be compatible with π_R in order to merge the two. The zero-knowledge variant of Valiant's mergeability that we just saw, however, has exactly this property! Indeed, after observing $\pi_R = (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, r_{\mathsf{mid}}, r_{\mathsf{fin}}, \tilde{\pi}_R)$, the adversary can run the honest prover for $x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{mid}}$ using r_{mid} as the randomness for encrypting x_{mid}. The resulting π_L then has $\mathsf{ct}_{\mathsf{mid}} = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{mid}}; r_{\mathsf{mid}})$ as its endpoint ciphertext, and can be merged with π_R.

We abstract and formalize, together with our attack, this specific type of post-hoc mergeability property and show that the IVC in [DJJ+25] satisfies it and is therefore malleable. In Appendix G we argue in more detail how, in general, variants of these Valiant-style designs satisfy post-hoc mergeability. We also show that our attack approach is robust: a weaker form of mergeability, which does not suffice to break newrt-SIM-SND, still suffices to break w-SIM-SND. Finally, we prove that these two notions are actually separated from one another, in that there exists a construction from falsifiable assumptions that satisfies newrt-SIM-SND but not w-SIM-SND (this intermediate construction can be thought of as a substantially simplified version of the one in our positive result and is described later in this overview as the one with "binding to the root").

[5] Here we simplify certain aspects and we keep our focus on what is relevant for this overview. We also note that this description, while morally capturing both [DJJ+25] and [AG25], does not perfectly match the setting in [AG25], since this work is for a more restricted setting than full-blown **NP**.

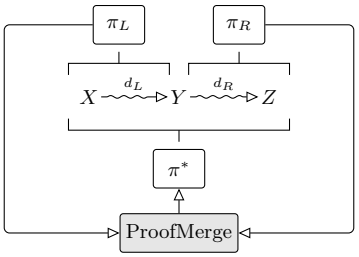


Fig. 2: Graphical intuition on Valiant-style mergeability (non zero-knowledge case): two given proofs π_L (proving Y is reachable from X) and π_R (proving Z is reachable from Y) are merged without the underlying witnesses into a proof π^* (proving Z is reachable from X).

2.3 (Weak) simulation-soundness via context chaining

As shown above, any IVC satisfying our notion of post-hoc mergeability is inherently malleable. To obtain a non-malleable construction, we therefore need to constrain which proofs may be merged. Our approach is to augment recursive proofs with a notion of *context*, so that a proof can only be reused or merged in a position consistent with the computation prefix in which it was originally created.

We instantiate this idea starting from the IVC of Datta et al. [DJJ+25]. Their construction follows the same proof-merging paradigm as above, while its recursive structure makes particularly clear where context is lost and how it can be restored. We first recall the aspects of their construction that are relevant for the attack, then describe two increasingly strong forms of context binding: root binding, which suffices for newrt-SIM-SND, and context chaining, which yields w-SIM-SND.

The [DJJ+25] IVC. At a high level, the construction of [DJJ+25] represents a computation $x_0 \to x_1 \to \cdots \to x_t$ through encrypted intermediate states $\mathsf{ct}_0, \ldots, \mathsf{ct}_t$. Each adjacent pair $(\mathsf{ct}_{i-1}, \mathsf{ct}_i)$ is certified by a succinct level-0 proof asserting that the ciphertexts open to configurations related by one valid computation step. These proofs are then merged recursively: a level-1 BARG certifies two adjacent level-0 proofs, a level-2 BARG certifies two adjacent level-1 proofs, and so on. Thus a level-ℓ proof succinctly certifies a block of 2^ℓ consecutive steps.

The recursive statement at level ℓ contains only the encrypted endpoints of the block. Its witness consists of a midpoint ciphertext together with a lower-level proof certifying the two child blocks. The prover maintains these proofs using a binary-counter merging schedule, so that after t steps only $O(\log t)$ recursive proofs remain. The simulator has the following structure. On a query (x, x', d), it encrypts the requested endpoints, uses encryptions of 0 for the interior states, simulates the level-0 proofs, and then constructs all higher-level BARGs honestly from the resulting verifying child proofs.

A specific version of the attack in [DJJ+25]. In [DJJ+25], the attack above takes a particularly simple form (see Fig. 3 for a depiction). Pick states $a \neq b$ and c such that $a \overset{2}{\rightsquigarrow} b$ and $a \overset{4}{\not\rightsquigarrow} c$ (hence also $b \overset{2}{\not\rightsquigarrow} c$). The adversary queries the simulator on the false statement $(b, c, 2)$ and obtains a level-1 proof on ciphertexts $(\mathsf{ct}_0^q, \mathsf{ct}_1^q, \mathsf{ct}_2^q)$. It then computes an honest two-step proof from a to b whose final ciphertext is ct_0^q, and merges the two proofs. In the resulting four-step proof for the false statement $(a, c, 4)$, the simulated level-1 proof appears verbatim as the right half: its local ciphertext statement is unchanged, and only its position in the proof tree differs. Since the query is rooted at b and the target at a, this violates even newrt-SIM-SND. The underlying problem is that recursive proofs are bound to their local ciphertext sub-statements, but not to the computation context in which those sub-statements occur.

A first fix attempt: binding proofs to the root. The attack above exploits post-hoc mergeability across computations with different roots: a proof created inside a computation beginning at b can be reused inside one *beginning at* a. This suggests restricting mergeability by binding every recursive proof to the *root* x_{in}, the starting state of the computation whose proof is being simulated or generated honestly.

At level 0, we replace the NIZK by a *NIZK signature of knowledge* (NIZK-SoK). Informally, a NIZK-SoK is a NIZK proof that is additionally verified relative to an external message m. The message is, intuitively, a

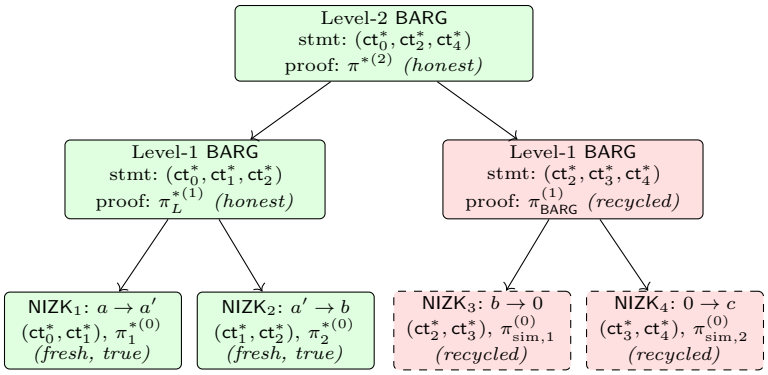


Fig. 3: **Structure of the forgery Π^* against the IVC of [DJJ+25] (Appendix C.1).** *The honest construction* of an IVC proof has a tree shape: each level-0 leaf is a NIZK proof certifying one computation step on a pair of adjacent ciphertexts (indirectly, through a SNARG); two adjacent level-ℓ proofs are merged by a level-(ℓ+1) BARG; and the root is a single succinct BARG proof—here, a 4-step computation with two BARG levels. *The forgery* reuses this exact shape. *Green* nodes are produced honestly by $\mathcal{A}$: the two level-0 NIZK leaves on the left (fresh, true one-step proofs) and the level-1 BARG proof $\pi_L^{*(1)}$ merging them. *Red* nodes are *recycled* from a single simulator query: the right level-1 BARG proof $\pi_{\mathsf{BARG}}^{(1)}$ is copied verbatim from the simulator's response $\Pi^{(1)}$, and its level-0 NIZK leaves exist only as the simulator's witnesses, unknown to $\mathcal{A}$. Finally, $\mathcal{A}$ honestly merges the green and red level-1 proofs into the level-2 BARG proof $\pi^{*(2)}$ that it outputs.

string the proof gets tied to. We use the root x_{in} as this message. At higher levels, we include x_{in} explicitly in every recursive BARG instance. Thus a recursive proof created inside a computation rooted at b certifies instances carrying root b; relocating it into a computation rooted at a changes the recursive statement itself. Following extraction down the recursion preserves the same root tag, until at level 0 one reaches a NIZK-SoK proof under message a. If no simulator query was rooted at a, this message is fresh and the NIZK-SoK knowledge guarantee applies.

In terms of the mergeability framework above, proofs remain mergeable only when, in addition to matching at their boundary, they agree on the root context. This already rules out the relocation attack above and, in fact, suffices to obtain newrt-SIM-SND, our weakest notion of non-malleability. We give the full construction and proof in Appendix D.

Root binding does not give weak simulation soundness. Binding only the root, however, does not bind a proof to its *position* within a computation having that root. This leaves a simple attack against w-SIM-SND. Choose states a, c such that $a \to a$, $a \not\to c$, $c \not\to c$, $a \overset{2}{\not\rightsquigarrow} c$. The adversary queries the simulator on the false one-step statement $q = \left[a \overset{1}{\underset{?}{\rightsquigarrow}} c\right]$ and receives ciphertexts $\mathsf{ct}_0^q = \mathsf{Enc}(\mathsf{pk}, a; r_0)$ and $\mathsf{ct}_1^q = \mathsf{Enc}(\mathsf{pk}, c; r_1)$ together with a simulated level-0 proof $\pi_q^{(0)}$ under message $m = a$.

It now constructs a proof of the false two-step statement $a \overset{2}{\rightsquigarrow} c$. It chooses a fresh encryption $\mathsf{ct}_0^* = \mathsf{Enc}(\mathsf{pk}, a; r_0^*)$, sets $\mathsf{ct}_1^* := \mathsf{ct}_0^q$ and $\mathsf{ct}_2^* := \mathsf{ct}_1^q$, and honestly proves the first step $(\mathsf{ct}_0^*, \mathsf{ct}_1^*)$, corresponding to the valid transition $a \to a$. For the second step it simply reuses $\pi_q^{(0)}$. Its local ciphertext statement is still $(\mathsf{ct}_0^q, \mathsf{ct}_1^q)$, and its message is still a, since both the simulator query and the target computation have root a. The adversary can therefore honestly build the higher-level BARG from the two verifying child proofs.

This is a valid w-SIM-SND attack: the target $(a, c, 2)$ is false and was not queried to the simulator. The attack is not counted by newrt-SIM-SND, precisely because the simulator query and the target share the same root a.

Context-chaining. The preceding attack identifies what is missing from root tagging. Recall that in [DJJ+25] each recursive proof certifies a *block* of the computation: a level-ℓ proof covers 2^ℓ consecutive steps, forming an interval $[j, j + 2^\ell]$ with $2^\ell \mid j$ (a *dyadic* block), and it is obtained by merging the proofs for the two halves of this interval, its *child blocks* $[j, j + 2^{\ell-1}]$ and $[j + 2^{\ell-1}, j + 2^\ell]$. The statement of such a proof mentions only the encrypted endpoints ct_j and ct_{j+2^ℓ} of its block.

We need a recursive proof to be bound not only to this local ciphertext statement, but also to the IVC *context* in which it appears. By context we mean the portion of the computation preceding the block: its root and the encrypted states leading up to the block's left endpoint, which in particular fix the block's position. We bind each proof to its context by maintaining a hash chain across the blocks of the computation, as follows.

Let $H_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^\lambda$ be a collision-resistant hash function with public key hk. Consider a node spanning $[j, j + 2^\ell]$, with endpoint ciphertexts $(\mathsf{ct}_L, \mathsf{ct}_R)$. The node receives a *chain-in* value h_j summarizing the portion of the computation lying to its left, and computes a *chain-out* value by hashing its level, chain-in, and endpoint ciphertexts.

The next node to its right receives this value as its chain-in, and the chain starts from an initial value $\perp$. At levels $\ell \geq 1$, the chain-in h_j is included directly in the recursive BARG instance; at level 0, the chain-out is used as the message of the NIZK-SoK proof.

Thus a recursive proof is no longer bound only to its endpoint ciphertexts: it also carries a cryptographic digest of its left context. In particular, relocating a simulator-issued proof to a different point in the computation changes its chain-in. At higher levels this changes the *BARG statement* itself, while at level 0 it changes the NIZK-SoK message. Thus, by collision resistance, equality of the relevant chain values determine the corresponding left context; this is the anchoring property used in our soundness proof.

Furthermore, for the chain to pin a node's position, it is not enough to hash only its incoming chain value and endpoint ciphertexts. In the dyadic proof tree, the same tuple $(h, \mathsf{ct}_L, \mathsf{ct}_R)$ could, in principle, describe blocks at different levels. If the level were omitted from the hash, equality of two chain values would therefore not imply that the corresponding blocks have the same length. In our soundness proof, we would no longer be able to recover the same canonical dyadic prefix from a stale chain value, and hence could not pin the simulated and forged leaves to the same position. We therefore domain-separate the levels. We also separate the initial chain value from ordinary node hashes. Using a canonical encoding, we set $\perp := H_{\mathsf{hk}}(0)$, $\mathsf{Ch}_\ell(h, \mathsf{ct}_L, \mathsf{ct}_R) := H_{\mathsf{hk}}(1 \| \langle \ell \rangle \| h \| \mathsf{ct}_L \| \mathsf{ct}_R)$. Now equality of chain values, except via a hash collision, fixes not only the endpoint ciphertexts and the preceding chain value but also the level of the preceding dyadic block.

Using the chain in the IVC construction. We now incorporate these chain values into the IVC construction of [DJJ+25]. At level 0, the NIZK relation is unchanged: it still proves consistency of two endpoint ciphertexts with a valid SNARG proof. We instantiate it with a NIZK-SoK: a leaf with chain-in h and endpoints $(\mathsf{ct}_L, \mathsf{ct}_R)$ is proved under message $m = \mathsf{Ch}_0(h, \mathsf{ct}_L, \mathsf{ct}_R)$. Thus relocating a simulated leaf to a different context changes the message under which its proof must verify.

At higher levels, the chain-in becomes part of the recursive BARG instance. For a level-ℓ node with midpoint ct_M, the left child receives chain-in h, while the right child receives $h_M = \mathsf{Ch}_{\ell-1}(h, \mathsf{ct}_L, \mathsf{ct}_M)$, the chain-out of the left child. The recursive relation recomputes h_M and verifies the two child proofs under these respective contexts. Thus every merge enforces both *ciphertext adjacency* and *chain adjacency*: the right child must begin in exactly the context produced by the left child.

We next explain how this *context chaining* turns any reuse of simulated material into an anchor to an entire simulator prefix, which is the key to proving weak simulation soundness.

2.4 Proof overview of weak simulation soundness

From a false target to a false leaf. The soundness proof of [DJJ+25] starts with an accepting proof of a false computation. The verifier's adjacency checks imply that some root in the final dyadic frontier is bad: its endpoint ciphertexts either fail to decrypt to states or decrypt to states that are not related by the represented number of steps. A bad parent must have a bad child, since two valid child computations would concatenate to make the parent valid. Somewhere extraction from the BARG then supports a descent to an accepting bad level-0 leaf. Extracting its NIZK witness yields an accepting one-step SNARG proof of a false statement, contradicting SNARG soundness.

We use the same descent, but must additionally account for the simulated proofs available to our adversary. Suppose the adversary wins with a target $x^*_{\mathsf{in}} \overset{t^*}{\rightsquigarrow} x^*_{\mathsf{out}}$ and an accepting proof Π^*. A bad frontier root exists as above (Lemma E.6). At each recursive level, the BARG extracts only the child designated at its trapdoor setup. Averaging selects fixed directions, CRS indistinguishability allows switching from ordinary setups to

extraction setups, and index hiding permits changing the designated child. This gives a fixed-path descent producing an accepting bad leaf (Lemma E.7). If its message is *fresh*, that us, it was never submitted to the level-0 simulator, then NIZK simulation extractability yields its endpoint openings and an accepting one-step SNARG proof. Perfect PKE correctness and SNARG soundness rule out this fresh bad leaf (Lemma E.8). It now remains to handle a *stale* message.

A stale leaf identifies a unique query. Suppose the extracted leaf $[s^*, s^* + 1]$ has the same message as a recorded leaf in a simulator query. Since both messages are outputs of Ch_0, collision resistance binds the leaves to the same endpoint ciphertexts and incoming chain value. This chain value summarizes the canonical dyadic prefix preceding each leaf. For the target leaf, we obtain the prefix descriptors from the preceding frontier roots and, whenever the extraction path turns right, the left-child descriptor exposed by its parent. At each merge, the relation proved by the BARG requires the child proofs to verify under the endpoint ciphertexts and incoming chain values prescribed by the parent. Somewhere extraction guarantees a witness satisfying this relation, so each extracted child inherits precisely that context. Together with the target verifier's checks on the frontier roots, this ensures that the exposed descriptor list folds to the target leaf's incoming chain value.

To see why the common incoming chain value binds the entire prefix, compare the descriptors exposed by extraction with those in the simulation record corresponding to the simulation query. Starting from the common chain value, the final descriptor in each nonempty list supplies a hash input producing that value. By collision resistance, these inputs cannot differ except with negligible probability, so they identify the same block and preceding chain value. We can therefore repeat the comparison with the two shortened lists. Since the level is included in the hashed input, matching descriptors represent blocks of the same length. The two backwards comparisons must also reach the empty prefix together: its chain value is $\perp = H_{\mathsf{hk}}(0)$, whereas a non-empty prefix's chain value is obtained by hashing a block encoding beginning with 1. If one list became empty before the other, their common chain value would therefore have two distinct hash inputs. Collision resistance rules this out except with negligible probability, so the lists must terminate together.

Thus the comparison establishes equality of the entire descriptor lists: their block lengths pin the leaves to the same position, and their first endpoint identifies the same starting ciphertext when the prefix is nonempty; for an empty prefix, the matching leaf endpoints identify the starting ciphertext (Lemma E.5). The key is that both lists already supply the hash inputs needed for this comparison; so we do not need to extract from the proofs of the preceding blocks.

The preceding argument forces every query occurrence containing a leaf with this stale message to have the target's starting ciphertext. If two distinct occurrences matched, their starting ciphertexts would therefore collide. The simulator generates each starting ciphertext with fresh independent randomness, and CPA security together with perfect correctness bounds the probability of such a collision by a negligible quantity (Lemma E.9), even when the queries have the same root state. Hence the stale leaf identifies a unique query occurrence $q^* = [x^*_{\mathsf{in}} \overset{d^*}{\rightsquigarrow} y^*]_?$ whose prefix context agrees with the target's through that leaf.

Completing the inherited prefix. Suppose the target and q^* agree through an interior boundary $0 < r < d^*$. At this boundary, the simulator encrypts 0^n in $\mathsf{ct}_r^{q^*}$ but does not reveal the encryption coins. Unexplainability ensures that knowing the plaintext, even together with the secret key, does not enable an efficient algorithm to produce a valid opening of this ciphertext. This prevents the target from ending at r, since its public endpoint opening would supply precisely such an opening. It also prevents the target from continuing with a fresh message on the next leaf $[r, r + 1]$, since NIZK simulation extractability would make an opening of the same ciphertext extractable. Thus the target must continue with a stale message, which the preceding binding argument and query uniqueness identify with the next leaf of q^*. Hence agreement through r advances to agreement through $r + 1$.

To advance the matched prefix, we move from the extraction path for the leaf ending at r to the path for the next leaf, $[r, r + 1]$. These paths may select different children at some BARG levels. However, the prefix agreement can be checked from the public frontier and the descriptors exposed before the paths diverge, allowing index-hiding to justify this change while preserving the agreement already established (Lemma E.10). Applying the preceding argument at each interior boundary shows that the target must follow q^* all the way to its endpoint (Lemma E.11). It cannot stop earlier, and at the query endpoint it has the same ciphertext and hence the same state y^*, so $d^* \leq t^*$. Equality would make the target statement identical to the queried

statement, which the winning condition excludes. Therefore $d^* < t^*$, leaving a suffix from y^* to x^*_{out}. If this suffix were true, the target would be compliant with the promise leap supplied by q^*. Since the adversary wins, the suffix must be false: $y^* \overset{t^* - d^*}{\not\rightsquigarrow} x^*_{\mathsf{out}}$.

A second descent gives a fresh false leaf. To locate a bad block within the false suffix, consider the boundary view at d^* supplied by the completion argument. The descriptors in this view, together with the later frontier roots, give a partition of $[d^*, t^*]$ into adjacent dyadic blocks.

The BARG merge conditions and the target verifier's adjacency checks ensure that these blocks concatenate from the matched query endpoint to the target endpoint. If they all represented valid computations, their concatenation would prove the suffix true. Hence at least one block must be bad (Lemma E.12).

We now apply the same BARG extraction and CRS-switch argument used in the first descent to obtain an accepting bad leaf $[r, r+1]$ inside this suffix, so $r \geq d^*$. If its message were stale, the earlier hash-chain argument would place it at the same position in a simulator query with the target's starting ciphertext. Query uniqueness forces that occurrence to be q^*, but every leaf of q^* has position strictly less than d^*. This contradicts $r \geq d^*$. Hence the second bad leaf must have a fresh message, and leaf soundness gives the final contradiction (Lemma E.13).

Security losses and proof roadmap. Turning this argument into reductions incurs losses polynomial in T, the query bound, and 2^n, because the relevant blocks, paths, and instances must be fixed before setup. In particular, using non-adaptive SNARG soundness requires fixing the endpoint state pair in advance, at a cost of 2^{2n} in the success probability. Following [DJJ+25], we use sub-exponential primitive security and the parameter choice of Theorem 6.1 to absorb these losses even when T is exponential.

The reductions remain efficient because each uses fixed indices and performs only $O(\log T)$ recursive extractions. The prefix-completion argument combines guarantees for individual boundaries, while each reduction examines only one fixed boundary. Answering a simulation query of depth d requires processing its d steps, which is polynomial in the query's unary input length. Figure 4 summarizes the two descents; the formal bounds and reductions appear in Appendix E.2.

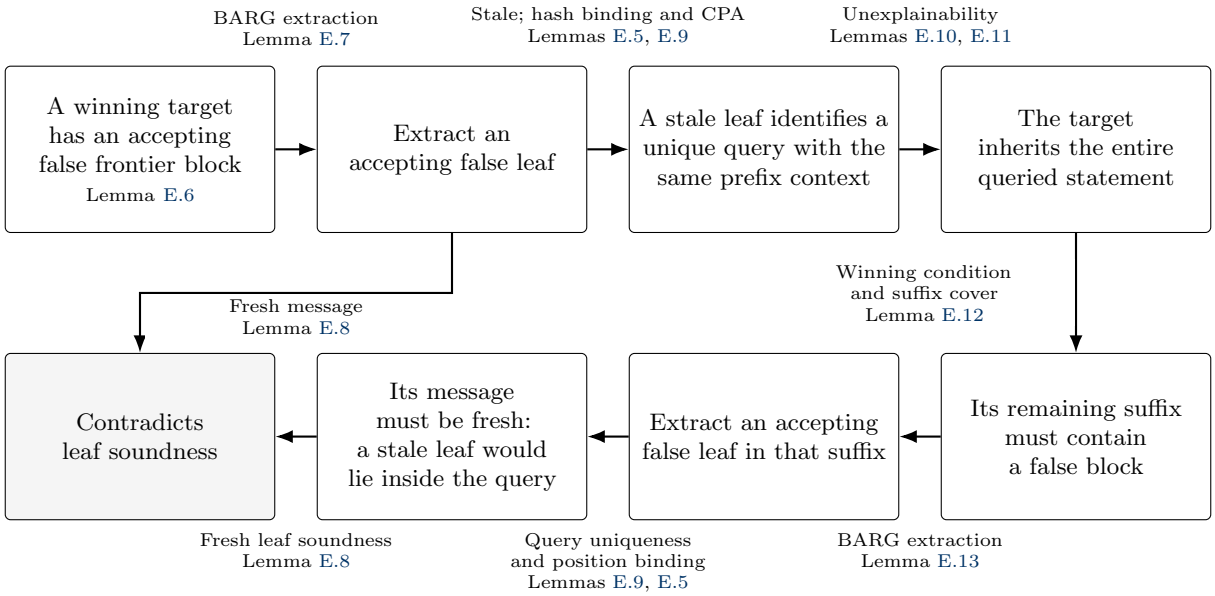


Fig. 4: The proof sketch with the two descents in the weak simulation-soundness proof.

2.5 Towards SIM-SND

Context-chaining prevents simulator-issued material being moved to a different position or context, which is exactly the additional protection needed to obtain w-SIM-SND. Full SIM-SND, however, must also rule out a new proof of the *same false, non-compliant queried statement*, and context-chaining does not by itself address this.

Indeed, w-SIM-SND alone does not exclude such proof modifications. To illustrate this distinction, take a w-SIM-SND IVC whose simulator returns accepting proofs for arbitrary well-formed queries, append a bit to each proof, and let the verifier ignore it. The prover and simulator append 0, while the incremental prover discards the appended bit of its input proof before invoking the original prover. This modification preserves w-SIM-SND, since any accepting forgery for a fresh statement yields an underlying forgery after removing the bit. On a false one-step simulator query $(x, y, 1)$, however, the adversary can change the returned proof from $\Pi\|0$ to $\Pi\|1$. With this as the only query, the target is false and non-compliant, and the statement–proof pair is fresh; thus the modified scheme fails SIM-SND.

In our context-chained construction, the NIZK-SoK definition already requires extraction for a fresh statement–proof–message tuple, even when the statement and message were previously queried. Therefore a different accepting level-0 proof for a false one-step simulator query would yield endpoint openings and an accepting SNARG proof of the false transition, contradicting SNARG soundness. Thus proof-freshness extraction at level 0 is already part of our assumptions. It does not, however, establish freshness guarantees for the entire IVC proof: a new recursive aggregate may still contain exactly the simulated child proofs.

The remaining gap from w-SIM-SND to SIM-SND is therefore qualitatively different from the gap between newrt-SIM-SND and w-SIM-SND: it concerns proof-string freshness, rather than relocation of simulated material. The gap can potentially be cloased by strengtheing the underlying primitives: *unique* accepting BARG proofs for each fixed batch and a *coin-binding* PKE. Informally, coin binding says that it is infeasible to find $r \neq r'$ such that $\mathsf{Enc}(\mathsf{pk}, x; r) = \mathsf{Enc}(\mathsf{pk}, x; r')$. It is needed because the endpoint coins are explicitly included in the IVC proof: a second opening would preserve the endpoint ciphertext and every recursive check, while producing a different proof string for the same queried statement. BARG proof uniqueness would similarly prevent changing a recursive aggregate proof while keeping its batch fixed.

Establishing full SIM-SND would still require showing that any different accepting proof of a false, non-compliant queried statement yields a fresh extractable component that contradicts the security assumptions or violates the additional binding or uniqueness properties. We leave a systematic study of SIM-SND IVC, including minimal assumptions and concrete realizations, to future work.

3 Preliminaries

3.1 Non-interactive zero-knowledge proof (NIZK) and signatures of knowledge

We now define non-interactive zero-knowledge arguments (NIZKs) [BFM88] and NIZK signatures of knowledge (NIZK-SoK) [CL06]. Informally, a NIZK lets a prover convince a verifier that a statement x is true—non-interactively and without revealing a witness; a NIZK signature of knowledge additionally binds a message m to the proof, so that the proof doubles as a signature on m that only a party knowing a witness for x could have produced. We capture both notions with a single definition: a NIZK is precisely the special case of a NIZK signature of knowledge in which the message is fixed to $m = \perp$ (so that the message input plays no role). Consequently, we do not state the NIZK definition separately.

We now define a NIZK signature of knowledge (NIZK-SoK). The prover and the verifier additionally take a message $m \in \{0,1\}^*$, and security guarantees that the message m is bound to the proof in a non-malleable way. Here the proof π produced on a message m acts as a signature on m that can only be generated by a party who knows a witness w for the statement x.

Definition 3.1 (NIZK Signature of Knowledge [CL06]) *A NIZK signature of knowledge for an NP relation $\mathcal{R}$ is a tuple of algorithms* $\mathsf{NIZK\text{-}SoK} = (\mathsf{Setup}, \mathsf{TdSetup}, \mathcal{P}, \mathcal{V}, \mathsf{Sim}, \mathsf{E})$*:*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R})$*: on input the security parameter λ and an NP relation $\mathcal{R}$, outputs a common reference string* crs*.*
- $(\mathsf{tdcrs}, \tau, \mathsf{td}) \leftarrow \mathsf{TdSetup}(1^\lambda, \mathcal{R})$*: as in* NIZK*, outputs a trapdoor CRS, an extraction trapdoor τ, and a simulation trapdoor* td*.*

- $\pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w, m)$*: takes the additional input of a message* $m \in \{0,1\}^*$ *and outputs a proof* π *that binds the message* m *to the statement* x*.*
- $b \leftarrow \mathcal{V}(\mathsf{crs}, x, \pi, m)$*: takes the additional input of a message* $m \in \{0,1\}^*$ *and outputs a bit* $b \in \{0,1\}$*.*
- $\pi \leftarrow \mathsf{Sim}(\mathsf{tdcrs}, \mathsf{td}, x, m)$*: simulates a proof on input the simulation trapdoor and a message* m*, without using a witness.*
- $w^* \leftarrow \mathsf{E}(\tau, x, \pi, m)$*: extracts a witness from a valid proof under the trapdoor mode.*

Properties. The properties of a NIZK signature of knowledge are inherited from those of a (simulation-extractable) NIZK, with the message m *included in every invocation of* $\mathcal{P}$*,* $\mathcal{V}$*,* Sim *and* E*, and with the simulation oracle now keyed by* (x, m)*.*

1. **Completeness** *For all* $\lambda \in \mathbb{N}$*, every NP relation* $\mathcal{R}$*, every message* $m \in \{0,1\}^*$*, and every* $(x, w) \in \mathcal{R}$*,*

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

2. **Zero knowledge** *There exist PPT algorithms* $\mathsf{TdSetup}$ *and* Sim *such that for any non-uniform PPT distinguisher* $\mathcal{A}$*, the following two distributions are computationally indistinguishable: a real proof produced by* $\mathcal{P}(\mathsf{crs}, x, w, m)$ *under an honest* crs*, and a simulated proof produced by* $\mathsf{Sim}(\mathsf{tdcrs}, \mathsf{td}, x, m)$ *under a trapdoor* tdcrs*, where the distinguisher may adaptively choose* (x, w, m) *with* $(x, w) \in \mathcal{R}$*. Formally, for any non-uniform PPT adversary* $\mathcal{A}$*, the following is negligible in* λ*:*

$$\left| \Pr\left[\mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1 \ : \ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R})\right] - \Pr\left[\mathcal{A}^{\mathsf{Sim}'(\mathsf{td},\cdot,\cdot,\cdot)}(\mathsf{tdcrs}) = 1 \ : \ (\mathsf{tdcrs}, \mathsf{td}, \tau) \leftarrow \mathsf{TdSetup}(1^\lambda, \mathcal{R})\right] \right|$$

where both $\mathcal{P}(\mathsf{crs}, x, w, m)$ *and* $\mathsf{Sim}'(\mathsf{td}, x, w, m)$ *return* $\perp$ *if* $(x, w) \notin \mathcal{R}$*. If* $(x, w) \in \mathcal{R}$*,* Sim' *invokes* $\mathsf{Sim}(\mathsf{tdcrs}, \mathsf{td}, x, m)$*.*

3. **Simulation Extractability.** *A NIZK signature of knowledge is simulation extractable if there exists a PPT extractor* E *such that for any non-uniform PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the following probability is negligible in* λ*:*

$$\Pr\left[\begin{array}{l} \mathcal{V}(\mathsf{tdcrs}, x, \pi, m) = 1 \\ \wedge\ (x, \pi, m) \notin \mathcal{Q} \\ \wedge\ (x, w^*) \notin \mathcal{R} \end{array} \ : \ \begin{array}{l} (\mathcal{R}, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\mathsf{tdcrs}, \tau, \mathsf{td}) \leftarrow \mathsf{TdSetup}(1^\lambda, \mathcal{R}) \\ (x, \pi, m) \leftarrow \mathcal{A}_2^{\mathsf{Sim}(\mathsf{tdcrs},\mathsf{td},\cdot,\cdot)}(\mathsf{st}, \mathsf{tdcrs}) \\ w^* \leftarrow \mathsf{E}(\tau, x, \pi, m) \end{array}\right],$$

where $\mathcal{Q}$ *is the list of query-response tuples* $\{(x^{(j)}, \pi_j, m_j)\}_{j \in |\mathcal{Q}|}$ *consisting of the statement* $x^{(j)}$ *and the message* m_j *queried by the adversary to the simulator, together with the simulator's response* π_j*.*

3.2 Succinct non-interactive argument (SNARG)

Definition 3.2 (SNARG) *Let* $R : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ *be an NP relation. A* succinct non-interactive argument *(SNARG) for* R *is a tuple of PPT algorithms*

$$\mathsf{SNARG} = (\mathsf{SNARG.Gen}, \mathsf{SNARG.P}, \mathsf{SNARG.V})$$

with the following syntax:

- $\mathsf{SNARG.Gen}(1^\lambda, 1^n, 1^m, R)$ *outputs* $(\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V})$*.*
- $\mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, x, w)$*, for* $x \in \{0,1\}^n$ *and* $w \in \{0,1\}^m$*, outputs a proof* π*.*
- $\mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, x, \pi)$ *outputs a bit.*

It satisfies the following properties.

Perfect completeness. *For every* (x, w) *such that* $R(x, w) = 1$*,*

$$\Pr\left[\mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, x, \pi) = 1 \ : \ \begin{array}{l} (\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}) \leftarrow \mathsf{SNARG.Gen}(1^\lambda, 1^n, 1^m, R), \\ \pi \leftarrow \mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, x, w) \end{array}\right] = 1.$$

Succinctness. *There exists a polynomial p such that $|\pi| = p(\lambda, \log|R|)$, and the verifier runs in time* $\mathsf{poly}(\lambda, n, \log|R|)$.

Non-adaptive soundness. *For every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the following is* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[x^\star \notin \mathcal{L}_R \ \wedge\ \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, x^\star, \pi^\star) = 1 \ :\ \begin{array}{l}(1^n, 1^m, R, x^\star, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda),\\ (\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}) \leftarrow \mathsf{SNARG.Gen}(1^\lambda, 1^n, 1^m, R),\\ \pi^\star \leftarrow \mathcal{A}_2(\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, x^\star, \mathsf{st})\end{array}\right]$$

3.3 Public-key encryption

Definition 3.3 (Public-key encryption) *A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *consists of the following PPT algorithms.*

- $\mathsf{PKE.Gen}(1^\lambda, 1^n)$ *outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{PKE.Enc}(\mathsf{pk}, x; r)$*, for a message $x \in \{0,1\}^n$ and randomness r, outputs a ciphertext* ct.
- $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct})$ *outputs a message in $\{0,1\}^n$ or $\perp$.*

It satisfies the following properties.

Perfect correctness. *For every $\lambda, n \in \mathbb{N}$ and every $x \in \{0,1\}^n$,*

$$\Pr\left[\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}) = x \ :\ \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda, 1^n),\\ \mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x)\end{array}\right] = 1.$$

CPA security. *For every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\left|\Pr\left[b' = 1 \ :\ \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda, 1^n),\\ (x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}),\\ \mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_0),\\ b' \leftarrow \mathcal{A}_2(\mathsf{ct}, \mathsf{st})\end{array}\right] - \Pr\left[b' = 1 \ :\ \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda, 1^n),\\ (x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}),\\ \mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_1),\\ b' \leftarrow \mathcal{A}_2(\mathsf{ct}, \mathsf{st})\end{array}\right]\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{A}_1$ outputs $x_0, x_1 \in \{0,1\}^n$.

3.4 Batch arguments (BARGs)

We consider the NP-complete language of Boolean circuit satisfiability. For a Boolean circuit $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ with t wires, we associate wires $1, \ldots, n$ with the bits of the statement $x_1, \ldots, x_n$, and wires $n+1, \ldots, n+m$ with the bits of the witness $w_1, \ldots, w_m$, respectively. Wire t is associated with the output wire, and we have $t \leq n + m + s$, where s is the size of C, that is, the number of gates in it.

Definition 3.4 (Circuit Satisfiability) *Let $\mathcal{L}_{\mathrm{CSAT}} = \{(C, x) \mid \exists w \in \{0,1\}^m : C(x, w) = 1\}$ be the language of Boolean circuit satisfiability, where $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ is a Boolean circuit and $x \in \{0,1\}^n$ is a statement. For a positive integer $T \in \mathbb{N}$, we define the* batch circuit satisfiability *language $\mathcal{L}_{\text{B-CSAT},T}$ as follows:*

$$\mathcal{L}_{\text{B-CSAT},T} = \{(C, x_1, \ldots, x_T) \mid \forall i \in [T] : \exists w_i \in \{0,1\}^m : C(x_i, w_i) = 1\},$$

where $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ is a Boolean circuit and $x_1, \ldots, x_T \in \{0,1\}^n$ are instances.

In our treatment we use the BARG definition of [CJJ21].

Definition 3.5 (Batch Argument for NP) *A non-interactive batch argument (BARG) for circuit-satisfiability is a tuple of algorithms $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ given by:*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^T, 1^s, 1^n)$. Setup *takes as input the security parameter λ, the number of instance $T \in \mathbb{N}$, an upper bound s on the size of the circuit, and the input length n, and outputs a common reference string* crs.
- $\pi \leftarrow \mathcal{P}(\mathsf{crs}, C, (x_1, \ldots, x_T), (w_1, \ldots, w_T))$. *$\mathcal{P}$ takes as input the CRS, a Boolean circuit $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, statements $x_1, \ldots, x_T \in \{0,1\}^n$, and witnesses $w_1, \ldots, w_T \in \{0,1\}^m$, and outputs a proof π.*

– $b \leftarrow \mathcal{V}(\mathsf{crs}, C, (x_1, \ldots, x_T), \pi)$. $\mathcal{V}$ *takes as input the CRS, a Boolean circuit* $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, *statements* $x_1, \ldots, x_T \in \{0,1\}^n$, *and a proof* π, *and outputs a bit* $b \in \{0,1\}$.

A BARG should satisfy the following properties.

Definition 3.6 (Completeness) *For all* $T, s = s(\lambda), n = n(\lambda) \in \mathbb{N}$, *circuit* $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ *of size* $|C| \leq s$, *all statements* x_i *and witnesses* w_i *such that* $C(x_i, w_i) = 1$ *for all* $i \in [T]$

$$\Pr\left[\mathcal{V}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 \ : \ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^T, 1^s, 1^n) \\ \pi \leftarrow \mathcal{P}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w}) \end{array}\right] = 1$$

where $\mathbf{x} = (x_1, \ldots, x_T), \mathbf{w} = (w_1, \ldots, w_T)$.

Definition 3.7 (Somewhere Argument of Knowledge) *A BARG is a somewhere argument of knowledge (sweBARG) if there exist PPT algorithm* $\mathsf{TdSetup}$ *and a deterministic polynomial-time algorithm* E *such that:*

– $(\mathsf{tdcrs}, \tau) \leftarrow \mathsf{TdSetup}(1^\lambda, 1^T, 1^s, 1^n, i^*)$. $\mathsf{TdSetup}$ *takes as input the security parameter* λ, *the number of instances* $T \in \mathbb{N}$, *an upper bound* s *on the size of the verification circuit of the relation, the input length* n, *and an index* i^*, *and outputs a common reference string* tdcrs, *and and extraction trapdoor* τ.
– $w^* \leftarrow \mathsf{E}(\tau, C, x_1, \ldots, x_T, \pi)$. E *takes as input the trapdoor* τ, *statements* $x_1, \ldots, x_T$, *and a proof* π *and outputs a witness* $w^* \in \{0,1\}^m$.

The following two properties should hold.

– **CRS indistinguishability:** *For any non-uniform PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *and any polynomials* $T = T(\lambda)$, $s = s(\lambda)$ *and* $n = n(\lambda)$, *the following is negligible in* λ:

$$\left|\Pr\left[\mathcal{A}_2(\mathsf{st}, \mathsf{crs}) = 1 \ : \ \begin{array}{c} (i^*, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, 1^T, 1^s, 1^n), \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^T, 1^s, 1^n) \end{array}\right] - \right.$$

$$\left.\Pr\left[\mathcal{A}_2(\mathsf{st}, \mathsf{tdcrs}) = 1 \ : \ \begin{array}{c} (i^*, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, 1^T, 1^s, 1^n), \\ (\mathsf{tdcrs}, \tau) \leftarrow \mathsf{TdSetup}(1^\lambda, 1^T, 1^s, 1^n, i^*) \end{array}\right]\right|$$

– **Somewhere extraction:** *For any non-uniform PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *any* $s = s(\lambda), n = n(\lambda) \in \mathbb{N}$, *and any polynomial* $T = T(\lambda)$, *the following probability is negligible in* λ:

$$\Pr\left[\begin{array}{c} \mathcal{V}(\mathsf{tdcrs}, C, (x_1, \ldots, x_T), \pi) = 1 \\ C(x_{i^*}, w^*) \neq 1 \end{array} \ : \ \begin{array}{c} (i^*, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, 1^T, 1^s, 1^n), \\ (\mathsf{tdcrs}, \tau) \leftarrow \mathsf{TdSetup}(1^\lambda, 1^T, 1^s, 1^n, i^*), \\ (C, x_1, \ldots, x_T, \pi) \leftarrow \mathcal{A}_2(\mathsf{tdcrs}, \mathsf{st}) \\ w^* \leftarrow \mathsf{E}(\tau, C, (x_1, \ldots, x_T), \pi) \end{array}\right]$$

Definition 3.8 (Succinctness) Π_{BARG} *is succinct if the following properties hold.*

– **Succinct CRS:** *CRSs have length* $\mathsf{poly}(\lambda, \log T, s)$ *(parameter* n *is assumed to be s.t.* $n \leq s$*).*
– **Succinct Proofs:** *Proofs have length* $\mathsf{poly}(\lambda, \log T, s)$. *A BARG is said to be rate-1 if there exists a fixed polynomial* p *such that, for every batch of* T *instances of a relation circuit of size at most* s, *each with witness length* m, *the proof length is at most* $m + p(\lambda, \log T, \log s)$).
– **Succinct Verification:** *The verification algorithm runs in time* $\mathsf{poly}(\lambda, T, n) + \mathsf{poly}(\lambda, \log T, s)$.

We say that a BARG $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ has *offline-online verification* if the verification algorithm $\mathcal{V}$ can be split into a preprocessing phase and an online phase as follows:

– **Preprocessing:** There exists a deterministic algorithm $\mathsf{PreVerify}$ that takes as input a CRS crs and T instances $x_1, \ldots, x_T$, and outputs a short digest d of length $|d| = \mathsf{poly}(\lambda, \log T, |x_1|)$.
– **Online Verification:** There exists an online verification algorithm $\mathsf{OnVerify}$ that takes as input the digest d, a circuit C, a BARG proof π, and outputs accept or reject. Moreover, the running time of $\mathsf{OnVerify}$ is $\mathsf{poly}(\lambda, \log T, |C|)$.

4 Defining Simulation-Soundness in IVC

In this section we explain our modeling of simulation-soundness of IVC. We start by providing some background and notation on incremental computations and incrementally verifiable computation. The material in this section is novel, with the exception of the background material in this subsection and in Section 4.1.

Background: Incremental Computations. We begin with some preliminaries on incremental computation.

Step functions. In the context of incremental computation we assume a fixed "step" circuit C, $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. We will often keep C—as well as the parameters n, m—implicit from context.
For $x, x' \in \{0,1\}^*$ we define the following "step" notation:

$$x \longrightarrow x' \iff x, x' \in \{0,1\}^n \ \wedge \ \exists w \in \{0,1\}^m \text{ such that } C(x,w) = x'.$$

Reachability. We will also define the following convenient notation for "reachability through multiple steps". We define: $x \overset{0}{\rightsquigarrow} x' \iff x = x' \ \wedge \ x, x' \in \{0,1\}^n$, and for $n > 1, n \in \mathbb{N}$, $x \overset{n}{\rightsquigarrow} x' \iff \exists x_0, x_1, \ldots, x_{n-1}, x_n$ with $x_0 := x, x_n := x'$ such that $x_{i-1} \longrightarrow x_i$ for all $i \in [n]$.

Statements for incremental computation. Intuitively, $x_0 \overset{t}{\rightsquigarrow} x_t$ parses as "we can reach x_t from x_0 by applying C t times". Given x_0, x_t and t, in incremental proof systems we are interested in determining and certifying the truth of reachability claims such as the one above. We denote such *incremental statements* (whether true or not) as triples (x_0, x_t, t). We also define the following language:

$$\mathcal{L}_{\mathsf{IVC}} = \{(\mathcal{C}, x_0, x_t, t) : \ \mathcal{C} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n \ \wedge \ x_0 \overset{t}{\rightsquigarrow} x_t\}$$

4.1 Background on IVC

Definition 4.1 (Incrementally Verifiable Computation (IVC) [Val08]) *A incrementally verifiable computation (IVC) scheme for the language $\mathcal{L}_{\mathsf{IVC}}$ consists of a tuple of PPT algorithms $\mathsf{IVC} = (\mathsf{IVC.Gen}, \mathsf{IVC}.\mathcal{P}, \mathsf{IVC}.\mathcal{V})$ with the following syntax.*

- $\mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)$ *takes as input the security parameter λ, an iteration bound T, input length n, witness length m, a circuit $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, and outputs a common reference string* crs.
- $\mathsf{IVC}.\mathcal{P}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \pi_{i-1}, w_i)$ *takes as input the CRS* crs*, inputs $x_0, x_{i-1} \in \{0,1\}^n$, an index $i-1 \in [T-1]$, a proof π_{i-1}, and a witness $w_i \in \{0,1\}^m$, and outputs a proof π_i.*
- $\mathsf{IVC}.\mathcal{V}(\mathsf{crs}, (x_0, x_t, t), \pi_t)$ *takes as input the CRS* crs*, inputs $x_0, x_t \in \{0,1\}^n$, an index $t \in [T]$, a proof π_t, and outputs a bit in $\{0,1\}$.*

The scheme must satisfy completeness, efficiency, soundness and (in our setting) zero knowledge.

Completeness. For all λ, all iteration bounds $T \leq 2^\lambda$, all $t \leq T$, all polynomially bounded n, m, all polynomial-size circuits C, all $x_0 \in \{0,1\}^n$, and all witnesses $w_1, \ldots, w_t \in \{0,1\}^m$, the following probability is 1:

$$\Pr\left[\mathsf{IVC}.\mathcal{V}(\mathsf{crs}, (x_0, x_t, t), \pi_t) = 1 \ : \ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C) \\ \forall i \in [t] : x_i := C(x_{i-1}, w_i) \\ \forall i \in [t] : \pi_i \leftarrow \mathsf{IVC}.\mathcal{P}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \pi_{i-1}, w_i) \end{array}\right]$$

Efficiency. There exist polynomials p_1, p_2 such that $\mathsf{IVC.Gen}, \mathsf{IVC}.\mathcal{P}, \mathsf{IVC}.\mathcal{V}$ run in time $p_1(\lambda, \log T, |C|)$, and the proof size $|\pi_i|$ is $p_2(\lambda, n, m, \log T, \log |C|)$.

An IVC scheme satisfies soundness and zero knowledge.

Soundness. For all $\lambda \in \mathbb{N}$, all iteration bounds $T \leq 2^\lambda$, all polynomials n, m, all polynomial size circuits $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} \mathsf{IVC}.\mathcal{V}(\mathsf{crs}, (x_0, x_t, t), \pi_t) = 1 \\ (C, x_0, x_t, t) \notin \mathcal{L}_{\mathsf{IVC}} \\ t \leq T \end{array} \ : \ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C) \\ (x_0, x_t, t, \pi_t) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda)$$

Zero knowledge. There exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for all $\lambda \in \mathbb{N}$, all iteration bounds $T \leq 2^\lambda$, all polynomially bounded n, m, all polynomial-size circuits $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, and all PPT adversaries $\mathcal{A}$, the following is negligible in λ:

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathrm{real}}(\mathsf{crs},\cdot)}(\mathsf{crs}) = 1 : \ \mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)\right] \\ \quad - \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathrm{sim}}(\mathsf{tdcrs},\mathsf{st},\cdot)}(\mathsf{tdcrs}) = 1 : \ (\mathsf{tdcrs}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, T, 1^n, 1^m, C)\right] \end{array} \right|.$$

The two oracles answer queries of the form $((x_0, x_t, 1^t), (w_1, \ldots, w_t))$. They return $\perp$ unless the query is well formed, $1 \leq t \leq T$, $x_0, x_t \in \{0,1\}^n$, every $w_i \in \{0,1\}^m$, and the states defined by $z_0 := x_0$ and $z_i := C(z_{i-1}, w_i)$ satisfy $z_t = x_t$. For a valid query:

- $\mathcal{O}_{\mathrm{real}}$ sets $\pi_0 := \perp$, and computes $\pi_i \leftarrow \mathsf{IVC}.\mathcal{P}(\mathsf{crs}, (x_0, z_{i-1}, i-1), \pi_{i-1}, w_i)$ for $i = 1, \ldots, t$. It returns only π_t.
- $\mathcal{O}_{\mathrm{sim}}$ returns $\mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{st}, x_0, x_t, 1^t)$.

4.2 Auxiliary notions: promise leaps and compliance

Before we provide the definition of simulation-soundness, below we formalize the notions of promise leaps and compliance.

Given $u, v \in \{0,1\}^n$ and $d \in \mathbb{N}$ we define a *promise leap* as the triple (u, v, d). We call this a "promise" leap because we will intuitively use it as an "assumed promise" for the statement "v can be reached from u in d steps". To distinguish a *promise* leap from the actual logical predicate "$u \overset{d}{\rightsquigarrow} v$", we will explicitly denote the promise leap (u, v, d) as $\left[u \overset{d}{\underset{!}{\rightsquigarrow}} v\right]$. Some intuitions around these notions, with examples of compliant and non-compliant statements, are in Appendix B.

Definition 4.2 (Set of promise leaps) *Let $\boldsymbol{u} = (u_i)_i$, $\boldsymbol{v} = (v_i)_i$ be two vectors with ℓ elements such that for all $i \in [\ell]$ $u_i, v_i \in \{0,1\}^n$. Let $(d_i)_i \in \mathbb{N}^\ell$. We call the unordered set $P = \left\{\left[u_1 \overset{d_1}{\underset{!}{\rightsquigarrow}} v_1\right], \ldots, \left[u_\ell \overset{d_\ell}{\underset{!}{\rightsquigarrow}} v_\ell\right]\right\}$ a set of promise leaps.*

Definition 4.3 *We say that a statement (x_0, x_t, t) is compliant via a set of promise leaps P if there exists $\left[x_0 \overset{d}{\underset{!}{\rightsquigarrow}} x'\right] \in P$ with $d < t$ and $x' \overset{t-d}{\rightsquigarrow} x_t$ holds.*

4.3 Simulation-soundness notions for IVC

We are now ready to formalize simulation-soundness for IVC. The notion comes in three strengths—SIM-SND, w-SIM-SND, and newrt-SIM-SND—which share a common experiment and differ only in the adversary's winning condition.

Definition 4.4 (Simulation-soundness for IVC) *Let $\mathsf{IVC} = (\mathsf{IVC.Gen}, \mathsf{IVC}.\mathcal{P}, \mathsf{IVC}.\mathcal{V})$ be an IVC scheme and let $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ be its simulator, exactly as in the zero-knowledge property of Definition 4.1. Let Sim' be the associated* simulation oracle*: on a query $\left[x \overset{d}{\underset{?}{\rightsquigarrow}} x''\right]$ for a (possibly false) reachability statement, $\mathsf{Sim}'(\mathsf{st}, \cdot)$ returns the simulated proof $\mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{st}, x, x'', 1^d)$. We say that IVC satisfies X-simulation-soundness, for $X \in \{\mathsf{SIM\text{-}SND}, \mathsf{w\text{-}SIM\text{-}SND}, \mathsf{newrt\text{-}SIM\text{-}SND}\}$, if for all $\lambda \in \mathbb{N}$, all iteration bounds $T \leq 2^\lambda$, all polynomials n, m, all polynomial size circuits $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, all PPT adversaries $\mathcal{A}$ the following probability is negligible in λ:*

$$\Pr\left[\begin{array}{l} \mathsf{IVC}.\mathcal{V}(\mathsf{tdcrs}, x^*, \pi^*) = 1 \\ \wedge \ \ \mathsf{WinCond}^X(x^*, \pi^*, Q_S) \end{array} \ : \ \begin{array}{l} (\mathsf{tdcrs}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, T, 1^n, 1^m, C) \\ (x^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Sim}'(\mathsf{st},\cdot)}(\mathsf{tdcrs}) \end{array}\right],$$

where the target statement is parsed as $x^* = \left[x^*_{in} \overset{d^*}{\underset{?}{\rightsquigarrow}} x^*_{out}\right]$; the set Q_S collects the query–response pairs $\left(\left[x_i \overset{d_i}{\underset{?}{\rightsquigarrow}} x''_i\right], \pi_i\right)$ produced in $\mathcal{A}$'s interaction with Sim'; The winning condition $\mathsf{WinCond}^X$ for each notion X is given in Figure 5.

Winning Conditions $\mathsf{WinCond}^X(x^*, \pi^*, Q_S)$ for IVC Simulation-Soundness

Parse x^* as $x^*_{\text{in}} \overset{d^*}{\rightsquigarrow} x^*_{\text{out}}$; define this set of promise leaps

$$P(x^*_{\text{in}}) \;:=\; \left\{ \left[x^*_{\text{in}} \underset{!}{\overset{d}{\rightsquigarrow}} x''\right] \;:\; \left(\left[x^*_{\text{in}} \underset{?}{\overset{d}{\rightsquigarrow}} x''\right], __\right) \in Q_S \right\}.$$

Winning condition for SIM-SND:

$$x^*_{\text{in}} \overset{d^*}{\not\rightsquigarrow} x^*_{\text{out}} \;\wedge\; x^*_{\text{in}} \overset{d^*}{\rightsquigarrow} x^*_{\text{out}} \textit{ not} \text{ compliant via } P(x^*_{\text{in}}) \;\wedge\; \left(\left[x^*_{\text{in}} \underset{?}{\overset{d^*}{\rightsquigarrow}} x^*_{\text{out}}\right], \pi^*\right) \notin Q_S{}^{1}$$

Winning condition for w-SIM-SND:

$$x^*_{\text{in}} \overset{d^*}{\not\rightsquigarrow} x^*_{\text{out}} \;\wedge\; x^*_{\text{in}} \overset{d^*}{\rightsquigarrow} x^*_{\text{out}} \textit{ not} \text{ compliant via } P(x^*_{\text{in}}) \;\wedge\; \left(\left[x^*_{\text{in}} \underset{?}{\overset{d^*}{\rightsquigarrow}} x^*_{\text{out}}\right], __\right) \notin Q_S{}^{2}$$

Winning condition for newrt-SIM-SND:

$$x^*_{\text{in}} \overset{d^*}{\not\rightsquigarrow} x^*_{\text{out}} \;\wedge\; \forall d.\; \left(\left[x^*_{\text{in}} \underset{?}{\overset{d}{\rightsquigarrow}} __\right], __\right) \notin Q_S{}^{3}$$

---

[1] π^* is not a proof for x^* obtained from the simulator.
[2] the proof π^* must be for a statement that has not been requested.
[3] no statement about the target root has ever been queried (this condition implicitly ensures non-compliance and so we do not add it explicitly).

Fig. 5: Winning conditions $\mathsf{WinCond}^X$ for the three IVC simulation-soundness notions SIM-SND, w-SIM-SND, and newrt-SIM-SND; see Definition 4.4. Q_S is the set of query–response pairs from the interaction with the simulator.

Remark 4.5 (Intuitions on our definitions) *The three notions share the same experiment and differ only in their winning condition.* SIM-SND *and* w-SIM-SND *rely on* non-compliance*: a successful attack must prove a statement whose truth cannot be explained by honestly extending the leaps the simulator was queried on (Definition 4.3); this is what discards the malleability that is inherent to incremental proofs. On top of this,*SIM-SND *counts as an attack even a fresh* proof *for an already-queried statement, thus also ruling out mauling of the proof itself. The weak variant* w-SIM-SND *instead disregards any target whose* statement *was queried to the simulator (in the spirit of weak simulation-soundness for NIZKs [FKMV12]), while* newrt-SIM-SND *only requires the target* root x^*_{in} *to be fresh—a relaxation specific to the incremental setting.*

5 A Framework for Mauling Attacks in IVC

Here we show that some form of "mergeability", a feature common to several IVC constructions, is a sufficient condition for malleability in IVC. We also show a separation result between newrt-SIM-SND and w-SIM-SND. Formal proofs for this section can be found in Appendix F.

Auxiliary notation. Consider a true statement $a \overset{d}{\rightsquigarrow} b$. We write $\boldsymbol{w} \in \mathcal{W}\left(a \overset{d}{\rightsquigarrow} b\right)$ if $\boldsymbol{w} = (w_0, \ldots, w_{d-1})$ with $w_i \in \{0,1\}^m$ for every i, $x_{i+1} := C(x_i, w_i)$ for $i \in \{0, \ldots, d-1\}$ with $x_0 = a$ and $x_d = b$.

For strings x_0, x_d and let $\boldsymbol{w} \in \mathcal{W}(x_0 \overset{d}{\rightsquigarrow} x_d)$, $\boldsymbol{r} = (r_0, \ldots, r_{d-1})$ we write $\mathsf{IVC.P}^d(\mathsf{crs}, x_0, \boldsymbol{w}; r_i)$ for the proof π_d obtained by running $\pi_i \leftarrow \mathsf{IVC.P}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \pi_{i-1}, w_{i-1})$ for $i \in [d]$ on the random tape r_i, the states $x_1, \ldots, x_{d-1}$ being the ones induced by $\boldsymbol{w}$.

5.1 Mergeable IVCs

We formalize notions of post-hoc mergeability described in the technical overview.

$$\mathsf{Exp}_{\mathrm{Mrg}}(x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}, \boldsymbol{w}_L, \boldsymbol{w}_R) := \Pr\left[\mathsf{IVC.V}\big(\mathsf{crs}, \mathsf{s}^*, \pi\big) = 1 : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C) \\ \pi_R \leftarrow \mathsf{IVC.P}^{d_R}\big(\mathsf{crs}, x_{\mathsf{mid}}, \boldsymbol{w}_R\big) \\ r_L \leftarrow \mathsf{GetHint}\big(\mathsf{crs}, \mathsf{s}_L, \mathsf{s}_R, \pi_R\big) \\ \pi_L := \mathsf{IVC.P}^{d_L}\big(\mathsf{crs}, x_{\mathsf{init}}, \boldsymbol{w}_L; r_L\big) \\ \pi \leftarrow \mathsf{Merge}\big(\mathsf{crs}, (x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}), \\ \qquad (d_L, d_R), \pi_L, \pi_R\big) \end{array}\right]$$

Fig. 6: The (post-hoc) mergeability experiment. $\mathsf{s}_L := (x_{\mathsf{init}}, x_{\mathsf{mid}}, d_L)$, $\mathsf{s}_R := (x_{\mathsf{mid}}, x_{\mathsf{fin}}, d_R)$, $\mathsf{s}^* := (x_{\mathsf{init}}, x_{\mathsf{fin}}, d_L + d_R)$ are the prefix/suffix/merged statement.

Definition 5.1 (Universal (Post-Hoc) Mergeability) *Let* $\mathsf{IVC} = (\mathsf{IVC.Gen}, \mathsf{IVC.P}, \mathsf{IVC.V})$ *be an IVC scheme and let* $d_L, d_R \geq 1$ *be integers. We say that* IVC *is universally (post-hoc)* (d_L, d_R)-mergeable *if there exist PPT algorithms* GetHint *and* Merge *such that the following holds for all* $\lambda \in \mathbb{N}$*, all iteration bounds* T *with* $d_L + d_R \leq T \leq 2^\lambda$*, all polynomially bounded* n, m*, and all polynomial-size circuits* $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$*: For all* $x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}} \in \{0,1\}^n$ *and all witness vectors* $\boldsymbol{w}_L \in \mathcal{W}\big(x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{mid}}\big)$ *and* $\boldsymbol{w}_R \in \mathcal{W}\big(x_{\mathsf{mid}} \overset{d_R}{\rightsquigarrow} x_{\mathsf{fin}}\big)$*,*

$$\mathsf{Exp}_{Mrg}(x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}, \boldsymbol{w}_L, \boldsymbol{w}_R) \;\geq\; 1 - \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Exp}_{Mrg}(x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}, \boldsymbol{w}_L, \boldsymbol{w}_R)$ *is defined in Fig. 6. We say that* IVC *is* universally mergeable *if it is universally* (d_L, d_R)*-mergeable for some* $d_L, d_R \geq 1$*.*

The following definition is a restricted version of the notion above and will be used for our result showing attacks for w-SIM-SND. In contrast to the previous notion which demands that we can "universally" merge any triple here we require a more relaxed form of mergeability for triples of a very specific pattern. In particular the two statements being glued share their first component. A universally mergeable scheme is clearly XXY mergeable, the notion defined below.

Definition 5.2 (XXY (Post-Hoc) Mergeability) *We say that an IVC scheme is* XXY[6] *(post-hoc)* (d_L, d_R)*-mergeable if there exist PPT algorithms* GetHint *and* Merge *such that, for all* λ, T, n, m, C *as in Definition 5.1, the following holds: For all* $x_{\mathsf{init}}, x_{\mathsf{fin}} \in \{0,1\}^n$ *and all witness vectors* $\boldsymbol{w}_L \in \mathcal{W}\big(x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{init}}\big)$ *and* $\boldsymbol{w}_R \in \mathcal{W}\big(x_{\mathsf{init}} \overset{d_R}{\rightsquigarrow} x_{\mathsf{fin}}\big)$*,* $\mathsf{Exp}_{Mrg}(x_{\mathsf{init}}, x_{\mathsf{init}}, x_{\mathsf{fin}}, \boldsymbol{w}_L, \boldsymbol{w}_R) \geq 1 - \mathsf{negl}(\lambda)$*, where the experiment* Exp_{Mrg} *is as in Fig. 6. We say that* IVC *is* XXY mergeable *if it is* XXY (d_L, d_R)*-mergeable for some* $d_L, d_R \geq 1$*.*

Remark 5.3 *Unless specified otherwise, to simplify the language, in the rest of this section we implicitly drop "post-hoc" whenever we talk about the mergeability from Definitions 5.1 and 5.2.*

5.2 Hard reachability problems

We define a form of hard-to-decide distributional problems, which can be loosely thought of as an incremental variant of hard subset-membership problems [GW11].

Definition 5.4 (Hard Reachability Problem) *A hard reachability problem on a family of circuits* $C = \big(C_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}\big)_{\lambda \in \mathbb{N}}$ *has:*

[6] Read "ex-ex-why". The three letters refer to the pattern in the merge.

– *Two efficiently sampleable distribution ensembles* $\mathcal{L} = (\mathcal{L}_{\lambda,d})_{\lambda,d\in\mathbb{N}}$ *and* $\overline{\mathcal{L}} = \left(\overline{\mathcal{L}}_{\lambda,d}\right)_{\lambda,d\in\mathbb{N}}$ *such that for all* $\lambda \in \mathbb{N}$*, polynomially bounded* d, d' *with* $d' \geq 1$*:*

$$\Pr\left[x \overset{d}{\rightsquigarrow} x' \ \wedge\ \bar{x} \overset{d'}{\not\rightsquigarrow} \bar{x}' \ :\ (x,x') \xleftarrow{\$} \mathcal{L}_{\lambda,d};\ (\bar{x},\bar{x}') \xleftarrow{\$} \overline{\mathcal{L}}_{\lambda,d}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where, above, reachability is with respect to C_λ*.*

– *an efficient sampling algorithm* $\mathsf{Sam}(1^\lambda, 1^d) \to ((x,x'), \boldsymbol{w})$ *such that, for all* d*, the projection to the first coordinate yields the distribution* $(\mathcal{L}_{\lambda,d})_\lambda$ *and whose second coordinate is in* $\mathcal{W}\left(x \overset{d}{\rightsquigarrow} x'\right)$ *with overwhelming prob. in* λ*.*

Hardness*: for every polynomially bounded* $d = d(\lambda)$ *the two ensembles indexed by the security parameter,* $\left(\mathcal{L}_{\lambda,d(\lambda)}\right)_{\lambda\in\mathbb{N}}$ *and* $\left(\overline{\mathcal{L}}_{\lambda,d(\lambda)}\right)_{\lambda\in\mathbb{N}}$*, are computationally indistinguishable.*

Domain*: For two families of sets* $S^{\mathsf{init}} = (S^{\mathsf{init}}_\lambda)_{\lambda\in\mathbb{N}}$ *and* $S^{\mathsf{fin}} = (S^{\mathsf{fin}}_\lambda)_{\lambda\in\mathbb{N}}$*, we say that a hard reachability problem has domain* $(S^{\mathsf{init}}, S^{\mathsf{fin}})$ *if for all* $\lambda \in \mathbb{N}$ *and all polynomially bounded* d*, the following probability is overwhelming:* $\Pr\left[x, \bar{x} \in S^{\mathsf{init}}_\lambda \ \wedge\ x', \bar{x}' \in S^{\mathsf{fin}}_\lambda \ :\ (x,x') \xleftarrow{\$} \mathcal{L}_{\lambda,d};\ (\bar{x},\bar{x}') \xleftarrow{\$} \overline{\mathcal{L}}_{\lambda,d}\right]$*.*

Definition 5.5 (Structure in reachability problems (loops and prefixes)) *A hard reachability problem over* $C = (C_\lambda)_{\lambda\in\mathbb{N}}$ *with domain* $\left(S^{\mathsf{init}}, S^{\mathsf{fin}}\right)$ *has ...*

(Loop property) *... easy loops on starting points and no loops for landing points if there is a deterministic polynomial-time algorithm* Loop *such that* $\mathsf{Loop}(1^\lambda, x, 1^\ell) \in \mathcal{W}\left(x \overset{\ell}{\rightsquigarrow} x\right)$ *for every* $x \in S^{\mathsf{init}}_\lambda$*. and if* $x' \overset{\ell}{\not\rightsquigarrow} x'$ *for every* $x' \in S^{\mathsf{fin}}_\lambda$ *and polynomially bounded* ℓ*.*

(Prefix property) *... easy "funneling" prefixes on starting points if there is a deterministic polynomial-time algorithm* Pre *such that for every* $x \in S^{\mathsf{init}}_\lambda$ *the output* $(\alpha, \boldsymbol{w}) := \mathsf{Pre}(1^\lambda, x, 1^\ell)$ *satisfies* $\alpha \neq x$ *and* $\boldsymbol{w} \in \mathcal{W}\left(\alpha \overset{\ell}{\rightsquigarrow} x\right)$*, and every chain out of* α *passes through* x *after* ℓ *steps: for all* $d' \geq 0$ *and all* y*:* $\alpha \overset{\ell+d'}{\rightsquigarrow} y \implies x \overset{d'}{\rightsquigarrow} y$*.*

5.3 Structured hard reachability problems from one-way functions.

We define a step circuit from PRGs.

Definition 5.6 *Let* $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *be a pseudorandom generator and let* $k \geq \lambda + 1$*. States are pairs* (x,c) *with* $x \in \{0,1\}^{2\lambda}$ *and* $c \in \mathbb{Z}_{2^k}$*, so* $n = 2\lambda + k$*; witnesses are pairs* $w = (b,s) \in \{0,1\} \times \{0,1\}^\lambda$*, so* $m = \lambda + 1$*. Define*

$$C_G\big((x,c),(b,s)\big) \ :=\ \begin{cases} (x,1) & \text{if } c = 0,\ b = 1 \text{ and } G(s) = x, \\ (x,0) & \text{if } c = 0 \text{ otherwise}, \\ (x,\ c+1 \bmod 2^k) & \text{if } c \neq 0. \end{cases}$$

Away from $c = 0$ the circuit is a counter that ignores its witness. At $c = 0$ the circuit idles, unless the witness certifies that x is in the range of G, in which case it may advance to $c = 1$.

Notation. Throughout, $-\ell$ denotes $2^k - \ell \in \mathbb{Z}_{2^k}$, and $w^\ell_\perp$ denotes the witness vector $\big((0,0^\lambda),\ldots,(0,0^\lambda)\big)$ of length ℓ (this is the witness certifying a chain of idle steps). For $s \in \{0,1\}^\lambda$ and $\ell \geq 1$ we write $\boldsymbol{w}^{(\ell)}_s := \big(\underbrace{(0,0^\lambda),\ldots,(0,0^\lambda)}_{\ell-1},\ (1,s)\big)$ for the witness vector that idles $\ell - 1$ times and then takes the gate with s; thus $\boldsymbol{w}^{(1)}_s = \big((1,s)\big)$.

Lemma 5.7 (Basic properties of C_G**)** *The following two properties hold for* C_G*: (1) For* $c \neq 0$ *the state* (x,c) *has the single successor* $(x, c+1 \bmod 2^k)$*, whatever the witness. (2) The successors of* $(x,0)$ *are* $(x,0)$*, reached with every witness* $(0,s)$*, and, if and only if* x *is in the range of* G*,* $(x,1)$*, reached with* $(1,s)$ *for any* s *with* $G(s) = x$*. Consequently, for every* x *and every* $1 \leq \ell < 2^k$*: a)* $w^\ell_\perp \in \mathcal{W}\big((x,0) \overset{\ell}{\rightsquigarrow} (x,0)\big)$ *and* $w^\ell_\perp \in \mathcal{W}\big((x,-\ell) \overset{\ell}{\rightsquigarrow} (x,0)\big)$*; b) every chain of* ℓ *steps from* $(x,-\ell)$ *ends at* $(x,0)$*, and* $(x,c) \overset{\ell}{\not\rightsquigarrow} (x,c)$ *if* $c \neq 0$*; c) if* $x = G(s)$ *then* $\boldsymbol{w}^{(\ell)}_s \in \mathcal{W}\big((x,0) \overset{\ell}{\rightsquigarrow} (x,1)\big)$*, and if* x *is outside the range of* G *then* $(x,0) \overset{\ell}{\not\rightsquigarrow} (x,1)$*.*

By leveraging the observations above, we can show the following lemma.

Lemma 5.8 (Hardness of C_G) *Let G be a pseudorandom generator and let*

$$\mathcal{L}_{\lambda,d} := \left\{ \big((G(s),0),(G(s),1)\big) : s \xleftarrow{\$} \{0,1\}^\lambda \right\},\ \overline{\mathcal{L}}_{\lambda,d} := \left\{ \big((x,0),(x,1)\big) : x \xleftarrow{\$} \{0,1\}^{2\lambda} \right\},$$

and let $\mathsf{Sam}(1^\lambda, 1^d)$ sample $s \xleftarrow{\$} \{0,1\}^\lambda$ and output $\big(((G(s),0),(G(s),1)),\ \boldsymbol{w}_s^{(d)}\big)$. Then $(\mathcal{L}, \overline{\mathcal{L}}, \mathsf{Sam})$ is a hard reachability problem on C_G with domain $(S^{\mathsf{init}}, S^{\mathsf{fin}})$ with $S^{\mathsf{init}}_\lambda := \{(x,0) : x \in \{0,1\}^{2\lambda}\}$ and $S^{\mathsf{fin}}_\lambda := \{(x,c) : x \in \{0,1\}^{2\lambda},\ c \neq 0\}$.

Theorem 5.9 (Hard reachability problems from one-way functions). *Assume one-way functions exist. Then there are a polynomial-size family of step circuits C, families of sets of states S^{init} and S^{fin}, and a hard reachability problem on C with domain $(S^{\mathsf{init}}, S^{\mathsf{fin}})$ such that C has easy loops and easy prefixes on S^{init} and is loop-free on S^{fin}, in the sense of Definition 5.5.*

5.4 Sufficient conditions for attacks against newrt-SIM-SND

The proof of the theorem below follows by applying the attack we sketched in the technical overview. Some of the technical aspects of the analysis are guaranteed by the additional properties we describe in Definitions 5.4 and 5.5 and shown to be implied by OWFs in Theorem 5.9. In particular, for a sampled false statement $x_{\mathsf{mid}} \overset{d_R}{\rightsquigarrow} x_{\mathsf{fin}}$, the attack can use easy prefixes to construct a distinct starting state $x_{\mathsf{init}} \neq x_{\mathsf{mid}}$ and witnesses for $x_{\mathsf{init}} \overset{d_L}{\rightsquigarrow} x_{\mathsf{mid}}$, allowing the adversary to generate the left proof honestly while keeping the target's root fresh. The funneling property ensures that every computation from x_{init} passes through x_{mid} after d_L steps, so the merged target $x_{\mathsf{init}} \overset{d_L+d_R}{\rightsquigarrow} x_{\mathsf{fin}}$ remains false (there is no "alternative path" and the suffix remains false). Finally, the fact that we can efficient sampling of true instances together with their witnesses lets us apply zero knowledge and hardness to transfer the acceptance guarantee of merging two honest proofs to the attack using a simulated proof of a false instance (Theorem F.1).

Theorem 5.10 (Universal mergeability breaks newrt-SIM-SND). *If OWFs exist, no universally mergeable zero-knowledge IVC scheme satisfies newrt-SIM-SND.*

As a direct implication of Theorem 5.10, any universally mergeable zero-knowledge IVC scheme then satisfies neither w-SIM-SND nor SIM-SND.

The construction of [DJJ+25] is universally mergeable. The hypothesis of Theorem 5.10 is met by the IVC construction of Fig. 10, for the shape $d_L = d_R = 2$. No component of a proof of that construction is tied to the root of the statement it belongs to, and this is what lets two halves with different roots be glued.

Theorem 5.11 ([DJJ+25] is universally post-hoc mergeable). *The IVC scheme from [DJJ+25] (Fig. 10 in the appendix) is universally $(2,2)$-mergeable.*

While the pattern (2,2) is sufficient to show that [DJJ+25] is malleable, in Remark F.2 we expand on how it actually exhibits the general pattern $(2^j, 2^j)$.

Remark 5.12 (Valiant-style ZK designs exhibit universal mergeability) *In Appendix G we abstract the common adaptation to zero knowledge of Valiant-style proofs and we prove how they generally imply our notion of mergeability.*

5.5 Sufficient conditions for attacks against w-SIM-SND

The following results illustrate that even if an IVC is not universally post-hoc mergeable, it would still not satisfy w-SIM-SND if it exhibits the more restricted property of XXY mergeability (Definition 5.2).

Theorem 5.13. *If one-way functions exist, no XXY mergeable zero-knowledge IVC scheme satisfies w-SIM-SND.*

A separation result. We show that w-SIM-SND does not collapse to newrt-SIM-SND.

Theorem 5.14. *There exists an IVC (the root-bound construction in the technical overview, formalized in the appendix in Figs. 11 to 13) is* $(2,2)$-XXY *mergeable and* $(1,1)$-XXY *mergeable.*

Corollary 5.15 (Separation of newrt-SIM-SND and w-SIM-SND) *Under subexponential iO and subexponential LWE (or subexponential bilinear maps) there exists an IVC scheme that satisfies* newrt-SIM-SND *and not* w-SIM-SND.

6 Context-chained IVC Construction

Our construction augments the proof-merging framework of Datta et al. [DJJ+25] with a hash chain that binds each recursive proof to the computation prefix preceding it. Each dyadic node receives a *chain-in* value, an incoming chain value summarizing its left context and produces a *chain-out*, an outgoing chain value that additionally commits to to the node itself, that is, its level and endpoint ciphertexts. At level 0, this chain-out value is the message of a NIZK signature of knowledge. At higher levels, the chain-in is included directly in the recursive BARG instance.

Fix a collision-resistant hash function H_{hk} with key hk, and assume a canonical injective encoding of its inputs. Let $\langle \ell \rangle$ denote a fixed-length encoding of $\ell \in \{0, \ldots, \ell_{\max}\}$. We initialize the chain by $v_0 := \bot := \mathsf{H}_{\mathsf{hk}}(0)$. For a level-$\ell$ node spanning the interval $[j, j+2^\ell]$, with endpoint ciphertexts $(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$ and chain-in v_j, define its chain-out as $h_{\mathsf{fin}} := \mathsf{Ch}_\ell(v_j, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) := \mathsf{H}_{\mathsf{hk}}(1 \| \langle \ell \rangle \| v_j \| \mathsf{ct}_{\mathsf{init}} \| \mathsf{ct}_{\mathsf{fin}})$. Thus the initial marker and ordinary chain values are domain separated, and the hash also binds the dyadic level of the node. The concatenation uses the setup's fixed field-length bounds.

Writing $h_{\mathsf{init}} := v_j$, a level-0 proof is generated and verified under the message $m := h_{\mathsf{fin}} = \mathsf{Ch}_0(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$. The message is determined entirely by public information available before the proof is generated. In particular, it binds the node's endpoint ciphertexts to the chain value representing its left context. The chain values are maintained according to the same binary-counter merge schedule as the recursive proof forest. For $j > 0$, let $\nu_2(j) := \max\{a \geq 0 : 2^a \mid j\}$. The rightmost block in the canonical dyadic decomposition of the prefix $[0, j]$ is $[j - 2^{\nu_2(j)}, j]$, and v_j is precisely the chain-out of this block. Equivalently, v_j is the chain value exposed by the current frontier at position j. Lemma 6.3 shows that this value is well defined and that every node subsequently created with left endpoint j receives the same chain-in v_j. Accordingly, an entry of $S^{(\ell)}$ has the form $(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{init}}, \pi^{(\ell)})$, where the $\mathsf{ct}_{\mathsf{mid}}$ slot contains a fixed dummy value at level 0. No chain-out is stored explicitly; it is recomputed as $\mathsf{Ch}_\ell(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$.

At level 0, the underlying NIZK statement remains the ciphertext pair $(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$, and the chain enters only through the NIZK-SoK message. At levels $\ell \geq 1$, the recursive statement additionally carries the single hash value h_{init}, which is needed to reconstruct the chain values and messages of its children. Together with the canonical frontier bookkeeping, these chain values bind every recursive proof to its position and left context.

We present the construction in Figs. 7 to 8. We denote in blue the components that are new or changed relative to the construction of Datta et al. [DJJ+25]. Our construction additionally requires the PKE scheme to satisfy an *unexplainability* property. In Section E.1, we show how to obtain such a scheme generically from perfectly correct CPA-secure PKE, and a PRG.

Setup. Let $\ell_{\max} := \lceil \log_2 T \rceil$ and use the security parameters in (1). The generator samples the SNARG parameters $(\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}) \leftarrow \mathsf{SNARG.Gen}(1^\kappa, 1^{2n}, 1^m, R_{C,1})$ and the encryption keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\kappa, 1^n)$. It also samples $\mathsf{hk} \leftarrow \{0,1\}^{\ell_K(\kappa)}$ for the collision resistant hash family and defines $\bot := \mathsf{H}_{\mathsf{hk}}(0)$ and $\mathsf{Ch}_\ell(h, \mathsf{ct}_L, \mathsf{ct}_R) := \mathsf{H}_{\mathsf{hk}}(1 \| \langle \ell \rangle \| h \| \mathsf{ct}_L \| \mathsf{ct}_R)$. It then generates a NIZK-SoK CRS $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}})$, a level-1 BARG CRS for $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$ if $\ell_{\max} \geq 1$, and, for every $2 \leq \ell \leq \ell_{\max}$, a level-$\ell$ BARG CRS for $R_{\ell,\mathsf{crs}_{\ell-1}}$, each with batch size 2.

The public parameters are

$$\mathsf{crs} = (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{hk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell=1}^{\ell_{\max}}).$$

The bound T is retained so that the prover and verifier can enforce their stated input ranges explicitly. In the figures, BARG calls omit their fixed relation argument R_ℓ, and batches are ordered pairs. All encodings use the public setup length bounds; and malformed inputs reject.

Languages and Relations

Every relation rejects malformed inputs and witnesses.

(a) **One step.** $L_{C,1} = \{(x, x') : \exists w,\ C(x, w) = x'\}$ and $R_{C,1}((x, x'), w) = 1$ iff $C(x, w) = x'$ (unchanged).

(b) **Level 0.** Let $X_0 = (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$ and $W_0 = (x_{\mathsf{init}}, x_{\mathsf{fin}}, r_{\mathsf{init}}, r_{\mathsf{fin}}, \pi_{\mathsf{SNARG}})$. Then $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}(X_0, W_0) = 1$ iff $\mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, (x_{\mathsf{init}}, x_{\mathsf{fin}}), \pi_{\mathsf{SNARG}}) = 1$, $\mathsf{ct}_{\mathsf{init}} = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{init}}; r_{\mathsf{init}})$, and $\mathsf{ct}_{\mathsf{fin}} = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{fin}}; r_{\mathsf{fin}})$. Set $L_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}} := \{X_0 : \exists W_0\ R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}(X_0, W_0) = 1\}$. A level-0 node with incoming value h_{init} uses message $m := \mathsf{Ch}_0(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$.

(c) **Level 1.** Let $X_1 = (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{init}})$ and $W_1 = \pi^{(0)}$. Then $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}(X_1, W_1) = 1$ iff

$$\mathsf{NIZK.V}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := \mathsf{Ch}_0(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})) = 1.$$

Set $L_{1,\mathsf{crs}_{\mathsf{NIZK}}} := \{X_1 : \exists W_1\ R_{1,\mathsf{crs}_{\mathsf{NIZK}}}(X_1, W_1) = 1\}$.

(d) **Level $\ell \geq 2$.** Let $X_\ell = (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{init}})$ and $W_\ell = (\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell-1)})$, set $h_{\mathsf{mid}} := \mathsf{Ch}_{\ell-2}(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}})$. Then $R_{\ell,\mathsf{crs}_{\ell-1}}(X_\ell, W_\ell) = 1$ iff

$$\mathsf{BARG}.\mathcal{V}\big(\mathsf{crs}_{\ell-1}, ((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, h_{\mathsf{init}}), (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{mid}})), \pi^{(\ell-1)}\big) = 1.$$

Set $L_{\ell,\mathsf{crs}_{\ell-1}} := \{X_\ell : \exists W_\ell\ R_{\ell,\mathsf{crs}_{\ell-1}}(X_\ell, W_\ell) = 1\}$.

Summary. Level 0 uses a NIZK-SoK; levels $\ell \geq 1$ use somewhere-extractable BARGs of batch size 2. The chain-in is carried explicitly in every higher-level BARG instance, while at level 0 it determines the NIZK-SoK message.

Fig. 7: Languages and relations for the context-chained IVC.

Theorem 6.1 (Context-chained IVC). *Fix $\lambda \geq 1$, $1 \leq T \leq 2^\lambda$, and polynomially bounded $n, m, |C|$, with $n \geq 1$. Circuit size includes its input and witness wires. Assume the following ingredients:*

1. *a CPA-secure PKE with perfect correctness and unexplainability using uniformly sampled λ_{PKE}-bit randomness;*
2. *a perfectly complete non-adaptively sound SNARG for $R_{C,1}$;*
3. *a perfectly complete simulation-extractable NIZK-SoK for $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$, with adaptive multi-theorem zero knowledge;*
4. *rate-1 BARGs for the recursive relations, with ordinary/trapdoor CRS indistinguishability and somewhere extraction; and*
5. *a collision-resistant hash family.*

For each required security property i, assume $(\mathsf{poly}(\kappa), 2^{-\kappa^{\delta_i}})$ security against non-uniform adversaries, for a constant $0 < \delta_i < 1$. Assume a fixed polynomial P bounds, at every level $\ell \leq \ell_{\max}$, the recursive relation and ordinary/trapdoor CRS sizes and the setup, proving, verification, and extraction costs by $P(\kappa, n, m, \ell_{\max}, |C|)$, with fixed public size bounds. Choose constants $0 < \delta' < \delta < \min_i \delta_i$ and set

$$\lambda_{\mathsf{PKE}} = \lambda_{\mathsf{SNARG}} = \lambda_{\mathsf{NIZK}} = \lambda_{\mathsf{BARG}} = \lambda_H = \kappa := \left\lceil (3n + 2\lambda)^{1/\delta'} \right\rceil. \tag{1}$$

Then the IVC scheme in Figs. 7 to 8 is perfectly complete, zero-knowledge, and satisfies w-SIM-SND. *Its proof size is* $\mathsf{poly}(\lambda, n, \log T, \log |C|)$, *and its CRS size and setup, update, and verification times are* $\mathsf{poly}(\lambda, \log T, |C|)$.

6.1 Completeness

We first establish the properties of the proof-merging structure used in the completeness proof.

Claim 6.2 (Binary-counter structure) *After the honest prover has processed i computation steps, $S^{(\ell)} \neq \emptyset$ iff bit ℓ of i is set. When nonempty, $S^{(\ell)}$ contains exactly one node spanning*

$$[j_\ell, j_\ell + 2^\ell], \qquad j_\ell := \sum_{\ell' > \ell : \mathrm{bit}_{\ell'}(i) = 1} 2^{\ell'}. \tag{2}$$

Context-Chained IVC — Prover and Verifier

$\mathsf{IVC.P}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \Pi_{i-1}, w_i)$.

1. Parse $\mathsf{crs} = (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{hk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell=1}^{\ell_{\max}})$. If $i-1 \notin \{0,\dots,T-1\}$, $x_0 \notin \{0,1\}^n$, $x_{i-1} \notin \{0,1\}^n$, or $w_i \notin \{0,1\}^m$, output $\perp$.
2. If $i = 1$: if $x_{i-1} \neq x_0$, output $\perp$; sample r_0 from the PKE coin space, set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$, $S^{(\ell)} := \emptyset$ for every level, and $h_{\mathsf{prev}} := \perp$.
3. If $i > 1$: parse $\Pi_{i-1} = (\mathsf{ct}_0, \mathsf{ct}_{i-1}, r_0, r_{i-1}, \{S^{(\ell)}\}_\ell)$. If $\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \Pi_{i-1}) = 0$, output $\perp$. Set $h_{\mathsf{prev}} := h_{\mathsf{curr}}$, the final chain value of that verification.
4. Set $x_i := C(x_{i-1}, w_i)$ and $\pi_{\mathsf{SNARG}} \leftarrow \mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, (x_{i-1}, x_i), w_i)$.
5. Sample fresh PKE coins r_i, set $\mathsf{ct}_i := \mathsf{PKE.Enc}(\mathsf{pk}, x_i; r_i)$ and $m_i := \mathsf{Ch}_0(h_{\mathsf{prev}}, \mathsf{ct}_{i-1}, \mathsf{ct}_i)$, and compute
$$\pi_i^{(0)} \leftarrow \mathsf{NIZK.P}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{i-1}, \mathsf{ct}_i), (x_{i-1}, x_i, r_{i-1}, r_i, \pi_{\mathsf{SNARG}}), m_i).$$
Add $(i-1, \mathsf{ct}_{i-1}, \perp, \mathsf{ct}_i, h_{\mathsf{prev}}, \pi_i^{(0)})$ into $S^{(0)}$.
6. **Merge.** For $\ell = 0, \dots, \ell_{\max} - 1$:
   (a) If $|S^{(\ell)}| < 2$, continue. Otherwise parse $S^{(\ell)} = \{E, \widetilde{E}\}$, where
   $$E = (i_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{init}}, \pi^{(\ell)}),$$
   $$\widetilde{E} = (\widetilde{i}_0, \widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{mid}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \widetilde{h}_{\mathsf{init}}, \widetilde{\pi}^{(\ell)}).$$
   Order the entries so that $i_0 \le \widetilde{i}_0$. Set $S^{(\ell)} := \emptyset$.
   (b) Adjacency: require $\mathsf{ct}_{\mathsf{fin}} = \widetilde{\mathsf{ct}}_{\mathsf{init}}$, $i_0 + 2^\ell = \widetilde{i}_0$, and $\widetilde{h}_{\mathsf{init}} = \mathsf{Ch}_\ell(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$; otherwise output $\perp$.
   (c) Set $X := (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{init}})$ and $\widetilde{X} := (\widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \widetilde{h}_{\mathsf{init}})$. If $\ell = 0$, compute
   $$\pi^{(1)} \leftarrow \mathsf{BARG.P}(\mathsf{crs}_1, (X, \widetilde{X}), (\pi^{(0)}, \widetilde{\pi}^{(0)})).$$
   If $\ell \ge 1$, compute
   $$\pi^{(\ell+1)} \leftarrow \mathsf{BARG.P}\big(\mathsf{crs}_{\ell+1}, (X, \widetilde{X}), ((\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell)}), (\widetilde{\mathsf{ct}}_{\mathsf{mid}}, \widetilde{\pi}^{(\ell)}))\big).$$
   Add $(i_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, h_{\mathsf{init}}, \pi^{(\ell+1)})$ into $S^{(\ell+1)}$.
7. Output $\Pi_i = (\mathsf{ct}_0, \mathsf{ct}_i, r_0, r_i, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

$\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_t, t), \Pi_t)$.

1. Parse $\mathsf{crs} = (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{hk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell=1}^{\ell_{\max}})$. If $t \notin \{1,\dots,T\}$ or $x_0 \notin \{0,1\}^n$ or $x_t \notin \{0,1\}^n$, output 0. Parse $\Pi_t = (\mathsf{ct}_0, \mathsf{ct}_t, r_0, r_t, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$; set $i_{\mathsf{curr}} := 0$, $\mathsf{ct}_{\mathsf{curr}} := \mathsf{ct}_0$, and $h_{\mathsf{curr}} := \perp$.
2. For $\ell = \ell_{\max}, \dots, 0$:
   (a) Let s_ℓ be bit ℓ of t. If $s_\ell = 0$: if $S^{(\ell)} \neq \emptyset$, output 0; otherwise continue to the next level. If $s_\ell = 1$, parse $S^{(\ell)} = \{(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{init}}, \pi^{(\ell)})\}$.
   (b) Require $i_{\mathsf{curr}} = i_{\mathsf{init}}$, $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_{\mathsf{init}}$, and $h_{\mathsf{curr}} = h_{\mathsf{init}}$; otherwise output 0. Then set $i_{\mathsf{curr}} := i_{\mathsf{curr}} + 2^\ell$, $\mathsf{ct}_{\mathsf{curr}} := \mathsf{ct}_{\mathsf{fin}}$, and $h_{\mathsf{curr}} := \mathsf{Ch}_\ell(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$.
   (c) If $\ell = 0$, require $\mathsf{ct}_{\mathsf{mid}} = \perp$ and $\mathsf{NIZK.V}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := h_{\mathsf{curr}}) = 1$. If $\ell \ge 1$, set $h_{\mathsf{mid}} := \mathsf{Ch}_{\ell-1}(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}})$ and require
   $$\mathsf{BARG}.\mathcal{V}\big(\mathsf{crs}_\ell, ((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, h_{\mathsf{init}}), (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, h_{\mathsf{mid}})), \pi^{(\ell)}\big) = 1.$$
   If the applicable proof check fails, output 0.
3. Require $i_{\mathsf{curr}} = t$, $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_t$, $\mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0) = \mathsf{ct}_0$, and $\mathsf{PKE.Enc}(\mathsf{pk}, x_t; r_t) = \mathsf{ct}_t$; otherwise output 0. Output 1.

Fig. 8: Context-chained IVC prover and verifier.

The nonempty stored nodes are precisely the canonical dyadic decomposition of $[0, i]$, ordered from left to right by decreasing level. Every node created at level ℓ has left endpoint divisible by 2^ℓ.

Proof. Initially all sets are empty. At the first step, the prover creates the leaf $[0, 1]$, matching the binary representation of 1.

Suppose the assertion holds after i steps. At step $i+1$, the prover appends the leaf $[i, i+1]$. If bit 0 of i is zero, no merge occurs at level 0. Otherwise, the two adjacent level-0 nodes merge into their level-1 parent. If level 1 was already occupied, the two nodes at that level merge, and so on.

Thus the merge loop performs precisely the carry propagation of the binary increment from i to $i+1$. Each successive occupied level is cleared, and the first unoccupied level receives the resulting parent. At a merge of level-ℓ nodes, the left endpoint is a sum of powers of two strictly larger than 2^ℓ, so the parent is aligned at level $\ell+1$. All higher occupied levels remain unchanged. Hence the resulting intervals and occupied levels are exactly those specified by (2).

Frozen chain values. Fix an honest execution reaching step t, with boundary ciphertexts $\mathsf{ct}_0, \ldots, \mathsf{ct}_t$. Set $v_0 := \perp$. For $j > 0$, let $a_j := \nu_2(j)$ and $B_j := [j - 2^{a_j}, j]$. By Claim 6.2, B_j is the last block in the canonical dyadic decomposition of $[0, j]$. Define

$$v_j := \mathsf{Ch}_{a_j}(v_{j-2^{a_j}}, \mathsf{ct}_{j-2^{a_j}}, \mathsf{ct}_j). \tag{3}$$

Lemma 6.3 (Chain well-definedness) *In every honest execution reaching step t, the following hold.*

(W1) *For every $j \in \{0, \ldots, t\}$, every node created with left endpoint j has incoming chain value v_j. The frontier fold after j steps equals v_j. In particular, whenever the prover starts the leaf $[j, j+1]$, it uses $h_{\mathsf{prev}} = v_j$.*

(W2) *Let N be a level-ℓ node, $\ell \geq 1$, with incoming value h_{init}, endpoints $(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}})$, and midpoint $\mathsf{ct}_{\mathsf{mid}}$. Its right child's incoming value is*

$$h_{\mathsf{mid}} = \mathsf{Ch}_{\ell-1}(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}). \tag{4}$$

Proof. We prove (W1) by strong induction on j. For $j = 0$, the empty frontier folds to $\perp = v_0$. The first leaf receives this value, and every later node with left endpoint 0 inherits the incoming value of its left child.

Now fix $j > 0$ and assume (W1) for smaller positions. By Claim 6.2, the frontier after j steps is the canonical dyadic decomposition of $[0, j]$. Every block $[p, p+2^a]$ in this frontier has $p < j$, so its stored incoming value is v_p by the induction hypothesis. Moreover, $\nu_2(p + 2^a) = a$. Starting at v_0, folding this block therefore produces $\mathsf{Ch}_a(v_p, \mathsf{ct}_p, \mathsf{ct}_{p+2^a}) = v_{p+2^a}$ by (3). Induction over the frontier blocks shows that its final fold is v_j and that all its chain-adjacency checks agree.

Whenever the leaf $[j, j+1]$ is created, the preceding proof has passed verification, and the prover uses that verification's final chain value. Hence the leaf receives v_j. Every later node with left endpoint j inherits this value through its left child. This proves (W1).

For (W2), let j be the left endpoint of N. By Claim 6.2, $2^\ell \mid j$. Its left child spans $[j, j + 2^{\ell-1}]$, and $\nu_2(j + 2^{\ell-1}) = \ell - 1$. Thus this child is the last canonical block ending at $j + 2^{\ell-1}$. By (3) and (W1), $v_{j+2^{\ell-1}} = \mathsf{Ch}_{\ell-1}(v_j, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}) = \mathsf{Ch}_{\ell-1}(h_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}})$. By (W1), this is the right child's incoming value.

Lemma 6.4 (Perfect completeness) *Assume perfect completeness of the SNARG, NIZK-SoK, and BARG. For every setup input of the construction, every $1 \leq t \leq T$, every $x_0 \in \{0,1\}^n$, and every sequence $w_1, \ldots, w_t \in \{0,1\}^m$, define $x_i := C(x_{i-1}, w_i)$. If the CRS and proofs are generated honestly as in Definition 4.1, then* $\Pr[\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_t, t), \Pi_t) = 1] = 1$.

Proof. We induct on the prover's steps, showing that each step completes and its output is accepted. At step $i = 1$, the initial state check succeeds and the prover sets $h_{\mathsf{prev}} = \perp = v_0$. At step $i > 1$, the preceding proof accepts by the induction hypothesis, and (W1) of Lemma 6.3 gives $h_{\mathsf{prev}} = v_{i-1}$. All input-domain checks hold because $i \leq t \leq T$ and the states and witnesses have their prescribed lengths.

Prover checks and proof generation. The new leaf receives incoming value v_{i-1}. Every parent subsequently created inherits its left child's incoming value, preserving the prescribed value v_p at its left endpoint p.

During a carry, the prover merges adjacent level-a blocks $[p, p + 2^a]$ and $[p + 2^a, p + 2^{a+1}]$. Their indices and shared ciphertext agree. By (2), the left block's initial index satisfies $2^{a+1} \mid p$, so $\nu_2(p + 2^a) = a$. Therefore

$v_{p+2^a} = \mathsf{Ch}_a(v_p, \mathsf{ct}_p, \mathsf{ct}_{p+2^a})$, which is exactly the merge procedure's chain-adjacency check. Hence these checks do not cause an abort.

The leaf's SNARG proof accepts by perfect completeness because $x_i = C(x_{i-1}, w_i)$. Together with the honestly generated ciphertext openings, it forms a valid witness for $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. NIZK-SoK completeness therefore makes the leaf proof accept under message $\mathsf{Ch}_0(v_{i-1}, \mathsf{ct}_{i-1}, \mathsf{ct}_i)$.

For each merge, the child proofs accept under their prescribed instances, either by the induction hypothesis or by an earlier merge in this step. At level 1, the witnesses are the accepting level-0 NIZK-SoK proofs and therefore satisfy $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$. At level $\ell \geq 2$, each witness consists of the child's midpoint ciphertext and accepting level-$(\ell-1)$ proof. Property (W2), applied to that child, identifies the $\mathsf{Ch}_{\ell-2}$ value recomputed inside $R_{\ell,\mathsf{crs}_{\ell-1}}$. Thus both witnesses satisfy the recursive relation. Perfect BARG completeness makes the parent proof accept. Every proof-generation call therefore supplies an accepting proof, and the step completes.

Frontier checks. By Claim 6.2, the resulting frontier is the canonical dyadic decomposition of $[0, i]$. Before processing a block $[p, p+2^\ell]$, the verifier's running values are $i_{\mathsf{curr}} = p$, $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_p$, and $h_{\mathsf{curr}} = v_p$. The stored block has these initial values by construction and (W1), so all adjacency checks succeed. Since $\nu_2(p + 2^\ell) = \ell$, its chain update produces v_{p+2^ℓ} by (3). The index and ciphertext updates identify the same right boundary. Starting from $0, \mathsf{ct}_0, v_0$, this proves the invariant throughout the frontier.

For a level-0 block, the verifier reconstructs exactly the NIZK-SoK message used by the prover; its dummy midpoint is $\perp$ by construction. For a higher-level block, (W2) ensures that the verifier reconstructs exactly the two instances used to generate its BARG proof. Thus every frontier proof check accepts.

Endpoint checks. After the final block, $i_{\mathsf{curr}} = i$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_i$. The retained openings satisfy $\mathsf{ct}_0 = \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$ and $\mathsf{ct}_i = \mathsf{PKE.Enc}(\mathsf{pk}, x_i; r_i)$. Hence every final check succeeds and $\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_i, i), \Pi_i) = 1$. Taking $i = t$ completes the proof.

6.2 Efficiency

We follow the size calculation of DJJ$^+$ [DJJ+25], accounting also for the chain values and hash computations. All lengths below denote fixed public upper bounds, with padding where required by the primitive syntax.

Proof size. The parameter choice in (1) gives $\kappa = \mathsf{poly}(\lambda, n)$. The ciphertext length $|\mathsf{ct}|$ and hash-output length $|h|$ are polynomial in (κ, n), and encryption randomness has length κ. SNARG succinctness gives $|\pi_{\mathsf{SNARG}}| = \mathsf{poly}(\kappa, \log(|C|+2))$ and a verifier circuit of size $\mathsf{poly}(\kappa, n, \log(|C|+2))$. The level-0 relation verifies this proof and two encryption openings. Hence its circuit size and the length of a level-0 NIZK-SoK proof are bounded by $\mathsf{poly}(\kappa, n, \log(|C|+2))$. The NIZK-SoK message has length $|h|$, which is covered by this bound.

At every BARG level, the batch contains two instances. Each instance contains two ciphertexts and one incoming chain value. At level 1, the per-instance witness is a level-0 NIZK-SoK proof. At level $\ell \geq 2$, it consists of a midpoint ciphertext and a level-$(\ell-1)$ proof. Thus the incoming chain value adds no witness field.

By rate 1, there is a fixed polynomial p such that $|\pi^{(1)}| \leq |\pi^{(0)}| + p(\kappa, \log(|R_1|+2))$ and, for $\ell \geq 2$, $|\pi^{(\ell)}| \leq |\pi^{(\ell-1)}| + |\mathsf{ct}| + p(\kappa, \log(|R_\ell|+2))$. Summing gives

$$|\pi^{(\ell)}| \leq |\pi^{(0)}| + (\ell-1)|\mathsf{ct}| + \sum_{a=1}^{\ell} p(\kappa, \log(|R_a|+2)) \qquad (1 \leq \ell \leq \ell_{\max}). \tag{5}$$

The uniform relation-size bound in Theorem 6.1, together with $n + m \leq |C|$, bounds every additive term by $\mathsf{poly}(\kappa, n, \ell_{\max}, \log(|C|+2))$. Therefore $|\pi^{(\ell)}| = \mathsf{poly}(\lambda, n, \ell_{\max}, \log(|C|+2))$.

By Claim 6.2, the final proof stores at most one node per level. Each entry contains an index of $O(\ell_{\max}+1)$ bits, a constant number of ciphertext and hash fields, and one proof. This also accounts for the level-0 dummy field. Including the two endpoint ciphertexts and their openings, we obtain

$$|\Pi_t| \leq 2|\mathsf{ct}| + 2\kappa + \sum_{\ell=0}^{\ell_{\max}} O\big(\ell_{\max} + 1 + |\mathsf{ct}| + |h| + |\pi^{(\ell)}|\big) = \mathsf{poly}(\lambda, n, \log T, \log |C|). \tag{6}$$

Thus the construction has witness-independent proof size.

CRS size and running times. A chain input has length polynomial in $(\kappa, n, \log(\ell_{\max}+2))$, so its hash can be evaluated by a circuit of polynomial size in these parameters. The level-1 relation computes the NIZK-SoK message; each higher-level relation computes the incoming chain value of the right child. These computations and the additional public instance field are covered by the recursive efficiency assumption in Theorem 6.1.

Setup generates the SNARG, PKE, NIZK-SoK, and hash parameters, followed by at most $\ell_{\max}$ BARG CRSs. An update verifies the preceding proof, evaluates one step of C, generates one SNARG proof and one NIZK-SoK proof, and performs at most $\ell_{\max}$ BARG merges. Verification checks at most $\ell_{\max}+1$ frontier proofs. Outside the primitive calls, an update or verification performs $O(\ell_{\max}+1)$ hash evaluations and bookkeeping operations.

The common polynomial bound on recursive relation sizes, CRS sizes, and primitive costs therefore gives CRS size and setup, update, and verification times $\mathsf{poly}(\kappa, n, m, \ell_{\max}, |C|)$. Using $\kappa = \mathsf{poly}(\lambda, n)$ and $n + m \le |C|$, these bounds become $\mathsf{poly}(\lambda, \log T, |C|)$. The same assumption gives polynomial cost for the trapdoor setups and paths of at most $\ell_{\max}$ extraction operations used in the security proof.

6.3 Zero knowledge

We prove zero knowledge using a hybrid argument similar to that of [DJJ+25]. The additional hash-chain computations to be performed due to the context chaining in our construction can be done from the ciphertexts and block descriptors, without knowing the intermediate states or encryption randomness.

The simulator. We define $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ for the construction parameters T, n, m, C, using the initial chain value $\perp$ and hash functions Ch_ℓ defined in the construction.

Sim_1. On input $(1^\lambda, T, 1^n, 1^m, C)$:

1. Generate the SNARG parameters, PKE keys, and hash key using the corresponding setup algorithms of $\mathsf{IVC.Gen}$, with the primitive security parameters specified in Theorem 6.1.
2. Sample $(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TdSetup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}})$.
3. For $\ell = 1, \ldots, \ell_{\max}$, generate crs_ℓ using the BARG setup algorithm and the setup inputs prescribed by $\mathsf{IVC.Gen}$, with $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$ in place of $\mathsf{crs}_{\mathsf{NIZK}}$. In particular, each recursive relation uses the previously generated public parameters.
4. Assemble the public IVC CRS $\widetilde{\mathsf{crs}}$ as in $\mathsf{IVC.Gen}$. Set $\mathsf{st} := \mathsf{td}_{\mathsf{NIZK}}$ and return $(\widetilde{\mathsf{crs}}, \mathsf{st})$.

Sim_2. On input $(\widetilde{\mathsf{crs}}, \mathsf{st}, u, v, 1^d)$, parse $\widetilde{\mathsf{crs}}$ as in $\mathsf{IVC.P}$. Return $\perp$ unless the query is well formed, $1 \le d \le T$, and $u, v \in \{0,1\}^n$. Otherwise:

1. Independently sample encryption randomness r_0, r_d and ρ_j for $0 < j < d$. Set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, u; r_0)$, $\mathsf{ct}_d := \mathsf{PKE.Enc}(\mathsf{pk}, v; r_d)$, and $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, 0^n; \rho_j)$ for every $0 < j < d$. Initialize $S^{(\ell)} = \emptyset$ for $0 \le \ell \le \ell_{\max}$ and $h_0 = \perp$.
2. For $j = 0, \ldots, d-1$, execute:
   (a) Set $m_j := \mathsf{Ch}_0(h_j, \mathsf{ct}_j, \mathsf{ct}_{j+1})$ and sample $\pi_j^{(0)} \leftarrow \mathsf{NIZK.Sim}(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \mathsf{st}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}), m_j)$.
   (b) Insert $(j, \mathsf{ct}_j, \perp, \mathsf{ct}_{j+1}, h_j, \pi_j^{(0)})$ into $S^{(0)}$. Execute the merge procedure of $\mathsf{IVC.P}$, using the BARG prover on the prescribed child instances and witnesses.
   (c) Set h_{j+1} to the chain value obtained by folding the descriptors of the resulting frontier from left to right, starting at $\perp$.
3. Return $\widetilde{\Pi}_d := (\mathsf{ct}_0, \mathsf{ct}_d, r_0, r_d, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

Every call uses fresh independent randomness, including repeated statements. Only the level-0 proofs are simulated; all BARG proofs are generated using the BARG prover.

Accepting NIZK simulation supplies valid witnesses for the level-1 BARG proofs. Inductively, perfect BARG completeness makes every higher merge accepting. The merge procedure preserves the prescribed ciphertext adjacency and chain values, and the returned openings match the endpoint ciphertexts. Hence $\widetilde{\Pi}_d$ verifies. The algorithm runs in time polynomial in its input length, including 1^d.

Hybrids. Fix a PPT adversary $\mathcal{A}$ in the zero-knowledge experiment of Definition 4.1. For a valid query $((x_0, x_t, 1^t), (w_1, \ldots, w_t))$, write $z_0 := x_0$ and $z_i := C(z_{i-1}, w_i)$ for $i \in [t]$, so $z_t = x_t$. Each query uses fresh independent randomness. Write $H_b^{\mathcal{A}}$ for $\mathcal{A}$'s output bit in hybrid H_b.

Hybrid H_0. Sample $\mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)$ and run $\mathcal{A}^{\mathcal{O}_{\mathsf{real}}(\mathsf{crs},\cdot)}(\mathsf{crs})$. This is the real zero-knowledge experiment.

Hybrid H_1. In $\mathsf{IVC.Gen}$, replace the NIZK setup call by $(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TdSetup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}})$. Generate the BARG CRSs in increasing order of level for the recursive relations defined using these public parameters.

For each valid query, initialize $S^{(\ell)} = \emptyset$ for all levels, $h_0 = \bot$, and $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$ with fresh encryption randomness r_0. For $j = 0, \ldots, t-1$, execute:

1. Generate $\pi_{\mathsf{SNARG},j} \leftarrow \mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, (z_j, z_{j+1}), w_{j+1})$.
2. Sample fresh encryption randomness r_{j+1} and set $\mathsf{ct}_{j+1} := \mathsf{PKE.Enc}(\mathsf{pk}, z_{j+1}; r_{j+1})$.
3. Set $m_j := \mathsf{Ch}_0(h_j, \mathsf{ct}_j, \mathsf{ct}_{j+1})$ and invoke the wrapper from the NIZK-SoK zero-knowledge definition: $\pi_j^{(0)} \leftarrow \mathsf{Sim}'(\mathsf{td}_{\mathsf{NIZK}}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}), (z_j, z_{j+1}, r_j, r_{j+1}, \pi_{\mathsf{SNARG},j}), m_j)$, with its associated CRS $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$.
4. Insert $(j, \mathsf{ct}_j, \bot, \mathsf{ct}_{j+1}, h_j, \pi_j^{(0)})$ into $S^{(0)}$ and execute the merge procedure of $\mathsf{IVC.P}$, using the BARG prover on the prescribed child instances and witnesses. Set h_{j+1} to the fold of the resulting frontier descriptors, starting at $\bot$.

Return $(\mathsf{ct}_0, \mathsf{ct}_t, r_0, r_t, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

Lemma 6.5 *For every PPT adversary $\mathcal{A}$ and all sufficiently large λ, $|\Pr[H_0^{\mathcal{A}} = 1] - \Pr[H_1^{\mathcal{A}} = 1]| \le 2^{-\kappa^\delta}$.*

Proof. Construct an adversary $\mathcal{B}$ against adaptive multi-theorem zero knowledge of the NIZK-SoK. Before receiving its challenge CRS, $\mathcal{B}$ generates the SNARG parameters, PKE keys, and hash key, thereby fixing $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. Given the challenge NIZK CRS, it generates the BARG CRSs in increasing order of level and runs $\mathcal{A}$ on the assembled IVC CRS.

For each valid query, $\mathcal{B}$ performs the operations of H_1, using its challenge proof oracle in Step 3. Every submitted witness satisfies $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ by perfect SNARG completeness and the definitions of the ciphertexts. It returns the resulting IVC proof and eventually outputs $\mathcal{A}$'s bit.

With the real NIZK oracle, the execution has distribution H_0; with the simulated oracle, it has distribution H_1. The statements and chain messages may depend on previous oracle answers, as permitted by adaptive multi-theorem zero knowledge. The reduction runs in polynomial time. Since $\delta < \min_i \delta_i$, the NIZK-SoK zero-knowledge guarantee in Theorem 6.1 bounds its distinguishing gap by $2^{-\kappa^\delta}$ for all sufficiently large λ.

Hybrid H_2. Use the setup of H_1. For each valid query, independently sample $r_0, \ldots, r_t$ and set $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, z_j; r_j)$ for $j = 0, \ldots, t$. Initialize $S^{(\ell)} = \emptyset$ for all levels and $h_0 = \bot$. For $j = 0, \ldots, t-1$, execute:

1. Set $m_j := \mathsf{Ch}_0(h_j, \mathsf{ct}_j, \mathsf{ct}_{j+1})$ and invoke $\pi_j^{(0)} \leftarrow \mathsf{NIZK.Sim}(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}), m_j)$.
2. Perform the insertion, merges, and chain update of Step 4 in H_1.

Return the same proof tuple as in H_1.

Lemma 6.6 *For every PPT adversary $\mathcal{A}$, its views in H_1 and H_2 have identical distributions.*

Proof. Every witness supplied to Sim' in H_1 satisfies $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. Hence the wrapper invokes $\mathsf{NIZK.Sim}$ on the same statement and message as in H_2. The SNARG proofs are not used elsewhere, so omitting their computation preserves the output distribution. Within each query, the supplied witness fixes the states $z_0, \ldots, z_t$, and all encryption randomness is independent of the proof-generation randomness. Sampling the ciphertexts before the leaf loop therefore also preserves the output distribution.

Hybrid H_3. Use the setup of H_1. For each valid query, independently sample $r_0, \ldots, r_t$ and set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$, $\mathsf{ct}_t := \mathsf{PKE.Enc}(\mathsf{pk}, x_t; r_t)$, and $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, 0^n; r_j)$ for $0 < j < t$. Initialize $S^{(\ell)} = \emptyset$ for all levels and $h_0 = \bot$. Execute the leaf loop of H_2 using these ciphertexts and return the same proof tuple. The setup and oracle answers are exactly those of Sim_1 and $\mathcal{O}_{\mathsf{sim}}$.

Lemma 6.7 *For every PPT adversary $\mathcal{A}$, let $M = M(\lambda)$ be a polynomial upper bound on the total number of interior encryption operations across its valid queries in any execution. Then, for all sufficiently large λ, $|\Pr[H_2^{\mathcal{A}} = 1] - \Pr[H_3^{\mathcal{A}} = 1]| \le M\, 2^{-\kappa^\delta}$.*

Proof. Number the interior encryption operations by query occurrence and then by position within each query. For $a \in \{0, \ldots, M\}$, define K_a by executing H_2 with its first a interior encryptions replaced by encryptions of 0^n. All chain values and proofs are computed from the resulting ciphertexts. Thus $K_0 = H_2$ and $K_M = H_3$.

Fix $a \in [M]$. A CPA adversary $\mathcal{B}_a$ receives the challenge public key and generates the remaining setup as in Sim_1, retaining the NIZK simulation trapdoor. It runs $\mathcal{A}$ and answers queries according to K_{a-1}. At the ath interior encryption, it submits $(z_j, 0^n)$ as its challenge messages and uses the challenge ciphertext as ct_j. It continues the experiment and outputs $\mathcal{A}$'s bit. If this operation is never reached, it makes an unused challenge query with identical messages.

Note that the reduction does not need an opening of the challenge ciphertext. It simulates the leaf proofs, computes the chain values from the ciphertexts and descriptors, and generates the BARG proofs using the prescribed child proofs and boundary ciphertexts. Only endpoint openings are returned. The two challenge choices give K_{a-1} and K_a, respectively. The reduction runs in polynomial time, so the CPA-security guarantee in Theorem 6.1, with $\delta < \min_i \delta_i$, bounds each adjacent distinguishing gap by $2^{-\kappa^\delta}$ for all sufficiently large λ. Summing over $a \in [M]$ proves the claim.

Proof (Proof of zero knowledge). Hybrid H_0 is the real experiment, and H_3 is the simulated experiment for $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$. The preceding lemmas bound their distinguishing gap by $(M+1)2^{-\kappa^\delta}$. Since $\mathcal{A}$ is PPT and query depths are given in unary, M is polynomially bounded in λ. The parameter choice of Theorem 6.1 makes this bound negligible in λ. Hence the construction satisfies zero knowledge.

6.4 Weak simulation soundness

We sketch the proof of weak simulation soundness; the full argument appears in Appendix E.2. As in [DJJ+25], an accepting proof of a false target statement must contain a frontier block whose endpoints do not represent a valid computation of the prescribed length. Somewhere extraction for the BARG lets us descend through such a block to an accepting false leaf. If its message was never used by the simulator, NIZK simulation extractability yields a witness for the leaf relation, contradicting SNARG soundness. The additional difficulty is that the adversary may reuse a simulated leaf, whose statement need not be true.

For a reused leaf, collision resistance binds its endpoint ciphertexts and incoming chain value to those of a recorded simulated leaf. The descriptors exposed along the extraction path and those retained in the simulation record supply the hash inputs needed to compare the two prefixes backwards. Collision resistance forces these inputs to agree, identifying the same descriptors, leaf position, and starting ciphertext (Lemma E.5). Moreover, CPA security and perfect correctness imply that distinct simulator queries have distinct starting ciphertexts, except with negligible probability (Lemma E.9). Thus the reused leaf identifies a unique query.

We next show that the target inherits this query's entire prefix. At an interior boundary of the query, the simulator's ciphertext has hidden encryption randomness. Unexplainability prevents the target from ending there, since its endpoint opening would explain that ciphertext. It also prevents a fresh accepting leaf from continuing from that boundary, since NIZK simulation extractability would supply such an opening. A reused leaf must instead belong to the same query, by context binding and query uniqueness, and therefore extends the inherited prefix. CRS indistinguishability justifies the CRS switches needed to compare extractions at successive boundaries, yielding agreement through the query's endpoint (Lemmas E.10 and E.11).

The winning condition now implies that the target extends beyond the inherited query and that the remaining suffix is false. The recursive decomposition supplies a consecutive cover of this suffix by blocks with matching boundary ciphertexts, so some suffix block must be false (Lemma E.12). A second descent yields an accepting false leaf in the suffix, whose message must be fresh: otherwise, context binding and query

uniqueness would place it inside the inherited query, contradicting its position in the suffix. Leaf soundness rules out this final case (Lemma E.13). The proof accounts for the averaging and CRS-switch losses using the sub-exponential security and parameter choice in Theorem 6.1.

Corollary 6.8 (Concrete instantiations) *Assume subexponentially secure indistinguishability obfuscation and either subexponential LWE or the subexponential bilinear-map assumptions against non-uniform adversaries. For polynomially bounded $n, m, |C|$, with $n \geq 1$, let $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be a step circuit, whose size includes its input and witness wires. For every depth bound $1 \leq T \leq 2^\lambda$, there exists a perfectly complete, zero-knowledge IVC scheme for C that satisfies* w-SIM-SND *against non-uniform PPT adversaries. Its proof size is* $\mathrm{poly}(\lambda, n, \log T, \log |C|)$*, and its CRS size and setup, update, and verification times are* $\mathrm{poly}(\lambda, \log T, |C|)$.

Proof. We instantiate the ingredients of Theorem 6.1. The non-adaptively sound SNARGs and rate-1 somewhere-extractable BARGs are instantiated as in [DJJ+25]. The chosen assumptions also provide perfectly correct CPA-secure PKE, PRG, collision-resistant hashing, and adaptively secure multi-theorem NIZKs for NP.

The simulation-extractable NIZK compiler of De Santis et al. [DDO+01], using a NIZK, PKE, and strongly unforgeable one-time signatures, gives a perfectly complete simulation-extractable NIZK with adaptive multi-theorem zero knowledge. For an NP relation R, apply this compiler to $\widehat{R}((X, M), W) := R(X, W)$, with the message M included in the public statement. An accepting tuple (X, π, M) that was not returned by the simulator corresponds exactly to a fresh statement–proof pair $((X, M), \pi)$ for the compiled NIZK. Thus the resulting NIZK-SoK satisfies the extraction condition of Definition 3.1, including when (X, M) was previously queried. These transformations incur only polynomial security losses, so subexponential security is preserved.

Apply Lemma E.2 to the perfectly correct CPA-secure PKE and a subexponentially secure PRG. At primitive security parameter κ, the compiled PKE uses a uniformly sampled κ-bit seed as its encryption randomness, while its key and ciphertext lengths and algorithm costs are polynomial in (κ, n). The CPA proof incurs only a constant number of PRG and PKE security losses. For every unexplainability adversary A, the distinguisher D constructed in Lemma E.2 gives

$$\Pr[A \text{ produces a valid explanation}] \leq \mathrm{Adv}_D^{\mathrm{PRG}}(\kappa) + 2^{-\kappa}.$$

Therefore the compiled PKE has subexponential CPA security and unexplainability, as required by Theorem 6.1.

We use the recursive rate-1 BARG instantiations of [DJJ+25] with their polynomial bounds throughout recursion. The compiled PKE and NIZK-SoK have polynomial size and cost at level 0, while context chaining adds a hash field and polynomial-size hash computations to the recursive instances and relations. With the corresponding fixed public size bounds, these instantiations satisfy the recursive efficiency condition of Theorem 6.1. Choosing the primitive security parameters as in that theorem therefore gives the claimed security and efficiency.

Acknowledgments

Matteo Campanelli was supported by the Estonian Research Council grant PRG2531.

References

ACDK26. Behzad Abdolmaleki, Matteo Campanelli, Quang Dao, and Hamidreza Khoshakhlagh. On the simulation-extractability of proof-carrying data. In *IACR International Conference on Public-Key Cryptography*. Springer, 2026.

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

AST24. Arasu Arun, Srinath T. V. Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 3–33. Springer, Cham, May 2024.

BBBF18. Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Cham, August 2018.

BBF19. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Cham, August 2019.

BCL+21. Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 681–710, Virtual Event, August 2021. Springer, Cham.

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

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

BMRS20. Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352, 2020.

CFP25a. 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, Report 2025/2086, 2025.

CFP25b. Matteo Campanelli, Dario Fiore, and Mahak Pancholi. When can we incrementally prove computations of arbitrary depth? Cryptology ePrint Archive, Report 2025/1413, 2025.

CHAK23. Matteo Campanelli, Mathias Hall-Andersen, and Simon Holmgaard Kamp. Curve trees: Practical and transparent zero-knowledge accumulators. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 4391–4408. USENIX Association, August 2023.

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

CL06. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Berlin, Heidelberg, August 2006.

CT10. Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 310–331. Tsinghua University Press, January 2010.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Berlin, Heidelberg, August 2001.

DEH25. Stefan Dziembowski, Shahriar Ebrahimi, and Parisa Hassanizadeh. VIMz: Private proofs of image manipulation using folding-based zkSNARKs. *PoPETs*, 2025(2):344–362, April 2025.

DJJ+25. Pratish Datta, Abhishek Jain, Zhengzhong Jin, Alexis Korb, Surya Mathialagan, and Amit Sahai. Incrementally verifiable computation for NP from standard assumptions. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 611–642. Springer, Cham, August 2025.

DW14. Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 313–326. Springer, Cham, September 2014.

FFHR19. Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 159–190. Springer, Cham, December 2019.

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Berlin, Heidelberg, December 2012.

FT26. Antonio Faonio and Lili Tong. Non-malleable reductions of knowledge. Cryptology ePrint Archive, Paper 2026/1822, 2026.

GMY03. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, May 2003.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

JMW26. Abhishek Jain, Surya Mathialagan, and Brent Waters. Incrementally verifiable computation without extraction. In *Annual International Cryptology Conference*, pages 3–34. Springer, 2026.

KST22. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 359–388. Springer, Cham, August 2022.

LLNW16. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Berlin, Heidelberg, May 2016.

MMZ25. Mary Maller, Nicolas Mohnblatt, and Aranxta Zapico. IVC in the open-and-sign random oracle model. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part VII*, volume 16251 of *LNCS*, pages 333–365. Springer, Singapore, December 2025.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Berlin, Heidelberg, August 2003.

NDC+24. Wilson D. Nguyen, Trisha Datta, Binyi Chen, Nirvan Tyagi, and Dan Boneh. Mangrove: A scalable framework for folding-based SNARKs. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 308–344. Springer, Cham, August 2024.

NT16. Assa Naveh and Eran Tromer. PhotoProof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy*, pages 255–271. IEEE Computer Society Press, May 2016.

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

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

Smy21. Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of helios. *Journal of Computer Security*, 29(6):551–611, 2021.

Val08. Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, March 2008.

A Additional Related Work

Non-malleability of reductions of knowledge. Faonio and Tong [FT26] study non-malleability for *reductions of knowledge*, an abstraction that captures folding schemes and, more generally, the step of turning a proof for one relation into a proof for another. They give conditions under which composing two such reductions, in particular two non-interactive ones obtained through Fiat–Shamir, preserves simulation extractability, isolating properties of the individual components that are easier to establish than simulation extractability of the composition. Their setting is orthogonal to ours in two respects: the composition they analyze is not incremental, in that it chains reductions for a single statement rather than extending a computation, and their guarantees are extractability-based, hence outside the falsifiable regime we target.

IVC without extraction. Jain, Mathialagan and Waters [JMW26] gave IVC constructions from non-extractable building blocks. Their main construction is for *deterministic* computations, for which our notions are not meaningful. They also give a construction for non-deterministic computations, which is however limited to two hops and is only non-adaptively sound, and is therefore not an informative case study for the attacks of Section 5.

B Examples of (Non) Compliance

Let us consider this toy example. Define a step circuit C over the integers as

$$C(x,w) := \begin{cases} x + 2(w+1) & \text{if } 0 \le x, w < 2^{32} \\ \bot & \text{otherwise} \end{cases}$$

Above we assume that any arithmetic operation on top of $\bot$ maps to $\bot$. Since every step adds an even amount, C preserves the parity of the state. In particular, from 0 only even states are reachable, so $0 \overset{d}{\not\rightsquigarrow} y$ for every d and every odd y. Moreover, every step strictly increases the state.

Remark B.1 *In Section 2.1 we showed that a naive adaptation of simulation-soundness to IVC is unachievable. The circuit C defined above gives a concrete instance. Recall what happens in the naive adaptation: an adversary obtains a simulated proof for a false statement $x_0 \overset{t}{\rightsquigarrow} x_t$, extends it by one honest step with some witness w, and outputs the resulting proof for $x_0 \overset{t+1}{\rightsquigarrow} C(x_t, w)$. The adversary wins the simulation-soundness experiment whenever the new statement is also false and has never been queried. With C as above, let $x_0 = 0$, $t = 3$ and $x_t = 121$. Since 121 is odd, the statement $0 \overset{3}{\rightsquigarrow} 121$ is false, and the adversary can obtain a simulated proof for it. Applying the honest prover with, e.g., $w = 4$ yields an accepting proof for $0 \overset{4}{\rightsquigarrow} 131$. This statement was never queried and it is false, since 131 is also odd.*

Examples of (non) compliance from C. We now give examples related to the notion of compliance (Definition 4.3). Take $P = \left\{ \left[0 \overset{3}{\underset{!}{\rightsquigarrow}} 121\right] \right\}$, a single (false) promise leap asserting that 121 can be reached from 0 in three steps. Then:

- *A compliant statement.* The statement $(0, 143, 5)$ is compliant via P: the leap $\left[0 \overset{3}{\underset{!}{\rightsquigarrow}} 121\right] \in P$ has $3 < 5$, and $121 \overset{2}{\rightsquigarrow} 143$ genuinely holds (witnessed, e.g., by $w_4 = 4$ and $w_5 = 5$).
- *A non-compliant statement.* The statement $(0, 121, 5)$ is not compliant via P: the statement by itself is false and the only leap available is $\left[0 \overset{3}{\underset{!}{\rightsquigarrow}} 121\right]$, which leaves 2 steps that are impossible to cover no matter the choice of the witness, since every step strictly increases the state.

C The IVC of [DJJ+25] and Concrete Malleability Attacks

In this section we describe malleability attacks on the IVC construction of [DJJ+25]. One of the attacks described here is a concrete instantiation of the more general attack described in the technical overview and is provided as a more detailed illustration. We also show an additional attack that is specific to [DJJ+25]. We first describe the [DJJ+25] construction.

C.1 Background: the IVC construction $\Pi^{\mathbf{IVC}}_{[\mathbf{DJJ+25}]}$

We begin by defining the NP languages and relations that underlie the construction. These formalize correctness of a single computation step, correctness of encrypted configurations, and correctness of recursively merged proofs. We then present the IVC construction of [DJJ+25].

In Figure 9 we show the NP languages and relations underlying the construction of [DJJ+25].

IVC for NP. We now present the full construction.

Building blocks (all with sub-exponential security):

- $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$: a perfectly correct public-key encryption scheme.
- $\mathsf{SNARG} = (\mathsf{SNARG.Gen}, \mathsf{SNARG.P}, \mathsf{SNARG.V})$: a non-adaptive succinct argument for **NP**.
- $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.TdSetup}, \mathsf{NIZK.P}, \mathsf{NIZK.V}, \mathsf{NIZK.Sim}, \mathsf{NIZK.E})$: a NIZK argument of knowledge.
- $\mathsf{BARG} = (\mathsf{BARG.Setup}, \mathsf{BARG}.\mathcal{P}, \mathsf{BARG}.\mathcal{V})$: a rate-1 somewhere extractable batch argument for **NP**.

IVC Construction from [DJJ+25]

$\underline{\mathsf{IVC.Gen}}(1^n, T, 1^n, 1^m, C)$:

1. Set $\ell_{\max} = \lceil \log T \rceil$.
2. **Setup SNARG** for one-step computation:
$$(\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}) \leftarrow \mathsf{SNARG.Gen}(1^{n_{\mathsf{SNARG}}}, 1^{2n}, 1^m, R_{C,1}).$$
3. **Setup PKE:** $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^{n_{\mathsf{PKE}}}, 1^n)$.
4. **Setup NIZK** for relation $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$:
$$\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^{n_{\mathsf{NIZK}}}, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}).$$
5. **Setup level-1 BARG** (batch size $k = 2$, relation $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$):
$$\mathsf{crs}_1 \leftarrow \mathsf{BARG.Setup}(1^{n_{\mathsf{sweBARG}}}, 2, 1^{n_1}, 1^{m_1}, R_{1,\mathsf{crs}_{\mathsf{NIZK}}}).$$
6. **Setup level-ℓ BARG** for each $\ell \in \{2, \ldots, \ell_{\max}\}$ (batch size $k = 2$, relation $R_{\ell,\mathsf{crs}_{\ell-1}}$):
$$\mathsf{crs}_\ell \leftarrow \mathsf{BARG.Setup}(1^{n_{\mathsf{sweBARG}}}, 2, 1^{n_\ell}, 1^{m_\ell}, R_{\ell,\mathsf{crs}_{\ell-1}}).$$
7. **Output:** $\mathsf{crs} := (\ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell \in [\ell_{\max}]})$.

$\underline{\mathsf{IVC.P}}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \Pi_{i-1}, w_i)$:

1. Parse $\mathsf{crs} = (\ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell \in [\ell_{\max}]})$.
2. **Base case:** If $i = 1$:
   (a) Sample $r_0 \leftarrow \{0,1\}^{n_{\mathsf{PKE}}}$ and compute $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$.
   (b) Initialize $S^{(\ell)} = \emptyset$ for all $\ell \in \{0, \ldots, \ell_{\max}\}$.
3. **Verify previous proof:** If $i > 1$:
   (a) Parse $\Pi_{i-1} = (\mathsf{ct}_0, \mathsf{ct}_{i-1}, r_0, r_{i-1}, \{S^{(\ell)}\}_{\ell \in \{0,\ldots,\ell_{\max}\}})$.
   (b) If $\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_{i-1}, i-1), \Pi_{i-1}) = 0$, output $\perp$.
4. **Compute next configuration and SNARG proof:**
   (a) $x_i = C(x_{i-1}, w_i)$.
   (b) $\pi_{\mathsf{SNARG}} \leftarrow \mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, (x_{i-1}, x_i), w_i)$.
5. **Encrypt and compute level-0 NIZK proof:**
   (a) Sample $r_i \leftarrow \{0,1\}^{n_{\mathsf{PKE}}}$ and compute $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_i; r_i)$.
   (b) $\pi_i^{(0)} \leftarrow \mathsf{NIZK.P}\big(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{i-1}, \mathsf{ct}_i), (x_{i-1}, x_i, r_{i-1}, r_i, \pi_{\mathsf{SNARG}})\big)$.
   (c) Add $(i-1, \mathsf{ct}_{i-1}, \perp, \mathsf{ct}_i, \pi_i^{(0)})$ to $S^{(0)}$.
6. **Merge proofs:** For $\ell \in \{0, \ldots, \ell_{\max} - 1\}$:
   (a) *Check if two proofs at level ℓ exist:*
   - If $|S^{(\ell)}| < 2$, continue to the next ℓ.
   - If $|S^{(\ell)}| = 2$, parse
   $S^{(\ell)} = \{(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, \pi^{(\ell)}), (\tilde{i}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{mid}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \widetilde{\pi}^{(\ell)})\}$,
   and set $S^{(\ell)} = \perp$.
   (b) *Verify adjacency:* If $\mathsf{ct}_{\mathsf{fin}} \neq \widetilde{\mathsf{ct}}_{\mathsf{init}}$ or $i_{\mathsf{init}} + 2^\ell \neq \tilde{i}_{\mathsf{init}}$, output $\perp$.
   (c) *Create level-(ℓ+1) proof:*
   - If $\ell = 0$: The two entries have $\mathsf{ct}_{\mathsf{mid}} = \perp$ and $\widetilde{\mathsf{ct}}_{\mathsf{mid}} = \perp$. Compute
   $$\pi^{(1)} \leftarrow \mathsf{BARG}.\mathcal{P}\big(\mathsf{crs}_1, \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), (\widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}})\}, \{\pi^{(0)}, \widetilde{\pi}^{(0)}\}\big),$$
   and add $(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \pi^{(1)})$ to $S^{(1)}$.
   - If $\ell \geq 1$: Compute
   $$\pi^{(\ell+1)} \leftarrow \mathsf{BARG}.\mathcal{P}\big(\mathsf{crs}_{\ell+1}, \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), (\widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}})\}, \{(\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell)}), (\widetilde{\mathsf{ct}}_{\mathsf{mid}}, \widetilde{\pi}^{(\ell)})\}\big),$$
   and add $(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \pi^{(\ell+1)})$ to $S^{(\ell+1)}$.
7. **Output:** $\Pi_i = (\mathsf{ct}_0, \mathsf{ct}_i, r_0, r_i, \{S^{(\ell)}\}_{\ell \in \{0,\ldots,\ell_{\max}\}})$.

(a) One-Step Language $L_{C,1}$ and Relation $R_{C,1}$:
The base relation for a single computation step.

$$L_{C,1} = \{(x, x') \ : \ \exists\, w \text{ s.t. } C(x,w) = x'\},$$

$$R_{C,1}\big((x,x'),\ w\big) = 1 \iff C(x,w) = x'.$$

(b) Level-0 Language $L_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ and Relation $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$:
A level-0 proof is a NIZK proof certifying that a SNARG proof of one computation step is valid and that the ciphertexts are well-formed. The SNARG proves the one-step relation $R_{C,1}$; the NIZK additionally checks ciphertext consistency.

$$L_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}} = \left\{ (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) \ : \ \begin{array}{l} \exists\,(x_{\mathsf{init}},\ x_{\mathsf{fin}},\ r_{\mathsf{init}},\ r_{\mathsf{fin}},\ \pi_{\mathsf{SNARG}}) \ \text{ s.t.} \\ \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V},\ (x_{\mathsf{init}}, x_{\mathsf{fin}}),\ \pi_{\mathsf{SNARG}}) = 1 \\ \wedge\ \mathsf{ct}_{\mathsf{init}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{init}};\ r_{\mathsf{init}}) \\ \wedge\ \mathsf{ct}_{\mathsf{fin}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{fin}};\ r_{\mathsf{fin}}) \end{array} \right\},$$

$$R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}\big((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}),\ \ (x_{\mathsf{init}},\ x_{\mathsf{fin}},\ r_{\mathsf{init}},\ r_{\mathsf{fin}},\ \pi_{\mathsf{SNARG}})\big) = 1$$
$$\iff \begin{array}{l} \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V},\ (x_{\mathsf{init}}, x_{\mathsf{fin}}),\ \pi_{\mathsf{SNARG}}) = 1 \\ \wedge\ \mathsf{ct}_{\mathsf{init}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{init}};\ r_{\mathsf{init}}) \\ \wedge\ \mathsf{ct}_{\mathsf{fin}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{fin}};\ r_{\mathsf{fin}}). \end{array}$$

This relation is proved by the NIZK.

(c) Level-1 Language $L_{1,\mathsf{crs}_{\mathsf{NIZK}}}$ and Relation $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$ (sweBARG, batch size $k = 2$):
A level-1 proof is a sweBARG proof verifying a batch of *two* level-0 NIZK proofs.

$$L_{1,\mathsf{crs}_{\mathsf{NIZK}}} = \left\{ (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) \ : \ \exists\, \pi^{(0)} \text{ s.t. } \mathsf{NIZK.V}\big(\mathsf{crs}_{\mathsf{NIZK}},\ (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}),\ \pi^{(0)}\big) = 1 \right\},$$

$$R_{1,\mathsf{crs}_{\mathsf{NIZK}}}\big((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}),\ \pi^{(0)}\big) = 1 \iff \mathsf{NIZK.V}\big(\mathsf{crs}_{\mathsf{NIZK}},\ (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}),\ \pi^{(0)}\big) = 1.$$

(d) Level-ℓ Language $L_{\ell,\mathsf{crs}_{\ell-1}}$ and Relation $R_{\ell,\mathsf{crs}_{\ell-1}}$ (for $2 \le \ell \le \ell_{\max}$, sweBARG, batch size $k = 2$):
A level-ℓ proof is a sweBARG proof verifying a batch of two level-$(\ell{-}1)$ sweBARG proofs, connected via a midpoint ciphertext $\mathsf{ct}_{\mathsf{mid}}$.

$$L_{\ell,\mathsf{crs}_{\ell-1}} = \left\{ (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) \ : \ \begin{array}{l} \exists\,(\mathsf{ct}_{\mathsf{mid}},\ \pi^{(\ell-1)}) \text{ s.t.} \\ \mathsf{sweBARG.V}\big(\mathsf{crs}_{\ell-1},\ \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}),\ (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})\},\ \pi^{(\ell-1)}\big) = 1 \end{array} \right\},$$

$$R_{\ell,\mathsf{crs}_{\ell-1}}\big((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}),\ (\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell-1)})\big) = 1$$
$$\iff \mathsf{sweBARG.V}\big(\mathsf{crs}_{\ell-1},\ \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}),\ (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})\},\ \pi^{(\ell-1)}\big) = 1.$$

Summary of Language Hierarchy:

Level	Primitive	Batch	What it certifies
0	se-NIZK	–	SNARG + ciphertext consistency
1	sweBARG	$k = 2$	Batch of two level-0 NIZK proofs
$\ell \ge 2$	sweBARG	$k = 2$	Batch of two level-$(\ell{-}1)$ sweBARG proofs

Fig. 9: The NP languages and relations underlying the IVC construction of [DJJ+25].

$\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_t, t), \Pi_t)$:

1. Parse $\mathsf{crs} = (\ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell\in[\ell_{\max}]})$ and $\Pi_t = (\mathsf{ct}_0, \mathsf{ct}_t, r_0, r_t, \{S^{(\ell)}\}_{\ell\in\{0,\ldots,\ell_{\max}\}})$.
2. Set $i_{\mathsf{curr}} = 0$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_0$.
3. **Verify proofs at each level:** For $\ell \in \{\ell_{\max}, \ell_{\max}-1, \ldots, 0\}$:
   (a) Let s_ℓ be the ℓ-th least significant bit of t.
   - If $s_\ell = 0$: check that $S^{(\ell)} = \emptyset$ and continue.
   - If $s_\ell = 1$: parse $S^{(\ell)} = \{(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, \pi^{(\ell)})\}$.

   (b) *Verify adjacency:* If $i_{\mathsf{curr}} \neq i_{\mathsf{init}}$ or $\mathsf{ct}_{\mathsf{curr}} \neq \mathsf{ct}_{\mathsf{init}}$, output 0. Update $i_{\mathsf{curr}} = i_{\mathsf{curr}} + 2^\ell$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_{\mathsf{fin}}$.
   (c) *Verify proof:*
   - If $\ell = 0$: check $\mathsf{NIZK.V}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}) = 1$.
   - If $\ell = 1$: check $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}_1, \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})\}, \pi^{(1)}) = 1$.
   - If $\ell \geq 2$: check $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}_\ell, \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})\}, \pi^{(\ell)}) = 1$.

   If any check fails, output 0.
4. **Verify ciphertexts:** If $\mathsf{ct}_{\mathsf{curr}} \neq \mathsf{ct}_t$, or $\mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0) \neq \mathsf{ct}_0$, or $\mathsf{PKE.Enc}(\mathsf{pk}, x_t; r_t) \neq \mathsf{ct}_t$, output 0.
5. **Output** 1.

Fig. 10: The IVC construction of [DJJ+25].

.

The IVC simulator. Following the zero-knowledge proof in Section 4.5 of [DJJ+25], define $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ as follows.

$\mathsf{Sim}_1(1^\lambda, T, 1^n, 1^m, C)$: Generate the SNARG parameters and PKE keys as in $\mathsf{IVC.Gen}$, using parameter κ. Sample $(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TdSetup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}})$. Generate the BARG CRSs in increasing order of level, using $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$ in the recursive relations. Set $\mathsf{tdcrs} := (\ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell=1}^{\ell_{\max}})$ and $\mathsf{st} := (T, \mathsf{td}_{\mathsf{NIZK}})$. Return $(\mathsf{tdcrs}, \mathsf{st})$.

$\mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{st}, x_0, x_t, 1^t)$: Parse $\mathsf{st} = (T, \mathsf{td}_{\mathsf{NIZK}})$. Return $\perp$ unless the query is well formed, $1 \leq t \leq T$, and $x_0, x_t \in \{0,1\}^n$. Otherwise, using fresh independent randomness:

1. Sample $r_0, r_t \leftarrow \{0,1\}^\kappa$ and set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$ and $\mathsf{ct}_t := \mathsf{PKE.Enc}(\mathsf{pk}, x_t; r_t)$. For $0 < j < t$, independently sample $\mathsf{ct}_j \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0^n)$. Initialize $S^{(\ell)} := \emptyset$ for every level.
2. For $j = 0, \ldots, t-1$, sample $\pi_j^{(0)} \leftarrow \mathsf{NIZK.Sim}(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}))$. Insert $(j, \mathsf{ct}_j, \perp, \mathsf{ct}_{j+1}, \pi_j^{(0)})$ into $S^{(0)}$, and execute the merge procedure of $\mathsf{IVC}.\mathcal{P}$ using the BARG prover on the prescribed child instances and witnesses.
3. Return $\Pi_t := (\mathsf{ct}_0, \mathsf{ct}_t, r_0, r_t, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

Only the level-0 proofs are simulated. The accepting simulated NIZK proofs supply valid witnesses for the first BARG merges, and BARG completeness makes every higher merge accepting. The returned openings match the endpoint ciphertexts.

C.2 IVC of [DJJ+25] is not SIM-SND

Consider the IVC construction of [DJJ+25], instantiated with a simulation-extractable NIZK, a sound SNARG, a perfectly correct PKE, and a standard (index-hiding, somewhere-extractable) sweBARG. If the sweBARG is *malleable*, i.e., there exists a polynomial-time algorithm Maul such that for any crs_ℓ, any statement $\boldsymbol{s}$, and any accepting proof $\pi^{(\ell)}$, the algorithm $\mathsf{Maul}(\mathsf{crs}_\ell, \boldsymbol{s}, \pi^{(\ell)})$ outputs $\pi'^{(\ell)} \neq \pi^{(\ell)}$ with $\mathsf{sweBARG.V}(\mathsf{crs}_\ell, \boldsymbol{s}, \pi'^{(\ell)}) = 1$, then the IVC scheme is *not* simulation-sound in the sense of Definition SIM-SND. We construct $\mathcal{A}$ as follows.

Adversary $\mathcal{A}^{\mathsf{Sim}'(\mathsf{st},\cdot)}(\mathsf{tdcrs})$:

1. Choose an arbitrary false statement $(x_0, x_t, t) \notin \mathcal{L}_C$.
2. Query the simulation oracle on $(x_0, x_t, 1^t)$, obtaining Π_t.
3. Parse $\Pi_t = (\mathsf{ct}_0, \mathsf{ct}_t, r_0, r_t, \{S^{(\ell)}\}_{\ell\in\{0,\ldots,\ell_{\max}\}})$.
4. Select any level $\ell^* \geq 1$ such that $S^{(\ell^*)}$ is non-empty. Parse $S^{(\ell^*)} = \{(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, \pi^{(\ell^*)})\}$.
5. Compute a mauled proof:

$$\pi'^{(\ell^*)} \leftarrow \mathsf{Maul}\big(\mathsf{crs}_{\ell^*},\ \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})\},\ \pi^{(\ell^*)}\big).$$

6. Define $S'^{(\ell^*)} = \{(i_{\text{init}}, \mathsf{ct}_{\text{init}}, \mathsf{ct}_{\text{mid}}, \mathsf{ct}_{\text{fin}}, \pi'^{(\ell^*)})\}$ and set

$$\Pi'_t \;=\; \big(\mathsf{ct}_0,\ \mathsf{ct}_t,\ r_0,\ r_t,\ \{S'^{(\ell)}\}_\ell\big),$$

   where $S'^{(\ell)} = S^{(\ell)}$ for all $\ell \neq \ell^*$.
7. Output (x_0, x_t, t, Π'_t).

We verify that $\mathcal{A}$ wins the simulation-soundness game:

1. **Verification passes.** The proof Π'_t differs from Π_t only in the sweBARG proof at level ℓ^*. All other components are identical, including:
   - the endpoint ciphertexts $\mathsf{ct}_0, \mathsf{ct}_t$ and randomness r_0, r_t,
   - all level-0 NIZK proofs,
   - all sweBARG proofs at levels $\ell \neq \ell^*$,
   - all intermediate ciphertexts and adjacency structure.

   The only modified check is $\mathsf{sweBARG}.\mathcal{V}(\mathsf{crs}_{\ell^*}, \boldsymbol{s}, \pi'^{(\ell^*)})$, which accepts by the guarantee of Maul. Therefore $\mathcal{V}(\mathsf{crs}, (x_0, x_t, t), \Pi'_t) = 1$.
2. **False statement.** $(x_0, x_t, t) \notin \mathcal{L}_C$ by construction.
3. **Fresh tuple.** The adversary made a single query (x_0, x_t, t) and received Π_t, so $\mathcal{Q} = \{(x_0, x_t, t, \Pi_t)\}$. Since $\pi'^{(\ell^*)} \neq \pi^{(\ell^*)}$, we have $\Pi'_t \neq \Pi_t$, and therefore $(x_0, x_t, t, \Pi'_t) \notin \mathcal{Q}$.
4. **Not compliant.** The only relevant simulator query is the target statement itself, at depth t, whereas compliance requires a query at some strictly smaller depth $d < t$ whose endpoint can be honestly extended for the remaining $t - d$ steps.

The attack exploits the design of the IVC construction: the proof has a layered structure where level-0 NIZK proofs form the "base layer" protected by simulation-extractability, while the BARG proofs at higher levels serve as an "aggregation layer" that compresses the base proofs.
The attack shows that simulation-extractability of the NIZK alone is insufficient: the adversary can maul only at the BARG layer, recycling all level-0 NIZK proofs from the simulation oracle. The attack relies on the adversary reusing the *same* false statement (x_0, x_t, t) that was queried to the simulator. Under a weaker simulation-soundness definition requiring only *statement freshness* (i.e., $(x_0^*, x_t^*, t^*) \neq (x_0^{(j)}, x_t^{(j)}, t^{(j)})$ for all queries j), this attack does not apply.
However, we describe a different attack that shows that even the weakest definition we introduce in this work, newrt-SIM-SND cannot be satisfied.

C.3 IVC of [DJJ+25] is not newrt-SIM-SND

Let $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ be a simulator described above.
Let

$$Q_{\mathsf{stmt}} = \{(x^{(j)}, x'^{(j)}, d^{(j)})\}_j, \quad \mathcal{Q}_{\mathsf{IVC}} = \{((x^{(j)}, x'^{(j)}, d^{(j)}), \Pi^{(j)})\}_j$$

denote the multi-sets of statements queried by $\mathcal{A}$ and of (statement, proof) pairs returned by Sim_2.
Recall that the compliance-based SIM-SND definition precisely captures *which statements the adversary should be able to produce proofs for, given its queries*. That is, a compliant statement is one that the adversary could already obtain by taking a queried IVC proof for (x, x', d) and honestly extending it from x' to x'' via $D - d$ true C-steps. The queried statement *must* start from the same first component x as the target: the adversary cannot pre-pend honest hops in front of a query. This is the natural notion for IVC because IVC proofs are incrementally extensible *forward* from a known intermediate state, not from an arbitrary earlier state. Concretely: given an IVC proof for (x, x', d) (honest or simulated), the honest $\mathcal{P}$ algorithm extends it one step at a time, producing proofs for $(x, x'', d+1), (x, x''', d+2), \ldots$. A trivially extended forgery thus has the form $(x, \cdot, \cdot)$ with the *same first* component as the original. Any non-compliant forgery, in particular one that changes the first component, is *not* a trivial extension of a query, and the simulation-soundness notion should rule it out. The SIM-SND definition in Section 4.3 captures this. We show that this natural definition is not satisfied by the construction of [DJJ+25].

Theorem C.1 (The IVC construction of [DJJ+25] is not newrt-SIM-SND). *There exists a PPT adversary $\mathcal{A}$ against the IVC construction of [DJJ+25, Section 4.2] that wins the* newrt-SIM-SND *game with overwhelming probability.*

We prove the theorem by exhibiting an explicit attack, that we call the Honest prefix attack. For clarity, we instantiate the attack at the smallest non-trivial parameter setting: $n^* = 4$ (the forgery's chain length) and $d^{(1)} = 2$ (the simulator query's chain length).

Setting. $\mathcal{A}$ chooses the circuit $C\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ such that:
- There exist values $a, a', b, c \in \{0,1\}^n$ and witnesses $w_{aa'}, w_{a'b} \in \{0,1\}^m$ with $C(a, w_{aa'}) = a'$ and $C(a', w_{a'b}) = b$, so $(a, b, 2) \in \mathcal{L}_C$ and the adversary knows witnesses for this.
- $(b, c, 2) \notin \mathcal{L}_C$: there is no C-chain of length 2 from b to c.
- $(a, c, 4) \notin \mathcal{L}_C$: the target statement of the forgery is false.

IVC for depth $n = 4$. For $n^* = 4$, the IVC proof Π^* has only $S^{*(2)}$ non-empty:

$$S^{*(2)} = (\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*, \pi^{*(2)})$$

where $\pi^{*(2)}$ is a level-2 BARG proof on $(\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*)$ asserting the existence of two valid level-1 BARG proofs:
- one on $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{ct}_2^*)$ (covering hops 1, 2),
- one on $(\mathsf{ct}_2^*, \mathsf{ct}_3^*, \mathsf{ct}_4^*)$ (covering hops 3, 4);

each in turn asserting two valid level-0 NIZK proofs on adjacent ciphertext pairs.

Simulator. For a simulator query on $(b, c, 2)$. $\mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{td}, b, c, 2)$, returns a proof $\Pi^{(1)} = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_2^{(1)}, r_0^{(1)}, r_2^{(1)}, \{S^{(\ell,1)}\}_\ell)$ Since $d^{(1)} = 2$ only $S^{(1,1)}$ is non-empty: $S^{(1,1)} = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)}, \mathsf{ct}_2^{(1)}, \pi^{(1)}_{\mathsf{sweBARG}})$. The internal ciphertext chain is $\mathsf{ct}_0^{(1)}$ (encrypts b with randomness $r_0^{(1)}$), $\mathsf{ct}_1^{(1)}$ (encrypts 0 with randomness $r_1^{(1)}$, *not part of the proof*), $\mathsf{ct}_2^{(1)}$ (encrypts c with randomness $r_2^{(1)}$). The level-0 NIZK proofs $\pi^{(0)}_{\mathrm{sim},1}$ on $(\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)})$ and $\pi^{(0)}_{\mathrm{sim},2}$ on $(\mathsf{ct}_1^{(1)}, \mathsf{ct}_2^{(1)})$ are produced by $\mathsf{NIZK.Sim}_2$ and serve as witnesses to $\pi^{(1)}_{\mathsf{sweBARG}}$, but are *not part of* the IVC proof.

What is visible to the adversary in $\Pi^{(1)}$:
- Endpoint ciphertexts $\mathsf{ct}_0^{(1)}, \mathsf{ct}_2^{(1)}$ and endpoint randomness $r_0^{(1)}, r_2^{(1)}$.
- Intermediate ciphertext $\mathsf{ct}_1^{(1)}$ (it is the "mid" field of the $S^{(1,1)}$ entry).
- The level-1 BARG proof $\pi^{(1)}_{\mathsf{sweBARG}}$.

Hidden from the adversary:
- Intermediate randomness $r_1^{(1)}$.
- Internal level-0 NIZK proofs $\pi^{(0)}_{\mathrm{sim},1}, \pi^{(0)}_{\mathrm{sim},2}$.

Adversary $\mathcal{A}^{\mathsf{Sim}_2(\mathsf{tdcrs},\cdot)}(\mathsf{crs})$: The adversary $\mathcal{A}$:

1. Receive tdcrs from Sim_1.
2. Query the simulator on $(b, c, 2)$ to receive $\Pi^{(1)} \leftarrow \mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{td}, b, c, 2)$. Parse $\Pi^{(1)}$ to obtain $\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)}, \mathsf{ct}_2^{(1)}, r_0^{(1)}, r_2^{(1)}, \pi^{(1)}_{\mathsf{sweBARG}}$.
3. Define the ciphertext chain $\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{ct}_2^*, \mathsf{ct}_3^*, \mathsf{ct}_4^*$ via:

$$\mathsf{ct}_0^* \coloneqq \mathsf{Enc}(\mathsf{pk}, a; r_0^*)(\mathcal{A}\text{'s fresh } r_0^*), \qquad \mathsf{ct}_1^* \coloneqq \mathsf{Enc}(\mathsf{pk}, a'; r_1^*)(\mathcal{A}\text{'s fresh } r_1^*),$$
$$\mathsf{ct}_2^* \coloneqq \mathsf{ct}_0^{(1)}(\text{simulated, encrypts } b), \qquad \mathsf{ct}_3^* \coloneqq \mathsf{ct}_1^{(1)}(\text{simulated intermediate ciphert.}),$$
$$\mathsf{ct}_4^* \coloneqq \mathsf{ct}_2^{(1)}(\text{simulated, encrypts } c).$$

   (We note that $\mathsf{ct}_1^{(1)}$ is read off from the $S^{(1,1)}$ entry of $\Pi^{(1)}$. The adversary does not need $r_1^{(1)}$.)
4. Produce two level-0 NIZK proofs for hops 1 and 2:
   - Hop 1: compute honest NIZK proof using the witness $(a, a', r_0^*, r_1^*, \pi_{\mathsf{SNARG},1})$ where $\pi_{\mathsf{SNARG},1}$ is an honestly produced SNARG proof asserting $C(a, w_{aa'}) = a'$. Call this $\pi_1^{*(0)}$. It verifies on $(\mathsf{ct}_0^*, \mathsf{ct}_1^*)$.

– Hop 2: compute honest NIZK proof using the witness $(a', b, r_1^*, r_0^{(1)}, \pi_{\mathsf{SNARG},2})$. The encryption randomness for $\mathsf{ct}_2^* = \mathsf{ct}_0^{(1)}$ is the simulator-revealed $r_0^{(1)}$, so $\mathcal{A}$ has it, and $\pi_{\mathsf{SNARG},1}$ is an honestly produced SNARG proof asserting $C(a', w_{a'b}) = b$. Call this $\pi_2^{*(0)}$. It verifies on $(\mathsf{ct}_1^*, \mathsf{ct}_2^*)$.

Note that, hops 3 and 4 are *implicit* since the adversary does not have explicit level-0 NIZK proofs for these. However, we show that the level-2 BARG proof exposed in Π^* that uses the level-0 NIZKs as witness suffices for the adversary.

5. Produce the level-1 BARG proofs (witnesses for the level-2 BARG):
   – *Left* level-1 BARG proof $\pi_L^{*(1)}$ on $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{ct}_2^*)$: run the honest BARG prover with witness $(\pi_1^{*(0)}, \pi_2^{*(0)})$.
   – *Right* level-1 BARG proof $\pi_R^{*(1)}$ on $(\mathsf{ct}_2^*, \mathsf{ct}_3^*, \mathsf{ct}_4^*) = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)}, \mathsf{ct}_2^{(1)})$: copy $\pi_{\mathsf{BARG}}^{(1)}$ from $S^{(1,1)}$ of $\Pi^{(1)}$. $\pi_R^{*(1)} \coloneqq \pi_{\mathsf{BARG}}^{(1)}$. Note that no knowledge of the internal $\pi_{\mathrm{sim},1}^{(0)}, \pi_{\mathrm{sim},2}^{(0)}$ is required.
6. Produce the level-2 BARG proof: run the honest BARG prover on the statement $(\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*)$ with witness $(\pi_L^{*(1)}, \pi_R^{*(1)})$. Call this $\pi^{*(2)}$.
7. Output the IVC proof for the statement $(a, c, 4)$:

$$\Pi^* = \left(\mathsf{ct}_0^*, \mathsf{ct}_4^*, r_0^*, r_2^{(1)}, \{S^{*(0)} = \bot, S^{*(1)} = \bot, S^{*(2)} = (\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*, \pi^{*(2)})\}\right).$$

We depict the structure of the forgery in Fig. 3. We verify that $\mathcal{A}$ wins the simulation-soundness game:

1. **Verification passes.** We check that Π^* is a valid IVC proof for $(a, c, 4)$. **Endpoint ciphertexts.** $\mathsf{ct}_0^* = \mathsf{Enc}(\mathsf{pk}, a; r_0^*)$ by construction; $\mathsf{ct}_4^* = \mathsf{ct}_2^{(1)} = \mathsf{Enc}(\mathsf{pk}, c; r_2^{(1)})$ since the simulator constructed $\mathsf{ct}_2^{(1)}$ this way and Π^* uses $r_2^{(1)}$ in the role of r_n^*. Both equations hold.
   BARG **check at level 2.** The IVC verifier reads $S^{*(2)} = (\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*, \pi^{*(2)})$ and runs $\mathsf{BARG.V}$ on $(\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*)$ with proof $\pi^{*(2)}$. $\pi^{*(2)}$ was constructed via the honest BARG prover on a witness pair $(\pi_L^{*(1)}, \pi_R^{*(1)})$, each of which itself is a verifying level-1 BARG proof on the appropriate sub-statement:
   – $\pi_L^{*(1)}$ on $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{ct}_2^*)$: produced by the honest level-1 BARG prover on witness $(\pi_1^{*(0)}, \pi_2^{*(0)})$. Both $\pi_1^{*(0)}, \pi_2^{*(0)}$ verify (they are honest NIZK proofs on true statements; the witnesses $w_{aa'}, w_{a'b}$ are valid by choice of C, the encryption equations hold by construction, and the SNARG proofs verify by SNARG completeness).
   – $\pi_R^{*(1)} = \pi_{\mathsf{BARG}}^{(1)}$ on $(\mathsf{ct}_2^*, \mathsf{ct}_3^*, \mathsf{ct}_4^*) = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)}, \mathsf{ct}_2^{(1)})$: this is exactly the verification check that the IVC verifier of $\Pi^{(1)}$ would perform. It passes because $\Pi^{(1)}$ was produced by Sim_2 and verifies by construction of the simulator.
   By BARG completeness, $\pi^{*(2)}$ verifies on $(\mathsf{ct}_0^*, \mathsf{ct}_2^*, \mathsf{ct}_4^*)$.
   Adjacency. $S^{*(2)}$ is the only non-empty $S^{*(\ell)}$ entry, so adjacency checks between entries are trivially satisfied.
   We conclude that Π^* is a verifying IVC proof for $(a, c, 4)$.
2. **False statement.** $(x_0, x_t, t) \notin \mathcal{L}_C$ by construction.
3. **No root query.** Finally, the no-root-query condition holds because the adversary's only simulator query is rooted at b, whereas the forged target is rooted at $a \neq b$. Thus the simulator was never queried on any statement whose root is the target root a.

D newrt-SIM-SND IVC Construction

In this section, we construct a root-binding IVC using a message-carrying NIZK-SoK at level 0 and somewhere-extractable BARGs at higher levels. The construction follows the same template as the IVC of [DJJ+25]. The root x_0 is used as the message of every level-0 NIZK proof and is carried explicitly as part of every higher-level recursive BARG instance. This structurally binds all recursive proofs to the root of the computation and yields newrt-SIM-SND.

D.1 Languages and relations for the construction

Languages and Relations

Every relation rejects malformed inputs and witnesses. All encodings use the public setup length bounds. As in Section 6, BARG prover and verifier calls omit the fixed relation argument: the relation associated with crs_1 is $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$, and that associated with crs_ℓ is $R_{\ell,\mathsf{crs}_{\ell-1}}$ for $\ell \geq 2$.

(a) One-Step Language $L_{C,1}$ and Relation $R_{C,1}$:

The base relation for a single computation step.

$$L_{C,1} = \{(x, x') \ : \ \exists\, w \text{ s.t. } C(x,w) = x'\},$$

$$R_{C,1}\big((x,x'),\ w\big) = 1 \iff C(x,w) = x'.$$

(b) Level-0 Language $L_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ and Relation $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$:

A level-0 proof is a NIZK proof certifying that a SNARG proof of one computation step is valid and that the ciphertexts are well-formed. The SNARG proves the one-step relation $R_{C,1}$; the NIZK additionally checks ciphertext consistency.

$$L_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}} = \left\{ (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) \ : \ \begin{array}{l} \exists\,(x_{\mathsf{init}},\ x_{\mathsf{fin}},\ r_{\mathsf{init}},\ r_{\mathsf{fin}},\ \pi_{\mathsf{SNARG}}) \ \text{ s.t.} \\ \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V},\ (x_{\mathsf{init}}, x_{\mathsf{fin}}),\ \pi_{\mathsf{SNARG}}) = 1 \\ \wedge\ \mathsf{ct}_{\mathsf{init}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{init}};\ r_{\mathsf{init}}) \\ \wedge\ \mathsf{ct}_{\mathsf{fin}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{fin}};\ r_{\mathsf{fin}}) \end{array} \right\},$$

$$R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}\big((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}),\ \ (x_{\mathsf{init}},\ x_{\mathsf{fin}},\ r_{\mathsf{init}},\ r_{\mathsf{fin}},\ \pi_{\mathsf{SNARG}})\big) = 1$$

$$\iff \begin{array}{l} \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V},\ (x_{\mathsf{init}}, x_{\mathsf{fin}}),\ \pi_{\mathsf{SNARG}}) = 1 \\ \wedge\ \mathsf{ct}_{\mathsf{init}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{init}};\ r_{\mathsf{init}}) \\ \wedge\ \mathsf{ct}_{\mathsf{fin}} = \mathsf{PKE.Enc}(\mathsf{pk},\ x_{\mathsf{fin}};\ r_{\mathsf{fin}}). \end{array}$$

This relation is proved by the NIZK.

(c) Level-1 Language $L_{1,\mathsf{crs}_{\mathsf{NIZK}}}$ and Relation $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$ (BARG, batch size $k = 2$):

A level-1 proof is a BARG proof verifying a batch of *two* level-0 NIZK proofs. The root x_0 is part of each BARG instance.

$$L_{1,\mathsf{crs}_{\mathsf{NIZK}}} = \left\{ (x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) \ : \ \exists\, \pi^{(0)} \text{ s.t. } \mathsf{NIZK.V}\big(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := x_0\big) = 1 \right\},$$

$$R_{1,\mathsf{crs}_{\mathsf{NIZK}}}\big((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}\big) = 1$$

$$\iff \mathsf{NIZK.V}\big(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := x_0\big) = 1.$$

(d) Level-ℓ Language $L_{\ell,\mathsf{crs}_{\ell-1}}$ and Relation $R_{\ell,\mathsf{crs}_{\ell-1}}$ (for $2 \leq \ell \leq \ell_{\max}$, BARG, batch size $k = 2$):

A level-ℓ proof is a BARG proof verifying a batch of two level-$(\ell-1)$ proofs, connected via a midpoint ciphertext $\mathsf{ct}_{\mathsf{mid}}$. The root x_0 is carried explicitly in both child instances.

$$L_{\ell,\mathsf{crs}_{\ell-1}} = \left\{ (x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}) \ : \ \begin{array}{l} \exists\,(\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell-1)}) \text{ s.t.} \\ \mathsf{BARG}.\mathcal{V}\big(\mathsf{crs}_{\ell-1}, ((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (x_0, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})), \pi^{(\ell-1)}\big) = 1 \end{array} \right\},$$

$$R_{\ell,\mathsf{crs}_{\ell-1}}\big((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), (\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell-1)})\big) = 1$$

$$\iff \mathsf{BARG}.\mathcal{V}\big(\mathsf{crs}_{\ell-1}, ((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (x_0, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})), \pi^{(\ell-1)}\big) = 1.$$

Summary of Language Hierarchy:

Level	Primitive	Batch	What it certifies
0	NIZK-SoK	–	SNARG + ciphertext consistency, under $m = x_0$
1	BARG	$k = 2$	Two level-0 NIZK proofs bound to root x_0
$\ell \geq 2$	BARG	$k = 2$	Two root-bound level-$(\ell-1)$ proofs

D.2 Root binding IVC construction

We now present the full construction.

Building blocks (all with subexponential security):

– $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$: a perfectly correct CPA-secure public-key encryption scheme using uniformly sampled κ-bit encryption randomness.

– $\mathsf{SNARG} = (\mathsf{SNARG.Gen}, \mathsf{SNARG.P}, \mathsf{SNARG.V})$: a non-adaptive succinct argument for **NP**.
– $\mathsf{NIZK\text{-}SoK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.TdSetup}, \mathsf{NIZK.P}, \mathsf{NIZK.V}, \mathsf{NIZK.Sim}, \mathsf{NIZK.E})$: a simulation-extractable NIZK signature of knowledge.
– $\mathsf{BARG} = (\mathsf{BARG.Setup}, \mathsf{BARG}.\mathcal{P}, \mathsf{BARG}.\mathcal{V})$: a rate-1 somewhere-extractable batch argument of knowledge for **NP** (Definition 3.5).

For each level $\ell \geq 1$, let n_ℓ be the encoded instance length and s_ℓ a public upper bound on the circuit size of the corresponding relation, fixed before setup. BARG batches and witness lists are ordered pairs. Malformed prover inputs return $\perp$, and malformed verifier inputs return 0.

Root binding IVC Construction — Setup

$\mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)$:

1. Set $\ell_{\max} = \lceil \log_2 T \rceil$ and use the security parameter κ in (1).
2. **Setup SNARG** for one-step computation:

$$(\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}) \leftarrow \mathsf{SNARG.Gen}(1^\kappa, 1^{2n}, 1^m, R_{C,1}).$$

3. **Setup PKE:** $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\kappa, 1^n)$.
4. **Setup NIZK-SoK** for relation $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$:

$$\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}).$$

5. **Setup level-1 BARG** if $\ell_{\max} \geq 1$ (batch size $k = 2$, relation $R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$):

$$\mathsf{crs}_1 \leftarrow \mathsf{BARG.Setup}(1^\kappa, 1^2, 1^{s_1}, 1^{n_1}).$$

6. **Setup level-ℓ BARG** for each $\ell \in \{2, \ldots, \ell_{\max}\}$ (batch size $k = 2$, relation $R_{\ell,\mathsf{crs}_{\ell-1}}$):

$$\mathsf{crs}_\ell \leftarrow \mathsf{BARG.Setup}(1^\kappa, 1^2, 1^{s_\ell}, 1^{n_\ell}).$$

7. **Output:** $\mathsf{crs} := (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell \in [\ell_{\max}]})$.

Fig. 11: Root Binding IVC Construction: setup algorithm $\mathsf{IVC.Gen}$.

D.3 Root binding IVC satisfies newrt-SIM-SND

Theorem D.1 (Root-binding IVC). *Fix $\lambda \geq 1$, $1 \leq T \leq 2^\lambda$, and polynomially bounded $n, m, |C|$, with $n \geq 1$, where $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. Circuit size includes its input and witness wires. Assume the following ingredients:*

1. *a CPA-secure PKE with perfect correctness and uniformly sampled κ-bit encryption randomness;*
2. *a perfectly complete, non-adaptively sound SNARG for $R_{C,1}$;*
3. *a perfectly complete, simulation-extractable NIZK-SoK for $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$, with adaptive multi-theorem zero knowledge; and*
4. *rate-1 BARGs for the recursive relations, with perfect completeness, CRS indistinguishability, and somewhere extraction.*

For each required security property i, assume $(\mathsf{poly}(\kappa), 2^{-\kappa^{\delta_i}})$ security against non-uniform adversaries, for a constant $0 < \delta_i < 1$. Assume a fixed polynomial P bounds, at every level $\ell \leq \ell_{\max}$, the recursive relation and ordinary/trapdoor CRS sizes and the setup, proving, verification, and extraction costs by $P(\kappa, n, m, \ell_{\max}, |C|)$, with fixed public size bounds. Choose $0 < \delta' < \delta < \min_i \delta_i$ and use the common parameter κ in (1). Then the IVC scheme in Figs. 11 to 13 is perfectly complete, zero-knowledge, and satisfies newrt-SIM-SND. *Its proof size is $\mathsf{poly}(\lambda, n, \log T, \log |C|)$, and its CRS size and setup, update, and verification times are $\mathsf{poly}(\lambda, \log T, |C|)$.*

Root Binding IVC Construction — Prover

$\mathsf{IVC.P}(\mathsf{crs}, (x_0, x_{i-1}, i{-}1), \Pi_{i-1}, w_i)$:

1. Parse $\mathsf{crs} = (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell\in[\ell_{\max}]})$. If $i{-}1 \notin \{0,\dots,T{-}1\}$, $x_0 \notin \{0,1\}^n$, $x_{i-1} \notin \{0,1\}^n$, or $w_i \notin \{0,1\}^m$, output $\perp$.
2. **Base case:** If $i = 1$:
   (a) If $x_{i-1} \neq x_0$, output $\perp$. Sample $r_0 \leftarrow \{0,1\}^\kappa$ and compute $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0)$.
   (b) Initialize $S^{(\ell)} = \emptyset$ for all $\ell \in \{0,\dots,\ell_{\max}\}$.
3. **Verify previous proof:** If $i > 1$:
   (a) Parse $\Pi_{i-1} = (\mathsf{ct}_0, \mathsf{ct}_{i-1}, r_0, r_{i-1}, \{S^{(\ell)}\}_{\ell\in\{0,\dots,\ell_{\max}\}})$.
   (b) If $\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_{i-1}, i{-}1), \Pi_{i-1}) = 0$, output $\perp$.
4. **Compute next configuration and SNARG proof:**
   (a) $x_i = C(x_{i-1}, w_i)$.
   (b) $\pi_{\mathsf{SNARG}} \leftarrow \mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, (x_{i-1}, x_i), w_i)$.
5. **Encrypt and compute level-0 NIZK proof:**
   (a) Sample $r_i \leftarrow \{0,1\}^\kappa$ and compute $\mathsf{ct}_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_i; r_i)$.
   (b) $\pi_i^{(0)} \leftarrow \mathsf{NIZK.P}\big(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{i-1}, \mathsf{ct}_i), (x_{i-1}, x_i, r_{i-1}, r_i, \pi_{\mathsf{SNARG}}), m := x_0\big)$.
   (c) Add $(i{-}1, \mathsf{ct}_{i-1}, \perp, \mathsf{ct}_i, \pi_i^{(0)})$ to $S^{(0)}$.
6. **Merge proofs:** For $\ell \in \{0,\dots,\ell_{\max}-1\}$:
   (a) *Check if two proofs at level ℓ exist:*
   – If $|S^{(\ell)}| < 2$, continue to the next ℓ.
   – If $|S^{(\ell)}| = 2$, parse
   $S^{(\ell)} = \{(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, \pi^{(\ell)}), (\tilde{i}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{mid}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \widetilde{\pi}^{(\ell)})\}$,
   ordered so that $i_{\mathsf{init}} \le \tilde{i}_{\mathsf{init}}$, and set $S^{(\ell)} = \emptyset$.
   (b) *Verify adjacency:* If $\mathsf{ct}_{\mathsf{fin}} \neq \widetilde{\mathsf{ct}}_{\mathsf{init}}$ or $i_{\mathsf{init}} + 2^\ell \neq \tilde{i}_{\mathsf{init}}$, output $\perp$.
   (c) *Create level-$(\ell{+}1)$ proof:*
   – If $\ell = 0$: The two entries have $\mathsf{ct}_{\mathsf{mid}} = \perp$ and $\widetilde{\mathsf{ct}}_{\mathsf{mid}} = \perp$. Compute

$$\pi^{(1)} \leftarrow \mathsf{BARG}.\mathcal{P}\big(\mathsf{crs}_1,\ ((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), (x_0, \widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}))\big),\ (\pi^{(0)}, \widetilde{\pi}^{(0)})\big),$$

   and add

$$(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \pi^{(1)})$$

   to $S^{(1)}$.
   – If $\ell \ge 1$: Compute

$$\pi^{(\ell+1)} \leftarrow \mathsf{BARG}.\mathcal{P}\big(\mathsf{crs}_{\ell+1}, ((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), (x_0, \widetilde{\mathsf{ct}}_{\mathsf{init}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}})), ((\mathsf{ct}_{\mathsf{mid}}, \pi^{(\ell)}), (\widetilde{\mathsf{ct}}_{\mathsf{mid}}, \widetilde{\pi}^{(\ell)}))\big),$$

   and add

$$(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}, \widetilde{\mathsf{ct}}_{\mathsf{fin}}, \pi^{(\ell+1)})$$

   to $S^{(\ell+1)}$.
7. **Output:** $\Pi_i = (\mathsf{ct}_0, \mathsf{ct}_i, r_0, r_i, \{S^{(\ell)}\}_{\ell\in\{0,\dots,\ell_{\max}\}})$.

Fig. 12: Root Binding IVC Construction: prover algorithm IVC.P.

Completeness. Fix an honest computation $x_j = C(x_{j-1}, w_j)$ for $j \in [t]$. Perfect SNARG completeness makes each generated one-step proof accepting. Together with the sampled endpoint openings, this gives a valid witness for every level-0 NIZK-SoK proof under message x_0. Perfect NIZK completeness therefore makes all leaf proofs accepting. Whenever two nodes merge, their accepting proofs and midpoint ciphertexts form valid witnesses for the two recursive instances, both with root x_0. Perfect BARG completeness makes the parent proof accepting. The binary-counter merge schedule preserves adjacency and leaves precisely the canonical dyadic frontier of $[0,t]$, as in Section 6.1. Its outer ciphertexts and the returned randomness open to x_0, x_t, so every IVC verification check succeeds. This proves perfect completeness for every $1 \le t \le T$.

Efficiency. Write $c := |\mathsf{ct}| = \mathsf{poly}(\kappa, n)$ and let a_ℓ bound the level-ℓ proof length. SNARG succinctness and the polynomial cost of the level-0 relation give $a_0 = \mathsf{poly}(\kappa, n, \log|C|)$; the NIZK-SoK message has length n. By rate 1, for a fixed polynomial p, $a_1 \le a_0 + p(\kappa, 1, \log s_1)$ and $a_\ell \le a_{\ell-1} + c + p(\kappa, 1, \log s_\ell)$ for $\ell \ge 2$. Summing these bounds and using the public relation-size bounds of Theorem D.1 gives $a_\ell = \mathsf{poly}(\lambda, n, \ell_{\max}, \log|C|)$, as in Section 6.2. The IVC proof contains two ciphertexts, two κ-bit openings, and at most one node at each level, each with three ciphertext slots, a position, and a recursive proof. Its total length is therefore

Root Binding IVC Construction — Verifier

$\mathsf{IVC.V}(\mathsf{crs}, (x_0, x_t, t), \Pi_t)$:

1. Parse $\mathsf{crs} = (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \mathsf{crs}_{\mathsf{NIZK}}, \{\mathsf{crs}_\ell\}_{\ell\in[\ell_{\max}]})$ and $\Pi_t = (\mathsf{ct}_0, \mathsf{ct}_t, r_0, r_t, \{S^{(\ell)}\}_{\ell\in\{0,\ldots,\ell_{\max}\}})$. If $t \notin \{1,\ldots,T\}$, $x_0 \notin \{0,1\}^n$, or $x_t \notin \{0,1\}^n$, output 0.
2. Set $i_{\mathsf{curr}} = 0$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_0$.
3. **Verify proofs at each level:** For $\ell \in \{\ell_{\max}, \ell_{\max}-1, \ldots, 0\}$:
   (a) Let s_ℓ be the ℓ-th least significant bit of t.
   - If $s_\ell = 0$: if $S^{(\ell)} \neq \emptyset$, output 0; otherwise continue to the next ℓ.
   - If $s_\ell = 1$: parse $S^{(\ell)} = \{(i_{\mathsf{init}}, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, \pi^{(\ell)})\}$.

   (b) *Verify adjacency:* If $i_{\mathsf{curr}} \neq i_{\mathsf{init}}$ or $\mathsf{ct}_{\mathsf{curr}} \neq \mathsf{ct}_{\mathsf{init}}$, output 0. Update $i_{\mathsf{curr}} = i_{\mathsf{curr}} + 2^\ell$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_{\mathsf{fin}}$.
   (c) *Verify proof:*
   - If $\ell = 0$: check $\mathsf{ct}_{\mathsf{mid}} = \perp$ and $\mathsf{NIZK.V}(\mathsf{crs}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := x_0) = 1$.
   - If $\ell = 1$: check
   $$\mathsf{BARG}.\mathcal{V}(\mathsf{crs}_1, ((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (x_0, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})), \pi^{(1)}) = 1.$$
   - If $\ell \geq 2$: check
   $$\mathsf{BARG}.\mathcal{V}(\mathsf{crs}_\ell, ((x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (x_0, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})), \pi^{(\ell)}) = 1.$$

   If any check fails, output 0.
4. **Verify ciphertexts:** If $\mathsf{ct}_{\mathsf{curr}} \neq \mathsf{ct}_t$, or $\mathsf{PKE.Enc}(\mathsf{pk}, x_0; r_0) \neq \mathsf{ct}_0$, or $\mathsf{PKE.Enc}(\mathsf{pk}, x_t; r_t) \neq \mathsf{ct}_t$, output 0.
5. **Output** 1.

Fig. 13: Root Binding IVC Construction: verifier algorithm $\mathsf{IVC.V}$.

$\mathsf{poly}(\lambda, n, \log T, \log|C|)$. Setup, an update, and verification each use $O(\ell_{\max}+1)$ primitive operations. The assumed size and running-time bounds, together with $\kappa = \mathsf{poly}(\lambda, n)$ and $n + m \leq |C|$, give the stated CRS-size and running-time bounds. For brevity, we write $\mu := 2^{-\kappa^\delta}$ in the following.

Lemma D.2 (Zero knowledge) *Under the assumptions of Theorem D.1, the IVC construction in Figs. 11 to 13 satisfies computational zero knowledge (as in Definition 4.1).*

The IVC simulator. We use the setup and simulation algorithms of the NIZK-SoK that satisfy both adaptive multi-theorem zero knowledge and simulation extractability. Define $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ as follows.

Sim_1. On input $(1^\lambda, T, 1^n, 1^m, C)$:

1. Set $\ell_{\max} := \lceil \log_2 T \rceil$ and use κ from Theorem D.1. Generate $(\mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}) \leftarrow \mathsf{SNARG.Gen}(1^\kappa, 1^{2n}, 1^m, R_{C,1})$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\kappa, 1^n)$.
2. Sample $(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TdSetup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}})$.
3. For $k = 1, \ldots, \ell_{\max}$, sample $\mathsf{crs}_k \leftarrow \mathsf{BARG.Setup}(1^\kappa, 2, 1^{s_k}, 1^{n_k})$ using the public size bounds of the construction. The level-1 relation uses $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$, and each subsequent relation uses the preceding BARG CRS.
4. Return $(\widetilde{\mathsf{crs}}, \mathsf{st})$, where $\mathsf{st} := \mathsf{td}_{\mathsf{NIZK}}$ and

$$\widetilde{\mathsf{crs}} := (T, \ell_{\max}, C, \mathsf{pk}, \mathsf{crs}_{\mathsf{SNARG},P}, \mathsf{crs}_{\mathsf{SNARG},V}, \widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \{\mathsf{crs}_k\}_{k=1}^{\ell_{\max}}). \tag{7}$$

The extraction trapdoor τ_{NIZK} is not part of the simulator's output.

Sim_2. On input $(\widetilde{\mathsf{crs}}, \mathsf{st}, u, v, 1^d)$, parse the public parameters as in (7) and set $\mathsf{td}_{\mathsf{NIZK}} := \mathsf{st}$. Return $\perp$ unless the query is well formed, $1 \leq d \leq T$, and $u, v \in \{0,1\}^n$. Otherwise:

1. Independently sample r_0, r_d and ρ_j for $0 < j < d$ from $\{0,1\}^\kappa$. Set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, u; r_0)$, $\mathsf{ct}_d := \mathsf{PKE.Enc}(\mathsf{pk}, v; r_d)$, and $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, 0^n; \rho_j)$ for $0 < j < d$. Initialize $S^{(\ell)} := \emptyset$ for $0 \leq \ell \leq \ell_{\max}$.
2. For $j = 0, \ldots, d-1$, generate $\pi_j^{(0)} \leftarrow \mathsf{NIZK.Sim}(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}), m := u)$, insert $(j, \mathsf{ct}_j, \perp, \mathsf{ct}_{j+1}, \pi_j^{(0)})$ into $S^{(0)}$, and execute the merge procedure of $\mathsf{IVC.P}$ with root $x_0 := u$. Each BARG proof is generated by the prescribed BARG prover on the ordered child instances and witnesses.
3. Return $\widetilde{\Pi}_d := (\mathsf{ct}_0, \mathsf{ct}_d, r_0, r_d, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

Every call uses fresh independent randomness, including repeated statements. Accepting proofs of NIZK simulation yieldss valid witnesses for the first BARG merge. Inductively, perfect BARG completeness makes all higher merges accepting under the ordinary BARG CRSs, and the returned endpoint openings and frontier satisfy the IVC verifier. The running time is $d \cdot \mathsf{poly}(\kappa, n, m, \ell_{\max}, |C|)$, polynomial in the simulator's input length.

Hybrids. Fix a non-uniform PPT adversary $\mathcal{A}$ in the zero-knowledge experiment of Definition 4.1. For a valid query $((u, v, 1^d), (w_1, \ldots, w_d))$, write $z_0 := u$ and $z_j := C(z_{j-1}, w_j)$ for $j \in [d]$, so $z_d = v$. Each query uses fresh independent randomness. Write $H_b^{\mathcal{A}}$ for $\mathcal{A}$'s output bit in hybrid H_b.

Hybrid H_0. Sample $\mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)$ and run $\mathcal{A}^{\mathcal{O}_{\mathsf{real}}(\mathsf{crs},\cdot)}(\mathsf{crs})$. This is the real zero-knowledge experiment.

Hybrid H_1. Generate $(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}})$ with Sim_1. For each valid query, initialize $S^{(\ell)} := \emptyset$ for all $0 \le \ell \le \ell_{\max}$, sample fresh $r_0 \leftarrow \{0,1\}^\kappa$, and set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, u; r_0)$. For $j = 0, \ldots, d-1$, execute:

1. Generate $\pi_{\mathsf{SNARG},j} \leftarrow \mathsf{SNARG.P}(\mathsf{crs}_{\mathsf{SNARG},P}, (z_j, z_{j+1}), w_{j+1})$.
2. Sample fresh $r_{j+1} \leftarrow \{0,1\}^\kappa$ and set $\mathsf{ct}_{j+1} := \mathsf{PKE.Enc}(\mathsf{pk}, z_{j+1}; r_{j+1})$.
3. Invoke the witness-validating wrapper from the NIZK-SoK zero-knowledge definition: $\pi_j^{(0)} \leftarrow \mathsf{Sim}'(\mathsf{td}_{\mathsf{NIZK}}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}), (z_j, z_{j+1}, r_j, r_{j+1}, \pi_{\mathsf{SNARG},j}), m := u)$, with its associated CRS $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$.
4. Insert $(j, \mathsf{ct}_j, \bot, \mathsf{ct}_{j+1}, \pi_j^{(0)})$ into $S^{(0)}$ and execute the merge procedure of $\mathsf{IVC.P}$ with root $x_0 := u$, using the BARG prover on the prescribed ordered child instances and witnesses.

Return $(\mathsf{ct}_0, \mathsf{ct}_d, r_0, r_d, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

Lemma D.3 *For every non-uniform PPT adversary $\mathcal{A}$ and all sufficiently large λ, $|\Pr[H_0^{\mathcal{A}} = 1] - \Pr[H_1^{\mathcal{A}} = 1]| \le \mu$.*

Proof. Construct an adversary $\mathcal{B}$ against adaptive multi-theorem zero knowledge of the NIZK-SoK. Before receiving its challenge CRS, $\mathcal{B}$ generates the SNARG parameters and PKE keys, thereby fixing $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. Given the challenge NIZK CRS, it generates the ordinary BARG CRSs in increasing order of level and runs $\mathcal{A}$ on the assembled public parameters.
For each valid query, $\mathcal{B}$ performs the operations of H_1, using its challenge proof oracle in Step 3 on the stated instance, witness, and message. Every submitted witness satisfies $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ by perfect SNARG completeness and the definitions of the ciphertexts. It returns the resulting IVC proof and eventually outputs $\mathcal{A}$'s bit.
With the real NIZK oracle, the execution has distribution H_0: the previous-proof checks performed by the incremental prover always succeed by completeness. With the simulated oracle, it has distribution H_1. The adaptive multi-theorem guarantee permits the proof queries to depend on previous answers and bounds the distinguishing gap by μ.

Hybrid H_2. Generate $(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}})$ with Sim_1. For each valid query, independently sample $r_0, \ldots, r_d \leftarrow \{0,1\}^\kappa$ and set $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, z_j; r_j)$ for $0 \le j \le d$. Initialize $S^{(\ell)} := \emptyset$ for all levels. For $j = 0, \ldots, d-1$, execute:

1. Generate $\pi_j^{(0)} \leftarrow \mathsf{NIZK.Sim}(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}, (\mathsf{ct}_j, \mathsf{ct}_{j+1}), m := u)$.
2. Perform the insertion and merges of Step 4 in H_1.

Return $(\mathsf{ct}_0, \mathsf{ct}_d, r_0, r_d, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

Lemma D.4 *For every non-uniform PPT adversary $\mathcal{A}$, its views in H_1 and H_2 have identical distributions.*

Proof. Every witness supplied to Sim' in H_1 satisfies $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. Hence the wrapper invokes $\mathsf{NIZK.Sim}$ on the same statement and message as in H_2. The SNARG proofs are not used elsewhere, so omitting their computation preserves the output distribution. Within each query, the supplied witnesses fix the states $z_0, \ldots, z_d$, and all encryption randomness is independent of the proof-generation randomness. Sampling the ciphertexts before the leaf loop therefore also preserves the distribution.

Hybrid H_3. Generate $(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}})$ with Sim_1. For each valid query, independently sample $r_0, \ldots, r_d \leftarrow \{0,1\}^\kappa$ and set $\mathsf{ct}_0 := \mathsf{PKE.Enc}(\mathsf{pk}, u; r_0)$, $\mathsf{ct}_d := \mathsf{PKE.Enc}(\mathsf{pk}, v; r_d)$, and $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, 0^n; r_j)$ for $0 < j < d$. Initialize $S^{(\ell)} := \emptyset$ for all levels, execute the leaf loop of H_2 using these ciphertexts, and return $(\mathsf{ct}_0, \mathsf{ct}_d, r_0, r_d, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$. The setup and oracle answers are exactly those of Sim_1 and $\mathcal{O}_{\mathsf{sim}}$ in Definition 4.1.

Lemma D.5 *For every non-uniform PPT adversary $\mathcal{A}$, let $M = M(\lambda)$ be a polynomial upper bound on the total number of interior encryption operations across its valid queries. For all sufficiently large λ, $|\Pr[H_2^{\mathcal{A}} = 1] - \Pr[H_3^{\mathcal{A}} = 1]| \le M\mu$.*

Proof. Number the interior encryption operations by query occurrence and then by position within the query. For $a = 0, \ldots, M$, define K_a by the algorithm for H_2, setting its first a interior ciphertexts to independent encryptions of 0^n and all subsequent interior ciphertexts to encryptions of their corresponding z_j. Endpoints and the leaf-and-merge loop are computed as in H_2 from the resulting ciphertexts. Thus $K_0 = H_2$ and $K_M = H_3$.
Fix $a \in [M]$. A CPA adversary $\mathcal{B}_a$ receives the challenge public key and generates the remaining setup as in Sim_1, retaining the NIZK simulation trapdoor. It runs $\mathcal{A}$ and answers queries according to K_{a-1}. At the a-th interior encryption, at position j of its query, it submits $(z_j, 0^n)$ as its challenge messages and uses the returned ciphertext as ct_j. It completes the current query, answers subsequent queries, and outputs $\mathcal{A}$'s bit. If this encryption operation is never reached, it makes an unused challenge query with identical messages.
The reduction simulates the NIZK-SoK leaf proofs and constructs all BARG proofs using the child proofs and boundary ciphertexts. These operations do not require the opening of ct_j, and only endpoint openings are returned. The challenge branches therefore give K_{a-1} and K_a, respectively. CPA security bounds each adjacent distinguishing gap by μ, and summing over $a \in [M]$ proves the claim.

Proof (Proof of Lemma D.2). Hybrid H_0 is the real experiment, and H_3 is the simulated experiment for $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$. Lemmas D.3–D.5 bound their distinguishing gap by $(M+1)\mu$. Since $\mathcal{A}$ is PPT and supplies each query depth in unary, M is polynomial in λ. The parameter choice of Theorem D.1 makes this bound negligible in λ, proving zero knowledge.

Proof of newrt-SIM-SND Throughout this proof, we refer to the root x_0 of a computation as its *source*, to avoid confusion with the roots of the proof tree. As in the soundness proof of DJJ$^+$ [DJJ+25], an accepting proof of a false target has a bad frontier block, and somewhere extraction lets us descend to a bad accepting leaf. Here every recursive instance carries the same source x_0. If that source was never queried to the IVC simulator, the resulting leaf's message x_0 was never submitted to the NIZK simulator. Simulation extraction and non-adaptive SNARG soundness then rule out the leaf.

Setup experiments and simulation queries. Fix $\sigma = (\sigma_1, \ldots, \sigma_{\ell_{\max}}) \in \{\perp, 1, 2\}^{\ell_{\max}}$ before setup. Let $\mathcal{H}_\sigma$ be the following experiment for the source-binding construction. Generate the PKE, SNARG, and joint NIZK parameters as in Sim_1, retaining sk, τ_{NIZK}, and $\mathsf{td}_{\mathsf{NIZK}}$ privately. Generate the BARG CRSs in increasing order of level. If $\sigma_k = \perp$, sample $\mathsf{crs}_k \leftarrow \mathsf{BARG.Setup}(1^\kappa, 1^2, 1^{s_k}, 1^{n_k})$ and set $\tau_k := \perp$. If $\sigma_k = b \in \{1, 2\}$, sample $(\mathsf{crs}_k, \tau_k) \leftarrow \mathsf{BARG.TdSetup}(1^\kappa, 1^2, 1^{s_k}, 1^{n_k}, b)$. Coordinate 1 is the left child and coordinate 2 the right child. At each level the recursive relation uses the public parameters already generated. Write $R_1 := R_{1,\mathsf{crs}_{\mathsf{NIZK}}}$ and $R_k := R_{k,\mathsf{crs}_{k-1}}$ for $k \ge 2$, with these parameters understood.
The experiment answers queries with $\mathsf{Sim}_2(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}}, \cdot)$. Let Q_{src} be the set of sources of its valid simulation queries, recorded before each query is answered. Every internal NIZK simulation call for a query with source u has message u. The simulator performs its prescribed leaf calls and merges in every mode, retaining every BARG prover output in its proof slot; these outputs need not accept in the auxiliary modes. For $\sigma = (\perp, \ldots, \perp)$, the public parameters and oracle answers have exactly the distribution of the simulation experiment. Write $\boldsymbol{\tau}_{>h} := (\tau_{h+1}, \ldots, \tau_{\ell_{\max}})$. The procedures below receive only the trapdoors specified in their statements; the NIZK simulation state is accessed through the oracle.

Nodes. For the analysis, a level-ℓ node is $N = (x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}}, \pi^{(\ell)})$. At level 0 its midpoint is $\perp$, and it accepts iff $\mathsf{NIZK.V}(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := x_0) = 1$. For $\ell \ge 1$, set $X_1(N) := (x_0, \mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}})$ and $X_2(N) := (x_0, \mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})$. The node accepts iff $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}_\ell, R_\ell, (X_1(N), X_2(N)), \pi^{(\ell)}) = 1$. These checks include the construction's encoding and domain checks; malformed nodes do not accept. Call N bad if an

endpoint decrypts outside $\{0,1\}^n$, or its decrypted endpoints a, z satisfy $a \overset{2^\ell}{\not\leftrightarrow} z$. Call its source fresh if $x_0 \notin Q_{\mathsf{src}}$.

Lemma D.6 (Fresh-source leaf soundness) *Fix σ before setup. Let $\mathcal{G}$ be any non-uniform PPT algorithm with input $(\widetilde{\mathsf{crs}}, \mathsf{sk}, \boldsymbol{\tau}_{>0})$, oracle access to $\mathsf{Sim}_2(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}}, \cdot)$, and output a level-0 node N_0 or $\perp$. On a node output, let $w \leftarrow \mathsf{NIZK.E}(\tau_{\mathsf{NIZK}}, (\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, m := x_0)$. Then the probability that N_0 accepts, $x_0 \notin Q_{\mathsf{src}}$, and w is not a valid witness for $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ is at most μ. Moreover,*

$$\Pr_{\mathcal{H}_\sigma}[N_0 \text{ accepts } \wedge \ N_0 \text{ is bad } \wedge \ x_0 \notin Q_{\mathsf{src}}] \le (1 + 2^{2n})\mu. \tag{8}$$

Proof. Let p denote the probability on the left-hand side of (8). We first show that NIZK extraction fails on an accepting fresh-source output of $\mathcal{G}$ with probability at most μ. Therefore, with probability at least $p - \mu$, $\mathcal{G}$ produces an accepting bad fresh-source leaf from which extraction returns a valid witness. We then use this witness to construct a non-adaptive SNARG adversary.
To bound extraction failure, define a NIZK simulation-extractability adversary $\mathcal{B}_{\mathsf{SE}} = (\mathcal{B}_{\mathsf{SE},1}, \mathcal{B}_{\mathsf{SE},2})$ as follows.

1. $\mathcal{B}_{\mathsf{SE},1}$ generates the PKE keys and SNARG parameters as in $\mathcal{H}_\sigma$. It outputs $(R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}, \mathsf{st}_{\mathsf{SE}})$, retaining the generated parameters and sk in $\mathsf{st}_{\mathsf{SE}}$.
2. Given the challenge NIZK CRS and $\mathsf{st}_{\mathsf{SE}}$, $\mathcal{B}_{\mathsf{SE},2}$ generates all BARG levels in modes σ and runs $\mathcal{G}$ with its prescribed inputs. It answers each IVC simulation query using Sim_2, obtaining every level-0 proof from its challenge simulation oracle on the prescribed ciphertext pair and source message. If $\mathcal{G}$ outputs an accepting leaf with $x_0 \notin Q_{\mathsf{src}}$, it outputs $((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), \pi^{(0)}, x_0)$; otherwise it outputs $\perp$.

The setup and oracle answers have exactly their distribution in $\mathcal{H}_\sigma$. Every message submitted to the challenge simulation oracle belongs to Q_{src}, so $x_0 \notin Q_{\mathsf{src}}$ ensures that the output tuple is absent from the query-response list in the NIZK simulation-extractability experiment. Its extractor uses the jointly generated τ_{NIZK}, exactly as in the lemma. Thus $\mathcal{B}_{\mathsf{SE}}$ succeeds with probability

$$\Pr_{\mathcal{H}_\sigma}\left[N_0 \text{ accepts } \wedge \ x_0 \notin Q_{\mathsf{src}} \ \wedge \ R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}((\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{fin}}), w) = 0\right].$$

Simulation extractability bounds this probability by μ, proving the first assertion.
It follows that, with probability at least $p - \mu$, $\mathcal{G}$ outputs an accepting bad fresh-source leaf and w is a valid witness. On this event, parse $w = (a, z, r_{\mathsf{init}}, r_{\mathsf{fin}}, \pi_{\mathsf{SNARG}})$. The relation $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ gives $\mathsf{ct}_{\mathsf{init}} = \mathsf{PKE.Enc}(\mathsf{pk}, a; r_{\mathsf{init}})$, $\mathsf{ct}_{\mathsf{fin}} = \mathsf{PKE.Enc}(\mathsf{pk}, z; r_{\mathsf{fin}})$, and $\mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, (a, z), \pi_{\mathsf{SNARG}}) = 1$. Perfect PKE correctness identifies $a, z \in \{0,1\}^n$ with the endpoint decryptions. Since the leaf is bad, we therefore have $(a, z) \notin L_{C,1}$.
We now construct a non-adaptive SNARG adversary $\mathcal{B}_{\mathsf{S}} = (\mathcal{B}_{\mathsf{S},1}, \mathcal{B}_{\mathsf{S},2})$ that obtains such a false accepting proof.

1. Before SNARG setup, $\mathcal{B}_{\mathsf{S},1}$ samples $(\widehat{a}, \widehat{z}) \leftarrow \{0,1\}^{2n}$ and outputs $(1^{2n}, 1^m, R_{C,1}, (\widehat{a}, \widehat{z}), \mathsf{st}_{\mathsf{S}})$, where $\mathsf{st}_{\mathsf{S}} := (\widehat{a}, \widehat{z})$.
2. Given the challenge SNARG parameters and this state, $\mathcal{B}_{\mathsf{S},2}$ generates the remaining setup as in $\mathcal{H}_\sigma$, retaining $\mathsf{sk}, \tau_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}$ and the BARG extraction trapdoors. It runs $\mathcal{G}$ with its prescribed inputs and answers its queries using Sim_2. If $\mathcal{G}$ does not output an accepting fresh-source leaf, it returns $\perp$. Otherwise, it computes w using $\mathsf{NIZK.E}$ as in the lemma. It returns $\perp$ unless w is a valid witness for $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. Finally, it parses $w = (a, z, r_{\mathsf{init}}, r_{\mathsf{fin}}, \pi_{\mathsf{SNARG}})$ and outputs π_{SNARG} if $(a, z) = (\widehat{a}, \widehat{z})$; otherwise it outputs $\perp$.

The setup, execution of $\mathcal{G}$, and extraction of w are independent of $(\widehat{a}, \widehat{z})$: the guessed pair is used only in the final equality check. Thus, whenever $\mathcal{G}$ produces an accepting bad fresh-source leaf with a valid extracted witness, the guess matches (a, z) with probability 2^{-2n}. On a match, $\mathcal{B}_{\mathsf{S}}$ outputs an accepting SNARG proof for its false instance, which was fixed before SNARG setup. Writing $\pi^\star$ for its output, we obtain

$$\Pr[(\widehat{a}, \widehat{z}) \notin L_{C,1} \ \wedge \ \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, (\widehat{a}, \widehat{z}), \pi^\star) = 1] \ge 2^{-2n}(p - \mu).$$

Non-adaptive SNARG soundness bounds the left-hand side by μ. Hence $p \le \mu + 2^{2n}\mu$, proving (8).

Lemma D.7 (Fresh-source soundness at every level) *For every $0 \le \ell \le \ell_{\max}$, every setup vector σ satisfying $\sigma_k = \perp$ for $1 \le k \le \ell$, and every non-uniform PPT algorithm $\mathcal{G}$ with input $(\widetilde{\mathsf{crs}}, \mathsf{sk}, \boldsymbol{\tau}_{>\ell})$, oracle access to $\mathsf{Sim}_2(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}}, \cdot)$, and output a level-$\ell$ node N_ℓ or $\perp$, the following holds for all sufficiently large λ:*

$$\Pr_{\mathcal{H}_\sigma}[N_\ell \text{ accepts} \wedge N_\ell \text{ is bad} \wedge x_0 \notin Q_{\mathsf{src}}] \le 2^\ell(3 \cdot 2^{2n} + 5)\mu. \tag{9}$$

Proof. We prove a stronger bound by induction on ℓ, simultaneously for all σ and $\mathcal{G}$ in the statement. Writing p for the probability in (9), we show

$$p \le \left[2^\ell(1 + 2^{2n}) + 2(2^\ell - 1)(2^{2n} + 2)\right]\mu. \tag{10}$$

For $\ell = 0$, this is Lemma D.6. Suppose $\ell \ge 1$ and the bound holds at level $\ell - 1$.

Selecting a bad half. For $b \in \{1, 2\}$, call half b bad if an endpoint in $X_b(N_\ell)$ decrypts outside $\{0,1\}^n$, or its two decrypted endpoints fail reachability in $2^{\ell-1}$ steps. A bad node has a bad half: if both halves were good, their computations would concatenate through the common midpoint; an invalid outer endpoint belongs to an incident half. Let E_b be the event that N_ℓ accepts, has fresh source, and half b is bad. Thus $p \le \Pr_{\mathcal{H}_\sigma}[E_1] + \Pr_{\mathcal{H}_\sigma}[E_2]$. Choose $b \in \{1, 2\}$ with $\Pr_{\mathcal{H}_\sigma}[E_b] \ge p/2$. Let σ' agree with σ except that $\sigma'_\ell = b$.

Switching the level-ℓ CRS. We compare E_b in $\mathcal{H}_\sigma$ and $\mathcal{H}_{\sigma'}$ using the fixed-pair argument of [DJJ+25]. Consider the following Boolean tests on the output of $\mathcal{G}$, each requiring that N_ℓ accepts and $x_0 \notin Q_{\mathsf{src}}$. The test $T_{b,\perp}$ checks whether either endpoint of half b decrypts outside $\{0,1\}^n$. For each $(a, z) \in \{0,1\}^{2n}$, the test $T_{b,a,z}$ checks whether these decryptions equal (a, z), and each test is polynomial time using sk.
For each test, define a CRS-indistinguishability adversary $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ as follows.

1. $\mathcal{D}_1$ generates the PKE, SNARG, and joint NIZK parameters, followed by ordinary BARG levels $1, \ldots, \ell - 1$, as in $\mathcal{H}_\sigma$. It outputs $(b, \mathsf{st}_{\mathcal{D}})$, retaining the generated parameters and their setup secrets in $\mathsf{st}_{\mathcal{D}}$.
2. Given the challenge CRS and $\mathsf{st}_{\mathcal{D}}$, $\mathcal{D}_2$ uses the challenge as crs_ℓ and generates levels $\ell + 1, \ldots, \ell_{\max}$ in their modes specified by σ, retaining $\boldsymbol{\tau}_{>\ell}$. It runs $\mathcal{G}$ on $(\widetilde{\mathsf{crs}}, \mathsf{sk}, \boldsymbol{\tau}_{>\ell})$, answers its queries using Sim_2 with the retained $\mathsf{td}_{\mathsf{NIZK}}$, and outputs the chosen test's bit.

The challenge uses the public bounds $(1^\kappa, 1^2, 1^{s_\ell}, 1^{n_\ell})$ and coordinate b. The ordinary and trapdoor challenge branches reproduce $\mathcal{H}_\sigma$ and $\mathcal{H}_{\sigma'}$, respectively: the input of $\mathcal{G}$ and the test do not require τ_ℓ. Hence CRS indistinguishability bounds each test's absolute probability gap by μ.

The event E_b is the disjoint union of $T_{b,\perp} = 1$ and $T_{b,a,z} = 1$ over pairs satisfying $a \overset{2^{\ell-1}}{\nrightarrow} z$. Summing the test bounds therefore gives $\Pr_{\mathcal{H}_{\sigma'}}[E_b] \ge p/2 - (2^{2n} + 1)\mu$. The false-reachability conditions are used only to select terms in this probability sum. Each distinguisher has a single pair (a, z) as advice and tests decryption equality.

Constructing a level-$(\ell - 1)$ adversary. Define $\mathcal{G}'$ with input $(\widetilde{\mathsf{crs}}, \mathsf{sk}, \boldsymbol{\tau}_{>\ell-1})$ and the same simulation oracle as follows. It runs $\mathcal{G}$ with $(\widetilde{\mathsf{crs}}, \mathsf{sk}, \boldsymbol{\tau}_{>\ell})$, forwarding its oracle queries. It returns $\perp$ unless the output N_ℓ accepts and $x_0 \notin Q_{\mathsf{src}}$. Otherwise it computes $w \leftarrow \mathsf{BARG.E}(\tau_\ell, R_\ell, (X_1(N_\ell), X_2(N_\ell)), \pi^{(\ell)})$ and returns $\perp$ unless $R_\ell(X_b(N_\ell), w) = 1$. For $\ell = 1$, it parses w as a NIZK-SoK proof and sets the child midpoint to $\perp$. For $\ell > 1$, it parses w as a child midpoint ciphertext and a level-$(\ell - 1)$ proof. It returns the resulting node $N_{\ell-1}$, taking its source and endpoint ciphertexts from $X_b(N_\ell)$.
To bound the failed relation check, construct a somewhere-extraction adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$. Its first stage performs the setup of $\mathcal{D}_1$ and outputs $(b, \mathsf{st}_{\mathcal{D}})$. Given its trapdoor-mode challenge CRS, its second stage completes the higher setup and runs $\mathcal{G}$ exactly as $\mathcal{D}_2$. If N_ℓ accepts and has fresh source, it outputs $(R_\ell, (X_1(N_\ell), X_2(N_\ell)), \pi^{(\ell)})$; otherwise it outputs $\perp$. The somewhere-extraction experiment applies its extractor with τ_ℓ to precisely this batch and proof. Its extraction-failure event is therefore the event that $\mathcal{G}'$ reaches its extraction step but $R_\ell(X_b(N_\ell), w) \ne 1$. Somewhere extraction bounds this event by μ.
A successful relation check ensures that the returned child accepts. Its source is the public value x_0 in $X_b(N_\ell)$: for $\ell > 1$, the recursive relation verifies the child BARG proof on two instances containing this same x_0; for $\ell = 1$, it verifies the child NIZK-SoK proof with message x_0. The child has the selected endpoint ciphertexts, so it is bad whenever E_b holds and extraction succeeds. The query-source set is unchanged after $\mathcal{G}$ returns, so

source freshness is preserved. Thus, writing p' for the probability that $\mathcal{G}'$ returns an accepting bad fresh-source node in $\mathcal{H}_{\sigma'}$, we have $p' \geq p/2 - (2^{2n}+2)\mu$.
Since $\sigma'_k = \bot$ for $1 \leq k \leq \ell - 1$, the induction hypothesis applies to $\mathcal{G}'$ in $\mathcal{H}_{\sigma'}$. Substituting its bound on p' into $p \leq 2p' + 2(2^{2n}+2)\mu$ proves (10) at level ℓ. Finally, $2^\ell - 1 \leq 2^\ell$ gives (9).

Proof (Proof of newrt-SIM-SND*).* Fix a non-uniform PPT adversary $\mathcal{A}$, and let ϵ denote its success probability in the newrt-SIM-SND experiment for $(\mathsf{Sim}_1, \mathsf{Sim}_2)$.
Consider a winning output $((x_0, x_t, t), \Pi)$. Verification gives a canonical dyadic frontier covering $[0, t]$, with at most one root at each level, accepting root proofs, equal ciphertexts at adjacent boundaries, and valid outer openings to x_0, x_t. If every root were good, its endpoint decryptions would satisfy a computation of the corresponding length. The shared boundary ciphertexts identify the states at adjacent endpoints, and perfect PKE correctness identifies the outer states with x_0, x_t. Concatenating these computations would give $x_0 \overset{t}{\rightsquigarrow} x_t$, contrary to the winning condition. Thus some frontier root is bad. Moreover, the winning condition gives $x_0 \notin Q_{\mathsf{src}}$, so every frontier root has fresh source.
For each $0 \leq \ell \leq \ell_{\max}$, define $\mathcal{G}_\ell$ with the inputs and oracle of Lemma D.7. It runs $\mathcal{A}$ with the simulation oracle, recording its query sources. On output $((x_0, x_t, t), \Pi)$, it checks $\mathsf{IVC.V}(\widetilde{\mathsf{crs}}, (x_0, x_t, t), \Pi) = 1$ and $x_0 \notin Q_{\mathsf{src}}$. If both checks hold and the frontier contains a level-ℓ root, it returns that root as a node with source x_0; otherwise it returns $\bot$. This procedure is polynomial time, since it uses only verification and the recorded query sources.
In $\mathcal{H}_{(\bot,\ldots,\bot)}$, the public parameters and simulation oracle have exactly their distribution in the newrt-SIM-SND experiment. Every winning output causes at least one $\mathcal{G}_\ell$ to return an accepting bad fresh-source node. Therefore a union bound and Lemma D.7 give

$$\begin{aligned} \epsilon &\leq \sum_{\ell=0}^{\ell_{\max}} 2^\ell (3 \cdot 2^{2n} + 5)\mu \\ &\leq 2(\ell_{\max}+1)T(3 \cdot 2^{2n} + 5)\, 2^{-\kappa^\delta}, \end{aligned} \tag{11}$$

where the second inequality uses $2^{\ell_{\max}} \leq 2T$.

Resources and parameter choice. Let τ bound the running time of $\mathcal{A}$, including writing its queries, and let D be the sum of its valid query depths. Since query depths are given in unary, $D \leq \tau$. Each reduction runs $\mathcal{A}$ once, generates the remaining public parameters, simulates D computation steps, and performs at most $\ell_{\max}$ BARG extractions and one NIZK extraction. Its total running time is $\mathsf{poly}(\tau, \kappa, n, m, \ell_{\max}, |C|)$. The child indices and fixed pairs used in the CRS tests require only polynomial advice length. Thus the reductions are polynomial-size adversaries for the underlying primitives, and their assumed security bounds apply.
Since $\ell_{\max} \leq \lambda$ and $T \leq 2^\lambda$, (11) gives $\epsilon \leq \mathsf{poly}(\lambda)\, 2^{2n+\lambda-\kappa^\delta}$. Our parameter choice gives $\kappa^\delta \geq (3n + 2\lambda)^{\delta/\delta'}$, where $\delta/\delta' > 1$. Hence $\kappa^\delta \geq 2n + 3\lambda$ for all sufficiently large λ, and therefore $\epsilon \leq \mathsf{poly}(\lambda)\, 2^{-2\lambda} \leq 2^{-\lambda}$. This proves newrt-SIM-SND.

E w-SIM-SND Proof of Context-chained IVC

For weak simulation soundness we need the following unexplainability property from the PKE. We first define this property next, and then prove w-SIM-SND.

E.1 Unexplainable encryption

Definition E.1 (Unexplainable encryption) *A public-key encryption scheme* $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *has* unexplainable encryptions *if, for every PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{PKE.Enc}(\mathsf{pk}, x; r) = \mathsf{ct}^\star \;:\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^n), \\ \rho \leftarrow \mathcal{R}_{\mathsf{PKE}}, \\ \mathsf{ct}^\star \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0; \rho), \\ (x, r) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}^\star) \end{array}\right] = \mathsf{negl}(n).$$

Here $\mathcal{R}_{\mathsf{PKE}}$ *denotes the randomness space of the encryption algorithm.*

For standard schemes such as ElGamal, unexplainability already holds under discrete-log hardness: any valid opening reveals the encryption exponent r from the ciphertext component g^r, even given the secret key.
More generally, any perfectly correct CPA-secure PKE can be compiled to have this property using a PRG. The compiled scheme encrypts the message with coins derived from a PRG seed and appends an encryption of the expanded coins under an independent key. Perfect correctness forces any valid explanation to supply a seed producing those coins, which contradicts pseudorandomness because the PRG's range is sparse. We give the full transformation and proof below.

Instantiating unexplainable encryption. We give a generic compiler from perfectly correct CPA-secure PKE and a PRG. The compiled scheme uses a short PRG seed as its encryption randomness. Its ciphertext contains an encryption of the message and, under an independent key, an encryption of part of the expanded randomness. Perfect correctness forces any valid explanation to recover a seed producing this value. Since the PRG's range is sparse, producing such an explanation contradicts pseudorandomness, even given both secret keys.
Let PKE be a perfectly correct CPA-secure encryption scheme, and let $r(\lambda, n)$ be its fixed encryption randomness length for n-bit messages. Set $R := \max\{r(\lambda, n), 2\lambda\}$. Let $G_{\lambda,n} : \{0,1\}^\lambda \to \{0,1\}^{R+r(\lambda,R)}$ be a PRG. Parse its output as $G_{\lambda,n}(s) = (\rho, \omega)$, where $|\rho| = R$ and $|\omega| = r(\lambda, R)$. Write ρ_{enc} for the first $r(\lambda, n)$ bits of ρ.
Define PKE' as follows.

1. $\mathsf{PKE}'.\mathsf{Gen}(1^\lambda, 1^n)$: independently sample $(\mathsf{pk}_{\mathsf{enc}}, \mathsf{sk}_{\mathsf{enc}}) \leftarrow \mathsf{PKE}.\mathsf{Gen}(1^\lambda, 1^n)$ and $(\mathsf{pk}_{\mathsf{tag}}, \mathsf{sk}_{\mathsf{tag}}) \leftarrow \mathsf{PKE}.\mathsf{Gen}(1^\lambda, 1^R)$. Return $\mathsf{pk}' := (\mathsf{pk}_{\mathsf{enc}}, \mathsf{pk}_{\mathsf{tag}})$ and $\mathsf{sk}' := (\mathsf{sk}_{\mathsf{enc}}, \mathsf{sk}_{\mathsf{tag}})$. The public keys include the parameters needed to determine these length bounds.
2. $\mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}', x; s)$: compute $(\rho, \omega) := G_{\lambda,n}(s)$ and return
$$\big(\mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{enc}}, x; \rho_{\mathsf{enc}}), \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{tag}}, \rho; \omega)\big).$$
3. $\mathsf{PKE}'.\mathsf{Dec}(\mathsf{sk}', (\mathsf{ct}, \gamma))$: return $\mathsf{PKE}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{enc}}, \mathsf{ct})$.

The encryption randomness of PKE' is $s \leftarrow \{0,1\}^\lambda$.

Lemma E.2 (Generic unexplainability) *If* PKE *is perfectly correct and CPA-secure and* G *is a PRG, then* PKE' *is perfectly correct, CPA-secure, and has unexplainability.*

Proof. Perfect correctness follows from that of PKE: decryption ignores the second ciphertext component and recovers the message from the first.

CPA security. Fix a CPA adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Each experiment below generates $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{PKE}'.\mathsf{Gen}(1^\lambda, 1^n)$ and obtains $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}')$, where $m_0, m_1 \in \{0,1\}^n$. For $b \in \{0,1\}$, consider the following experiments, each returning $\mathcal{A}_2(\mathsf{ct}^\star, \mathsf{st})$.

1. $H_{b,0}$: sample $s \leftarrow \{0,1\}^\lambda$, set $(\rho, \omega) := G_{\lambda,n}(s)$, and give $\mathcal{A}_2$ the ciphertext $\mathsf{ct}^\star := (\mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{enc}}, m_b; \rho_{\mathsf{enc}}), \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{tag}}, \rho; \omega))$.
2. $H_{b,1}$: sample (ρ, ω) uniformly and give $\mathcal{A}_2$ the same ciphertext expression as in $H_{b,0}$.
3. $H_{b,2}$: sample independent uniform ρ, ω and give $\mathcal{A}_2$ $\mathsf{ct}^\star := (\mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{enc}}, m_b; \rho_{\mathsf{enc}}), \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{tag}}, 0^R; \omega))$.

For each fixed b, PRG security makes $H_{b,0}$ and $H_{b,1}$ computationally indistinguishable: the reduction generates the keys, runs $\mathcal{A}_1$, and forms the ciphertext using its PRG challenge. CPA security under $\mathsf{pk}_{\mathsf{tag}}$ makes $H_{b,1}$ and $H_{b,2}$ computationally indistinguishable. Here the reduction generates the first key pair and samples ρ, allowing it to encrypt m_b with the known coins ρ_{enc} and submit $(\rho, 0^R)$ as its challenge messages.
Finally, $H_{0,2}$ and $H_{1,2}$ are computationally indistinguishable by CPA security under $\mathsf{pk}_{\mathsf{enc}}$. The reduction generates the second key pair and an independent encryption of 0^R, and uses its challenge encryption of m_0 or m_1 as the first component. Hence $H_{0,0}$ and $H_{1,0}$ are computationally indistinguishable, proving CPA security of PKE'.

Unexplainability. Let $\mathcal{A}$ attack unexplainability of PKE'. Construct a PRG distinguisher $\mathcal{D}$ that, on input $y = (\rho, \omega)$, proceeds as follows.

1. Generate $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{PKE}'.\mathsf{Gen}(1^\lambda, 1^n)$ and form $\mathsf{ct}^\star := (\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, 0^n; \rho_{\mathsf{enc}}), \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{tag}}, \rho; \omega))$.
2. Run $(x, s') \leftarrow \mathcal{A}(\mathsf{pk}', \mathsf{sk}', \mathsf{ct}^\star)$.
3. Output 1 iff $x \in \{0,1\}^n$, $s' \in \{0,1\}^\lambda$, and $\mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}', x; s') = \mathsf{ct}^\star$.

If $y = G_{\lambda,n}(s)$ for uniform $s \in \{0,1\}^\lambda$, the challenge has exactly the unexplainability distribution. Hence $\mathcal{D}$ outputs 1 with $\mathcal{A}$'s success probability.
If y is uniform, write $G_{\lambda,n}(s') = (\rho', \omega')$. A valid explanation gives $\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{tag}}, \rho'; \omega') = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{tag}}, \rho; \omega)$. By perfect correctness, $\rho' = \rho$. There are at most 2^λ possible values of ρ' as s' ranges over $\{0,1\}^\lambda$, whereas ρ is uniform in $\{0,1\}^R$. Thus $\mathcal{D}$ outputs 1 with probability at most $2^{\lambda - R} \le 2^{-\lambda}$. PRG security therefore bounds $\mathcal{A}$'s success probability by a negligible quantity. Since $\mathcal{D}$ generates both key pairs, it can give both secret keys to $\mathcal{A}$.

Application to our IVC. Instantiate the base PKE and PRG at the primitive security parameter κ specified in Theorem 6.1. The compiled scheme uses a uniformly sampled κ-bit seed as its encryption randomness. Its ciphertext contains two base-PKE ciphertexts, and its PRG output has length $R + r(\kappa, R)$, where $R = \max\{r(\kappa, n), 2\kappa\}$. These lengths, the key sizes, and the encryption and decryption costs are polynomial in (κ, n). Thus the compiler preserves the efficiency bounds used in Section 6.2; in particular, each endpoint opening has length κ.

Remark E.3 (Native unexplainability of ElGamal and Cramer–Shoup) *Several standard schemes are unexplainable without any modification, directly from the discrete-logarithm assumption, and this holds even given the secret key.*
For ElGamal over a group $\langle g \rangle$ of order q, a ciphertext is $\mathsf{PKE.Enc}(\mathsf{pk}, m; r) = (g^r,\ m \cdot \mathsf{pk}^r)$. The first component g^r determines r uniquely as $r = \log_g(g^r)$, so ElGamal is coin-injective. An opening (x, r) of a challenge $\mathsf{ct}^\star = (g^\rho, \mathsf{pk}^\rho)$ must satisfy $g^r = g^\rho$, hence $r = \rho$; producing ρ is exactly computing $\log_g(g^\rho)$. Knowledge of $\mathsf{sk} = \log_g(\mathsf{pk})$ is unrelated to $\log_g(g^\rho)$ and does not help. Thus ElGamal has unexplainable encryptions under discrete log, even given sk.
Cramer–Shoup encryption exposes the encryption exponent in the same way through its group elements g_1^r, g_2^r, so similar argument shows it is unexplainable under discrete log, again given sk.
Consequently, when the underlying PKE can be instantiated with such a scheme, no transformation is needed; the generic construction of Lemma E.2 is required only when the chosen base PKE does not already make its encryption coins hard to recover even given the secret key.

E.2 Weak simulation soundness

Throughout this proof, we refer to the root x_0 of a computation as its *source*, to avoid confusion with the roots of the proof tree.

Proof sketch. The soundness proof of DJJ$^+$ [DJJ+25] starts with an accepting proof of a false computation. The verifier's adjacency checks imply that some frontier block is false. Somewhere extraction from the BARG exposes an accepting child proof, and a false parent has a false child. Repeating this descent reaches an accepting false leaf; extracting its NIZK witness then contradicts one-step SNARG soundness. In our experiment the adversary also receives simulated proofs, whose leaves may certify false statements. Therefore NIZK simulation extractability gives this contradiction only when the extracted leaf's message is fresh. If it is stale, hash binding identifies a simulator query with the same leaf position, preceding prefix descriptors, and starting ciphertext, without extracting the proofs of the preceding blocks. CPA security and perfect correctness imply that distinct query occurrences have distinct starting ciphertexts, except with negligible probability. Therefore the stale leaf identifies a unique query occurrence with the same prefix context.
We next show that the target must inherit the entire queried statement. At an interior boundary of that query, terminating the target or extracting a fresh next leaf would reveal an opening of a simulator ciphertext and violate unexplainability. By query uniqueness, a stale next leaf must continue the same query. This advances the matched prefix to the query endpoint, where the winning condition requires a false remaining suffix. A

second descent extracts an accepting false leaf $[r, r+1]$ with $r \geq d_q$, where d_q is the matched query's depth. If its message were stale, hash binding and query uniqueness would place this leaf inside query q, forcing $r < d_q$, contrary to $r \geq d_q$. Hence its message is fresh, contradicting leaf soundness.

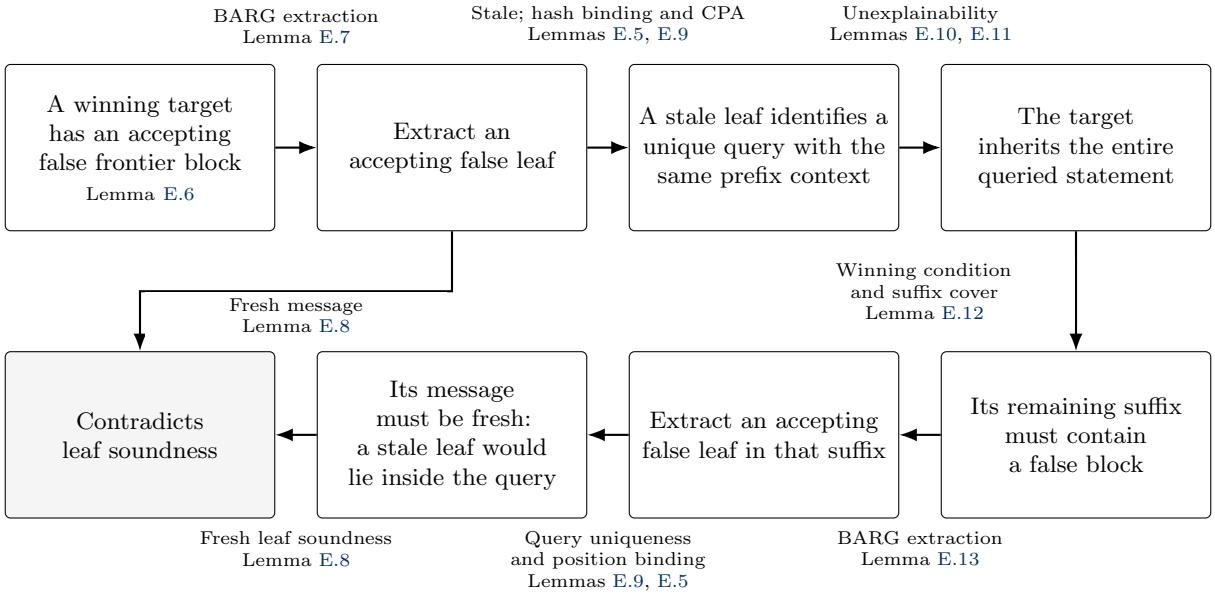


Fig. 14: The proof sketch with the two descents in the weak simulation-soundness proof.

The simulator and setup experiments We define $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ for the fixed construction parameters T, n, m, C. We use the initial chain value $\perp$ and functions Ch_ℓ defined in Section 6.

Setup algorithm Sim_1. On input $(1^\lambda, T, 1^n, 1^m, C)$, use the same security parameters as $\mathsf{IVC.Gen}$ to generate the SNARG parameters, PKE keys, and hash key honestly. Replace the NIZK-SoK setup by $(\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}, \tau_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TdSetup}(1^\kappa, R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}})$, where κ is the construction's NIZK security parameter. Then generate all BARG CRSs honestly, bottom-up, for the recursive relations defined using $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$. Return $(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}})$. We use the joint setup and simulation algorithms of the NIZK-SoK that satisfy both adaptive multi-theorem zero knowledge and simulation extractability. The simulation state is $\mathsf{st} := \mathsf{td}_{\mathsf{NIZK}}$; the extraction trapdoor τ_{NIZK} is generated jointly but is not needed by Sim_2.

Proof algorithm Sim_2. On input $(\widetilde{\mathsf{crs}}, \mathsf{td}_{\mathsf{NIZK}}, u, v, 1^d)$, return $\perp$ unless the query is well formed, $1 \le d \le T$, and $u, v \in \{0,1\}^n$. Otherwise proceed as follows, with fresh independent randomness on every call, including repeated statements. There is no requirement that $u \overset{d}{\rightsquigarrow} v$.

1. Sample $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, u; r_0)$ and $\mathsf{ct}_d \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, v; r_d)$, and independently sample $\mathsf{ct}_j \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0^n; \rho_j)$ for $0 < j < d$. Initialize $S^{(\ell)} = \emptyset$ for all $0 \le \ell \le \ell_{\max}$ and $h_0 = \perp$.
2. For $j = 0, \ldots, d-1$, set $m_j := \mathsf{Ch}_0(h_j, \mathsf{ct}_j, \mathsf{ct}_{j+1})$. Generate $\pi_j^{(0)}$ for the leaf $[j, j+1]$ by calling $\mathsf{NIZK.Sim}$ with CRS $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$, trapdoor $\mathsf{td}_{\mathsf{NIZK}}$, statement $(\mathsf{ct}_j, \mathsf{ct}_{j+1})$, and message m_j. Insert $(j, \mathsf{ct}_j, \perp, \mathsf{ct}_{j+1}, h_j, \pi_j^{(0)})$ into $S^{(0)}$, and execute the merge procedure of $\mathsf{IVC.P}$ verbatim, using the honest BARG prover on the prescribed child instances and witnesses. Set h_{j+1} to the chain value at the right endpoint of the resulting merged frontier, obtained by folding its descriptors from left to right starting at $\perp$.
3. Return $\widetilde{\Pi}_d := (\mathsf{ct}_0, \mathsf{ct}_d, r_0, r_d, \{S^{(\ell)}\}_{\ell=0}^{\ell_{\max}})$.

The simulated leaf proofs verify, so they serve as valid child-proof witnesses for the first BARG merges. Perfect BARG completeness then ensures, inductively, that every higher merge accepts and the returned IVC proof verifies. Only the level-0 proofs are simulated; the BARG proofs are generated honestly. The returned openings are those of the two endpoints.

Simulation records. Fix a polynomially bounded query bound $\widetilde{N} \ge 1$ before setup. Let $N \le \widetilde{N}$ be the number of valid query occurrences in an execution, indexed in order by $q = 1, \ldots, N$. Write query q as $[x_0^q \overset{d_q}{\rightsquigarrow} x_{d_q}^q]_?$ and its ciphertext boundaries as $\mathsf{ct}_0^q, \ldots, \mathsf{ct}_{d_q}^q$. Let $\mathcal{L}_q$ be the complete simulation record of query occurrence q. It contains the queried statement, the returned proof $\widetilde{\Pi}^q$, all ciphertext boundaries, every node created by the append-and-merge routine indexed by level and left endpoint, and the canonical frontier descriptor list after each of the d_q steps, including the initial empty list. In particular, it records every internal simulated leaf $(q, j, m_j^q, \pi_j^{q,(0)})$, where $m_j^q = \mathsf{Ch}_0(h_j^q, \mathsf{ct}_j^q, \mathsf{ct}_{j+1}^q)$ for $0 \le j < d_q$. Write N_j^q for this leaf. These entries are retained after the leaves have been merged into higher-level nodes. Thus matching a message to an entry identifies a query occurrence q, a leaf position j, its endpoints and incoming chain value, and the descriptor list preceding that leaf. The completed log is $\mathcal{L} = (\mathcal{L}_q)_{q=1}^N$, with one record per query occurrence. Each $\mathcal{L}_q$ includes the endpoint openings returned in the proof; setup secrets and interior encryption coins are kept separately. Let $Q := \{m_j^q : 1 \le q \le N,\ 0 \le j < d_q\}$ be the set of messages submitted to the NIZK simulator. A leaf is fresh if its message is outside Q, and stale otherwise.
A query adversary $\mathcal{A}$ receives $\widetilde{\mathsf{crs}}$ and the proof answers returned by Sim_2, and makes at most $\widetilde{N}$ valid queries. A reduction running $\mathcal{A}$ forwards each proof $\widetilde{\Pi}^q$ and retains $\mathcal{L}_q$. For the algorithms in the lemmas, write $\mathcal{O}_{\mathrm{rec}}$ for this oracle returning $(\widetilde{\Pi}^q, \mathcal{L}_q)$ on successful simulation and $\perp$ when Sim_2 fails. Every record reference uses the log of completed calls in that execution.
As in DJJ$^+$ [DJJ+25], Definition 3.2, the simulator receives the query depth in unary. If τ is the adversary's running time, including writing its queries, then $D := \sum_q d_q \le \tau$. The global bound T and the final target depth are binary and may be exponential.

BARG setup modes. To extract selected children in the reductions, we use setup experiments with a designated child at each required level. Fix a vector $\sigma = (\sigma_1, \ldots, \sigma_{\ell_{\max}}) \in \{\perp, 1, 2\}^{\ell_{\max}}$ before setup, where σ_k specifies the setup mode at level k. The experiment $\mathcal{H}_\sigma$ samples the PKE, SNARG, hash, and joint NIZK

simulation/extraction parameters as in Sim_1. It then generates the BARG CRSs for $k = 1, \ldots, \ell_{\max}$, in this order. At level 1, the relation R_1 checks a child NIZK-SoK proof using the already sampled $\widetilde{\mathsf{crs}}_{\mathsf{NIZK}}$. At level $k \geq 2$, the relation R_k checks a child level-$(k-1)$ BARG proof using the already sampled crs_{k-1}. These are the construction's recursive relations, with their public parameters fixed by this execution. If $\sigma_k = \perp$, let crs_k be the output of ordinary $\mathsf{BARG.Setup}$ and set $\tau_k = \perp$. If $\sigma_k = b \in \{1, 2\}$, let (crs_k, τ_k) be the output of $\mathsf{BARG.TdSetup}$ designated to child b, where $b = 1$ is the left child and $b = 2$ is the right child. Both setups use security parameter κ, batch size 2, and the construction's public input- and circuit-size bounds. The τ_k are private extraction trapdoors; each lemma specifies which of them its procedure receives.
The experiment answers queries using Sim_2 and keeps $\mathcal{L}$. In an auxiliary mode, every recursive prover output is retained in its node's proof slot and the append-and-merge schedule continues even if that output rejects. Thus all prescribed nodes, descriptors, and internal NIZK simulation calls are recorded. When $\sigma = (\perp, \ldots, \perp)$ and only proof answers are forwarded to $\mathcal{A}$, this is the actual simulation experiment. In a CRS switch at level k, levels $1, \ldots, k-1$ have already been generated when the challenge CRS is inserted as crs_k. Levels $k+1, \ldots, \ell_{\max}$ are then generated in their prescribed modes using the recursive relations defined by that CRS.

Security bounds. Set $\mu := 2^{-\kappa^\delta}$, with κ and δ as in Theorem 6.1. Since $\delta < \min_i \delta_i$, for sufficiently large λ every primitive failure probability and absolute distinguishing gap used below is at most μ. All algorithms below are non-uniform PPT. Their fixed indices and state pairs are advice chosen before setup. Thus the primitive bounds apply uniformly over these choices, at the resources established at the end of the proof.

Checked paths and context binding

Nodes and child instances. A target $(x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*, \Pi^*)$ is accepting if its states have length n, $0 \leq t^* \leq T$, and $\mathsf{IVC.V}(\widetilde{\mathsf{crs}}, (x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*), \Pi^*) = 1$. Write $\mathsf{ct}^*_0, \mathsf{ct}^*_{t^*}$ and $r^*_0, r^*_{t^*}$ for the endpoint ciphertexts and their openings in Π^*. All target acceptance checks below include the stated domain checks. Verification fixes the frontier intervals to the canonical dyadic decomposition of $[0, t^*]$, with at most one root at each level.
A level-ℓ node is $N = (j, \mathsf{ct}_L, \mathsf{ct}_M, \mathsf{ct}_R, h_N, \pi^{(\ell)})$, with interval $I(N) = [j, j + 2^\ell]$ and descriptor $\mathsf{Desc}(N) = (\ell, h_N, \mathsf{ct}_L, \mathsf{ct}_R)$. At level 0, set $\mathsf{ct}_M = \perp$; the node accepts if its NIZK-SoK proof verifies for $(\mathsf{ct}_L, \mathsf{ct}_R)$ with message $m(N) := \mathsf{Ch}_0(h_N, \mathsf{ct}_L, \mathsf{ct}_R)$. At level $\ell \geq 1$, define $X_L(N) := (\mathsf{ct}_L, \mathsf{ct}_M, h_N)$ and $X_R(N) := (\mathsf{ct}_M, \mathsf{ct}_R, \mathsf{Ch}_{\ell-1}(h_N, \mathsf{ct}_L, \mathsf{ct}_M))$. The node accepts if its level-ℓ BARG proof verifies for the batch $(X_L(N), X_R(N))$ and recursive relation R_ℓ. Here R_k denotes the recursive relation with its public parameters fixed by the current setup. Call N bad if an endpoint decrypts to $\perp$, or if its decrypted endpoint pair (a, z) satisfies $a \overset{2^\ell}{\rightsquigarrow} z$. The same endpoint condition defines badness of a descriptor.

Extraction along a path. Fix $0 \leq h \leq \ell \leq \ell_{\max}$ and $\rho = (\rho_\ell, \ldots, \rho_{h+1}) \in \{L, R\}^{\ell-h}$; set $\iota(L) = 1$ and $\iota(R) = 2$. The deterministic algorithm $\mathsf{Open}_\rho(\widetilde{\mathsf{crs}}, N_\ell, \{\tau_k\}_{h<k\leq\ell})$ returns $\perp$ or a transcript $(N_\ell, \ldots, N_h)$. It operates as follows.

1. Verify N_ℓ; return $\perp$ if it is malformed or rejects.
2. For $k = \ell, \ldots, h+1$, compute $w_k := \mathsf{BARG.E}(\tau_k, R_k, (X_L(N_k), X_R(N_k)), \pi^{(k)})$. Return $\perp$ unless $R_k(X_{\rho_k}(N_k), w_k) = 1$. If $k = 1$, parse w_k as the child NIZK-SoK proof and set its midpoint to $\perp$. If $k > 1$, parse it as the child midpoint and level-$(k-1)$ proof. Set the child's endpoint and chain-in triple to $X_{\rho_k}(N_k)$ and its interval to the selected half of $I(N_k)$; this defines N_{k-1}.
3. Return $(N_\ell, \ldots, N_h)$.

The relation check includes witness parsing and child verification. The path is complete when $h = 0$ and truncated otherwise. We suppress public parameters and trapdoor arguments in subsequent calls to Open_ρ. A path transcript is *checked* if every node accepts and, at each turn, the child's interval is the selected dyadic half and its endpoint and incoming-chain triple equals $X_{\rho_k}(N_k)$. These conditions are publicly checkable. An accepted target fixes the intervals of its frontier roots; a checked transcript from such a root therefore fixes its leaf position. The same definition applies to a truncated transcript. Write $\mathsf{CheckPath}(\Pi^*, \mathcal{P}, N_0)$ for the public check that $\mathcal{P}$ is a complete checked transcript starting at a frontier root of Π^* and ending at N_0; the current public parameters are implicit inputs. It returns 0 on malformed inputs.

In the two descents of Fig. 14, $\mathcal{G}$ represents the procedure that runs the adversary and selects or extracts the current node, from which Open_ρ continues extraction.

Lemma E.4 (Extraction along a fixed path) *Fix $0 \le h \le \ell \le \ell_{\max}$, a path $\rho \in \{L,R\}^{\ell-h}$, and a setup vector σ before setup, with $\sigma_k = \iota(\rho_k)$ for $h < k \le \ell$. For every PPT algorithm $\mathcal{G}^{\mathcal{O}_{\mathrm{rec}}}$ with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$ and output (J, N_J, ζ) or $\perp$, where $h \le J \le \ell$ and N_J is a level-J node, the following holds. Here $\boldsymbol{\tau}_{>\ell} := (\tau_{\ell+1}, \ldots, \tau_{\ell_{\max}})$, ζ is an auxiliary string, and $\rho|_J := (\rho_J, \ldots, \rho_{h+1})$.*

$$\Pr_{\mathcal{H}_\sigma}[N_J \text{ accepts} \ \wedge \ \mathsf{Open}_{\rho|_J}(N_J) = \perp] \le (\ell - h)\mu. \tag{12}$$

Whenever $\mathsf{Open}_{\rho|_J}$ returns a transcript, it is checked and ends at level h.

Proof. For $h < k \le \ell$, let F_k be the event that N_J accepts, $J \ge k$, all witness checks at levels $J, \ldots, k+1$ succeed, and $R_k(X_{\rho_k}(N_k), w_k) = 0$ for the extractor output w_k. These events are disjoint, and their union is the event on the left of (12).
For each k, construct an adversary $\mathcal{B}_k = (\mathcal{B}_{k,1}, \mathcal{B}_{k,2})$ for somewhere extraction (Definition 3.7), with batch size 2 and the construction's fixed public input- and circuit-size bounds.

1. $\mathcal{B}_{k,1}$ samples the SNARG, PKE, hash, and joint NIZK setup, then generates BARG levels $1, \ldots, k-1$ in their modes σ_i. It outputs $(\iota(\rho_k), \mathsf{st})$, where st contains this setup, its retained secrets, and the resulting relation R_k.
2. $\mathcal{B}_{k,2}$ receives the challenge CRS and st. It inserts that CRS as crs_k and generates levels $k+1, \ldots, \ell_{\max}$ in modes σ_i. It runs $\mathcal{G}^{\mathcal{O}_{\mathrm{rec}}}(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$, answering and recording simulation queries with its retained NIZK simulation trapdoor. It aborts unless $J \ge k$ and N_J accepts. It then runs $\mathsf{Open}_{(\rho_J, \ldots, \rho_{k+1})}(N_J)$ and aborts if this call fails. For its final node N_k, it outputs $(R_k, X_L(N_k), X_R(N_k), \pi^{(k)})$.

All trapdoors given to $\mathcal{G}$ and used in this truncated extraction belong to levels greater than k and are generated by $\mathcal{B}_{k,2}$. Thus the reduction reproduces $\mathcal{H}_\sigma$ through the construction of N_k using the challenge CRS alone at level k. On F_k, its output batch accepts, while the level-k extractor returns w_k with $R_k(X_{\rho_k}(N_k), w_k) = 0$. The somewhere-extraction bound therefore gives $\Pr[F_k] \le \mu$. Summing over $h < k \le \ell$ proves (12). Each successful relation check verifies the reconstructed child; the reconstruction fixes its interval and public triple. Induction on the number of successful checks proves the checked-transcript assertion.

The prefix exposed by a path. For a checked transcript $\mathcal{P}$ from a target frontier root to $[r, r+1]$, define $\mathsf{PathPref}(\Pi^*, \mathcal{P})$ by listing the descriptors of preceding frontier roots, followed by the left-child descriptor at each right turn, in root-to-leaf order. At a right turn from level k, that descriptor is $(k-1, h_{N_k}, \mathsf{ct}_L(N_k), \mathsf{ct}_M(N_k))$. Its fields are read directly from the parent. The prefix through $r+1$ is computed by appending $\mathsf{Desc}(N_0)$ and performing the construction's carries. A carry replaces $(k, h, \mathsf{ct}_A, \mathsf{ct}_B)$ and $(k, \mathsf{Ch}_k(h, \mathsf{ct}_A, \mathsf{ct}_B), \mathsf{ct}_B, \mathsf{ct}_C)$ by $(k+1, h, \mathsf{ct}_A, \mathsf{ct}_C)$. Write $\mathsf{Pref}^q(r)$ for the canonical descriptor list after the simulator has processed r steps of occurrence q. Its records determine this list at every $0 \le r \le d_q$, including boundaries hidden by subsequent merges.

Lemma E.5 (Stale leaf context binding) *Fix a setup vector σ before setup. For every PPT algorithm $\mathcal{G}^{\mathcal{O}_{\mathrm{rec}}}$ with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>0})$ and output $(N_0, x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*, \Pi^*, \mathcal{P})$ or $\perp$ in $\mathcal{H}_\sigma$, the following holds with probability at least $1-\mu$. If the output target is accepting and $\mathsf{CheckPath}(\Pi^*, \mathcal{P}, N_0) = 1$, write $[r, r+1]$ for the interval of N_0. Then for every query occurrence $q \in [N]$ and $0 \le j < d_q$ with $m(N_0) = m^q_j$, all of the following hold:*

(i) $r = j < d_q$ *and* $\mathsf{Desc}(N_0) = \mathsf{Desc}(N^q_j)$;
(ii) $\mathsf{PathPref}(\Pi^*, \mathcal{P}) = \mathsf{Pref}^q(r)$;
(iii) the prefix through $r+1$ computed from $\mathcal{P}$ equals $\mathsf{Pref}^q(r+1)$;
(iv) $\mathsf{ct}^*_0 = \mathsf{ct}^q_0$ *and* $x^*_{\mathsf{in}} = x^q_0$.

Proof. For an accepting target, a checked path to N_0, and a matching recorded leaf (q, j), we construct a deterministic polynomial-time algorithm $\mathsf{Bind}(\Pi^*, \mathcal{P}, \mathcal{L}_q, j)$. The public parameters and target statement are implicit inputs. It returns match with items (i)–(iv) holding, or distinct strings α, β with $\mathsf{H}_{\mathsf{hk}}(\alpha) = \mathsf{H}_{\mathsf{hk}}(\beta)$.

The preceding prefix. Write $\mathcal{P} = (N_\ell, \ldots, N_0)$ and let j_k be the left endpoint of N_k. After reaching N_k from the root, the accumulated descriptors form the canonical dyadic cover of $[0, j_k]$ and fold from $\perp$ to h_{N_k}. We prove this invariant by induction on the number of edges traversed. At N_ℓ, the accumulated list consists of the preceding frontier descriptors; the target's adjacency and chain checks establish the invariant. If the next child is left, its left endpoint and incoming chain value equal those of its parent, so the list is unchanged. If the next child is right, append the parent's left-child descriptor. This extends the covered interval from $[0, j_k]$ to $[0, j_k + 2^{k-1}]$. The appended level is smaller than every preceding level, and the checked child triple makes its chain-out equal to the right child's chain-in. Thus the invariant is preserved in either case. At N_0, the list is $\mathsf{PathPref}(\Pi^*, \mathcal{P})$, covers $[0, r]$, and folds to h_{N_0}. Similarly, induction on the number of simulated steps, using the append-and-merge rule, shows that $\mathsf{Pref}^q(j)$ is the canonical dyadic cover of $[0, j]$. Its fold is h_j^q by the simulator's definition of that chain value.

Binding the descriptors. First compare the encodings of $(1, 0, h_{N_0}, \mathsf{ct}_L(N_0), \mathsf{ct}_R(N_0))$ and $(1, 0, h_j^q, \mathsf{ct}_j^q, \mathsf{ct}_{j+1}^q)$. Their hashes are equal by the hypothesis. Distinct inputs are therefore a collision; otherwise the leaf descriptors agree. Starting from their equal chain-ins, compare the two prefix lists backwards. For each nonempty list, its final descriptor supplies a known preimage of the current chain value. If both preimages agree, injective encoding identifies the final descriptors and their incoming chain values, and the comparison continues with those values. If they differ, output the two preimages. If exactly one list is empty, compare its initial preimage 0 with the other's final chain input; these are distinct by domain separation and have equal hashes. Thus termination without a collision implies entrywise equality of the lists. The sum of their dyadic lengths gives $r = j$ and proves items (i) and (ii). The first descriptor gives the starting-ciphertext equality; when the prefix is empty, the leaf supplies it directly. Perfect correctness and the valid source openings give the source-state equality in item (iv).

Including the leaf. Append the equal leaf descriptors to the equal preceding lists and apply the deterministic carry rule. Equal inputs give equal outputs at every carry. The query's merge routine uses this rule, so the resulting lists agree, proving item (iii). This completes the deterministic algorithm: it compares the known hash inputs and returns either a collision or match. All compared hash inputs are determined by the supplied path descriptors and the recorded query descriptors; the leaf comparisons use their incoming chain values and endpoints.

To bound the failure probability, a collision-resistance adversary generates the remaining setup using its challenge hash key and runs $\mathcal{G}$ with its prescribed oracle and trapdoor inputs. It checks target verification and $\mathsf{CheckPath}(\Pi^*, \mathcal{P}, N_0) = 1$, then scans the records with $m(N_0) = m_j^q$ and runs Bind on each. It outputs the first returned collision. If any matching pair (q, j) violates an item of the lemma, Bind must return distinct α, β with $\mathsf{H}_{\mathsf{hk}}(\alpha) = \mathsf{H}_{\mathsf{hk}}(\beta)$. There are D recorded leaf entries, so this scan is polynomial time. Collision resistance bounds the probability of finding such a collision by μ. Therefore, except with probability at most μ, all four items hold for every matching pair.

The first descent

Lemma E.6 (A winning target has a bad frontier block) *In every execution of $\mathcal{H}_\sigma$, an accepting target satisfying $x^*_{\mathsf{in}} \overset{t^*}{\rightsquigarrow} x^*_{\mathsf{out}}$ has $t^* > 0$ and at least one accepting bad frontier node.*

Proof. Enumerate the frontier nodes from left to right as $N_1, \ldots, N_b$, with lengths $2^{\ell_1}, \ldots, 2^{\ell_b}$. Verification gives accepting node proofs, identical ciphertexts at each adjacent boundary, valid outer openings, and $\sum_i 2^{\ell_i} = t^*$. If no node is bad, every boundary has a valid decryption and these states satisfy the corresponding exact-length computations. Perfect correctness identifies the outer states with $x^*_{\mathsf{in}}, x^*_{\mathsf{out}}$. Concatenating the computations gives $x^*_{\mathsf{in}} \overset{t^*}{\rightsquigarrow} x^*_{\mathsf{out}}$, a contradiction. If $t^* = 0$, verification identifies the two endpoint ciphertexts; their valid openings and perfect correctness imply $x^*_{\mathsf{in}} = x^*_{\mathsf{out}}$, again a contradiction.

CRS switches. To switch level k between ordinary mode and trapdoor mode designated to child b, use a distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ for the CRS-indistinguishability condition (Definition 3.7), using the construction's fixed public size bounds.

1. $\mathcal{D}_1$ samples the PKE, SNARG, hash, and joint NIZK setup, then generates BARG levels $1, \ldots, k-1$ in their prescribed modes. It outputs (b, st), retaining these parameters, their setup secrets, and the resulting relation R_k in st.

2. $\mathcal{D}_2$ inserts the challenge CRS as crs_k, generates levels $k+1,\dots,\ell_{\max}$ in their prescribed modes, and runs the specified algorithm and Boolean test. It answers simulation queries with the retained NIZK simulation trapdoor and keeps their records $\mathcal{L}$.

The algorithm receives the inputs specified in the relevant lemma. The test uses only locally generated trapdoors and, where needed, the retained PKE secret key. In each application below, we verify that this computation requires no level-k trapdoor. In the ordinary branch this is the experiment with $\sigma_k = \perp$; in the trapdoor branch it is the experiment with $\sigma_k = b$. The CRS-indistinguishability bound therefore makes the difference between their output-1 probabilities at most μ. A change of designated index from 1 to 2 goes through ordinary mode and has total error at most 2μ. We refer to this consequence as index hiding. The tests used below are specified by literal equality, verification, and, when explicitly stated, decryption to a fixed pair; the reachability facts selecting those pairs enter only the probability analysis.

Lemma E.7 (Bad-node descent) *Let $0 \le \ell \le \ell_{\max}$ and let σ be a setup vector. Let $\mathcal{G}^{\mathcal{O}_{\mathrm{rec}}}$ be a non-uniform PPT algorithm with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$ and output $(N_\ell, x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*, \Pi^*, \mathcal{P})$ or $\perp$. Whenever it returns a tuple, N_ℓ is a level-ℓ node, the target is accepting, and $\mathcal{P}$ is a checked transcript from a frontier root to N_ℓ.*
For $\rho \in \{L,R\}^\ell$, define $\mathcal{G}_\rho^{\mathcal{O}_{\mathrm{rec}}}$ with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>0})$ as follows. It runs $\mathcal{G}$ with its prescribed inputs and the same oracle, returning $\perp$ if $\mathcal{G}$ does so. It then runs $\mathsf{Open}_\rho(N_\ell)$ using $(\tau_k)_{k=1}^{\ell}$ and returns $\perp$ if this call fails. Otherwise, it appends the extracted nodes $N_{\ell-1},\dots,N_0$ to $\mathcal{P}$, obtaining $\mathcal{P}'$, and outputs $(N_0, x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*, \Pi^*, \mathcal{P}')$.
For all sufficiently large λ, there exists a path ρ such that, for σ' defined by $\sigma'_k = \iota(\rho_k)$ for $1 \le k \le \ell$ and $\sigma'_k = \sigma_k$ for $k > \ell$,

$$\begin{aligned}&\Pr_{\mathcal{H}_{\sigma'}}[\mathcal{G}_\rho \text{ outputs an accepting bad leaf}]\\ &\quad\ge 2^{-\ell}\Pr_{\mathcal{H}_\sigma}[\mathcal{G} \text{ outputs an accepting bad node}] - (4\cdot 2^{2n}+6)\mu.\end{aligned} \tag{13}$$

Proof. Write p for the accepting bad-node probability of $\mathcal{G}$ in $\mathcal{H}_\sigma$, and set $c := 2\cdot 2^{2n}+3$. We prove by induction on ℓ the stronger conclusion with error $c\mu\sum_{i=0}^{\ell-1}2^{-i}$. For $\ell = 0$, take the empty path and $\sigma' = \sigma$. Then $\mathcal{G}_\rho$ returns the output of $\mathcal{G}$, so the assertion holds. Suppose $\ell \ge 1$ and the stronger conclusion holds at level $\ell - 1$.

Selecting a bad half. For $b \in \{L,R\}$, call half b bad if an endpoint in $X_b(N_\ell)$ decrypts to $\perp$, or its decrypted endpoints fail reachability in $2^{\ell-1}$ steps. A bad node has a bad half: two valid half-computations concatenate through their common midpoint, while an invalid outer endpoint belongs to an incident half. Let E_b be the event that $\mathcal{G}$ returns a tuple and half b is bad. Since every returned node accepts, $p \le \Pr_{\mathcal{H}_\sigma}[E_L] + \Pr_{\mathcal{H}_\sigma}[E_R]$. Choose $b \in \{L,R\}$ with $\Pr_{\mathcal{H}_\sigma}[E_b] \ge p/2$. Let $\widehat{\sigma}$ agree with σ except that $\widehat{\sigma}_\ell = \iota(b)$.

Switching the level-ℓ CRS. As in the fixed-pair argument of! [DJJ+25], compare the two experiments using the following Boolean tests. Each test requires that $\mathcal{G}$ returns a tuple. The test $T_{b,\perp}$ checks whether either endpoint of half b decrypts to $\perp$. For a fixed $(a,z) \in \{0,1\}^{2n}$, the test $T_{b,a,z}$ checks whether these endpoint decryptions equal (a,z).
Instantiate the preceding CRS-switch reduction with each test. It generates levels below ℓ in their modes σ_k before receiving the challenge CRS, inserts the challenge at level ℓ, and generates the higher levels in their modes σ_k. It runs $\mathcal{G}$ with $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$ and oracle $\mathcal{O}_{\mathrm{rec}}$, and evaluates the test using its privately retained PKE secret key. The computation requires no τ_ℓ. Changing the level-ℓ mode to $\iota(b)$ costs at most 2μ for each test, by switching through ordinary mode when necessary.

The event E_b is the disjoint union of $T_{b,\perp} = 1$ and $T_{b,a,z} = 1$ over pairs satisfying $a \overset{2^{\ell-1}}{\rightsquigarrow} z$. Summing the test bounds therefore gives $\Pr_{\mathcal{H}_{\widehat{\sigma}}}[E_b] \ge p/2 - 2(2^{2n}+1)\mu$. The false-reachability conditions select terms only in this sum; each fixed-pair distinguisher has one pair as advice and tests decryption equality.

Constructing a child-producing algorithm. Define $\mathcal{G}'^{\mathcal{O}_{\mathrm{rec}}}$ with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell-1})$ as follows. It runs $\mathcal{G}$ with $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$, forwarding its oracle queries, and returns $\perp$ if $\mathcal{G}$ does so. It applies the one-step procedure $\mathsf{Open}_{(b)}(N_\ell)$, with terminal level $\ell - 1$ and trapdoor τ_ℓ. If this fails, it returns $\perp$; otherwise, it returns $(N_{\ell-1}, x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*, \Pi^*, \mathcal{P} \parallel (N_{\ell-1}))$.

To bound extraction failure, apply Lemma E.4 in $\mathcal{H}_{\widehat{\sigma}}$ with $h = \ell - 1$, upper level ℓ, and path (b). Its node-producing algorithm runs $\mathcal{G}$ and returns (ℓ, N_ℓ, ζ), where $\zeta = (x^*_{\mathsf{in}}, x^*_{\mathsf{out}}, t^*, \Pi^*, \mathcal{P})$. This algorithm has exactly the required input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$. Thus the probability that $\mathcal{G}$ returns its accepting node but the one-step extraction fails is at most μ.

On success, the child accepts, has the endpoint and incoming-chain triple $X_b(N_\ell)$, and extends the checked transcript. Hence it is bad whenever E_b holds and extraction succeeds. Writing p' for the accepting bad-node probability of $\mathcal{G}'$ in $\mathcal{H}_{\widehat{\sigma}}$, we obtain $p' \geq p/2 - c\mu$.

Apply the induction hypothesis to $\mathcal{G}'$ and $\widehat{\sigma}$ at level $\ell - 1$. It gives a path $\bar{\rho}$ from level $\ell - 1$ to level 0 and a final setup vector σ' in which the accepting bad-leaf probability is at least $2^{-(\ell-1)}p' - c\mu \sum_{i=0}^{\ell-2} 2^{-i}$. Substituting the bound on p' gives at least $2^{-\ell}p - c\mu \sum_{i=0}^{\ell-1} 2^{-i}$. Set $\rho_\ell = b$ and take the remaining directions from $\bar{\rho}$. The resulting procedure is exactly $\mathcal{G}_\rho$: it runs $\mathcal{G}$ once and performs the same deterministic checked extractions in sequence. Its target is the one returned in that execution, and its transcript extends the corresponding $\mathcal{P}$. The extraction steps make no oracle calls, so the query log is also the one from that run of $\mathcal{G}$. Finally, $\sum_{i=0}^{\ell-1} 2^{-i} \leq 2$ proves (13).

Lemma E.8 (Fresh leaf extraction and soundness) *Let σ be a setup vector and let $\mathcal{G}^{\mathcal{O}_{\mathrm{rec}}}$ be a non-uniform PPT algorithm with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>0})$ and output (N_0, ζ) or $\perp$, where N_0 is a leaf and ζ is an auxiliary string. On a leaf output, let $w \leftarrow \mathsf{NIZK.E}(\tau_{\mathsf{NIZK}}, (\mathsf{ct}_L(N_0), \mathsf{ct}_R(N_0)), \pi^{(0)}, m(N_0))$. For all sufficiently large λ, $\Pr_{\mathcal{H}_\sigma}[N_0$ accepts $\wedge\, m(N_0) \notin Q \wedge R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}((\mathsf{ct}_L(N_0), \mathsf{ct}_R(N_0)), w) = 0] \leq \mu$, and*

$$\Pr_{\mathcal{H}_\sigma}[N_0 \text{ accepts} \wedge N_0 \text{ is bad} \wedge m(N_0) \notin Q] \leq (1 + 2^{2n})\mu. \tag{14}$$

Proof. Let p denote the probability on the left-hand side of (14). We first bound the probability that NIZK extraction fails on an accepting fresh-message output of $\mathcal{G}$. This leaves probability at least $p - \mu$ of an accepting bad fresh leaf with a valid extracted witness, which we then use to construct a non-adaptive SNARG adversary.

Define a NIZK simulation-extractability adversary $\mathcal{B}_{\mathsf{SE}} = (\mathcal{B}_{\mathsf{SE},1}, \mathcal{B}_{\mathsf{SE},2})$ as follows.

1. $\mathcal{B}_{\mathsf{SE},1}$ generates the PKE, SNARG, and hash parameters as in $\mathcal{H}_\sigma$. It outputs $(R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}, \mathsf{st}_{\mathsf{SE}})$, retaining these parameters and the PKE secret key in $\mathsf{st}_{\mathsf{SE}}$.
2. Given the challenge NIZK CRS and $\mathsf{st}_{\mathsf{SE}}$, $\mathcal{B}_{\mathsf{SE},2}$ generates all BARG levels in modes σ and runs $\mathcal{G}$ with its prescribed inputs. It implements $\mathcal{O}_{\mathrm{rec}}$ using the simulator's algorithm and bookkeeping, obtaining every level-0 proof from its challenge simulation oracle on the prescribed ciphertext pair and message. If $\mathcal{G}$ returns an accepting leaf with $m(N_0) \notin Q$, it outputs $((\mathsf{ct}_L(N_0), \mathsf{ct}_R(N_0)), \pi^{(0)}, m(N_0))$; otherwise it outputs $\perp$.

The setup, oracle answers, and simulation records have exactly their distribution in $\mathcal{H}_\sigma$. Every message submitted to the challenge simulation oracle belongs to Q. Hence $m(N_0) \notin Q$ ensures that the output tuple is absent from its query-response list. The simulation-extractability experiment applies the jointly generated extractor to this tuple, exactly as in the lemma. Thus $\mathcal{B}_{\mathsf{SE}}$ succeeds with probability $\Pr_{\mathcal{H}_\sigma}[N_0$ accepts$\wedge m(N_0) \notin Q \wedge R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}((\mathsf{ct}_L(N_0), \mathsf{ct}_R(N_0)), w) = 0]$. Simulation extractability bounds this probability by μ, proving the first assertion.

Therefore, with probability at least $p - \mu$, $\mathcal{G}$ returns an accepting bad fresh leaf and w is a valid witness. On this event, parse $w = (a, z, r_L, r_R, \pi_{\mathsf{SNARG}})$. The relation gives $\mathsf{ct}_L(N_0) = \mathsf{PKE.Enc}(\mathsf{pk}, a; r_L)$, $\mathsf{ct}_R(N_0) = \mathsf{PKE.Enc}(\mathsf{pk}, z; r_R)$, and $\mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, (a, z), \pi_{\mathsf{SNARG}}) = 1$. Perfect PKE correctness identifies $a, z \in \{0,1\}^n$ with the endpoint decryptions. Since the leaf is bad, $(a, z) \notin L_{C,1}$.

We now construct a non-adaptive SNARG adversary $\mathcal{B}_{\mathsf{S}} = (\mathcal{B}_{\mathsf{S},1}, \mathcal{B}_{\mathsf{S},2})$ to obtain such a false accepting proof.

1. Before SNARG setup, $\mathcal{B}_{\mathsf{S},1}$ samples $(\widehat{a}, \widehat{z}) \leftarrow \{0,1\}^{2n}$ and outputs $(1^{2n}, 1^m, R_{C,1}, (\widehat{a}, \widehat{z}), \mathsf{st}_{\mathsf{S}})$, where $\mathsf{st}_{\mathsf{S}} := (\widehat{a}, \widehat{z})$.
2. Given the challenge SNARG parameters and st_{S}, $\mathcal{B}_{\mathsf{S},2}$ generates the remaining setup as in $\mathcal{H}_\sigma$, retaining the joint NIZK simulation and extraction trapdoors. It runs $\mathcal{G}$ with its prescribed inputs, answering and recording its simulation queries using the locally generated NIZK simulation trapdoor. It returns $\perp$ unless $\mathcal{G}$ outputs an accepting leaf with $m(N_0) \notin Q$. Otherwise, it computes w using $\mathsf{NIZK.E}$ as in the lemma and returns $\perp$ unless $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}((\mathsf{ct}_L(N_0), \mathsf{ct}_R(N_0)), w) = 1$. It parses $w = (a, z, r_L, r_R, \pi_{\mathsf{SNARG}})$ and outputs π_{SNARG} if $(a, z) = (\widehat{a}, \widehat{z})$; otherwise it outputs $\perp$.

The guessed pair is used only in the final equality check. Thus the setup, execution of $\mathcal{G}$, and extraction of w are independent of it. Whenever $\mathcal{G}$ returns an accepting bad fresh leaf with a valid extracted witness, the guess matches (a, z) with probability 2^{-2n}. On a match, $\mathcal{B}_{\mathsf{S}}$ outputs an accepting proof of its false instance, which was fixed before SNARG setup. Writing $\pi^\star$ for its output, we obtain

$$\Pr[(\widehat{a}, \widehat{z}) \notin L_{C,1} \ \wedge \ \mathsf{SNARG.V}(\mathsf{crs}_{\mathsf{SNARG},V}, (\widehat{a}, \widehat{z}), \pi^\star) = 1] \geq 2^{-2n}(p - \mu). \tag{15}$$

Non-adaptive SNARG soundness bounds the left-hand side by μ. Hence $p \leq \mu + 2^{2n}\mu$, proving (14).

Lemma E.9 (Distinct-query starting ciphertexts) *Fix σ and a polynomially bounded $\widetilde{N} \geq 1$ before setup. For every PPT adversary $\mathcal{A}$ with input $\widetilde{\mathsf{crs}}$ and at most $\widetilde{N}$ valid simulation queries, answered with the proofs returned by Sim_2 in $\mathcal{H}_\sigma$, let $N \leq \widetilde{N}$ denote the actual number of valid queries in that execution. Then*

$$\Pr_{\mathcal{H}_\sigma}[\exists\, 1 \leq q < q' \leq N : \mathsf{ct}_0^q = \mathsf{ct}_0^{q'}] \leq \widetilde{N}\mu. \tag{16}$$

Proof. Fix a PPT adversary $\mathcal{A}$ as in the statement. Perfect correctness implies that equal starting ciphertexts encrypt the same state. We guess the later query of the first collision and request a CPA challenge on its source state and a different state. Comparison with the earlier ciphertexts then distinguishes the two branches. Define $\mathcal{B}_{\mathsf{CPA}} = (\mathcal{B}_{\mathsf{CPA},1}, \mathcal{B}_{\mathsf{CPA},2})$. Its first stage performs steps 1–2 below; its second performs step 3.

1. Given the CPA public key, sample a uniform $I \in \{1, \ldots, \widetilde{N}\}$ and generate the remaining setup in mode σ, using the challenge public key as pk. Run $\mathcal{A}(\widetilde{\mathsf{crs}})$ and answer valid calls preceding I using Sim_2, retaining their records. Abort if two preceding starting ciphertexts collide or if $\mathcal{A}$ terminates before call I.
2. When valid call I with source u is submitted, choose $\bar{u}$ by flipping u's first bit and output $(u, \bar{u}, \mathsf{st})$, where st retains u and the preceding query records.
3. Given $(\mathsf{ct}^\star, \mathsf{st})$, output 1 iff some preceding call has source u and starting ciphertext $\mathsf{ct}^\star$.

If the first stage aborts, it outputs $(0^n, 0^n, \perp)$ and the second stage outputs 0. This uses only the CPA public key: all simulated openings before call I are locally sampled, and the challenge call is unanswered. Consider a full simulation run of $\mathcal{A}$, and let J be the later ordinal of its first starting-ciphertext collision, with $J = \infty$ if none occurs. In the encryption-of-u branch, couple the challenge encryption with the starting ciphertext of call I in this full run. The reduction outputs 1 exactly when $I = J$: earlier choices have no matching ciphertext, and later choices abort upon the first collision. Since I is independent of the full run, this probability is $\Pr[J < \infty]/\widetilde{N}$. In the encryption-of-$\bar{u}$ branch, output 1 is impossible by perfect correctness. The absolute distinguishing gap is $\Pr[J < \infty]/\widetilde{N}$. CPA security bounds this gap by μ. Hence $\Pr[J < \infty] \leq \widetilde{N}\mu$, proving (16).

In particular, if all query starts are distinct and Bind returns match for every recorded leaf with message $m(N_0)$, then a checked stale leaf determines exactly one query occurrence: every matching record has starting ciphertext ct_0^* by Lemma E.5.

Completing the inherited query For $1 \leq r \leq T$, let $\mathcal{H}_r^-$ denote the experiment with $\sigma_k = 1 + \mathrm{bit}_{k-1}(r-1)$ for every BARG level k. For $r < T$, define $\mathcal{H}_r^+$ with $\sigma_k = 1 + \mathrm{bit}_{k-1}(r)$. Thus $\mathcal{H}_r^+ = \mathcal{H}_{r+1}^-$. On any aligned frontier root containing the indicated leaf, these bits prescribe the paths to $[r-1, r]$ and $[r, r+1]$, respectively. Fix a PPT query adversary $\mathcal{A}$ and an ordinal $q \in \{1, \ldots, \widetilde{N}\}$ before setup. In this matching-query argument, Φ abbreviates one of the following conditions on the target and query record $\mathcal{L}_q$:

1. $\mathsf{ct}_0^* = \mathsf{ct}_0^q$ and $d_q > t^*$;
2. $\mathsf{ct}_0^* = \mathsf{ct}_0^q$, $d_q \leq t^*$, and $(x_{d_q}^q, x_{\mathsf{out}}^*, t^* - d_q) = (a, z, s)$, for a fixed $(a, z, s) \in \{0,1\}^{2n} \times \{0, \ldots, T\}$ with $a \overset{s}{\not\rightsquigarrow} z$.

The choice of condition and, in the second case, the triple are fixed before setup. Set $\Phi = 0$ if query q is absent. The completion lemmas below use only these two matching-query conditions. These checks are polynomial-time: the second tests equality to the fixed triple, while its false-reachability property is used only in the analysis.

The shared-boundary equality test. For fixed q, r, the deterministic test $\mathsf{Eq}_{q,r}$ takes the public parameters, target, log, and the BARG trapdoors used by the following algorithm.

1. Reject unless the target accepts, query q exists, and $1 \le r \le \min(d_q, t^*)$. Set $k := \nu_2(r) + 1$.
2. If r is a frontier boundary, form the descriptor list of frontier roots preceding that boundary; if $r = t^*$, use the full frontier.
3. Otherwise, let N_J be the frontier root containing both adjacent leaves. Their common level-k parent is reached by $\rho = (\rho_J, \dots, \rho_{k+1})$, where $\iota(\rho_i) = 1 + \mathrm{bit}_{i-1}(r-1)$. Run $\mathsf{Open}_\rho(N_J)$ and reject if it returns $\perp$. Form the list of preceding frontier descriptors, followed by the left-child descriptors at right turns of this transcript, then append $(k-1, h_{N_k}, \mathsf{ct}_L(N_k), \mathsf{ct}_M(N_k))$.
4. Accept iff the resulting list is $\mathsf{Pref}^q(r)$.

The retained transcript and public frontier entries are its *boundary view*. It uses only $\boldsymbol{\tau}_{>k}$; in the frontier-boundary case it uses no trapdoor.

To verify this construction, observe that the appended block is $[r - 2^{k-1}, r]$, so it has level $k - 1 = \nu_2(r)$. All lower turns of the ending-leaf path are right turns. At each carry, the checked child triples identify the result with its parent descriptor. Its prefix through r is therefore exactly the list in the test. The starting-leaf path takes the right child of N_k and then only left turns, so its preceding prefix is the same list. Successful full extractions of these paths restrict to the test's transcript because the BARG extractor is deterministic. At a frontier boundary the list is public and the verifier fixes its boundary ciphertext. In particular, whenever a full ending or starting path gives the corresponding prefix equality, the equality test accepts. For $r < T$, the vectors for $\mathcal{H}_r^-$ and $\mathcal{H}_r^+$ agree above k, and differ only at levels at most k.

In each experiment, evaluate $\mathsf{Eq}_{q,r}$ after $\mathcal{A}$ returns its target, using that experiment's log and prescribed trapdoors. Define $\mathsf{Anc}_{q,r} := \{\mathsf{Eq}_{q,r} = 1 \ \wedge \ \Phi(\widetilde{\mathsf{crs}}, \mathrm{target}, \mathcal{L}) = 1\}$. These events are efficiently decidable from their stated inputs.

Lemma E.10 (Advancing a matched prefix) *Fix $q \in \{1, \dots, \widetilde{N}\}$, $r \in \{1, \dots, T\}$, and either case condition Φ above, all before setup. For every PPT query adversary $\mathcal{A}$, the following hold.*

1. $\Pr_{\mathcal{H}_r^-}[\mathsf{Anc}_{q,r} \wedge t^* = r < d_q] \le \mu$.
2. *If $r < T$, then*

$$\Pr_{\mathcal{H}_r^+}[\mathsf{Anc}_{q,r+1}] \ge \Pr_{\mathcal{H}_r^-}[\mathsf{Anc}_{q,r} \wedge r < \min(d_q, t^*)] - (3\ell_{\max} + \widetilde{N} + 3)\mu. \tag{17}$$

Proof. For the first assertion, define an unexplainability adversary $\mathcal{B}_{\mathrm{term}}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}^\star)$ as follows.

1. Generate the remaining setup in mode $\mathcal{H}_r^-$, retaining the joint NIZK trapdoors.
2. Run $\mathcal{A}$ and answer its queries using the simulator code. In valid occurrence q, if $r < d_q$, replace the interior ciphertext ct_r^q by $\mathsf{ct}^\star$. Sample all other ciphertexts independently as prescribed by Sim_2.
3. If $\mathsf{Anc}_{q,r} \wedge t^* = r < d_q$ holds, output $(x^*_{\mathsf{out}}, r^*_{t^*})$; otherwise output $\perp$.

The challenge is a fresh encryption of 0^n. Since $r < d_q$ on the tested event, the public parameters, query records, and target have exactly their distribution in $\mathcal{H}_r^-$ there. The adjacent NIZK proofs are simulated, and every subsequent merge uses the prescribed public instances and child proofs. Equality of the full target prefix with $\mathsf{Pref}^q(r)$ gives $\mathsf{ct}^*_{t^*} = \mathsf{ct}_r^q = \mathsf{ct}^\star$. Target verification makes the returned pair a valid opening of the challenge. Unexplainability therefore bounds this event by μ.

For the second assertion, let $U := \mathsf{Anc}_{q,r} \wedge r < \min(d_q, t^*)$. Switch the changed coordinates from $\mathcal{H}_r^-$ to $\mathcal{H}_r^+$ one at a time. The test for U runs $\mathcal{A}$, evaluates Φ, and computes $\mathsf{Eq}_{q,r}$ from its boundary view. It uses trapdoors only above $k = \nu_2(r)+1$. Every changed coordinate is at most k, so the two-stage CRS distinguisher can run this entire test. Index hiding gives $\Pr_{\mathcal{H}_r^+}[U] \ge \Pr_{\mathcal{H}_r^-}[U] - 2\ell_{\max}\mu$.

In $\mathcal{H}_r^+$, select the frontier root containing $[r, r+1]$ from the target and run the full extraction prescribed by the bits of r. This root selector uses only public target data and the records of its simulation calls. Lemma E.4 with $h = 0$, $\ell = \ell_{\max}$, and its output root level J bounds extraction failure by $\ell_{\max}\mu$. Intersecting with U preserves this bound. On success, the checked path and boundary equality give $\mathsf{ct}_L(N_0) = \mathsf{ct}_r^q$ for the extracted leaf.

Apply the first bound of Lemma E.8 to the procedure that runs $\mathcal{A}$, checks U, and returns this extracted leaf only when its message is fresh. It has input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>0})$ and accesses only $\mathcal{O}_{\mathrm{rec}}$. Hence the probability that its NIZK witness extraction fails the $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$ check is at most μ.
For valid fresh extractions, define a second unexplainability adversary with the same challenge substitution at (q, r) as above, now in mode $\mathcal{H}_r^+$. It generates joint NIZK trapdoors, evaluates U, and extracts the next leaf. On a fresh message, it applies $\mathsf{NIZK.E}$ and checks its witness against $R_{0,\mathsf{pk},\mathsf{crs}_{\mathsf{SNARG},V}}$. A valid witness contains an opening of the leaf's left ciphertext $\mathsf{ct}_r^q = \mathsf{ct}^\star$; output that opening, and abort otherwise. The fresh case thus has probability at most 2μ. On U, the substituted boundary is interior to query q. Thus the challenge substitution has the same joint distribution as the ordinary local encryption, including all public parameters, records, trapdoors, and NIZK extraction randomness used in this test.
For a stale next leaf, run Bind on the matching log entries. Lemma E.5 bounds a returned collision by μ. Otherwise any matching entry (q', j) has $j = r$, $\mathsf{ct}_0^{q'} = \mathsf{ct}_0^*$, and a matching prefix through $r+1$. The equality at r already gives $\mathsf{ct}_0^* = \mathsf{ct}_0^q$. Lemma E.9 bounds distinct occurrences with this common starting ciphertext by $\widetilde{N}\mu$. Outside that event $q' = q$, and the shared-boundary construction gives $\mathsf{Eq}_{q,r+1} = 1$ in $\mathcal{H}_r^+ = \mathcal{H}_{r+1}^-$. The target and log are unchanged by extraction and binding, so Φ remains true. Thus $\mathsf{Anc}_{q,r+1}$ holds. Adding the switch, extraction, fresh, CR, and CPA losses proves (17).

Lemma E.11 (Completion of a queried prefix) *Fix $q \in \{1, \ldots, \widetilde{N}\}$, $1 \le r_0 \le T$, and either case condition Φ above, all before setup. For every PPT query adversary $\mathcal{A}$, there exists $d^* \in \{r_0, \ldots, T\}$, depending only on $\mathcal{A}$ and the fixed experiment parameters and independent of the experiment's randomness, such that*

$$\Pr_{\mathcal{H}_{d^*}^-}[\mathsf{Anc}_{q,d^*} \wedge d_q = d^*] \ge \frac{\Pr_{\mathcal{H}_{r_0}^-}[\mathsf{Anc}_{q,r_0}]}{T} - (3\ell_{\max} + \widetilde{N} + 4)\mu. \tag{18}$$

On this event, $x_{\mathsf{in}}^* = x_0^q$, $d_q \le t^*$, and the target boundary ciphertext at d_q equals $\mathsf{ct}_{d_q}^q$.

Proof. For each r, the event $\mathsf{Anc}_{q,r}$ implies $r \le d_q$ and $r \le t^*$. We partition this event according to the three disjoint cases $d_q = r$, $t^* = r < d_q$, and $r < \min(d_q, t^*)$. The first part of Lemma E.10 bounds the probability of the second case by μ. Applying its second part to the third case, and using $\mathcal{H}_r^+ = \mathcal{H}_{r+1}^-$, gives, for $r < T$,

$$\Pr_{\mathcal{H}_{r+1}^-}[\mathsf{Anc}_{q,r+1}] \ge \Pr_{\mathcal{H}_r^-}[\mathsf{Anc}_{q,r}] - \Pr_{\mathcal{H}_r^-}[\mathsf{Anc}_{q,r} \wedge d_q = r] - (3\ell_{\max} + \widetilde{N} + 4)\mu. \tag{19}$$

Sum (19) over $r = r_0, \ldots, T-1$. For every $r_0 < r < T$, the term $\Pr_{\mathcal{H}_r^-}[\mathsf{Anc}_{q,r}]$ appears on both sides. Moreover, $\mathsf{Anc}_{q,T}$ implies $d_q = T$, since every query has depth at most T. Rearranging therefore gives

$$\sum_{d=r_0}^{T} \Pr_{\mathcal{H}_d^-}[\mathsf{Anc}_{q,d} \wedge d_q = d] \ge \Pr_{\mathcal{H}_{r_0}^-}[\mathsf{Anc}_{q,r_0}] - (T - r_0)(3\ell_{\max} + \widetilde{N} + 4)\mu. \tag{20}$$

Choose $d^* \in \{r_0, \ldots, T\}$ maximizing the summands on the left-hand side. There are at most T summands, so (20) implies (18). This choice depends only on $\mathcal{A}$ and the fixed experiment parameters.
On $\mathsf{Anc}_{q,d^*} \wedge d_q = d^*$, the equality test Eq_{q,d^*} accepts. By its definition, $d_q = d^* \le t^*$ and the target's canonical prefix through d^* equals $\mathsf{Pref}^q(d^*)$. In particular, their starting ciphertexts agree, and the target boundary ciphertext at d_q is $\mathsf{ct}_{d_q}^q$. The valid source openings and perfect correctness then give $x_{\mathsf{in}}^* = x_0^q$.

The second descent, in the false suffix

Lemma E.12 (Suffix cover and its bad block) *Fix $q \in \{1, \ldots, \widetilde{N}\}$ and $1 \le d \le T$. For every execution of $\mathcal{H}_d^-$ satisfying $\mathsf{Anc}_{q,d} \wedge d_q = d < t^*$, its boundary view determines a consecutive dyadic cover of $[d, t^*]$ as follows: the right-child descriptors at left turns of the ending-leaf path, ordered by increasing level, followed by the later frontier roots in their target order. Every required parent occurs in the boundary view. Adjacent ciphertexts agree, the first endpoint is ct_d^q, and the last endpoint is the target's final ciphertext. There are at most $2\ell_{\max} + 1$ candidates, uniquely indexed by kind (right sibling or frontier root) and level. If $x_d^q \overset{t^*}{\not\rightsquigarrow}{}^{d} x_{\mathsf{out}}^*$, at least one candidate has invalid endpoint decryption or decrypted endpoints failing reachability in its stated dyadic length.*

Proof. If d is a frontier boundary, use the later frontier roots. Otherwise set $k = \nu_2(d) + 1$ and let N_k be the common parent in the boundary view. Its right-child interval begins at d. Append that descriptor, followed by the right-child descriptors at left turns above k, deepest first, and then later frontier roots. Below k the ending-leaf path has only right turns. Thus this list is exactly the stated cover. At every parent, its two child intervals are adjacent and cover its interval; induction up the partial path proves that the listed intervals concatenate from d to the selected root's right endpoint. The outer frontier continues from there to t^*. Checked child triples and target adjacency checks identify every shared ciphertext. The last descriptor of $\mathsf{Pref}^q(d)$ supplies ct_d^q at the initial boundary. There is at most one right sibling at each level below $\ell_{\max}$ and one frontier root at each level at most $\ell_{\max}$. If every candidate had valid decryptions and a genuine computation, their concatenation, using the query's final opening and the target's final opening, would give $x_d^q \overset{t^*-d}{\rightsquigarrow} x^*_{\mathsf{out}}$ by perfect correctness. This contradicts the hypothesis.

Lemma E.13 (Soundness beyond a completed query) *Fix $q \in \{1, \ldots, \widetilde{N}\}$, $d^* \in \{1, \ldots, T\}$, and $(a, z, s) \in \{0,1\}^{2n} \times \{0, \ldots, T\}$ with a $\overset{s}{\nrightarrow} z$, all before setup. Let Φ be the second case condition above for this q and (a, z, s). For every PPT query adversary $\mathcal{A}$, put $\gamma := \Pr_{\mathcal{H}_{d^*}^-}[\mathsf{Anc}_{q,d^*} \wedge d_q = d^*]$. Then*

$$\frac{\gamma}{2\ell_{\max} + 1} \leq T(7 \cdot 2^{2n} + \widetilde{N} + 11)\mu. \tag{21}$$

Proof. The bound is immediate if $\gamma = 0$; assume $\gamma > 0$. Write $d = d^*$. On the completed event, if $d = t^*$, the matched final ciphertext and perfect correctness give $x_d^q = x^*_{\mathsf{out}}$. This contradicts the fixed false triple with $s = 0$. Hence $d < t^*$. By Lemma E.12, on the completed event at least one block in the suffix cover is bad. This cover contains at most one right sibling at each level $0, \ldots, \ell_{\max} - 1$ along the extracted path, and at most one later frontier root at each level $0, \ldots, \ell_{\max}$. There are therefore at most $2\ell_{\max} + 1$ choices of block type and level. Fix one such choice, at level ℓ, for which the completed event occurs and the corresponding block is bad with probability at least $\gamma/(2\ell_{\max} + 1)$.

The completed event lower-bounds the success probability of the following selector, with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell+1})$ and oracle $\mathcal{O}_{\mathrm{rec}}$. Run $\mathcal{A}$ and check target acceptance, existence of q, $d_q = d$, and Φ. For a frontier-root index, return that root if its interval lies in $[d, t^*]$, and return $\perp$ otherwise. For a right-sibling index, let N_J be the frontier root containing $[d-1, d]$. Abort unless $J \geq \ell + 1$ and $\mathrm{bit}_\ell(d-1) = 0$; these conditions specify the indexed left turn. Set $\rho = (\rho_J, \ldots, \rho_{\ell+2})$ with $\iota(\rho_i) = 1 + \mathrm{bit}_{i-1}(d-1)$. Abort if the call $\mathsf{Open}_\rho(N_J)$ fails, or the resulting right-child interval is outside $[d, t^*]$. Return the parent and its right-child descriptor. This selector uses only $\boldsymbol{\tau}_{>\ell+1}$. On the completed event its transcript is a restriction of the boundary view, so determinism makes all these extractions succeed. Thus the selector's unconditional bad-descriptor probability is at least $\gamma/(2\ell_{\max} + 1)$.

A selected frontier root already has an accepting proof. For a right sibling, its parent designation in $\mathcal{H}_d^-$ is left. Apply the invalid-endpoint and fixed-pair CRS tests from Lemma E.7 to its right-half descriptor and switch that designation to right. These tests run the selector with only trapdoors above the challenged parent level. Their total loss is at most $2(2^{2n} + 1)\mu$. Extract the right-child witness and check its recursive relation; Lemma E.4 bounds failure by μ. This yields an algorithm with input $(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$, access to $\mathcal{O}_{\mathrm{rec}}$, and accepting bad suffix-node probability p satisfying $p \geq \gamma/(2\ell_{\max} + 1) - (2 \cdot 2^{2n} + 3)\mu$. It returns the same target and checked transcript and enforces $d_q = d$, Φ, and interval containment in $[d, t^*]$.

Lemma E.7 gives a fixed final experiment in which this procedure returns an accepting bad leaf with probability at least $p/2^\ell - (4 \cdot 2^{2n} + 6)\mu$. Every returned leaf $[r, r+1]$ has $r \geq d = d_q$ and $\mathsf{ct}_0^* = \mathsf{ct}_0^q$. In this experiment, accepting bad outputs with fresh messages have probability at most $(2^{2n} + 1)\mu$ by Lemma E.8. For a stale message, run Bind on the matching records. A returned collision has probability at most μ. Otherwise a matching occurrence q' satisfies $r < d_{q'}$ and $\mathsf{ct}_0^{q'} = \mathsf{ct}_0^* = \mathsf{ct}_0^q$. Since $r \geq d_q$, we have $q' \neq q$. The equality $\mathsf{ct}_0^{q'} = \mathsf{ct}_0^q$ is a collision in $\mathcal{A}$'s query log in the current setup experiment. Lemma E.9 bounds this event by $\widetilde{N}\mu$. Hence the probability of returning an accepting bad leaf in this experiment is at most $(2^{2n} + \widetilde{N} + 2)\mu$. Combining the lower and upper bounds gives $p \leq 2^\ell(5 \cdot 2^{2n} + \widetilde{N} + 8)\mu$. The selected interval lies within a target of depth at most T, so $2^\ell \leq T$ whenever it is returned. Therefore $\gamma/(2\ell_{\max}+1) \leq (2 \cdot 2^{2n}+3)\mu + T(5 \cdot 2^{2n} + \widetilde{N} + 8)\mu$, which implies (21).

Proof of w-SIM-SND

Proof. Fix a PPT query adversary $\mathcal{A}$ with winning probability ε in $\mathcal{H}_{(\bot,\dots,\bot)}$. We may assume $\varepsilon > 0$, since otherwise the bound is immediate.

Case selection and the initial block. Partition winning executions as follows. If ct_0^* equals no query start, assign the execution to the no-match class. Otherwise let q be the first matching occurrence. Its source is x_{in}^* by the valid openings and perfect correctness. If $d_q > t^*$, assign the execution to class q with this inequality. If $d_q < t^*$, non-compliance gives $x_{d_q}^q \overset{t^*-d_q}{\rightsquigarrow} x_{\mathsf{out}}^*$. If $d_q = t^*$, the requirement that the target statement was not queried gives $x_{d_q}^q \neq x_{\mathsf{out}}^*$, since the source states already agree. This is the same false-suffix condition at length 0. In either case, assign the execution to the class indexed by q and the false triple $(a, z, s) = (x_{d_q}^q, x_{\mathsf{out}}^*, t^* - d_q)$. For the no-match class, define Φ to test that every query start differs from ct_0^*. This class is resolved by the first descent and does not invoke the completion lemmas. For the two matching classes, Φ is respectively the checks $\mathsf{ct}_0^* = \mathsf{ct}_0^q$ and $d_q > t^*$; or the checks $\mathsf{ct}_0^* = \mathsf{ct}_0^q$, $d_q \le t^*$, and literal equality to the fixed triple (a, z, s). Each check is a PPT function of the target and completed log. The fixed false fact $a \overset{s}{\rightsquigarrow} z$ is used only in the analysis. The check need not test that q is the first match: every execution assigned to its class still satisfies it. There are at most $1 + \widetilde{N} + \widetilde{N}2^{2n}(T+1)$ classes. Lemma E.6 implies that every winning execution has a bad frontier node at one of $\ell_{\max}+1$ levels. Put $B := 2^{2n}$ and $K := (\ell_{\max}+1)(1+\widetilde{N}+\widetilde{N}B(T+1))$. By averaging over the classes and levels, fix one class and level ℓ whose intersection with an accepting bad frontier node has probability at least ε/K. Define $\mathcal{G}^{\mathcal{O}_{\mathrm{rec}}}(\widetilde{\mathsf{crs}}, \boldsymbol{\tau}_{>\ell})$ to run $\mathcal{A}(\widetilde{\mathsf{crs}})$, forwarding only the proof answers. It returns $\bot$ unless the target accepts, Φ holds, and the frontier has a root N_ℓ at level ℓ. Otherwise it returns $(N_\ell, x_{\mathsf{in}}^*, x_{\mathsf{out}}^*, t^*, \Pi^*, (N_\ell))$. This selection uses only public parameters, the target, and the query records, and satisfies the interface in Lemma E.7. Every assigned winning execution with the selected bad root is in its accepting bad-node event.

The first false leaf and its query. Lemma E.7 fixes a path ρ and an experiment whose accepting bad-leaf probability is at least $\varepsilon/(KT) - (4B+6)\mu$; the selected block has length $2^\ell \le T$. The procedure retains target acceptance, Φ, and a checked transcript. Accepting bad outputs with fresh messages have probability at most $(B+1)\mu$ by Lemma E.8. For an accepting bad stale output, run Bind against the matching records. A returned collision has probability at most μ by Lemma E.5.
In the no-match class, a matching record returning match gives $\mathsf{ct}_0^* = \mathsf{ct}_0^{q'}$ for some query q', contradicting Φ. Thus this class gives $\varepsilon/(KT) \le (5B+8)\mu$. In either matching class, every matching record returning match has start $\mathsf{ct}_0^{q'} = \mathsf{ct}_0^* = \mathsf{ct}_0^q$. Lemma E.9 bounds $q' \neq q$ by $\widetilde{N}\mu$. Let β denote the probability of an accepting bad stale leaf for which binding succeeds and every matching occurrence is q. Then $\beta \ge \varepsilon/(KT) - (5B + \widetilde{N} + 8)\mu$. If $\beta = 0$, this already yields a bound using only the first term of (22); assume $\beta > 0$ below.
Among at most T leaf positions, fix s^* with probability at least β/T of this event, and put $r_0 = s^* + 1$. Consider instead the efficiently checked event that the target and transcript verify, Φ holds, the leaf position is s^*, and a matching record of q returns match. Its probability is at least β/T, and it gives equality of the prefixes through r_0. On this event, alignment of the frontier root to its dyadic length forces $\iota(\rho_k) = 1 + \mathrm{bit}_{k-1}(s^*)$ for every $1 \le k \le \ell$. The current designations already match $\mathcal{H}_{r_0}^-$ at these levels. Switch the ordinary levels $k > \ell$ to its remaining designations, one level at a time. The test runs the root procedure, its extraction at levels at most ℓ, the public path and position checks, and binding. At each challenged level $k > \ell$, the trapdoors used for this test belong to levels $1, \dots, \ell$ and were generated before the level-k challenge. The root selector uses public inputs. The two-stage CRS distinguisher therefore implements the test, with loss at most μ per switch. Its successful checked path implies $\mathsf{Eq}_{q,r_0} = 1$. Writing $\alpha := \Pr_{\mathcal{H}_{r_0}^-}[\mathsf{Anc}_{q,r_0}]$, we obtain $\alpha \ge \beta/T - \ell_{\max}\mu$.

Completion and the false suffix. Lemma E.11 fixes d^* and a completion probability γ satisfying $\gamma \ge \alpha/T - (3\ell_{\max} + \widetilde{N} + 4)\mu$. In the longer-query class, completion entails $d_q \le t^*$ and hence contradicts Φ; therefore $\gamma = 0$. In the remaining class, Φ implies the fixed false suffix triple and all hypotheses of Lemma E.13. It follows that $\gamma \le (2\ell_{\max}+1)T(7B + \widetilde{N} + 11)\mu$.

Quantitative bound. Combining the bounds on β, α, γ gives

$$\begin{aligned}\varepsilon \le K\big[&T(5B + \widetilde{N} + 8) + T^2\ell_{\max} + T^3(3\ell_{\max} + \widetilde{N} + 4) \\ &+ (2\ell_{\max}+1)T^4(7B + \widetilde{N} + 11)\big]\mu.\end{aligned} \tag{22}$$

The no-match class satisfies this bound using only its first term, and the longer-query class satisfies it using only its first three terms. Thus the bound holds for every possible selected class. Since $T, \widetilde{N}, B \geq 1$, we have $K \leq 4(\ell_{\max}+1)\widetilde{N}BT$, $5B + \widetilde{N} + 8 \leq 14B\widetilde{N}$, $3\ell_{\max} + \widetilde{N} + 4 \leq 8(\ell_{\max}+1)\widetilde{N}$, and $7B + \widetilde{N} + 11 \leq 19B\widetilde{N}$. Substituting these inequalities into (22), and recalling that $B = 2^{2n}$ and $\mu = 2^{-\kappa^\delta}$, gives

$$\varepsilon \leq 256(\ell_{\max}+1)^2 \widetilde{N}^2 T^5\, 2^{4n-\kappa^\delta}. \tag{23}$$

Reduction resources and parameter choice. Let τ bound the running time of $\mathcal{A}$, including writing its queries, and let D be the sum of its valid query depths. Since query depths are given in unary, $D \leq \tau$. Each reduction runs $\mathcal{A}$ once, generates the remaining public parameters, simulates at most D computation steps, and performs $O(\ell_{\max}+1)$ BARG extractions and at most one NIZK extraction. The simulation records contain $O(D)$ nodes; maintaining these records and comparing their descriptors takes polynomial time in D and the node size. By the efficiency hypothesis of Theorem 6.1, the total running time is $\mathsf{poly}(\tau, \kappa, n, m, \ell_{\max}, |C|)$. The selected levels, query ordinals, positions, and states require only polynomial advice length. In particular, each reduction used in the telescoping argument works with a single boundary, represented using $O(\log T)$ bits. Thus the reductions are polynomial-size adversaries for the underlying primitives, and their assumed security bounds apply for all sufficiently large λ.

Since $\ell_{\max} \leq \lambda$, $T \leq 2^\lambda$, and $\widetilde{N}$ is polynomially bounded, (23) gives $\varepsilon \leq \mathsf{poly}(\lambda)\, 2^{4n+5\lambda-\kappa^\delta}$. Our parameter choice gives $\kappa^\delta \geq (3n+2\lambda)^{\delta/\delta'}$, where $\delta/\delta' > 1$. Hence $\kappa^\delta \geq 4n + 7\lambda$ for all sufficiently large λ, and therefore $\varepsilon \leq \mathsf{poly}(\lambda)\, 2^{-2\lambda} \leq 2^{-\lambda}$. This proves weak simulation soundness. The simulator's running time on $(u, v, 1^d)$ is $d \cdot \mathsf{poly}(\kappa, \lambda, n, m, \ell_{\max}, |C|)$, which is polynomial in its input length.

F Additional Formal Treatment of Mergeability

F.1 ZK simulators and hard reachability problems

We state this simple fact on zero-knowledge simulators in the context of hard reachability problems. We will use this in our attacks.

Theorem F.1. *Let $(\mathcal{L}, \overline{\mathcal{L}}, \mathsf{Sam})$ be a hard reachability problem on C, let IVC be a zero-knowledge IVC scheme with simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$, let d and $T \geq d$ be polynomially bounded, and write n, m for the state and witness lengths of C_λ. Consider the two experiments*

$\mathsf{Hon}_d(1^\lambda)$:
- $\big((x, x'), \boldsymbol{w}\big) \leftarrow \mathsf{Sam}(1^\lambda, 1^d)$
- $\mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C_\lambda)$
- $\pi \leftarrow \mathsf{IVC.P}^d(\mathsf{crs}, x, \boldsymbol{w})$
- *return* $\big(\mathsf{crs}, (x, x'), \pi\big)$

$\mathsf{Sml}_d(1^\lambda)$:
- $(\bar{x}, \bar{x}') \xleftarrow{\$} \overline{\mathcal{L}}_{\lambda,d}$
- $(\mathsf{tdcrs}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, T, n, m, C_\lambda)$
- $\pi \leftarrow \mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{st}, \bar{x}, \bar{x}', d)$
- *return* $\big(\mathsf{tdcrs}, (\bar{x}, \bar{x}'), \pi\big)$

Then:

1. *the output distributions of Hon_d and Sml_d are computationally indistinguishable;*
2. *in particular the verifier accepts the simulated proof of the false instance:*

$$\Pr\Big[\mathsf{IVC.V}\big(\mathsf{tdcrs}, (\bar{x}, \bar{x}', d), \pi\big) = 1 : \big(\mathsf{tdcrs}, (\bar{x}, \bar{x}'), \pi\big) \leftarrow \mathsf{Sml}_d\Big] \geq 1 - \mathsf{negl}(\lambda),$$

although $(\bar{x}, \bar{x}', d) \notin \mathcal{L}_{C_\lambda}$ except with negligible probability.

Proof. For item 1 consider the intermediate experiment Hyb_d that samples $\big((x, x'), \boldsymbol{w}\big) \leftarrow \mathsf{Sam}(1^\lambda, 1^d)$, ignores $\boldsymbol{w}$, and then proceeds as Sml_d on the instance (x, x'): $(\mathsf{tdcrs}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, T, n, m, C_\lambda)$, $\pi \leftarrow \mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{st}, x, x', d)$, return $(\mathsf{tdcrs}, (x, x'), \pi)$. Let D be any PPT distinguisher of triples.

Hon_d versus Hyb_d: zero knowledge. Let A be the zero-knowledge adversary that, given a reference string σ and an oracle, samples $\big((x, x'), \boldsymbol{w}\big) \leftarrow \mathsf{Sam}(1^\lambda, 1^d)$, aborts with output 0 if $\boldsymbol{w} \notin \mathcal{W}(x \overset{d}{\rightsquigarrow} x')$, and otherwise queries the oracle d times along the chain induced by $\boldsymbol{w}$, on $\big((x, x_{i-1}, i-1), \pi_{i-1}, w_{i-1}\big)$ for $i \in [d]$, and

outputs $D\big(\sigma,(x,x'),\pi_d\big)$. Since $\boldsymbol{w}$ is a valid witness vector, every statement A queries is true, so neither oracle answers $\perp$, and the simulation oracle answers the last query with $\mathsf{Sim}_2(\mathsf{tdcrs},\mathsf{st},x,x',d)$. Against the honest prover A therefore outputs D applied to the output of Hon_d, and against the simulation oracle D applied to the output of Hyb_d, in both cases up to the abort, whose probability is negligible by the second clause of Definition 5.4. Zero knowledge bounds the difference of the two output probabilities of A, hence of D on the two experiments, by $\mathsf{negl}(\lambda)$.

Hyb_d *versus* Sml_d*: indistinguishability of the instances.* Let D' be the algorithm that, on input a pair of states (y,y'), runs $(\mathsf{tdcrs},\mathsf{st})\leftarrow\mathsf{Sim}_1(1^\lambda,T,n,m,C_\lambda)$ and $\pi\leftarrow\mathsf{Sim}_2(\mathsf{tdcrs},\mathsf{st},y,y',d)$ and outputs $D\big(\mathsf{tdcrs},(y,y'),\pi\big)$. D' is PPT, since the simulator takes only public inputs and receives no witness. On $(y,y')\xleftarrow{\$}\mathcal{L}_{\lambda,d}$, the distribution of the instance output by Sam, the output of D' is that of D on Hyb_d; on $(y,y')\xleftarrow{\$}\overline{\mathcal{L}}_{\lambda,d}$ it is that of D on Sml_d. The indistinguishability of $\mathcal{L}_{\lambda,d}$ and $\overline{\mathcal{L}}_{\lambda,d}$ at the depth d fixed above bounds the difference by $\mathsf{negl}(\lambda)$, and item 1 follows.

For item 2, apply item 1 to the distinguisher $D\big(\sigma,(x,x'),\pi\big):=\mathsf{IVC.V}\big(\sigma,(x,x',d),\pi\big)$. On the output of Hon_d it returns 1 except with negligible probability, because π is an honest proof of the statement (x,x',d), which is true whenever $\boldsymbol{w}$ is a valid witness vector, and completeness then applies; so it returns 1 on the output of Sml_d except with negligible probability as well. Falsity of $(\bar{x},\bar{x}',d)$ except with negligible probability is the first clause of Definition 5.4.

F.2 Proof of Lemma 5.7

Proof. Items 1 and 2 read off the definition of C_G. For (a), idling at $(x,0)$ uses item 2, and from $(x,-\ell)$ the counter runs through the nonzero values $-\ell,\dots,-1$ by item 1 and reaches 0 after exactly ℓ steps. For (b), the first claim is item 1 applied ℓ times, since the counter is nonzero throughout. For the second, from (x,c) with $c\neq 0$ the counter takes the values $c+1,\dots,2^k-1,0$ during the first 2^k-c steps, all different from c; once at 0 it either stays there or moves to 1 and then climbs, so returning to c takes at least $(2^k-c)+1+(c-1)=2^k>\ell$ steps. For (c), the first $\ell-1$ witnesses of $\boldsymbol{w}_s^{(\ell)}$ keep the chain at $(x,0)$ by item 2, and the last one moves it to $(x,1)$ by item 2 again, s being a preimage of x; if instead x is outside the range of G, item 2 makes $(x,0)$ absorbing, so no chain out of it leaves the state.

F.3 Proof of Lemma 5.8

Proof. The instance output by Sam is distributed as $\mathcal{L}_{\lambda,d}$, and its witness vector is valid with probability 1 by Lemma 5.7(c). A pair from $\overline{\mathcal{L}}_{\lambda,d}$ has an x outside the range of G except with probability $2^{-\lambda}$, and for such an x the second part of Lemma 5.7(c) gives $(x,0)\overset{d'}{\not\rightsquigarrow}(x,1)$ for every $d'\geq 1$. For indistinguishability, the map $y\mapsto\big((y,0),(y,1)\big)$ is efficiently computable, so any distinguisher between $\mathcal{L}_{\lambda,d}$ and $\overline{\mathcal{L}}_{\lambda,d}$ yields one between $G(s)$ for $s\xleftarrow{\$}\{0,1\}^\lambda$ and a uniform $y\in\{0,1\}^{2\lambda}$ with the same advantage, which pseudorandomness bounds by $\mathsf{negl}(\lambda)$. Finally every instance of either ensemble has first coordinate in $S^{\mathsf{init}}_\lambda$ and second coordinate $(x,1)\in S^{\mathsf{fin}}_\lambda$, so the domain clause holds with probability 1.

F.4 Proof of Theorem 5.9

Proof. By [HILL99] the assumption yields a pseudorandom generator $G:\{0,1\}^\lambda\to\{0,1\}^{2\lambda}$. Take $C:=C_G$ of Definition 5.6 with $k:=\lambda+1$, a family of circuits of size $\mathsf{poly}(\lambda)$ on $n=3\lambda+1$ bit states, and take $S^{\mathsf{init}},S^{\mathsf{fin}}$ and the problem $(\mathcal{L},\overline{\mathcal{L}},\mathsf{Sam})$ of Lemma 5.8, which is a hard reachability problem on C with domain $(S^{\mathsf{init}},S^{\mathsf{fin}})$. It remains to exhibit the structure of Definition 5.5. Set

$$\mathsf{Loop}\big(1^\lambda,(x,0),1^\ell\big):=w_\perp^\ell,\qquad \mathsf{Pre}\big(1^\lambda,(x,0),1^\ell\big):=\big((x,-\ell),\,w_\perp^\ell\big);$$

both are deterministic and run in time polynomial in λ and ℓ, and every polynomially bounded ℓ satisfies $\ell<2^k=2^{\lambda+1}$, so Lemma 5.7 applies to it. Easy loops on S^{init} are Lemma 5.7(a). For easy prefixes on S^{init}, we have $(x,-\ell)\neq(x,0)$ because $0<\ell<2^k$, the witness vector is valid by (a), and the funnel condition follows from (b): a chain of $\ell+d'$ steps from $(x,-\ell)$ is at $(x,0)$ after ℓ steps, and its last d' steps form a chain of d' steps from $(x,0)$ to its endpoint. Loop-freeness on S^{fin} is the second part of (b), every state of S^{fin}_λ having a nonzero counter.

F.5 Proof of Theorem 5.10

Proof. Let IVC be zero-knowledge with simulator Sim and universally (d_L, d_R)-mergeable with algorithms $\mathsf{GetHint}$ and Merge, for some $d_L, d_R \geq 1$. By Theorem 5.9, one-way functions give a polynomial-size family of step circuits C with easy prefixes on S^{init}, witnessed by an algorithm Pre, that carries a hard reachability problem with domain $(S^{\mathsf{init}}, S^{\mathsf{fin}})$ $(\mathcal{L}, \overline{\mathcal{L}}, \mathsf{Sam})$; Theorem F.1 applies to it. Fix λ, set $T := d_L + d_R$, and write n, m for the state and witness lengths of C_λ, which are polynomially bounded. We exhibit a PPT adversary $\mathcal{A}$ that makes a single query to Sim' and wins the $\mathsf{newrt\text{-}SIM\text{-}SND}$ experiment of Definition 4.4 for C_λ, T, n, m with probability at least $1 - \mathsf{negl}(\lambda)$. Since the definition requires this probability to be negligible for every choice of these parameters and every PPT adversary, IVC is not $\mathsf{newrt\text{-}SIM\text{-}SND}$ with respect to Sim. On input tdcrs, where $(\mathsf{tdcrs}, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda, T, n, m, C_\lambda)$, the adversary $\mathcal{A}$ proceeds as follows.

1. Sample $(\bar{x}, \bar{x}') \xleftarrow{\$} \overline{\mathcal{L}}_{\lambda, d_R}$ and compute $(\alpha, \boldsymbol{w}_L) := \mathsf{Pre}(1^\lambda, \bar{x}, 1^{d_L})$. In the roles of $x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}$ take $\alpha, \bar{x}, \bar{x}'$, so that $\mathsf{s}_L = (\alpha, \bar{x}, d_L)$, $\mathsf{s}_R = (\bar{x}, \bar{x}', d_R)$ and $\mathsf{s}^* = (\alpha, \bar{x}', d_L + d_R)$.
2. Query Sim' on the leap $\big[\bar{x} \overset{d_R}{\underset{?}{\rightsquigarrow}} \bar{x}'\big]$ and receive π_R.
3. Compute $r_L \leftarrow \mathsf{GetHint}(\mathsf{tdcrs}, \mathsf{s}_L, \mathsf{s}_R, \pi_R)$.
4. Compute $\pi_L := \mathsf{IVC.P}^{d_L}(\mathsf{tdcrs}, \alpha, \boldsymbol{w}_L; r_L)$.
5. Compute $\pi^* \leftarrow \mathsf{Merge}\big(\mathsf{tdcrs}, (\alpha, \bar{x}, \bar{x}'), (d_L, d_R), \pi_L, \pi_R\big)$.
6. Output the target $x^* := \mathsf{s}^*$ and the proof π^*.

$\mathcal{A}$ is uniform, runs in polynomial time and makes one oracle query. By the domain clause, $\bar{x} \in S^{\mathsf{init}}_\lambda$ except with negligible probability, so $\alpha \neq \bar{x}$ and $\boldsymbol{w}_L \in \mathcal{W}(\alpha \overset{d_L}{\rightsquigarrow} \bar{x})$ by easy prefixes.
The proof that $\mathcal{A}$ outputs is accepting. Let $D(\sigma, (x, x'), \pi)$ be the algorithm that computes $(\alpha, \boldsymbol{w}_L) := \mathsf{Pre}(1^\lambda, x, 1^{d_L})$, runs items 3 to 5 of $\mathcal{A}$ with σ in place of tdcrs and π in place of π_R, and outputs $\mathsf{IVC.V}\big(\sigma, (\alpha, x', d_L + d_R), \pi^*\big)$; it is PPT. The run of $\mathcal{A}$ is D applied to the output of the experiment Sml_{d_R} of Theorem F.1, since the simulation oracle of Definition 4.4 answers the query with $\mathsf{Sim}_2(\mathsf{tdcrs}, \mathsf{st}, \bar{x}, \bar{x}', d_R)$ whether or not the leap is true. Consider D on the output of Hon_{d_R} instead: the instance (x, x') comes from Sam with a witness vector $\boldsymbol{w}_R \in \mathcal{W}(x \overset{d_R}{\rightsquigarrow} x')$, the reference string is drawn from $\mathsf{IVC.Gen}$, and π is the honest proof $\mathsf{IVC.P}^{d_R}(\mathsf{crs}, x, \boldsymbol{w}_R)$. With $x_{\mathsf{init}} := \alpha$, $x_{\mathsf{mid}} := x$, $x_{\mathsf{fin}} := x'$ and the vectors $\boldsymbol{w}_L$ from Pre and $\boldsymbol{w}_R$ from Sam, items 3 to 5 are the last three lines of $\mathsf{Exp}_{\mathrm{Mrg}}$ in Fig. 6, and $x \in S^{\mathsf{init}}_\lambda$ by the domain clause, except with negligible probability, so $\boldsymbol{w}_L$ is valid by easy prefixes. Universal (d_L, d_R)-mergeability, which quantifies over all states and all valid witness vectors, gives $\mathsf{Exp}_{\mathrm{Mrg}} \geq 1 - \mathsf{negl}(\lambda)$ for each such choice, so D outputs 1 on Hon_{d_R} except with negligible probability, after averaging over Sam and discarding the negligible event that $\boldsymbol{w}_R$ is invalid or that $x \notin S^{\mathsf{init}}_\lambda$. By Theorem F.1(1), D outputs 1 on Sml_{d_R} except with negligible probability as well, that is, $\mathsf{IVC.V}(\mathsf{tdcrs}, x^*, \pi^*) = 1$ except with negligible probability.
It remains to check the winning condition. Condition on $\bar{x} \in S^{\mathsf{init}}_\lambda$ and on $\bar{x} \overset{d_R}{\not\rightsquigarrow} \bar{x}'$, which fails with negligible probability by the domain clause and by the first clause of Definition 5.4 at $d' := d_R$. The target $x^* = (\alpha, \bar{x}', d_L + d_R)$ is false: if $\alpha \overset{d_L + d_R}{\rightsquigarrow} \bar{x}'$, the funnel condition of easy prefixes gives $\bar{x} \overset{d_R}{\rightsquigarrow} \bar{x}'$, which we excluded. The only query is $\big[\bar{x} \overset{d_R}{\underset{?}{\rightsquigarrow}} \bar{x}'\big]$, whose root $\bar{x}$ differs from the target root α, so no statement rooted at x^*_{in} was queried. Both conjuncts of $\mathsf{WinCond}^{\mathsf{newrt\text{-}SIM\text{-}SND}}(x^*, \pi^*, Q_S)$ from Fig. 5 hold, and together with the acceptance just shown and a union bound $\mathcal{A}$ wins with probability at least $1 - \mathsf{negl}(\lambda)$.

F.6 Proof of Theorem 5.11

Proof. Fix parameters λ, $T \geq 4$, n, m and a circuit C as in Definition 5.1, and let

$$\mathsf{crs} = \big(\ell_{\max},\, C,\, \mathsf{pk},\, \mathsf{crs}_{\mathsf{SNARG},P},\, \mathsf{crs}_{\mathsf{SNARG},V},\, \mathsf{crs}_{\mathsf{NIZK}},\, \{\mathsf{crs}_\ell\}_{\ell \in [\ell_{\max}]}\big)$$

be in the support of $\mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)$; since $\ell_{\max} = \lceil \log T \rceil \geq 2$, the string crs_2 is present. Fix states $x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}} \in \{0,1\}^n$ and witness vectors $\boldsymbol{w}_L \in \mathcal{W}(x_{\mathsf{init}} \overset{2}{\rightsquigarrow} x_{\mathsf{mid}})$ and $\boldsymbol{w}_R \in \mathcal{W}(x_{\mathsf{mid}} \overset{2}{\rightsquigarrow} x_{\mathsf{fin}})$. Nothing below depends on C, on the parameters, or on these choices.
We first record the shape of an honest two-step proof. Let $\boldsymbol{w} = (w_0, w_1) \in \mathcal{W}(a \overset{2}{\rightsquigarrow} b)$, let $a_0 = a, a_1, a_2 = b$ be the chain it induces, and let Π be the output of $\mathsf{IVC.P}^2(\mathsf{crs}, a, \boldsymbol{w}; r)$ for a tape r. Reading the prover of

Fig. 10: its first invocation samples r_0 in the base case and r_1 in step 5a, sets $\mathsf{ct}_j := \mathsf{PKE.Enc}(\mathsf{pk}, a_j; r_j)$ for $j \in \{0,1\}$, and puts $(0, \mathsf{ct}_0, \bot, \mathsf{ct}_1, \pi_1^{(0)})$ into $S^{(0)}$, where $\pi_1^{(0)}$ is a NIZK proof for $(\mathsf{ct}_0, \mathsf{ct}_1)$; the merge loop does nothing, $S^{(0)}$ having one entry. Its second invocation samples r_2 in step 5a, sets $\mathsf{ct}_2 := \mathsf{PKE.Enc}(\mathsf{pk}, a_2; r_2)$ and adds $(1, \mathsf{ct}_1, \bot, \mathsf{ct}_2, \pi_2^{(0)})$ to $S^{(0)}$. Now $|S^{(0)}| = 2$ and the two entries are adjacent, $\mathsf{ct}_{\mathsf{fin}} = \mathsf{ct}_1 = \widetilde{\mathsf{ct}}_{\mathsf{init}}$ and $i_{\mathsf{init}} + 2^0 = 1 = \widetilde{i}_{\mathsf{init}}$, so the loop empties $S^{(0)}$ and puts $(0, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \pi^{(1)})$ into $S^{(1)}$, with

$$\pi^{(1)} \leftarrow \mathsf{sweBARG.P}\big(\mathsf{crs}_1,\ \{(\mathsf{ct}_0, \mathsf{ct}_1), (\mathsf{ct}_1, \mathsf{ct}_2)\},\ \{\pi_1^{(0)}, \pi_2^{(0)}\}\big).$$

At level 1 the loop stops, one entry being present. Hence

$$\Pi = \big(\mathsf{ct}_0, \mathsf{ct}_2, r_0, r_2, \{S^{(\ell)}\}_\ell\big), \qquad S^{(1)} = \{(0, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \pi^{(1)})\},$$

and $S^{(\ell)} = \emptyset$ for every $\ell \neq 1$. In particular a two-step proof exposes its endpoint randomness r_0, r_2, its three ciphertexts and its level-1 proof.

We can now give the two algorithms. Both parse π_R as $\big(\mathsf{ct}_0^R, \mathsf{ct}_2^R, r_0^R, r_2^R, \{S_R^{(1)}\}\big)$ with $S_R^{(1)} = \{(0, \mathsf{ct}_0^R, \mathsf{ct}_1^R, \mathsf{ct}_2^R, \pi_R^{(1)})\}$, which the shape above justifies for the honest π_R of Fig. 6.

$\mathsf{GetHint}(\mathsf{crs}, \mathsf{s}_L, \mathsf{s}_R, \pi_R)$ outputs the tape that is uniform everywhere except on the segment read in step 5a by the second invocation of IVC.P, which it sets to r_0^R. That segment is determined by crs alone: the randomness complexity of each algorithm the prover calls is a fixed polynomial of the parameters, and the number of calls made by each invocation is determined by its index.

$\mathsf{Merge}(\mathsf{crs}, (x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}), (2,2), \pi_L, \pi_R)$ parses π_L as $\big(\mathsf{ct}_0^L, \mathsf{ct}_2^L, r_0^L, r_2^L, \{S_L^{(1)}\}\big)$ with $S_L^{(1)} = \{(0, \mathsf{ct}_0^L, \mathsf{ct}_1^L, \mathsf{ct}_2^L, \pi_L^{(1)})\}$, computes

$$\pi^{(2)} \leftarrow \mathsf{sweBARG.P}\Big(\mathsf{crs}_2,\ \big\{(\mathsf{ct}_0^L, \mathsf{ct}_2^L), (\mathsf{ct}_0^R, \mathsf{ct}_2^R)\big\},\ \big\{(\mathsf{ct}_1^L, \pi_L^{(1)}), (\mathsf{ct}_1^R, \pi_R^{(1)})\big\}\Big)$$

and outputs

$$\pi := \big(\mathsf{ct}_0^L, \mathsf{ct}_2^R, r_0^L, r_2^R, \{S^{(\ell)}\}_\ell\big), \qquad S^{(2)} := \{(0, \mathsf{ct}_0^L, \mathsf{ct}_2^L, \mathsf{ct}_2^R, \pi^{(2)})\},$$

with $S^{(\ell)} := \emptyset$ for every $\ell \neq 2$. Both algorithms run in polynomial time.

The junction ciphertexts agree. The second invocation of the prefix prover encrypts the last state of its chain, which is x_{mid}, with the randomness it reads in step 5a, which GetHint set to r_0^R; the base case of the suffix prover encrypts its initial state, also x_{mid}, with r_0^R. As $\mathsf{PKE.Enc}(\mathsf{pk}, \cdot\,; \cdot)$ is a function of the message and the randomness,

$$\mathsf{ct}_2^L = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{mid}}; r_0^R) = \mathsf{ct}_0^R.$$

The two level-1 proofs verify. The statements $(x_{\mathsf{init}}, x_{\mathsf{mid}}, 2)$ and $(x_{\mathsf{mid}}, x_{\mathsf{fin}}, 2)$ are true, being witnessed by $\boldsymbol{w}_L$ and $\boldsymbol{w}_R$. Completeness of the scheme (Definition 4.1) holds with probability 1 over the prover's coins, hence for every tape, in particular for the one GetHint outputs; so IVC.V accepts π_L on $(x_{\mathsf{init}}, x_{\mathsf{mid}}, 2)$ and π_R on $(x_{\mathsf{mid}}, x_{\mathsf{fin}}, 2)$. For $t = 2$ the only proof check the verifier performs is the one at level 1, whence

$$\mathsf{sweBARG.V}\big(\mathsf{crs}_1, \{(\mathsf{ct}_0^L, \mathsf{ct}_1^L), (\mathsf{ct}_1^L, \mathsf{ct}_2^L)\}, \pi_L^{(1)}\big) = 1,$$
$$\mathsf{sweBARG.V}\big(\mathsf{crs}_1, \{(\mathsf{ct}_0^R, \mathsf{ct}_1^R), (\mathsf{ct}_1^R, \mathsf{ct}_2^R)\}, \pi_R^{(1)}\big) = 1.$$

By the level-2 relation $R_{2,\mathsf{crs}_{2-1}}$ of Fig. 9, these two equalities say precisely that $(\mathsf{ct}_1^L, \pi_L^{(1)})$ witnesses $(\mathsf{ct}_0^L, \mathsf{ct}_2^L) \in L_{2,\mathsf{crs}_{2-1}}$ and that $(\mathsf{ct}_1^R, \pi_R^{(1)})$ witnesses $(\mathsf{ct}_0^R, \mathsf{ct}_2^R) \in L_{2,\mathsf{crs}_{2-1}}$, so the batch Merge hands to sweBARG.P is a true one with a valid witness.

Finally, run $\mathsf{IVC.V}\big(\mathsf{crs}, (x_{\mathsf{init}}, x_{\mathsf{fin}}, 4), \pi\big)$. The depth $t = 4$ has $s_2 = 1$ and $s_\ell = 0$ for $\ell \neq 2$, so the verifier finds $S^{(\ell)} = \emptyset$ at every level other than 2, as required, and at level 2 reads the single entry of $S^{(2)}$. Adjacency holds: $i_{\mathsf{curr}} = 0 = i_{\mathsf{init}}$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_0 = \mathsf{ct}_0^L = \mathsf{ct}_{\mathsf{init}}$; the verifier then sets $i_{\mathsf{curr}} = 4$ and $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_2^R$. The proof check at level 2 is

$$\mathsf{sweBARG.V}\big(\mathsf{crs}_2, \{(\mathsf{ct}_{\mathsf{init}}, \mathsf{ct}_{\mathsf{mid}}), (\mathsf{ct}_{\mathsf{mid}}, \mathsf{ct}_{\mathsf{fin}})\}, \pi^{(2)}\big)$$
$$= \mathsf{sweBARG.V}\big(\mathsf{crs}_2, \{(\mathsf{ct}_0^L, \mathsf{ct}_2^L), (\mathsf{ct}_2^L, \mathsf{ct}_2^R)\}, \pi^{(2)}\big),$$

and by the agreement of the junction ciphertexts the batch $\{(\mathsf{ct}_0^L, \mathsf{ct}_2^L), (\mathsf{ct}_2^L, \mathsf{ct}_2^R)\}$ is the one Merge proved, so the check passes by completeness of $\mathsf{sweBARG}$, its witnesses having just been shown valid. Finally $\mathsf{ct}_{\mathsf{curr}} = \mathsf{ct}_2^R = \mathsf{ct}_t$, while $\mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{init}}; r_0^L) = \mathsf{ct}_0^L$ by the base case of the prefix prover and $\mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{fin}}; r_2^R) = \mathsf{ct}_2^R$ by step 5a of the suffix prover's second invocation. The verifier outputs 1.

Everything above holds with probability 1 except the completeness of the level-2 $\mathsf{sweBARG}$ proof, so $\mathsf{Exp}_{\mathrm{Mrg}}(x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}}, \boldsymbol{w}_L, \boldsymbol{w}_R) \geq 1 - \mathsf{negl}(\lambda)$ for every choice of the quantified objects, which is Definition 5.1 for $d_L = d_R = 2$.

Remark F.2 (Universal mergeability for power-of-two depths) *For every $j \geq 1$ with $2^{j+1} \leq T$ the scheme of Fig. 10 is universally $(2^j, 2^j)$-mergeable, by easily extending the proof above. An honest proof for a depth that is a power of two carries a single entry, since the prover's merge loop empties every level below: for depth 2^j it is $S^{(j)} = \{(0, \mathsf{ct}_0, \mathsf{ct}_{2^{j-1}}, \mathsf{ct}_{2^j}, \pi^{(j)})\}$, with $S^{(\ell)} = \emptyset$ for $\ell \neq j$. $\mathsf{GetHint}$ is unchanged: it plants r_0^R in the segment that the last invocation of the prefix prover reads in step 5a, so that $\mathsf{ct}_{2^j}^L = \mathsf{ct}_0^R$. Merge makes one call to the level-$(j{+}1)$ batch prover, on the batch $\{(\mathsf{ct}_0^L, \mathsf{ct}_{2^j}^L), (\mathsf{ct}_0^R, \mathsf{ct}_{2^j}^R)\}$ with witnesses $(\mathsf{ct}_{2^{j-1}}^L, \pi_L^{(j)})$ and $(\mathsf{ct}_{2^{j-1}}^R, \pi_R^{(j)})$, and outputs the single level-$(j{+}1)$ entry $(0, \mathsf{ct}_0^L, \mathsf{ct}_{2^j}^L, \mathsf{ct}_{2^j}^R, \pi^{(j+1)})$ together with the endpoint fields $\mathsf{ct}_0^L, \mathsf{ct}_{2^j}^R, r_0^L, r_{2^j}^R$.*

F.7 Proof of Theorem 5.13

Proof. The proof follows that of Theorem 5.10 with the prefix chosen to return to its root; we indicate the changes. Let IVC be zero-knowledge with simulator Sim and XXY (d_L, d_R)-mergeable with algorithms $\mathsf{GetHint}$ and Merge, for some $d_L, d_R \geq 1$. Theorem 5.9 gives a polynomial-size family C with easy loops on S^{init}, witnessed by Loop, and loop-free on S^{fin}, carrying a hard reachability problem with domain $(S^{\mathsf{init}}, S^{\mathsf{fin}})$ $(\mathcal{L}, \overline{\mathcal{L}}, \mathsf{Sam})$ to which Theorem F.1 applies. Fix λ, set $T := d_L + d_R$, and write n, m for the state and witness lengths of C_λ. As before, it suffices to exhibit a PPT adversary that makes one query to Sim' and wins the $\mathsf{w\text{-}SIM\text{-}SND}$ experiment for C_λ, T, n, m with probability at least $1 - \mathsf{negl}(\lambda)$.

On input tdcrs, the adversary $\mathcal{A}$ samples $(\bar{x}, \bar{x}') \xleftarrow{\$} \overline{\mathcal{L}}_{\lambda, d_R}$, computes $\boldsymbol{w}_L := \mathsf{Loop}(1^\lambda, \bar{x}, 1^{d_L})$, and sets $x_{\mathsf{init}} = x_{\mathsf{mid}} := \bar{x}$ and $x_{\mathsf{fin}} := \bar{x}'$, so that $\mathsf{s}_L = (\bar{x}, \bar{x}, d_L)$, $\mathsf{s}_R = (\bar{x}, \bar{x}', d_R)$ and $\mathsf{s}^* = (\bar{x}, \bar{x}', d_L + d_R)$. It queries Sim' on $\left[\bar{x} \overset{d_R}{\underset{?}{\rightsquigarrow}} \bar{x}'\right]$ to receive π_R, computes $r_L \leftarrow \mathsf{GetHint}(\mathsf{tdcrs}, \mathsf{s}_L, \mathsf{s}_R, \pi_R)$, $\pi_L := \mathsf{IVC.P}^{d_L}(\mathsf{tdcrs}, \bar{x}, \boldsymbol{w}_L; r_L)$ and $\pi^* \leftarrow \mathsf{Merge}\big(\mathsf{tdcrs}, (\bar{x}, \bar{x}, \bar{x}'), (d_L, d_R), \pi_L, \pi_R\big)$, and outputs $x^* := \mathsf{s}^*$ with π^*. By the domain clause, $\bar{x} \in S_\lambda^{\mathsf{init}}$ except with negligible probability, and then $\boldsymbol{w}_L \in \mathcal{W}(\bar{x} \overset{d_L}{\rightsquigarrow} \bar{x})$ by easy loops.

The proof that $\mathcal{A}$ outputs is accepting. Let $D(\sigma, (x, x'), \pi)$ compute $\boldsymbol{w}_L := \mathsf{Loop}(1^\lambda, x, 1^{d_L})$, run the three lines above with σ and π in place of tdcrs and π_R, and output $\mathsf{IVC.V}\big(\sigma, (x, x', d_L + d_R), \pi^*\big)$; it is PPT. The run of $\mathcal{A}$ is D applied to the output of Sml_{d_R}. On the output of Hon_{d_R}, with $x_{\mathsf{init}} = x_{\mathsf{mid}} := x$, $x_{\mathsf{fin}} := x'$, $\boldsymbol{w}_L$ from Loop and $\boldsymbol{w}_R$ from Sam, the three lines are the last three lines of $\mathsf{Exp}_{\mathrm{Mrg}}$, and $\boldsymbol{w}_L$ is valid because $x \in S_\lambda^{\mathsf{init}}$, which the domain clause gives except with negligible probability. XXY (d_L, d_R)-mergeability gives that D outputs 1 on Hon_{d_R} except with negligible probability, and Theorem F.1(1) transfers this to Sml_{d_R}: $\mathsf{IVC.V}(\mathsf{tdcrs}, x^*, \pi^*) = 1$ except with negligible probability.

It remains to check the winning condition. Condition on $\bar{x} \in S_\lambda^{\mathsf{init}}$, $\bar{x}' \in S_\lambda^{\mathsf{fin}}$ and $\bar{x} \overset{d_L + d_R}{\not\rightsquigarrow} \bar{x}'$, which fails with negligible probability by the domain clause and by the first clause of Definition 5.4 at $d' := d_L + d_R$. The target $(\bar{x}, \bar{x}', d_L + d_R)$ is false by the last of these. For compliance, Q_S consists of the single pair $\left(\left[\bar{x} \overset{d_R}{\underset{?}{\rightsquigarrow}} \bar{x}'\right], \pi_R\right)$, so $P(x_{\mathrm{in}}^*) = P(\bar{x})$ contains the single leap $\left[\bar{x} \overset{d_R}{\underset{!}{\rightsquigarrow}} \bar{x}'\right]$; by Definition 4.3, x^* is compliant via $P(\bar{x})$ only if $d_R < d_L + d_R$, which holds, and $\bar{x}' \overset{d_L}{\rightsquigarrow} \bar{x}'$, which loop-freeness on S^{fin} excludes. Finally the only statement ever queried is $(\bar{x}, \bar{x}', d_R)$, which differs from x^* because $d_L \geq 1$, so $(x^*, _) \notin Q_S$. All three conjuncts of $\mathsf{WinCond}^{\mathsf{w\text{-}SIM\text{-}SND}}(x^*, \pi^*, Q_S)$ hold, and $\mathcal{A}$ wins with probability at least $1 - \mathsf{negl}(\lambda)$.

F.8 Proof of Theorem 5.14

Proof. We give $\mathsf{GetHint}$ and Merge for the pair $(2, 2)$; the pair $(1, 1)$ is the same argument one level down, with the level-1 BARG prover in place of the level-2 one and the two level-0 NIZK-SoK proofs, which a proof for a one-step statement exposes in $S^{(0)}$, as its witnesses. Fix crs in the support of $\mathsf{IVC.Gen}$, a closed chain

$x_{\mathsf{init}} = x_0, x_1, x_2 = x_{\mathsf{init}}$ with witnesses w_1, w_2, a state x_{fin}, and an accepting proof π_R for $(x_{\mathsf{init}}, x_{\mathsf{fin}}, 2)$. Since the verifier of Fig. 13 accepted π_R, it has the form $\pi_R = (\mathsf{ct}_0, \mathsf{ct}_2, r_0, r_2, \{S^{(1)}\})$ with $S^{(1)} = (0, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2, \pi^{(1)})$, where $\mathsf{ct}_0 = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{init}}; r_0)$, $\mathsf{ct}_2 = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{fin}}; r_2)$, and $\pi^{(1)}$ is a level-1 BARG proof on $(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2)$ that verifies with message $m := x_{\mathsf{init}}$.

- $\mathsf{GetHint}(\mathsf{crs}, (x_{\mathsf{init}}, x_{\mathsf{init}}, 2), (x_{\mathsf{init}}, x_{\mathsf{fin}}, 2), \pi_R)$ outputs a random tape that is uniform except at one position: the coins the honest prover draws in its second step as encryption randomness for the state $x_2 = x_{\mathsf{init}}$ are set to r_0. Running $\mathsf{IVC.P}^2$ on this tape with root x_{init} produces $\pi_L = (\mathsf{ct}_0^L, \mathsf{ct}_2^L, r_0^L, r_0, \{S_L^{(1)}\})$ with $\mathsf{ct}_2^L = \mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{init}}; r_0) = \mathsf{ct}_0$. The honest prover signs its two level-0 proofs and its level-1 BARG proof $\pi_L^{(1)}$ with the message x_{init}, because x_{init} is the root of the statement it is proving.
- $\mathsf{Merge}(\mathsf{crs}, (x_{\mathsf{init}}, x_{\mathsf{init}}, x_{\mathsf{fin}}), (2, 2), \pi_L, \pi_R)$ reads $\pi_L^{(1)}$ and $\pi^{(1)}$ off the two level-1 entries, runs the level-2 BARG prover on the pair of statements $\{(\mathsf{ct}_0^L, \mathsf{ct}_0), (\mathsf{ct}_0, \mathsf{ct}_2)\}$ with witnesses $(\mathsf{ct}_1^L, \pi_L^{(1)})$ and $(\mathsf{ct}_1, \pi^{(1)})$ and message $m := x_{\mathsf{init}}$, obtaining $\pi^{(2)}$, and outputs $\big(\mathsf{ct}_0^L, \mathsf{ct}_2, r_0^L, r_2, \{S^{(2)} = (0, \mathsf{ct}_0^L, \mathsf{ct}_0, \mathsf{ct}_2, \pi^{(2)})\}\big)$.

The verifier, run on the statement $(x_{\mathsf{init}}, x_{\mathsf{fin}}, 4)$, checks the single entry $S^{(2)}$: the adjacency conditions hold with $i_{\mathsf{init}} = 0$ and $\mathsf{ct}_{\mathsf{init}} = \mathsf{ct}_0^L$; the level-2 BARG proof verifies with $m := x_{\mathsf{init}}$ by BARG completeness, because both witnesses are level-1 BARG proofs that verify with the same message x_{init}, the left one by completeness of the components the honest prover assembled and the right one because the verifier of π_R ran that very check; and the endpoint equations $\mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{init}}; r_0^L) = \mathsf{ct}_0^L$ and $\mathsf{PKE.Enc}(\mathsf{pk}, x_{\mathsf{fin}}; r_2) = \mathsf{ct}_2$ hold by construction. The argument uses only the shape that the verifier's acceptance of π_R forces, so it covers the honest π_R of Definition 5.2 and any other accepting proof as well.

G Valiant-Style Proofs in the Zero-Knowledge Setting

In this section we abstract the common adaptation to zero knowledge of Valiant-style proofs and we prove how they generally imply our notion of post-hoc universal mergeability.

Definition G.1 (Explicit input encodings) *Let* $\mathsf{IVC} = (\mathsf{IVC.Gen}, \mathsf{IVC.P}, \mathsf{IVC.V})$ *be an IVC scheme. We say that* IVC *has* explicit input encodings *if there exist efficient deterministic algorithm* Enc, ChkX, ChkY*, and PPT algorithms* $\mathsf{IVC.P}'$ *and* $\mathsf{IVC.V}'$ *such that, for all* λ, T, n, m, C *and every* crs *in the support of* $\mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C)$*, the following hold.*

Shape of a proof. Every proof π *for a statement* $x \overset{d}{\rightsquigarrow} y$ *is a tuple*

$$\pi = (e_x,\, e_y,\, p_x,\, p_y,\, \tilde{\pi})\,,$$

where e_x *and* e_y *are encodings of the two endpoints,* p_x *and* p_y *are the respective openings, and* $\tilde{\pi}$ *is the residual proof.*

Verification. On a statement $x \overset{d}{\rightsquigarrow} y$, IVC.V *parses the proof it receives as a tuple of the above shape and accepts if and only if*

$$\mathsf{ChkX}(e_x, p_x, x) = 1 \;\wedge\; \mathsf{ChkY}(e_y, p_y, y) = 1 \;\wedge\; \mathsf{IVC.V}'\big(\mathsf{crs},\, e_x,\, e_y,\, x,\, y,\, d,\, \tilde{\pi}\big) = 1.$$

Proof generation. IVC.P *encodes the two endpoints and then invokes* $\mathsf{IVC.P}'$*: writing* inp *for the inputs* IVC.P *receives, it samples*

$$(e_\alpha, p_\alpha) \leftarrow \mathsf{Enc}(\mathsf{crs}, \alpha) \qquad \text{for } \alpha \in \{x, y\}\,,$$

computes $\tilde{\pi} \leftarrow \mathsf{IVC.P}'\big(\mathsf{inp},\, e_x,\, e_y,\, p_x,\, p_y\big)$ *and returns*

$$\pi = (e_x,\, e_y,\, p_x,\, p_y,\, \tilde{\pi})\,.$$

where $\tilde{\pi}$ *are all the inputs to the IVC prover in the standard syntax.*

Definition G.2 (Plain (d_L, d_R)-mergeability (or, ZK-Valiant-Style IVC)) *We say that an IVC scheme with explicit input encodings is (plain)* (d_L, d_R)*-mergeable, for* $d_L, d_R \in \mathbb{N}$*, if there exists an algorithm* Merge' *such that the following holds for all* $\lambda \in \mathbb{N}$*, all iteration bounds* T *with* $d_L + d_R \le T \le 2^\lambda$*, all polynomially bounded* n, m*, and all polynomial-size circuits* $C : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$*: For all PPT* M*,*

$$\mathsf{Exp}_{Mrg'}(M) \;\ge\; 1 - \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Exp}_{Mrg'}(M)$ *is defined in Fig. 15.*

$$\mathsf{Exp}_{\mathrm{Mrg}'}(M) := \Pr\left[\begin{array}{l} \big[\mathsf{IVC.V}\big(\mathsf{crs}, \mathsf{s}_L, \pi_L\big) = 1 \\ \quad \wedge\ \mathsf{IVC.V}\big(\mathsf{crs}, \mathsf{s}_R, \pi_R\big) = 1\big] \\ \quad \Longrightarrow\ \mathsf{IVC.V}\big(\mathsf{crs}, \mathsf{s}^*, \pi^*\big) = 1 \end{array} : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{IVC.Gen}(1^\lambda, T, 1^n, 1^m, C) \\ (x, y, z, e_x, e_y, e_z, \rho_x, \rho_y, \rho_z, \tilde{\pi}_L, \tilde{\pi}_R) \leftarrow M(\mathsf{crs}) \\ \pi_L := (e_x, e_y, \rho_x, \rho_y, \tilde{\pi}_L) \\ \pi_R := (e_y, e_z, \rho_y, \rho_z, \tilde{\pi}_R) \\ \tilde{\pi}^* \leftarrow \mathsf{Merge}'\big(\mathsf{crs}, (x, y, z), (d_L, d_R), \pi_L, \pi_R\big) \\ \pi^* := (e_x, e_z, \rho_x, \rho_z, \tilde{\pi}^*) \end{array}\right]$$

Fig. 15: The plain mergeability experiment for IVC with explicit input encodings. $\mathsf{s}_L := (x, y, d_L)$, $\mathsf{s}_R := (y, z, d_R)$, $\mathsf{s}^* := (x, z, d_L + d_R)$ are the prefix, the suffix and the merged statement.

Theorem G.3. *Let* IVC *be an IVC scheme with explicit input encodings (Definition G.1). If* IVC *is plain* (d_L, d_R)*-mergeable (Definition G.2), then it is universally (post-hoc)* (d_L, d_R)*-mergeable (Definition 5.1).*

Proof (Sketch). Write $(x, y, z) := (x_{\mathsf{init}}, x_{\mathsf{mid}}, x_{\mathsf{fin}})$, let GetHint return a uniform tape, and let Merge parse its two input proofs as in Definition G.1, run $\tilde{\pi}^* \leftarrow \mathsf{Merge}'\big(\mathsf{crs}, (x, y, z), (d_L, d_R), \pi_L, \pi_R\big)$ and output $\pi^* := (e_x, e_z, \rho_x, \rho_z, \tilde{\pi}^*)$. Since Enc is deterministic, the prefix proof π_L and the suffix proof π_R of Fig. 6 carry the same mid encoding and opening (e_y, ρ_y), hence (π_L, π_R) is an output of the machine M that has x, y, z and the two witness vectors hardwired and runs the two provers on them; completeness makes both π_L and π_R accept, so plain mergeability makes π^* accept on s^* except with negligible probability.

H Ordering of the Simulation-Soundness Notions

We formalize the fact that the three winning conditions of Fig. 5 are ordered.

Proposition H.1 *Let* IVC *be an IVC scheme and* Sim *a zero-knowledge simulator for it. If* IVC *is* SIM-SND *with respect to* Sim *then it is* w-SIM-SND *with respect to* Sim*, and if it is* w-SIM-SND *then it is* newrt-SIM-SND.

Proof. The three experiments differ only in the winning condition, so it suffices to show, as predicates of (x^*, π^*, Q_S), that

$$\mathsf{WinCond}^{\text{newrt-SIM-SND}} \Longrightarrow \mathsf{WinCond}^{\text{w-SIM-SND}} \Longrightarrow \mathsf{WinCond}^{\text{SIM-SND}},$$

since then an adversary winning the experiment for a weaker notion wins the experiment for a stronger one with at least the same probability. All notions require a false statement. For the second implication, the falsity and non-compliance conjuncts are shared, and $\left(\left[x^*_{\mathrm{in}} \overset{d^*}{\underset{?}{\rightsquigarrow}} x^*_{\mathrm{out}}\right], _\right) \notin Q_S$ implies $\left(\left[x^*_{\mathrm{in}} \overset{d^*}{\underset{?}{\rightsquigarrow}} x^*_{\mathrm{out}}\right], \pi^*\right) \notin Q_S$. For the first, assume $\mathsf{WinCond}^{\text{newrt-SIM-SND}}$: no statement rooted at x^*_{in} was queried, hence $P(x^*_{\mathrm{in}}) = \emptyset$ and, by Definition 4.3, x^* is not compliant via $P(x^*_{\mathrm{in}})$, there being no leap to use; moreover any query whose statement is x^* would be rooted at x^*_{in}, so $\left(\left[x^*_{\mathrm{in}} \overset{d^*}{\underset{?}{\rightsquigarrow}} x^*_{\mathrm{out}}\right], _\right) \notin Q_S$.

I Example Applications

I.1 Proofs of authorized image manipulation succinct in the number of edits

We now describe applications of simulation-sound IVC to the setting of image manipulation introduced by [NT16]. In the same setting, the recent work of [DEH25] has shown the advantage of incremental proving. We give an adapted description of it, then explain which of our notions each of its security goals requires.

Step circuit $C_{\mathsf{pk},\mathsf{Adm}}(x, w)$
State: $x = (I, \mathsf{cm}, \mathsf{md}, \sigma)$ with I an image or $\perp$. *Witness:* $w = (T,\ I_0,\ r)$.

1. $b_{\text{sig}} := \mathsf{Vfy}\big(\mathsf{pk},\ I_0 \| \mathsf{md},\ \sigma\big)$, for the σ carried by the state.
2. $b_{\text{com}} := \big[\mathsf{cm} = \mathsf{Com}(I_0;\ r)\big]$.
3. $b_{\text{trans}} := \mathsf{Adm}(T)$.
4. If $b_{\text{sig}} \wedge b_{\text{com}} \wedge b_{\text{trans}}$: let $I' := I_0$ if $I = \perp$ and $I' := I$ otherwise, and output $\big(T(I'), \mathsf{cm}, \mathsf{md}, \sigma\big)$. Otherwise output the failure state $\perp$.

Fig. 16: The step circuit for authorized image manipulation. The failure state $\perp$ is absorbing: $C_{\mathsf{pk},\mathsf{Adm}}(\perp, \cdot) = \perp$.

Setting. A trusted source, say a camera, holds a signing key with verification key pk. When it captures an image I_0 it also records metadata md (for instance the time and place of the capture) and signs the two together, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, I_0 \| \mathsf{md})$.[7] An editor, who may be the photographer or someone the image was passed on to, applies a sequence of transformations $T_1, \ldots, T_t$ and obtains a final image $I_{\text{final}} = T_t \circ \cdots \circ T_1(I_0)$. Which transformations are allowed (cropping, resizing, rotation, brightness and contrast adjustments, and so on) is fixed by a predicate Adm that takes a description of a transformation and accepts if it is admissible. A client receives

$$(I_{\text{final}},\ \mathsf{cm},\ \mathsf{md},\ \sigma),$$

where $\mathsf{cm} = \mathsf{Com}(I_0; r)$ is a commitment to the original image, and should be convinced that I_{final} was obtained from an image that the source signed together with md by applying admissible transformations only. The signature is passed along in the clear, which spares the circuit an opening of cm to it. The client should learn nothing else: not the original (which may show for example what was cropped out) and not the transformations (for example, we may not want to show what which parts were specifically cropped out). The commitment scheme is computationally hiding, for privacy, and perfectly binding, so that the original an authenticated image refers to is well defined. As a consequence we do not need to rely on extractability.
Two features of the setting make IVC the natural tool. Image transformations are commonly applied one at a time. It is also plausible they are applied by different parties and at different times (a photo used in newspaper reporting may have some filters applied by the source or staff photographer and then subsequently transformed prepress). An IVC permits solutions where what is shown to the client does not grow with the number of edits.

The step circuit. We describe a step circuit exemplifying how this could work. The computation state is a tuple $x = (I, \mathsf{cm}, \mathsf{md}, \sigma)$ consisting of the current image, the commitment, the metadata and the signature; the last three never change along the chain, and the first may take a placeholder value $\perp$ for "no image yet". The circuit is parametrized by the verification key pk of the source and by the predicate Adm, and is described in Fig. 16. Its witness carries the transformation applied in this step together with an opening of cm to the original image.
Three remarks on the design. First, the circuit re-checks the signature and the opening of cm at every step. Only the first step needs them, since that is where the committed original enters the image slot; the later checks are redundant but keep the circuit uniform, and they cost nothing in proof size. Second, the original never appears in a public state: the chain starts from

$$x_0 := (\perp, \mathsf{cm}, \mathsf{md}, \sigma),$$

and the first step replaces the placeholder by $T_1(I_0)$ using the I_0 in its witness, which the check on cm ties to the committed original. Third, by perfect binding, all steps of an accepting chain open cm to the same image I_0.

[7] The signature travels in the clear in what follows, while the image it is computed on stays hidden, so we assume a scheme whose signatures leak little about the signed message: what a signature reveals should be of no use to an adversary unable to guess the original image in the first place. Such schemes exist. In BLS [BLS01] a signature is $H(m)^{\mathsf{sk}}$ for a hash function H modelled as a random oracle, which an adversary can use only to test a guess for m, by checking the pairing equation against the candidate.

The statement. The editor's claim is the IVC statement

$$\big(x_0,\ x_t,\ t\big) \quad \text{with} \quad x_0 = (\bot, \mathsf{cm}, \mathsf{md}, \sigma), \qquad x_t = (I_{\text{final}}, \mathsf{cm}, \mathsf{md}, \sigma).$$

Unwinding the definition of L_{IVC} and using perfect binding, this statement is true if and only if the unique opening I_0 of cm satisfies $\mathsf{Vfy}(\mathsf{pk}, I_0 \| \mathsf{md}, \sigma) = 1$ and there are $T_1, \ldots, T_t$ with $\mathsf{Adm}(T_i) = 1$ for all i and $I_{\text{final}} = T_t \circ \cdots \circ T_1(I_0)$. The editor proves it with the honest IVC prover, one transformation per step, and passes the proof along with the image; a later editor extends the proof by further steps. The client checks the proof against (x_0, x_t, t), which it can form from the tuple it received.

Non-malleability. The image manipulation scenario is one naturally requiring a form of non-malleability: an adversary observing a valid image manipulation proof—intuitively certifying $\mathsf{cm} \overset{I}{\rightsquigarrow}_{\text{final}}$ for a commitment to an image cm—should not be able use that to forge a proof certifying a non-admissible or plainly false transformation of any kind. Natural applications settings plausibly would not require resistance to re-randomizing proofs and so a weak form of simulation-soundness may suffice.
In some settings newrt-SIM-SND may even be sufficient. For example, whenever we envision the adversary's main target being (committed) images for which it has already seen valid proofs[8].

I.2 Zero-knowledge (non-malleable) shuffles with sublinear proof-size

Consider a setting where n ciphertexts must be shuffled and rerandomized so that the output ciphertexts cannot be linked to the input ones, and where a mauled shuffle proof would let a party claim credit for someone else's shuffle. Natural applications of shuffle arguments often require simulation-soundness [Smy21, FFHR19].
We realize the shuffle as an IVC computation that processes one ciphertext per step. The public input of the step function consists of two accumulators $A_{\mathsf{in}}, A_{\mathsf{out}}$ with efficient insertion; its witness consists of one input ciphertext ct and the coins used to rerandomize it. The accumulators are order-insensitive: their value depends on the multiset of what has been inserted and not on the order of insertion, which is what lets the prover consume the input ciphertexts in the order of its choosing. The step function rerandomizes ct into ct' and outputs the updated accumulators $\mathsf{Insert}(A_{\mathsf{in}}, \mathsf{ct})$ and $\mathsf{Insert}(A_{\mathsf{out}}, \mathsf{ct}')$. The proof consists of a single IVC proof, hence its size is sublinear in n. Accumulators with sublinear-time insertion can be instantiated from several cryptographic assumptions: from hidden-order groups [BBF19], from elliptic curves [CHAK23], and from lattices [LLNW16].
The permutation itself is never named in the computation. The prover consumes the input ciphertexts in permuted order, and what the verifier checks is that the two multisets the chain accumulated are the ones it accumulates itself from the published lists. Duplicating one ciphertext and omitting another changes the first multiset, so the check on A_{in} catches it.

The step function. We can now give the step function in full; see Fig. 17. Write $\mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}; \rho)$ for the rerandomization algorithm of the encryption scheme and Insert for insertion into an accumulator. The state carries the step counter and the two accumulator values, and the circuit refuses to extend a chain that has already reached n steps.
Per step the circuit performs one rerandomization and two insertions. Every value the verifier relies on is an *output* of the computation, compared against a value it computes itself from public data. No part of the statement asserts that an opening of a public digest exists, which is what would force us to extract a witness in order to conclude anything from an accepting proof: a computationally binding accumulator admits openings to any value, and ruling them out requires knowledge soundness rather than the membership soundness we assume.

[8] A more concrete example: a media outlet covering social protests periodically publishes manipulated images I_{final}'s whose (commitment to the) original is authenticated by trusted sources. Consider a malicious actor—e.g., a corrupt government—whose main goal were to discredit such media organization by spreading false versions of I_{final} together with valid proofs (to show what is "truly" contained in the image captured by the trusted sources). This type of attacks, which would target the same root statement as the one in the observed proofs, are prevented by newrt-SIM-SND.

Step circuit $C_{\mathsf{pk},n}(x, w)$
State: $x = (i,\ A_{\mathsf{in}},\ A_{\mathsf{out}})$ with $i \in [0, n]$. *Witness:* $w = (\mathsf{ct},\ \rho)$.

1. $\mathsf{ct}' := \mathsf{ReRand}(\mathsf{pk},\ \mathsf{ct};\ \rho)$.
2. If $i < n$, output
$$(i + 1,\ \mathsf{Insert}(A_{\mathsf{in}}, \mathsf{ct}),\ \mathsf{Insert}(A_{\mathsf{out}}, \mathsf{ct}'));$$
otherwise output the failure state $\perp$.

Fig. 17: The step function for the shuffle. The failure state is absorbing, $C_{\mathsf{pk},n}(\perp, \cdot) = \perp$. Step $i + 1$ consumes one input ciphertext, inserts it into A_{in}, and inserts its rerandomization into A_{out}.

What the verifier checks. The mixer publishes the output list, and the claim is the IVC statement (x_0, x_n, n) with

$$x_0 := (0,\ \varnothing,\ \varnothing), \qquad x_n := (n,\ D_{\mathsf{in}},\ D_{\mathsf{out}}).$$

The verifier computes D_{in} and D_{out} by accumulating the published input and output lists itself, one ciphertext at a time, forms the two states above, and accepts if the IVC proof verifies for (x_0, x_n, n) and $t = n$.
Truth of (x_0, x_n, n) is the shuffle relation: there are ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and coins $\rho_1, \ldots, \rho_n$ such that inserting $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ into the first accumulator and $\mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_1; \rho_1), \ldots, \mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_n; \rho_n)$ into the second yields D_{in} and D_{out}. Since the accumulators are order-insensitive and binding on multisets, this says that the published output list is, up to reordering, a rerandomization of the published input list, one output ciphertext per input ciphertext.

J Simulation-Extractability for PCD [ACDK26]

This section contains the notion of simulation-extractability for PCDs from [ACDK26] and necessary preliminaries.

J.1 Proof-carrying data

Definition J.1 (Proof-Carrying Transcript) *A* transcript T *is a directed acyclic graph where each vertex* $u \in V(\mathsf{T})$ *is labeled by local data* $w^{(u)}_{\mathsf{loc}} \in \{0,1\}^* \cup \{\perp\}$*, and each edge* $e \in E(\mathsf{T})$ *is labeled by a message* $z^{(e)} \neq \perp$*. The* output *of* T*, denoted* $\mathsf{o}(\mathsf{T})$*, is the message* $z^{(e)}$ *corresponding to the lexicographically first sink of* T*.*
A proof-carrying transcript PCT *has the same topology and format as* T*, except that each edge* $e \in E(\mathsf{PCT})$ *is further labeled with a proof* $\pi^{(e)}$*.*

The *fan-in* of PCT is the maximum number m of incoming edges of any vertex. The classes of compliance predicates F below are such that every $\Psi \in \mathsf{F}$ has the form $\Psi : \{0,1\}^* \times (\{0,1\}^* \cup \{\perp\})^{m+1} \to \{0,1\}$, i.e. it takes as input a tuple $(z, w_{\mathsf{loc}}, z_1, \ldots, z_{m'}, \perp, \ldots, \perp)$ for $m' \leq m$ and outputs accept/reject.

Definition J.2 (Compliant Proof-Carrying Transcript) *Let* PCT *be a proof-carrying transcript. A vertex* $u \in V(\mathsf{PCT})$ *is* $(\Psi, \overline{\mathbb{V}}_{\mathsf{pp}})$-compliant *for a compliance predicate* $\Psi \in \mathsf{F}$ *and verifier circuit* $\overline{\mathbb{V}}_{\mathsf{pp}}$ *if, for all outgoing edges* $e = (u, v) \in E(\mathsf{PCT})$*, letting* $e_1, \ldots, e_{m'}$ *(with* $m' \leq m$*) be the incoming edges of* u*:*

i) $\Psi\big(z^{(e)}, w^{(u)}_{\mathsf{loc}}, z^{(e_1)}, \ldots, z^{(e_{m'})}, \perp, \ldots, \perp\big) = 1$,
ii) $\overline{\mathbb{V}}_{\mathsf{pp}}\big(\Psi, z^{(e)}, \pi^{(e)}\big) = 1$.

We say that PCT *is* $(\Psi, \overline{\mathbb{V}}_{\mathsf{pp}})$-compliant *if all of its vertices are* $(\Psi, \overline{\mathbb{V}}_{\mathsf{pp}})$*-compliant.*

Definition J.3 (PCD) *A* proof-carrying data *(PCD) scheme for a class of predicates* F *is a tuple of PPT algorithms* $\mathsf{PCD} = (\mathbb{G}, \mathbb{I}, \mathbb{P}, \mathbb{V})$ *with the following syntax:*
– $\mathbb{G}(1^\lambda) \to (\mathsf{pp}, \mathsf{td})$*. On input the security parameter, outputs public parameters* pp *and a trapdoor* td*.*

– $\mathbb{I}(\mathsf{pp}, \Psi) \to (\mathsf{ipk}, \mathsf{ivk})$. *On input the public parameters* pp *and a predicate* Ψ*, deterministically outputs a proving key* ipk *and a verifying key* ivk.
– $\mathbb{P}\big(\mathsf{ipk}, z, w_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m\big) \to \pi$. *On input the proving key* ipk*, outgoing data* z*, local data* w_{loc}*, and incoming data-proof pairs* $[z_i, \pi_i]_{i=1}^m$*, outputs an outgoing proof* π.
– $\mathbb{V}(\mathsf{ivk}, z, \pi) \to b \in \{0,1\}$. *On input the verifying key* ivk*, outgoing data* z*, and outgoing proof* π*, outputs a bit indicating accept/reject.*
– *The* combined *verifier* $\overline{\mathbb{V}}_{\mathsf{pp}}(\Psi, z, \pi) \to b \in \{0,1\}$ *has* pp *hardwired and runs both* $\mathbb{I}(\mathsf{pp}, \Psi) \to (\mathsf{ipk}, \mathsf{ivk})$ *and* $\mathbb{V}(\mathsf{ivk}, z, \pi) \to b$.

PCD *is required to satisfy* completeness *and* knowledge soundness*;* zero knowledge *is an optional notion.*

– ***Completeness.*** *For every (unbounded) adversary* $\mathbb{A}$*,*

$$\Pr\left[\begin{array}{c|l} \Psi \in \mathsf{F} & (\mathsf{pp}, \mathsf{td}) \leftarrow \mathbb{G}(1^\lambda) \\ \wedge\ \Psi(z, w_{\mathsf{loc}}, z_1, \ldots, z_m) = 1 & \big(\Psi, z, w_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m\big) \leftarrow \mathbb{A}(\mathsf{pp}) \\ \wedge\ \mathbb{V}(\mathsf{ivk}, z_i, \pi_i) = 1\ \forall i \in [m'] & (\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathbb{I}(\mathsf{pp}, \Psi) \\ \Downarrow & \pi \leftarrow \mathbb{P}\big(\mathsf{ipk}, z, w_{\mathsf{loc}}, [z_i, \pi_i]_{i=1}^m\big) \\ \mathbb{V}(\mathsf{ivk}, z, \pi) = 1 & \end{array}\right] = 1.$$

– ***Knowledge soundness.*** PCD *is knowledge sound with respect to an auxiliary input distribution* $\mathcal{D}$ *if for every expected polynomial-time adversary* $\mathbb{A}$ *there exists an expected polynomial-time extractor* $\mathbb{E}_{\mathbb{A}}$ *such that*

$$\Pr\left[\mathsf{PCD\text{-}KS}^{\mathbb{A},\mathbb{E}_{\mathbb{A}}}_{\mathsf{PCD},\mathcal{D}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

for the game of Fig. 18.

– ***Zero knowledge.*** PCD *satisfies computational (resp. statistical) zero knowledge if there exists a PPT simulator* $\mathbb{S}$ *with the syntax*

- $\mathbb{S}(\mathsf{td}, (\mathsf{pp}, \Psi, z)) \to \pi$*, returning a proof* π *on input* pp*, a predicate* Ψ*, and data* z*,*

such that for every expected polynomial-time (resp. unbounded) adversary $\mathbb{A}$ *the following is negligible in* λ*:*

$$\left|\Pr\left[\mathsf{PCD\text{-}ZK}^{\mathbb{A}}_{0,\mathsf{PCD},\mathsf{F}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{PCD\text{-}ZK}^{\mathbb{A},\mathbb{S}}_{1,\mathsf{PCD},\mathsf{F}}(1^\lambda) = 1\right]\right|,$$

for the games of Fig. 19.

Game $\mathsf{PCD\text{-}KS}^{\mathbb{A},\mathbb{E}_{\mathbb{A}}}_{\mathsf{PCD},\mathcal{D}}(1^\lambda)$

$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathbb{G}(1^\lambda)$; $\mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp})$
$(\Psi, \mathsf{o}, \pi) \leftarrow \mathbb{A}(\mathsf{pp}, \mathsf{ai})$
$(\mathsf{ipk}, \mathsf{ivk}) \leftarrow \mathbb{I}(\mathsf{pp}, \Psi)$
$\mathsf{PCT} \leftarrow \mathbb{E}_{\mathbb{A}}(\mathsf{Trans}_{\mathbb{A}})$
return $\left(\begin{array}{c} \Psi \in \mathsf{F} \wedge \\ \mathbb{V}(\mathsf{ivk}, \mathsf{o}, \pi) = 1 \wedge \\ \big(\mathsf{PCT} \text{ is not } (\Psi, \overline{\mathbb{V}}_{\mathsf{pp}})\text{-compliant} \ \vee\ \mathsf{o}(\mathsf{PCT}) \neq \mathsf{o}\big) \end{array}\right)$

Fig. 18: Knowledge soundness for proof-carrying data.

J.2 Simulation-extractability for PCD

Definition J.4 (Relaxed Compliance) *Let* PCT *be a proof-carrying transcript, let* $\overline{\mathbb{V}}_{\mathsf{pp}}$ *be the combined PCD verifier of Definition J.3, and let* $\mathcal{Q}_{\mathbb{S}}$ *be the set of tuples* (Ψ', z', π') *where* (Ψ', z') *are inputs to the simulation oracle* $\widetilde{\mathbb{S}}$ *and* π' *is its corresponding output. A vertex* $u \in V(\mathsf{PCT})$ *is* $(\overline{\mathbb{V}}_{\mathsf{pp}}, \mathcal{Q}_{\mathbb{S}})$-partially compliant *if for all outgoing edges* $e = (u, v) \in E(\mathsf{PCT})$ *labeled* $(z^{(e)}, \pi^{(e)})$ *the following conditions hold* ***unless*** $(\Psi_u, z^{(e)}, \pi^{(e)}) \in \mathcal{Q}_{\mathbb{S}}$*, where* Ψ_u *is the predicate associated with vertex* u*:*

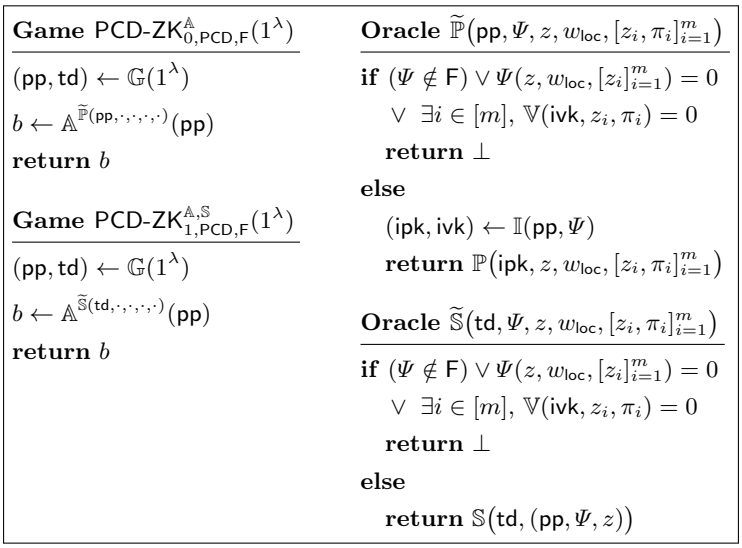


Fig. 19: Zero knowledge for proof-carrying data.

1. *letting* $e_1,\ldots,e_{m'}$ *be the incoming edges of* u*, labeled* $(z^{(e_i)},\pi^{(e_i)})$*, the local data* $w_{\mathsf{loc}}^{(u)}$ *satisfies*

$$\Psi_u\big(z^{(e)},w_{\mathsf{loc}}^{(u)},z^{(e_1)},\ldots,z^{(e_{m'})}\big) = 1;$$

2. *the combined verifier accepts the outgoing proof:*

$$\overline{\mathbb{V}}_{\mathsf{pp}}\big(\Psi_u,z^{(e)},\pi^{(e)}\big) = 1.$$

We say that PCT *is* $(\overline{\mathbb{V}}_{\mathsf{pp}},\mathcal{Q}_{\mathbb{S}})$-partially compliant *if all its vertices satisfy the condition above.*

Definition J.5 (Simulation-Extractability for PCD) *Let* PCD *be a zero-knowledge PCD scheme for a class of predicates* F*, with simulator* $\mathbb{S}$*. We say that* PCD *satisfies* simulation extractability *if for every auxiliary input distribution* $\mathcal{D}$ *and every expected polynomial-time adversary* $\mathbb{A}$ *there exists an expected polynomial-time extractor* $\mathbb{E}_{\mathbb{A}}$ *such that*

$$\Pr\Big[\mathsf{PCD\text{-}SE}^{\mathbb{A},\mathbb{S},\mathbb{E}_{\mathbb{A}}}_{\mathsf{PCD},\mathsf{F},\mathcal{D}}(1^\lambda) = 1\Big] \leq \mathsf{negl}(\lambda),$$

where the game is defined in Fig. 20 and $\mathcal{Q}_{\mathbb{S}}$ *consists of the statement-proof pairs queried by* $\mathbb{A}$ *to* $\mathbb{S}$*.*

Remark J.6 *The notion of relaxed compliance from Definition J.4 is strongly coupled to the PCD setting and to the presence of an extractor and hence it cannot be simply "ported" to the simulation-soundness setting in IVC. This is essentially because it is described in terms of properties of the proofs provided by the extractor, which are part of the transcript provided by the same (in PCD, proofs are an inherent part of the "witness"; see notion of transcript in Definition J.1). The requirement of relaxed compliance for the extracted transcript essentially boils down to "the transcript reaches source node or a node with a proof given by the simulator (and satisfies local predicates everywhere else)". Thus, moving to simulation-soundness requires a new and different formulation of admissible continuations of simulated statements, since no extracted proof-carrying transcript is available to identify where simulated proofs enter the computation.*

Game $\mathsf{PCD\text{-}SE}^{\mathbb{A},\mathbb{S},\mathbb{E}_{\mathbb{A}}}_{\mathsf{PCD},\mathsf{F},\mathcal{D}}(1^\lambda)$

$(\mathsf{pp},\mathsf{td}) \leftarrow \mathbb{G}(1^\lambda);\ \mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp});\ \mathcal{Q}_{\mathbb{S}} := \emptyset$

$(\Psi^*, \mathsf{o}^*, \pi^*, \mathsf{ao}) \leftarrow \mathbb{A}^{\widetilde{\mathbb{S}}(\mathsf{td},\cdot,\cdot)}(\mathsf{pp},\mathsf{ai})$

$(\mathsf{ipk},\mathsf{ivk}) \leftarrow \mathbb{I}(\mathsf{pp},\Psi^*)$

$\mathsf{PCT}^* \leftarrow \mathbb{E}_{\mathbb{A}}(\mathsf{Trans}_{\mathbb{A}})$

$b_{\mathsf{compl}} \leftarrow \big(\mathsf{PCT}^* \text{ is } (\overline{\mathbb{V}}_{\mathsf{pp}}, \mathcal{Q}_{\mathbb{S}})\text{-partially compliant}\big)$

$b_{\mathsf{output}} \leftarrow \big(\mathsf{o}(\mathsf{PCT}^*) = \mathsf{o}^*\big)$

$b_{\mathbb{V}} \leftarrow \mathbb{V}(\mathsf{ivk},\mathsf{o}^*,\pi^*)$

return $\begin{pmatrix} (\Psi^* \in \mathsf{F}) \ \wedge\ (\Psi^*,\mathsf{o}^*,\pi^*) \notin \mathcal{Q}_{\mathbb{S}} \\ \wedge\ b_{\mathbb{V}} \ \wedge\ \neg(b_{\mathsf{compl}} \wedge b_{\mathsf{output}}) \end{pmatrix}$

Oracle $\widetilde{\mathbb{S}}(\mathsf{td},\Psi,z)$

if $\Psi \notin \mathsf{F}$ **then return** $\perp$

$\pi \leftarrow \mathbb{S}\big(\mathsf{td},(\mathsf{pp},\Psi,z)\big)$

$\mathcal{Q}_{\mathbb{S}} := \mathcal{Q}_{\mathbb{S}} \cup \{(\Psi,z,\pi)\}$

return π

Fig. 20: Simulation extractability for proof-carrying data.