Revisiting the Nova Proof System on a Cycle of Curves

Wilson D. Nguyen
Stanford University, CA, USA

Dan Boneh
Stanford University, CA, USA

Srinath Setty
Microsoft Research, Redmond, WA, USA

Abstract

Nova is an efficient recursive proof system built from an elegant folding scheme for (relaxed) R1CS statements. The original Nova paper (CRYPTO'22) presented Nova using a single elliptic curve group of order p. However, for improved efficiency, the implementation of Nova alters the scheme to use a 2-cycle of elliptic curves. This altered scheme is only described in the code and has not been proven secure. In this work, we point out a soundness vulnerability in the original implementation of the 2-cycle Nova system. To demonstrate this vulnerability, we construct a convincing Nova proof for the correct evaluation of 2^{75} rounds of the Minroot VDF in only 116 milliseconds. We then present a modification of the 2-cycle Nova system and formally prove its security. The modified system also happens to be more efficient than the original implementation. In particular, the modification eliminates an R1CS instance-witness pair from the recursive proof. The implementation of Nova has now been updated to use our optimized and secure system. In addition, we show that the folding mechanism at the core of Nova is malleable: given a proof for some statement z, an adversary can construct a proof for a related statement z', at the same depth as z, without knowledge of the witness for z'.

2012 ACM Subject Classification Security and privacy → Cryptanalysis and other attacks

Keywords and phrases Cryptographic Protocols, Recursive Proof Systems, Folding, Vulnerability

Digital Object Identifier 10.4230/LIPIcs.AFT.2023.18

Related Version *Full Version*: `https://eprint.iacr.org/2023/969`

Supplementary Material *Software (Demo Code)*: `https://github.com/MercysJest/NovaBreakingTheCycleAttack`, archived at `swh:1:dir:ca6b1263b64aa9e429f687369b2df7a68bdc4a7f`

Funding This work was partially funded by NSF, DARPA, the Simons Foundation, and NTT Research. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

1 Introduction

In a recent work, Kothapalli, Setty, and Tzialla introduced an elegant folding scheme for relaxed R1CS statements [12]. The scheme leads to the Nova proof system: an efficient and succinct proof system for incrementally verifiable computation, or IVC [21]. This proof system has many applications in the blockchain space, such as verifiable delay functions [20], a Nova-based ZK virtual machine [15], and outsourced computation.

The description and analysis of Nova in [12] restricts itself to a single chain of incremental computation, namely a series of identical computation steps that produce an output which is fed directly into the next step. At every step, a single application of some function F is applied, and a statement about the validity of the prior step is folded into an ongoing statement of validity. We refer to this as a single *IVC chain*.

To improve efficiency, the implementation of Nova [14] uses a 2-cycle of elliptic curves. This leads to a proof system that uses two parallel IVC chains that must be linked together. Until this work, the 2-cycle Nova system was only described in the implementation code and there was no public proof of security.

In this paper, we present two security concerns that affect the 2-cycle Nova system. First, in Section 7 we describe a soundness issue that enables an attacker to produce proofs for false statements. For example, we compute a convincing proof for an evaluation of 2^{75} rounds of the Minroot VDF [10] in only 116 milliseconds on a single laptop. The core issue that this attack exploits is that the 2-cycle Nova system produces an IVC proof that contains an additional R1CS instance-witness pair that is not sufficiently constrained by the verifier.

To fix this issue we first formally describe the two IVC chains approach used in the Nova implementation. Instead of describing the original scheme, we present in Sections 4 and 5 a modified version of the system that fixes the vulnerability and results in a shorter IVC proof. We present the scheme as a compiler that compilers a Nova-like folding scheme into an IVC proof using a cycle of curves. In Section 6 we prove knowledge soundness of this modified system. We followed responsible disclosure best practice and coordinated patches with the Nova authors. The Nova implementation has now been updated [19] to use this optimized and secure system.

Second, in Section 8, we show that Nova's IVC proofs are malleable, which can lead to a security vulnerability in some applications. We also discuss strategies to mitigate this issue.

We begin by establishing in Sections 2 and 3 the terminology needed to describe the 2-cycle Nova system (i.e the 2-cycle Nova IVC Scheme).

2 Preliminaries

2.1 Incrementally Verifiable Computation (IVC)

Incrementally verifiable computation, or IVC, was introduced by Valiant [21]. For a function $\mathsf{F} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^a$, and some public values $z_0, z_i \in \{0,1\}^a$, an IVC scheme lets a prover generate a succinct proof that it knows auxiliary values $\mathsf{aux}_0, \ldots, \mathsf{aux}_{i-1} \in \{0,1\}^b$ such that

$$\begin{tikzcd}
& \mathsf{aux}_0 \arrow[d] & \mathsf{aux}_1 \arrow[d] & & \mathsf{aux}_{i-1} \arrow[d] & \\
z_0 \arrow[r] & \boxed{\mathsf{F}} \arrow[r] & \boxed{\mathsf{F}} \arrow[r] & \cdots \arrow[r] & \boxed{\mathsf{F}} \arrow[r] & z_i
\end{tikzcd}$$

The following definition gives the syntax and security properties for an IVC scheme. The prover $\mathcal{P}$ in this definition computes a proof for one step in the IVC chain. Iterating the prover will produce a proof π_i for the entire chain of length i.

▶ **Definition 1** (IVC [21]). *An **IVC Scheme** is a tuple of efficient algorithms* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *with the following interface:*

- $\mathsf{Setup}(1^\lambda, n) \to \mathsf{pp}$: *Given a security parameter* 1^λ*, a poly-size bound* $n \in \mathbb{N}$*, outputs public parameters* pp.
- $\mathcal{P}(\mathsf{pp}, \mathsf{F}, (i, z_0, z_i), \mathsf{aux}_i, \pi_i) \to \pi_{i+1}$: *Given public parameters* pp*, a function* $\mathsf{F} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^a$ *computable by a circuit of size at most* n*, an index* $i \in \mathbb{N}$*, an initial input* $z_0 \in \{0,1\}^a$*, a claimed output* $z_i \in \{0,1\}^a$*, advice* $\mathsf{aux}_i \in \{0,1\}^b$*, and an IVC proof* π_i*, outputs a new IVC proof* π_{i+1}.
- $\mathcal{V}(\mathsf{pp}, \mathsf{F}, (i, z_0, z_i), \pi_i) \to 0/1$: *Given public parameters* pp*, a function* F*, an index* i*, an initial input* z_0*, a claimed output* z_i*, and an IVC proof* π_i*, outputs* 0 *(reject) or* 1 *(accept).*

An IVC Scheme satisfies the following properties:

Completeness. *For every poly-size bound $n \in \mathbb{N}$, for every* pp *in the output space of* $\mathsf{Setup}(1^\lambda, n)$*, for every function* $\mathsf{F} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^a$ *computable by a circuit within the poly-size bound n, for every collection of elements $(i \in \mathbb{N}, z_0, z_i \in \{0,1\}^a)$,* $\mathsf{aux}_i \in \{0,1\}^b$*, and IVC proof π_i,*

$$\Pr\left[\begin{array}{c}\mathcal{V}(\mathsf{pp}, \mathsf{F}, (i, z_0, z_i), \pi_i) = 1 \\ \Downarrow \\ \mathcal{V}(\mathsf{pp}, \mathsf{F}, (i+1, z_0, z_{i+1}), \pi_{i+1}) = 1\end{array} \quad : \quad \begin{array}{l}\pi_{i+1} \leftarrow \mathcal{P}(\mathsf{pp}, \mathsf{F}, (i, z_0, z_i), \mathsf{aux}_i, \pi_i), \\ z_{i+1} \leftarrow F(z_i, \mathsf{aux}_i)\end{array}\right] = 1$$

Knowledge Soundness. *Let $n \in \mathbb{N}$ be a poly-size bound and $\ell(\lambda)$ be a polynomial in the security parameter. Let $\mathcal{F}$ be an efficient function sampling adversary that outputs a function* $\mathsf{F} : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}^a$ *computable by a circuit within the poly-size bound n. Then for every efficient IVC prover P^*, there exists an efficient extractor $\mathcal{E}$ such that the probability*

$$\Pr\left[\begin{array}{c}\mathcal{V}(\mathsf{pp}, \mathsf{F}, (i, z_0, z_i), \pi_i) = 1 \\ \Big\Downarrow \\ z_i = \mathsf{F}(\mathbf{z}_{i-1}, \mathsf{aux}_{i-1}) \ \wedge \\ (i = 1 \ \Rightarrow \ \mathbf{z}_{i-1} = z_0) \ \wedge \\ (i > 1 \ \Rightarrow \ \mathcal{V}(\mathsf{pp}, \mathsf{F}, (i-1, z_0, \mathbf{z}_{i-1}), \pi_{i-1}) = 1)\end{array} \quad : \quad \begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n), \\ \rho \leftarrow \{0,1\}^{\ell(\lambda)}, \\ \mathsf{F} \leftarrow \mathcal{F}(\mathsf{pp}; \rho), \\ (i, z_0, z_i, \pi_i) \leftarrow P^*(\mathsf{pp}; \rho), \\ (\mathbf{z}_{i-1}, \mathsf{aux}_{i-1}), \pi_{i-1} \leftarrow \mathcal{E}(\mathsf{pp}; \rho)\end{array}\right]$$

is greater than or equal to $1 - \mathsf{negl}(\lambda)$.

▶ Remark 2 (Full Extraction). Our definition of knowledge soundness implies other notions of IVC knowledge soundness, which require the extraction of all the auxiliary values in the execution chain [12, 11, 1, 2]. Informally, consider some ρ and pp sampled at random, and an adversary $P^*(\mathsf{pp}; \rho)$ that outputs a proof for i iterations of the IVC. Then the knowledge extractor $\mathcal{E}$ can be used to construct an IVC prover for a proof of $i-1$ iterations. Applying the definition again to the prover derived from $\mathcal{E}$ implies that there is a knowledge extractor $\mathcal{E}'$ that outputs a valid $(z_{i-2}, \mathsf{aux}_{i-2})$, π_{i-2} with all but negligible probability. We can repeat this argument inductively to extract a vector of auxiliary values $(\mathsf{aux}_{i-1}, \ldots, \mathsf{aux}_0)$ that shows that the z_i output by P^* is computed correctly from z_0. Note that if $\mathrm{time}(\mathcal{E}) > c \cdot \mathrm{time}(P^*)$ for some constant $c > 1$, then this argument only works for $O(\log \lambda)$ steps before the running time of the extractor becomes super-polynomial in λ. We use this sequential IVC model for consistency with the original Nova [12, 14]. In certain applications, a tree-like IVC proof system might be preferable.

▶ Remark 3 (Zero Knowledge). In some settings one also wants the IVC scheme to be zero knowledge, but in this writeup we focus on knowledge soundness of the scheme.

2.2 Committed Relaxed R1CS over a Ring

The Nova Proof system over a cycle of curves (2-cycle Nova system) makes use of two finite fields $\mathbb{F}_1$ and $\mathbb{F}_2$ simultaneously. As such, it is convenient to treat the primitives used in Nova as operating on the finite commutative ring $\mathrm{R} := \mathbb{F}_1 \times \mathbb{F}_2$, where addition and multiplication are defined component wise. That is, for $a = (a_1, a_2)$ and $b = (b_1, b_2)$ in R, we define $a + b = (a_1 + a_2, b_1 + b_2)$ and $a \cdot b = (a_1 b_1, a_2 b_2)$.

▶ **Definition 4** (Commitment Scheme). *Let* R *be a finite commutative ring. A commitment scheme for vectors over* R *is a pair of efficient algorithms* $(\mathsf{Setup}_{\mathsf{com}}, \mathsf{Commit})$ *with the following interface:*

- $\mathsf{Setup}_{\mathsf{com}}(1^\lambda, \mathrm{R}, n) \to \mathsf{pp}_{\mathsf{com}}$: *Given a security parameter* $1^\lambda \in 1^{\mathbb{N}}$*, a description of a ring* R*, and a poly-size bound* $n \in \mathbb{N}$*, outputs public parameters* $\mathsf{pp}_{\mathsf{com}}$.
- $\mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x) \to c$: *Given public parameters* $\mathsf{pp}_{\mathsf{com}}$ *and input* $x \in \mathrm{R}^n$*, outputs a commitment* c.

These algorithms need to satisfy the following properties.

- *Binding: Let* $n \in \mathbb{N}$ *be a poly-size bound. For every efficient adversary* $\mathcal{A}$ *and for every finite commutative ring* R *whose size is at most exponential in* λ,

$$\Pr\left[\begin{array}{c} \mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x_0) = \mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x_1) \wedge \\ x_0 \neq x_1 \end{array} : \begin{array}{l} \mathsf{pp}_{\mathsf{com}} \leftarrow \mathsf{Setup}_{\mathsf{com}}(1^\lambda, \mathrm{R}, n) \\ (x_0, x_1) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{com}}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- *Additively Homomorphic: Given two commitments* $c \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x)$, $c' \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x')$ *to vectors* $x,\ x' \in \mathrm{R}^n$ *(not necessarily distinct), there is an efficient homomorphism* $\oplus$ *on commitments such that* $c \oplus c' = \mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x + x')$.
- *Succinct: For any* $x \in \mathrm{R}^n$*, the commitment* $c \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{com}}, x)$ *must have size* $|c| \leq \mathsf{poly}(\lambda, \log(n))$.

▶ **Definition 5** (Committed Relaxed R1CS over a Ring). *Consider* $m, n, \ell \in \mathbb{N}$ *where* $m > \ell$ *and a finite commutative ring* R*. Further, consider a commitment scheme* Commit *for vectors over* R*, where* pp_W *and* pp_E *are commitment parameters for vectors of size* $m - \ell - 1$ *and* n *respectively.*

- *A* ***committed relaxed R1CS instance*** *is a tuple* $\mathbb{U} := (\bar{E}, s, \bar{W}, x)$*, where* $\bar{E}$ *and* $\bar{W}$ *are commitments,* $s \in \mathrm{R}$*, and* $x \in \mathrm{R}^\ell$.
- *A committed relaxed R1CS instance* $\mathbb{U} = (\bar{E}, s, \bar{W}, x)$ *is* ***satisfiable*** *with respect to an* ***R1CS constraint system*** $\mathsf{R1CS} := (A, B, C \in \mathrm{R}^{n \times m})$ *if there exist a relaxed witness* $\mathbb{W} := (E \in \mathrm{R}^n,\ W \in \mathrm{R}^{m-\ell-1})$ *such that*

$$\bar{E} = \mathsf{Commit}(\mathsf{pp}_E, E), \quad \bar{W} = \mathsf{Commit}(\mathsf{pp}_W, W), \quad \textit{and} \quad (A \cdot Z) \circ (B \cdot Z) = s \cdot (C \cdot Z) + E$$

  where $Z = (W, x, s)$*. We refer to* E *as the* ***error vector*** *and* W *as the* ***extended witness****.*
- *An instance-witness pair* $(\mathbb{U}, \mathbb{W})$ ***satisfies*** *a constraint system* R1CS *if* $\mathbb{W}$ *is a satisfying relaxed witness for* $\mathbb{U}$*. An instance-witness pair* $(\mathbb{u}, \mathbb{w})$ *pair* ***strictly satisfies*** *an R1CS constraint system* R1CS *if* (1) *the pair satisfies* R1CS *and* (2) $\mathbb{u}.\bar{E} = \bar{0}$ *is the commitment to the zero vector and* $s = 1$.

▶ Remark 6 (Trivially Satisfiable Instance-Witness Pairs). A committed instance-witness pair $(\mathbb{U}_\perp, \mathbb{W}_\perp)$ will denote a trivially satisfying pair for an R1CS constraint system R1CS over R. In Nova [12], this pair is constructed by setting $E, W,$ and x to appropriately sized zero vectors, $\bar{E}, \bar{W}$ to be commitments to the zero vectors, and s equal to 0.

2.3 A Folding Scheme for Committed Relaxed R1CS over a Ring

Folding schemes give an efficient approach to IVC. In recent years, several works [2, 12, 11, 1, 13, 17] constructed efficient folding schemes for different problems. Nova [12] introduces an elegant folding scheme, for folding two committed relaxed R1CS instances and their witnesses. Nova's folding scheme is a public-coin, one-round interactive protocol that is made non-interactive in the random oracle model using the Fiat-Shamir transform. Additionally, Nova heuristically instantiates the random oracle with a concrete hash function and assumes that this heuristic produces a protocol that is knowledge sound. A similar assumption is used in other recursive proof systems [2, 11, 1].

▶ **Definition 7.** *A* ***Non-Interactive Folding Scheme for Committed Relaxed R1CS*** *consists of an underlying commitment scheme* $(\mathsf{Setup}_{\mathsf{com}}, \mathsf{Commit})$ *(Definition 4) for committed relaxed instances (Definition 5) and a tuple of efficient algorithms* $(\mathsf{Fold}_{\mathsf{Setup}}, \mathsf{Fold}_{\mathcal{K}}, \mathsf{Fold}_{\mathcal{P}}, \mathsf{Fold}_{\mathcal{V}})$ *with the following interface:*

- $\mathsf{Fold}_{\mathsf{Setup}}(1^\lambda, n) \to \mathsf{pp}$: *Given a security parameter* $1^\lambda \in 1^{\mathbb{N}}$*, a poly-size bound* $n \in \mathbb{N}$*, outputs public parameters* pp *which contain the description of a finite commutative ring* R *and commitment parameters* $\mathsf{pp}_{\mathsf{com}}$ *for vectors over* R *within the size bound* n*.*
- $\mathsf{Fold}_{\mathcal{K}}(\mathsf{pp}, \mathsf{R1CS}) \to (\mathsf{pk}, \mathsf{vk})$ *Given public parameters* pp*, an R1CS constraint system* $\mathsf{R1CS}$ *over* R *within the poly-size bound* n*, outputs proving key* pk *and verifier key* vk*.*
- $\mathsf{Fold}_{\mathcal{P}}(\mathsf{pk}, (\mathbb{u}, \mathbb{w}), (\mathbb{U}, \mathbb{W})) \to (\bar{\mathsf{T}}, (\mathbb{U}', \mathbb{W}'))$: *Given a proving key* pk*, two committed relaxed R1CS instance-witness pairs* $(\mathbb{u}, \mathbb{w}), (\mathbb{U}, \mathbb{W})$*, outputs a folding proof* $\bar{\mathsf{T}}$ *in the commitment space, and a new committed relaxed R1CS instance-witness pair* $(\mathbb{U}', \mathbb{W}')$*.*
- $\mathsf{Fold}_{\mathcal{V}}(\mathsf{vk}, \mathbb{u}, \mathbb{U}, \bar{\mathsf{T}}) \to \mathbb{U}'$: *Given a verification key* vk*, two committed relaxed R1CS instances* $\mathbb{u}, \mathbb{U}$*, and a folding proof* $\bar{\mathsf{T}}$*, outputs a new committed relaxed R1CS instance* $\mathbb{U}'$*.*

These algorithms need to satisfy the following properties:

Completeness. *For every poly-size bound* $n' \in \mathbb{N}$*, for every* pp *in the output space of* $\mathsf{Fold}_{\mathsf{Setup}}(1^\lambda, n')$*, for every poly-size* $m, n, \ell \in \mathbb{N}$ *where* $m > \ell$*,* $n' > m - \ell - 1$*,* $n' > n$*, for every R1CS constraint system* $\mathsf{R1CS} := (A, B, C \in \mathrm{R}^{n \times m})$*, for every committed relaxed instance-witness pair* $(\mathbb{u}, \mathbb{w}), (\mathbb{U}, \mathbb{W})$ *for* $\mathsf{R1CS}$*,*

$$\Pr\left[\begin{array}{c} \mathbb{U}' = \mathbb{U}'' \\ \wedge \\ (\mathbb{u}, \mathbb{w}), (\mathbb{U}, \mathbb{W}) \text{ satisfy } \mathsf{R1CS} \\ \implies (\mathbb{U}', \mathbb{W}') \text{ satisfies } \mathsf{R1CS} \end{array} : \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Fold}_{\mathcal{K}}(\mathsf{pp}, \mathsf{R1CS}), \\ (\bar{\mathsf{T}}, (\mathbb{U}', \mathbb{W}')) \leftarrow \mathsf{Fold}_{\mathcal{P}}(\mathsf{pk}, (\mathbb{u}, \mathbb{w}), (\mathbb{U}, \mathbb{W})), \\ \mathbb{U}'' \leftarrow \mathsf{Fold}_{\mathcal{V}}(\mathsf{vk}, \mathbb{u}, \mathbb{U}, \bar{\mathsf{T}}) \end{array}\right] = 1$$

Knowledge Soundness. *Let* $n \in \mathbb{N}$ *be a poly-size bound and* $\ell(\lambda)$ *be a polynomial in the security parameter. For every efficient adversary* $\mathcal{P}^*$*, there exist an efficient extractor* $\mathcal{E}$ *such that the probability*

$$\Pr\left[\begin{array}{c} \mathbb{U}' = \mathsf{Fold}_{\mathcal{V}}(\mathsf{vk}, \mathbb{u}, \mathbb{U}, \bar{\mathsf{T}}) \wedge \\ (\mathbb{U}', \mathbb{W}') \text{ satisfies } \mathsf{R1CS} \\ \Downarrow \\ (\mathbb{u}, \mathbb{w}), (\mathbb{U}, \mathbb{W}) \text{ satisfy } \mathsf{R1CS} \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Fold}_{\mathsf{Setup}}(1^\lambda, n), \\ \rho \leftarrow \{0,1\}^{\ell(\lambda)}, \\ (\mathsf{R1CS}, (\mathbb{u}, \mathbb{U}, \bar{\mathsf{T}}), (\mathbb{U}', \mathbb{W}')) \leftarrow \mathcal{P}^*(\mathsf{pp}; \rho), \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Fold}_{\mathcal{K}}(\mathsf{pp}, \mathsf{R1CS}), \\ (\mathbb{w}, \mathbb{W}) \leftarrow \mathcal{E}(\mathsf{pp}; \rho) \end{array}\right]$$

is $\geq 1 - \mathsf{negl}(\lambda)$*. In words, the definition of knowledge soundness states that if an adversary* $\mathcal{P}^*$ *can create a folded statement* $\mathbb{U}'$ *of two statements* $\mathbb{u}$ *and* $\mathbb{U}$ *and a satisfying witness* $\mathbb{W}'$ *for* $\mathbb{U}'$*, then an extractor* $\mathcal{E}$ *for* $\mathcal{P}^*$ *can produce satisfying witnesses* $\mathbb{w}$ *for* $\mathbb{u}$ *and* $\mathbb{W}$ *for* $\mathbb{U}$*.*

Collision resistance. The Nova construction also uses collision resistant hash functions. To be comprehensive, we define these next.

▶ **Definition 8** (Collision Resistant Hash Functions). *Let* R *be a finite commutative ring such that* $|\mathrm{R}| \approx 2^\lambda$. *A hash function for* R *is a pair of efficient algorithms* $(\mathsf{Setup_H}, \mathsf{H})$ *with the following interface*:

- $\mathsf{Setup_H}(1^\lambda, \mathrm{R}) \to \mathsf{pp_H}$: *Given a security parameter* $1^\lambda \in 1^{\mathbb{N}}$ *and a description of* R, *outputs public parameters* $\mathsf{pp_H}$.
- $\mathsf{H}(\mathsf{pp_H}, x) \to h$: *Given public parameters* $\mathsf{pp_H}$ *and input* $x \in \mathrm{R}^*$, *outputs a hash* $h \in \mathrm{R}$.

A hash function is ***collision resistant*** *if for every efficient adversary* $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}\mathsf{H}(\mathsf{pp_H}, m_0) = \mathsf{H}(\mathsf{pp_H}, m_1) \wedge \\ m_0 \neq m_1\end{array} : \begin{array}{c}\mathsf{pp_H} \leftarrow \mathsf{Setup_H}(1^\lambda, \mathrm{R}) \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pp_H})\end{array}\right] \le \mathsf{negl}(\lambda)$$

3 The Nova Proof System over a Cycle of Curves: Preliminary Details

In this section and the next, we describe details about the underlying primitives in the 2-cycle Nova System [14]. Section 5 describes the explicit operation of the modified IVC verifier and modified IVC prover.

Cycle of Elliptic Curves. To reduce the number of constraints related to group operations, the implementation of Nova uses a cycle of elliptic curves for which the discrete log problem is hard. Specifically, the Nova implementation is generic over any cycle of elliptic curves that implements certain Rust traits (Nova implements those traits for the pasta cycle of two curves [16]). We denote the elliptic curve groups as $\mathbb{G}_1$ and $\mathbb{G}_2$. We refer to the *scalar field* of an elliptic curve group $\mathbb{G}$ as the field $\mathbb{F}$ whose order is $|\mathbb{G}|$, and the *base field* of $\mathbb{G}$ as the field $\mathbb{F}'$ over which the elliptic curve is defined (i.e. the points have the form $(x, y) \in \mathbb{F}' \times \mathbb{F}'$).

The group $\mathbb{G}_1$ has scalar field $\mathbb{F}_1$ and base field $\mathbb{F}_2$, while $\mathbb{G}_2$ has scalar field $\mathbb{F}_2$ and base field $\mathbb{F}_1$. Group operations for $\mathbb{G}_1$ can be efficiently expressed as constraints over the base field $\mathbb{F}_2$. Symmetrically, group operations for $\mathbb{G}_2$ can be efficiently expressed as constraints over the base field $\mathbb{F}_1$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ will be the commitment spaces for Pedersen vector commitments for vectors over $\mathbb{F}_1$ and $\mathbb{F}_2$ respectively.

Groups and Rings. We define the ring $\mathrm{R} := \mathbb{F}_1 \times \mathbb{F}_2$ as the set of tuples with one element in $\mathbb{F}_1$ and another in $\mathbb{F}_2$. We can naturally define the ring operations as the component-wise field operations. Similarly, define the group $\mathbb{G} := \mathbb{G}_1 \times \mathbb{G}_2$ and it's group operation as the component-wise group operation.

Commitments. In Nova, the folding procedure requires additively homomorphic commitments to vectors over a field $\mathbb{F}$. Their specific construction [12] uses Pedersen vector commitments belonging to a group $\mathbb{G}$ of order $|\mathbb{F}|$, for which the discrete log problem is hard. Nova's implementation [14] is generic over the commitment scheme and one can supply a different commitment scheme for vectors, but we restrict our attention to Pedersen vector commitments in this paper.

We generalize the Pedersen vector commitment to the ring $\mathrm{R} := \mathbb{F}_1 \times \mathbb{F}_2$ by composing a Pedersen vector commitment over $\mathbb{F}_1$ with commitment space $\mathbb{G}_1$ and a Pedersen vector commitment over $\mathbb{F}_2$ with commitment space $\mathbb{G}_2$. We write $x^{(1)} \in \mathbb{F}_1^n$ and $x^{(2)} \in \mathbb{F}_2^n$ for the left and right projections of a vector $x \in \mathrm{R}^n$. Then, the commitment to x is a pair of commitments: a commitment to $x^{(1)} \in \mathbb{F}_1^n$ and a commitment to $x^{(2)} \in \mathbb{F}_2^n$. Concretely, this commitment to the vector $x \in \mathrm{R}^n$ will be an element in $\mathbb{G} := \mathbb{G}_1 \times \mathbb{G}_2$.

Committed relaxed instances. Consider two R1CS constraint systems

$$\mathsf{R1CS}^{(1)} := (A_1, B_1, C_1 \in \mathbb{F}_1^{n_1 \times m_1}) \quad \text{and} \quad \mathsf{R1CS}^{(2)} := (A_2, B_2, C_2 \in \mathbb{F}_2^{n_2 \times m_2})$$

defined over $\mathbb{F}_1$ and $\mathbb{F}_2$, respectively. A committed relaxed instance for $\mathsf{R1CS}^{(1)}$ is a tuple

$$\mathbb{U}^{(1)} := \left(\bar{\mathsf{E}}^{(1)},\ s^{(1)},\ \bar{\mathsf{W}}^{(1)},\ x^{(1)}\right) \qquad \text{where} \qquad \bar{\mathsf{E}}^{(1)}, \bar{\mathsf{W}}^{(1)} \in \mathbb{G}_1, \quad s^{(1)} \in \mathbb{F}_1, \quad x^{(1)} \in \mathbb{F}_1^{\ell_1}.$$

The corresponding relaxed witness $\mathbb{W}^{(1)} = (E^{(1)}, W^{(1)})$ has an error vector $E^{(1)} \in \mathbb{F}_1^{n_1}$ and extended witness $W^{(1)} \in \mathbb{F}_1^{m_1 - \ell_1 - 1}$. Symmetrically, a committed relaxed instance for $\mathsf{R1CS}^{(2)}$ is a tuple

$$\mathbb{U}^{(2)} := \left(\bar{\mathsf{E}}^{(2)},\ s^{(2)},\ \bar{\mathsf{W}}^{(2)},\ x^{(2)}\right) \qquad \text{where} \qquad \bar{\mathsf{E}}^{(2)}, \bar{\mathsf{W}}^{(2)} \in \mathbb{G}_2, \quad s^{(2)} \in \mathbb{F}_2, \quad x^{(2)} \in \mathbb{F}_2^{\ell_2}.$$

The corresponding relaxed witness $\mathbb{W}^{(2)} = (E^{(2)}, W^{(2)})$ has error vector $E^{(2)} \in \mathbb{F}_2^{n_2}$ and $W^{(2)} \in \mathbb{F}_2^{m_2 - \ell_2 - 1}$.

The two constraint systems $\mathsf{R1CS}^{(1)}$ over $\mathbb{F}_1$ and $\mathsf{R1CS}^{(2)}$ over $\mathbb{F}_2$ can be treated as a single constraint system $\mathsf{R1CS} := (A, B, C \in \mathrm{R}^{n \times m})$ over $\mathrm{R} := \mathbb{F}_1 \times \mathbb{F}_2$. The constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ are simply the left and right projections of $\mathsf{R1CS}$. A strict projection of $\mathsf{R1CS}$ would require the dimensions of $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ to be identical to the dimensions of $\mathsf{R1CS}$. In practice, $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ can have different dimensions. When abstractly combining the constraint systems to obtain $\mathsf{R1CS}$, we can pad the systems with dummy rows and columns so that $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ have the same dimension. In particular, $m = \max(m_1, m_2)$, $n = \max(n_1, n_2)$, and $l = \max(l_1, l_2)$. Similarly, we can treat instance-witness pairs $(\mathbb{U}^{(1)}, \mathbb{W}^{(1)})$, $(\mathbb{U}^{(2)}, \mathbb{W}^{(2)})$ as the left and right projection of an instance-witness pair $(\mathbb{U}, \mathbb{W})$ for $\mathsf{R1CS}$.

Hash Functions. Hash functions $\mathsf{H}_1 : \mathbb{F}_1^* \to \mathbb{F}_1$ and $\mathsf{H}_2 : \mathbb{F}_2^* \to \mathbb{F}_2$ are collision resistant hash functions that take as input an arbitrary number of field elements and output a single field element which encodes the hash. In Nova, this single field element can be represented as a scalar whose bit representation is at most 250 bits long. Thus, the output hash has a unique representation in both fields, whose elements are 256 bits.[1]

Concretely, define $h_1 := \mathsf{H}_1(\dots)$ as the output of H_1 for some arbitrary input elements $(\dots) \in \mathbb{F}_1^*$. The hash can be expressed as $h_1 = \sum_{i \le 250} b_i^{(1)} \cdot (2^{(1)})^i$ where $2^{(1)} \in \mathbb{F}_1$ and for all $i \le 250$, $b_i^{(1)} \in \mathbb{F}_1$ are bits in $\{0, 1\}$. The hash output $h_1 := \sum_{i \le 250} b_i^{(1)} \cdot (2^{(1)})^i$ in $\mathbb{F}_1$ can be represented as an element h_1' in $\mathbb{F}_2$. To do so, define $h_1' := \sum_{i \le 250} b_i^{(2)} \cdot (2^{(2)})^i$ where for all i, the bit $b_i^{(2)} \in \mathbb{F}_2$ is the same the bit $b_i^{(1)} \in \mathbb{F}_1$ (i.e. if $b_i^{(1)} = 1^{(1)}$, we define $b_i^{(2)} = 1^{(2)}$ otherwise $b_i^{(2)} = 0^{(2)}$). Symmetrically, a hash output $h_2 := \sum_{i \le 250} b_i^{(2)} \cdot (2^{(2)})^i$ in $\mathbb{F}_2$ can be represented as an element h_2' in $\mathbb{F}_1$ in the same way.

Similarly, the hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is a collision resistant hash function whose outputs can be represented uniquely in both fields. We omit the hash parameters for H for ease of presentation. The Nova implementation [14] uses the Poseidon hash function [8] for H_1 and H_2 and SHA-3 [6] for H.

[1] The size of a digest is configurable, but a digest length of 250 bits was chosen to support a variety of popular curve cycles e.g., secp/secq, pallas/vesta (pasta curves), BN254/Grumpkin.

3.1 Folding over a Cycle of Curves

In Nova [12], a non-interactive folding scheme in the random oracle model is constructed by applying the Fiat-Shamir transform [7] to an interactive folding scheme. By instantiating the random oracle with an appropriate cryptographic hash function, they heuristically obtain a non-interactive folding scheme in the plain model. The construction described in [12] is limited to an R1CS constraint system $\mathsf{R1CS}$ defined over a field $\mathbb{F}$ with commitments belonging to a group $\mathbb{G}$ (with scalar field $\mathbb{F}$).

We extend the construction to R1CS constraint systems $\mathsf{R1CS}$ defined over a ring $\mathrm{R} := \mathbb{F}_1 \times \mathbb{F}_2$ by composing a folding scheme for R1CS constraint systems defined over $\mathbb{F}_1$ and a folding scheme for R1CS constraint systems defined over $\mathbb{F}_2$. When we fold committed relaxed instances for $\mathsf{R1CS}^{(1)}$, we implicitly mean run the folding scheme for systems over $\mathbb{F}_1$. Symmetrically, when we fold committed relaxed instances for $\mathsf{R1CS}^{(2)}$, we implicitly mean run the folding scheme for systems over $\mathbb{F}_2$. However, the random oracle calls used in both folding scheme will need to take in an argument vk, which is derived from both systems. We describe this in more detail in the description of $\mathsf{Fold}_{\mathcal{K}}$.

3.1.1 Folding Setup

$\mathsf{Fold}_{\mathsf{Setup}}$ takes in as input:
- A security parameter 1^λ.
- A poly-size bound $n \in \mathbb{N}$.

The algorithm performs the following steps:
1. Sample a cycle of elliptic curves $(\mathbb{G}_1, \mathbb{F}_1, \mathbb{G}_2, \mathbb{F}_2) \leftarrow \mathsf{SampleCycle}(1^\lambda)$.
2. Sample collision resistant hash parameters $\mathsf{pp}_{\mathsf{H}_1} \leftarrow \mathsf{Setup}_{\mathsf{H}}(1^\lambda, \mathbb{F}_1)$, $\mathsf{pp}_{\mathsf{H}_2} \leftarrow \mathsf{Setup}_{\mathsf{H}}(1^\lambda, \mathbb{F}_2)$.
3. Sample commit params $\mathsf{pp}_{\mathsf{com}_1} \leftarrow \mathsf{Setup}_{\mathsf{com}}(1^\lambda, \mathbb{F}_1, n)$ and $\mathsf{pp}_{\mathsf{com}_2} \leftarrow \mathsf{Setup}_{\mathsf{com}}(1^\lambda, \mathbb{F}_2, n)$. [2]
4. Output $\mathsf{pp} := \big((\mathbb{G}_1, \mathbb{F}_1, \mathbb{G}_2, \mathbb{F}_2),\ \ \mathsf{pp}_{\mathsf{H}_1}, \mathsf{pp}_{\mathsf{H}_2}, \mathsf{pp}_{\mathsf{H}},\ \ \mathsf{pp}_{\mathsf{com}_1}, \mathsf{pp}_{\mathsf{com}_2}\big)$.

3.1.2 Folding Keygen

$\mathsf{Fold}_{\mathcal{K}}$ takes in as input:
- Public parameters pp
- An R1CS constraint system $\mathsf{R1CS}$ over R within the poly-size bound n.

The algorithm performs the following steps:
1. Assign the verification key vk to a hash digest of the public parameters and constraint systems
$$\mathsf{vk} \leftarrow \mathsf{H}\big(\mathsf{pp},\ \mathsf{R1CS} := (\mathsf{R1CS}^{(1)}, \mathsf{R1CS}^{(2)})\big) \tag{1}$$
2. Assign the proving key $\mathsf{pk} \leftarrow \mathsf{pp}$ to be the public parameters.
3. Output $(\mathsf{pk}, \mathsf{vk})$.

The Verification Key. The Nova folding scheme is derived from an interactive protocol via the Fiat-Shamir transform [7]. As such, queries to the random oracle must include a description of the entire environment. Concretely, let H be an appropriate cryptographic hash function that heuristically instantiates a random oracle and whose outputs can be represented uniquely in both fields. The vk element (assigned in (1)) denotes a hash digest of the

[2] The commitment parameters pp_E, pp_W will be prefixes of $\mathsf{pp}_{\mathsf{com}}$ where the length is $\max\big(|E|,\ |W|\big)$.

environment. $\mathsf{Fold}_{\mathcal{V}}$ incorporates the elements vk, $\mathbb{u}, \mathbb{U}, \bar{\mathsf{T}}$ as arguments to its random oracle. We stress that this is needed to preserve the soundness of the Fiat-Shamir transform [4], as these digest elements represent inputs to the folding verifier when viewed as an interactive protocol.

4 The Augmented Constraint Systems Used in Nova

The 2-cycle Nova IVC Scheme operates on a pair of functions F_1 and F_2, one for each field. Abstractly, one can treat Nova as an IVC scheme for the combined function $\mathsf{F} : (\mathbb{F}_1^{a_1} \times \mathbb{F}_2^{a_2}) \times (\mathbb{F}_1^{b_1} \times \mathbb{F}_2^{b_2}) \to (\mathbb{F}_1^{a_1} \times \mathbb{F}_2^{a_2})$ of the form

$$\left((z^{(1)},\ z^{(2)}),\ (\mathsf{aux}^{(1)},\ \mathsf{aux}^{(2)})\right) \ \xmapsto{\mathsf{F}}\ \left(\mathsf{F}_1(z^{(1)},\ \mathsf{aux}^{(1)}),\ \mathsf{F}_2(z^{(2)},\ \mathsf{aux}^{(2)})\right)$$

where $\mathsf{F}_1 : \mathbb{F}_1^{a_1} \times \mathbb{F}_1^{b_1} \to \mathbb{F}_1^{a_1}$ and $\mathsf{F}_2 : \mathbb{F}_2^{a_2} \times \mathbb{F}_2^{b_2} \to \mathbb{F}_2^{a_2}$ are poly-size arithmetic circuits over $\mathbb{F}_1$ and $\mathbb{F}_2$ respectively.

The 2-cycle Nova IVC scheme aims to prove that $(z_i^{(1)}, z_i^{(2)})$ is the result of iterating the function $\mathsf{F} = (\mathsf{F}_1, \mathsf{F}_2)$ a total of i times starting from the input $(z_0^{(1)}, z_0^{(2)})$ and using some auxiliary inputs. Every iteration of the IVC uses two R1CS constraint systems, one over $\mathbb{F}_1$ and one over $\mathbb{F}_2$, to verify that the functions F_1 and F_2 were evaluated correctly in that iteration. However, Nova augments these core constraint systems with additional constraints to verify that folding is done correctly at every iteration, and that the outputs of the previous iteration are properly forward to the current iteration. In this section we describe the two augmented constraint systems in detail.

The augmented constraint systems. The 2-cycle Nova IVC Scheme defines two augmented R1CS constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ over $\mathbb{F}_1$ and $\mathbb{F}_2$. As noted in Section 3, a group operation for $\mathbb{G}_1$ can be efficiently expressed as constraints in the base field $\mathbb{F}_2$. Since the folding operation requires group operations in $\mathbb{G}_1$, the Nova implementation does the folding of the committed instances $\mathbb{u}^{(1)}$ and $\mathbb{U}^{(1)}$ for $\mathsf{R1CS}^{(1)}$ in the constraints of $\mathsf{R1CS}^{(2)}$. Symmetrically, the Nova implementation does the folding of the committed instances $\mathbb{u}^{(2)}$ and $\mathbb{U}^{(2)}$ for $\mathsf{R1CS}^{(2)}$ in the constraints of $\mathsf{R1CS}^{(1)}$.

The constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ are defined as follows:

- let $\mathsf{R1CS}^{(1)}$ be the R1CS constraint system for the relation $\mathcal{R}_1$ defined in Figure 1a.
- let $\mathsf{R1CS}^{(2)}$ be the $\mathsf{R1CS}$ constraint system for the relation $\mathcal{R}_2$ defined in Figure 1b.

Intuitively, each constraint system applies one step of its function $z_{i+1} := \mathsf{F}(z_i,\ \mathsf{aux}_i)$, folds a prior committed instance $\mathbb{u}$ into a running committed instance $\mathbb{U}$ for the opposite constraint system, maintains the original inputs z_0, and updates the iteration index i. The public inputs $\mathbb{u}^{(1)}.x := (x_0, x_1)$ and $\mathbb{u}^{(2)}.x := (x_0, x_1)$ denote hashes that can be uniquely represented in both fields. We will explain these constraint systems in more detail when we describe the operation of the prover in Section 5.3.

Representation of Non-native Field elements and Arithmetic. Folding two committed instances $\mathbb{u}^{(1)}$ and $\mathbb{U}^{(1)}$ requires not only group operations over $\mathbb{G}_1$, but also field operations over $\mathbb{F}_1$. However, the R1CS constraint system $\mathsf{R1CS}^{(2)}$ over $\mathbb{F}_2$ has to encode the folding operation as constraints over $\mathbb{F}_2$. To account for this, $\mathbb{F}_1$ elements are encoded appropriately as $\mathbb{F}_2$ elements such that non-native arithmetic can be expressed as $\mathbb{F}_2$ constraints. The same strategy is symmetrically applied for folding constraints in $\mathsf{R1CS}^{(1)}$.

$$\mathcal{R}_1 := \left\{ \begin{array}{l} \left(\begin{array}{l} \mathbb{u}^{(1)}_{i+1}.x := (x_0, x_1 \in \mathbb{F}_1)\ ; \\ \hat{w}^{(1)}_{i+1} := \big(\mathsf{vk} \in \mathbb{F}_1,\ \ i^{(1)} \in \mathbb{F}_1,\ \ z^{(1)}_0, z^{(1)}_i \in \mathbb{F}_1^{a_1}, \\ \qquad\qquad \mathsf{aux}^{(1)}_i \in \mathbb{F}_1^{b_1},\ \ \mathbb{U}^{(2)}_i, \mathbb{u}^{(2)}_i \in \mathcal{U}^{(2)},\ \ \bar{\mathsf{T}}^{(2)}_i \in \mathbb{G}_2\big) \\ \qquad \text{where } \mathcal{U}^{(2)} := \mathbb{G}_2 \times \mathbb{F}_2 \times \mathbb{G}_2 \times \mathbb{F}_2^2 \end{array} \right) : \\ \\ \text{If } i^{(1)} = 0^{(1)} : \\ \quad \text{Then set } \mathbb{U}^{(2)}_{i+1} := \mathbb{U}^{(2)}_{\perp} \\ \quad \text{Else set } \mathbb{U}^{(2)}_{i+1} := \mathsf{Fold}_{\mathcal{V}}\big(\mathsf{vk}, \mathbb{u}^{(2)}_i, \mathbb{U}^{(2)}_i, \bar{\mathsf{T}}^{(2)}_i\big) \\ \\ \text{Accept if} : \\ \quad \text{If } i^{(1)} = 0^{(1)} \text{ then } z^{(1)}_i = z^{(1)}_0 \\ \quad \mathbb{u}^{(2)}_i.\bar{E} = \bar{0}^{(2)} \\ \quad \mathbb{u}^{(2)}_i.s = 1^{(2)} \\ \quad \mathbb{u}^{(2)}_i.x_0 = \mathsf{H}_1\big(\mathsf{vk},\ i^{(1)},\ z^{(1)}_0,\ z^{(1)}_i,\ \mathbb{U}^{(2)}_i\big) \\ \quad x_0 = \mathbb{u}^{(2)}_i.x_1 \\ \quad x_1 = \mathsf{H}_1\big(\mathsf{vk},\ (i+1)^{(1)},\ z^{(1)}_0,\ z^{(1)}_{i+1} := \mathsf{F}_1(z^{(1)}_i,\ \mathsf{aux}^{(1)}_i),\ \mathbb{U}^{(2)}_{i+1}\big) \end{array} \right\}$$

(a) The relation $\mathcal{R}_1$ defining the R1CS constraint system $\mathsf{R1CS}^{(1)}$ on instance-witness pairs $\big(\mathbb{u}^{(1)}_{i+1}.x\ ;\ \hat{w}^{(1)}_{i+1}\big)$.

$$\mathcal{R}_2 := \left\{ \begin{array}{l} \left(\begin{array}{l} \mathbb{u}^{(2)}_{i+1}.x := (x_0, x_1 \in \mathbb{F}_2)\ ; \\ \hat{w}^{(2)}_{i+1} := \big(\mathsf{vk} \in \mathbb{F}_2,\ \ i^{(2)} \in \mathbb{F}_2,\ \ z^{(2)}_0, z^{(2)}_i \in \mathbb{F}_2^{a_2}, \\ \qquad\qquad \mathsf{aux}^{(2)}_i \in \mathbb{F}_2^{b_2},\ \ \mathbb{U}^{(1)}_i, \mathbb{u}^{(1)}_{i+1} \in \mathcal{U}^{(1)},\ \ \bar{\mathsf{T}}^{(1)}_i \in \mathbb{G}_1\big) \\ \qquad \text{where } \mathcal{U}^{(1)} := \mathbb{G}_1 \times \mathbb{F}_1 \times \mathbb{G}_1 \times \mathbb{F}_1^2 \end{array} \right) : \\ \\ \text{If } i^{(2)} = 0^{(2)} : \\ \quad \text{Then set } \mathbb{U}^{(1)}_{i+1} := \mathbb{u}^{(1)}_{i+1} \\ \quad \text{Else set } \mathbb{U}^{(1)}_{i+1} := \mathsf{Fold}_{\mathcal{V}}\big(\mathsf{vk}, \mathbb{u}^{(1)}_{i+1}, \mathbb{U}^{(1)}_i, \bar{\mathsf{T}}^{(1)}_i\big) \\ \\ \text{Accept if} : \\ \quad \text{If } i^{(2)} = 0^{(2)} \text{ then } z^{(2)}_i = z^{(2)}_0 \\ \quad \mathbb{u}^{(1)}_{i+1}.\bar{E} = \bar{0}^{(1)} \\ \quad \mathbb{u}^{(1)}_{i+1}.s = 1^{(1)} \\ \quad \mathbb{u}^{(1)}_{i+1}.x_0 = \mathsf{H}_2\big(\mathsf{vk},\ i^{(2)},\ z^{(2)}_0,\ z^{(2)}_i,\ \mathbb{U}^{(1)}_i\big) \\ \quad x_0 = \mathbb{u}^{(1)}_{i+1}.x_1 \\ \quad x_1 = \mathsf{H}_2\big(\mathsf{vk},\ (i+1)^{(2)},\ z^{(2)}_0,\ z^{(2)}_{i+1} := \mathsf{F}_2(z^{(2)}_i,\ \mathsf{aux}^{(2)}_i),\ \mathbb{U}^{(1)}_{i+1}\big) \end{array} \right\}$$

(b) The relation $\mathcal{R}_2$ defining the R1CS constraint system $\mathsf{R1CS}^{(2)}$ on instance-witness pairs $\big(\mathbb{u}^{(2)}_{i+1}.x\ ;\ \hat{w}^{(2)}_{i+1}\big)$.

Hash parameters. The hash parameters $\mathsf{pp}_{\mathsf{H}_1}$ and $\mathsf{pp}_{\mathsf{H}_2}$ for H_1 and H_2 are hard-coded in the respective constraint systems. We omit the hash parameters in our paper for ease of notation, but implicitly call the hash function with their respective parameters generated in the IVC Setup.

Symmetry. If we omit the base case constraints, $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ are essentially symmetric constraint systems. The difference in indexing, $\mathbb{u}_i^{(2)}$ versus $\mathbb{u}_{i+1}^{(1)}$, is a notional choice that does not affect the symmetry. Additionally, we want to highlight that the only constraint on $\mathbb{u}_{i+1}^{(1)}.x_0$ and $\mathbb{u}_{i+1}^{(2)}.x_0$ are that they equal $\mathbb{u}_i^{(2)}.x_1$ and $\mathbb{u}_{i+1}^{(1)}.x_1$ respectively. As described in Section 3, hash values can be represented in both fields uniquely; thus, this equality is well-defined. Essentially, these *copy* constraints *pass* along the hashes meant for the public IO of the opposite instance. We will describe this strategy in more detail in Section 5.3.

5 The Modified Nova IVC Scheme

This section describes a modification to the prior (vulnerable) 2-cycle Nova proof system. In Section 6, we prove our modified system is knowledge sound (Definition 1).

5.1 Setup

The Nova Setup algorithm Setup takes in as input:
- A security parameter 1^λ.
- A poly-size bound $n \in \mathbb{N}$.

The algorithm outputs $\mathsf{pp} \leftarrow \mathsf{Fold}_{\mathsf{Setup}}(1^\lambda, n)$.

5.2 The Modified Nova Verifier

In this section, we describe a modified version of the 2-cycle Nova IVC verifier that patches a vulnerability found in the prior implementation. The algorithm is similar to the prior (vulnerable) 2-cycle Nova IVC verifier (Section 7.1), but the input IVC proof π_i omits a pair $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$,which caused the original vulnerability. We provide a proof of knowledge soundness of our modified scheme in Section 6.

The Nova Verifier $\mathcal{V}$ takes in as input:
- IVC public parameters pp,
- a description of functions $\mathsf{F}_1 : \mathbb{F}_1^{a_1} \times \mathbb{F}_1^{b_1} \to \mathbb{F}_1^{a_1}$ and $\mathsf{F}_2 : \mathbb{F}_2^{a_2} \times \mathbb{F}_2^{b_2} \to \mathbb{F}_2^{a_2}$,
- an index $i \in \mathbb{N}$,
- starting values $z_0^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_0^{(2)} \in \mathbb{F}_2^{a_2}$,
- claimed evaluations $z_i^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_i^{(2)} \in \mathbb{F}_2^{a_2}$, and
- an IVC Proof for iteration i, namely $\pi_i := \big((\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)}),\ (\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)}),\ (\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})\big)$.

The verifier first runs the following initial procedure, which can be treated as a preprocessing phase:
1. Given functions F_1 and F_2, deterministically generate augmented R1CS constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ which implement relations $\mathcal{R}_1$ and $\mathcal{R}_2$ from Figures 1a and 1b
2. Compute the folding verification key

$$(\,\cdot\,, \mathsf{vk}) \leftarrow \mathsf{Fold}_{\mathcal{K}}\big(\mathsf{pp},\ \mathsf{R1CS} := (\mathsf{R1CS}^{(1)}, \mathsf{R1CS}^{(2)})\big)$$

Then, the verifier accepts if the following six conditions are met:

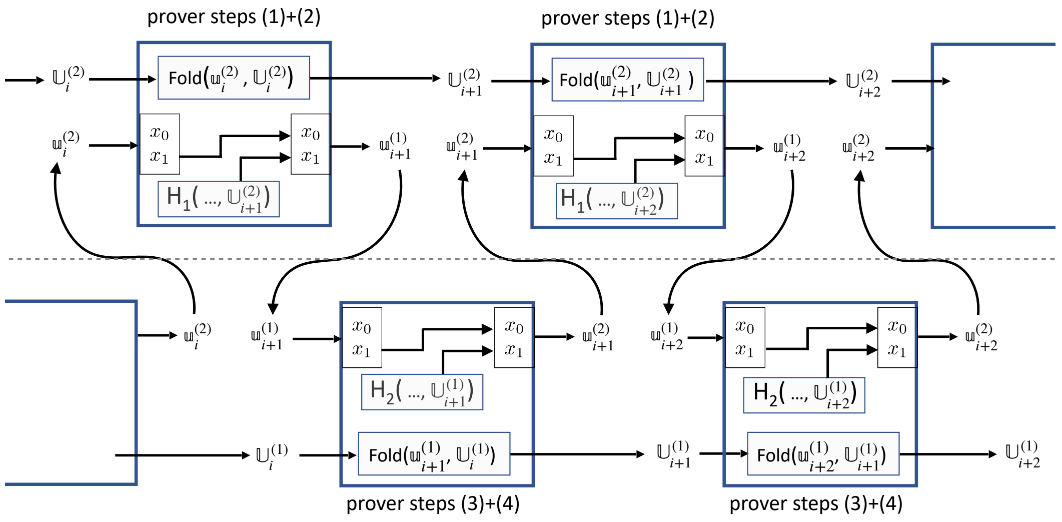
prover steps (1)+(2)
prover steps (1)+(2)
prover steps (3)+(4)
prover steps (3)+(4)
Fold($\mathbb{u}_i^{(2)}$, $\mathbb{U}_i^{(2)}$)
Fold($\mathbb{u}_{i+1}^{(2)}$, $\mathbb{U}_{i+1}^{(2)}$)
H$_1$(..., $\mathbb{U}_{i+1}^{(2)}$)
H$_1$(..., $\mathbb{U}_{i+2}^{(2)}$)
H$_2$(..., $\mathbb{U}_{i+1}^{(1)}$)
H$_2$(..., $\mathbb{U}_{i+2}^{(1)}$)
Fold($\mathbb{u}_{i+1}^{(1)}$, $\mathbb{U}_i^{(1)}$)
Fold($\mathbb{u}_{i+2}^{(1)}$, $\mathbb{U}_{i+1}^{(1)}$)
x_0
x_1
$\mathbb{U}_i^{(2)}$
$\mathbb{u}_i^{(2)}$
$\mathbb{U}_{i+1}^{(2)}$
$\mathbb{u}_{i+1}^{(1)}$
$\mathbb{u}_{i+1}^{(2)}$
$\mathbb{U}_{i+2}^{(2)}$
$\mathbb{u}_{i+2}^{(1)}$
$\mathbb{u}_{i+2}^{(2)}$
$\mathbb{U}_i^{(1)}$
$\mathbb{U}_{i+1}^{(1)}$
$\mathbb{U}_{i+2}^{(1)}$

Figure 2 An illustration of the key parts of the prover's operation in the non-base case.

1. The index i must be greater than 0.
2. $\mathbb{u}_i^{(2)}.x_0 = \mathsf{H}_1\big(\mathsf{vk}, i^{(1)}, z_0^{(1)}, z_i^{(1)}, \mathbb{U}_i^{(2)}\big)$
3. $\mathbb{u}_i^{(2)}.x_1 = \mathsf{H}_2\big(\mathsf{vk}, i^{(2)}, z_0^{(2)}, z_i^{(2)}, \mathbb{U}_i^{(1)}\big)$
4. The pair $(\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})$ satisfies $\mathsf{R1CS}^{(1)}$.
5. The pair $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ satisfies $\mathsf{R1CS}^{(2)}$.
6. The pair $(\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)})$ **strictly** satisfies $\mathsf{R1CS}^{(2)}$.

5.3 The Modified Nova Prover

In this section, we describe a modified 2-cycle Nova IVC prover. The algorithm is similar to the prior 2-cycle Nova IVC prover, but the generated IVC proof π_{i+1} omits a pair $(\mathbb{u}_{i+1}^{(1)}, \mathbb{w}_{i+1}^{(1)})$, which caused the original vulnerability (Section 7.1). We first describe an initial procedure, then the base case step of the Nova prover, and then the recursive step as illustrated in Figure 2.

5.3.1 Initial Procedure

The prover performs an initial procedure identical to the initial procedure of the verifier:

1. Given functions F_1 and F_2, deterministically generate augmented R1CS constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ which implement relations $\mathcal{R}_1$ and $\mathcal{R}_2$.
2. Compute the folding prover and verifier key

$$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Fold}_{\mathcal{K}}\big(\mathsf{pp},\ \mathsf{R1CS} := (\mathsf{R1CS}^{(1)}, \mathsf{R1CS}^{(2)})\big)$$

5.3.2 The Base Case

The Nova Prover $\mathcal{P}$ takes in as input:

- IVC public parameters pp.
- A description of functions $\mathsf{F}_1 : \mathbb{F}_1^{a_1} \times \mathbb{F}_1^{b_1} \to \mathbb{F}_1^{a_1}$ and $\mathsf{F}_2 : \mathbb{F}_2^{a_2} \times \mathbb{F}_2^{b_2} \to \mathbb{F}_2^{a_2}$.
- Starting values $z_0^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_0^{(2)} \in \mathbb{F}_2^{a_2}$.
- Auxiliary inputs $\mathsf{aux}_0^{(1)} \in \mathbb{F}_1^{b_1}$ and $\mathsf{aux}_0^{(2)} \in \mathbb{F}_2^{b_2}$.

The prover proceeds as follows:

- **Compute New Pair for $\mathsf{R1CS}^{(1)}$:** Compute the new committed pair $(\mathbbm{u}_1^{(1)}, \mathbbm{w}_1^{(1)})$ for $\mathsf{R1CS}^{(1)}$ as follows:
  - Define an initial dummy instance as $\mathbbm{u}_0^{(2)} := \big(\bar{0}^{(2)},\ 1^{(2)},\ \bar{0}^{(2)},\ x := (x_0,\ x_1)\big)$ where

    $$x_0 := \mathsf{H}_1\big(\mathsf{vk},\ 0^{(1)},\ z_0^{(1)},\ z_0^{(1)},\ \mathbb{U}_\perp^{(2)}\big) \qquad \text{and} \qquad x_1 := \mathsf{H}_2\big(\mathsf{vk},\ 0^{(2)},\ z_0^{(2)},\ z_0^{(2)},\ \mathbb{U}_\perp^{(1)}\big)$$

    This instance will not be folded into any running instance.
  - Define $\hat{w}_1^{(1)} := (\mathsf{vk}, 0^{(1)}, z_0^{(1)}, z_0^{(1)}, \mathsf{aux}_0^{(1)}, \mathbb{U}_\perp^{(2)}, \mathbbm{u}_0^{(2)}, \bar{0}^{(2)})$ as the relation witness for $\mathcal{R}_1$. Then, compute the extended witness $w_1^{(1)}$ by performing the computation on $\hat{w}_1^{(1)}$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(1)}$.
  - Commit to the extended witness $\bar{\mathsf{w}}_1^{(1)} \leftarrow \mathsf{Commit}\big(\mathrm{pp}_W^{(1)}, w_1^{(1)}\big)$.
  - Define $\mathbb{U}_1^{(2)} := \mathbb{U}_\perp^{(2)}$ and $\mathbb{W}_1^{(2)} := \mathbb{W}_\perp^{(2)}$.
  - Define $x_0 := \mathbbm{u}_0^{(2)}.x_1$ and $x_1 := \mathsf{H}_1\big(\mathsf{vk}, 1^{(1)}, z_0^{(1)}, z_1^{(1)} := \mathsf{F}_1(z_0^{(1)}, \mathsf{aux}_0^{(1)}), \mathbb{U}_1^{(2)}\big)$.
  - Assign $\mathbbm{u}_1^{(1)} := \big(\bar{0}^{(1)}, 1^{(1)}, \bar{\mathsf{w}}_1^{(1)}, (x_0, x_1)\big)$ and $\mathbbm{w}_1^{(1)} := \big(\vec{0}^{(1)}, w_1^{(1)}\big)$.
- **Compute New Pair for $\mathsf{R1CS}^{(2)}$:** Compute the new committed pair $(\mathbbm{u}_1^{(2)}, \mathbbm{w}_1^{(2)})$ for $\mathsf{R1CS}^{(2)}$ as follows:
  - Define $\hat{w}_1^{(2)} := (\mathsf{vk}, 0^{(2)}, z_0^{(2)}, z_0^{(2)}, \mathsf{aux}_0^{(2)}, \mathbb{U}_\perp^{(1)}, \mathbbm{u}_1^{(1)}, \bar{0}^{(1)})$ as the relation witness for $\mathcal{R}_2$. Then, compute the extended witness $w_1^{(2)}$ by performing the computation on $\hat{w}_1^{(2)}$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(2)}$.
  - Commit to the extended witness $\bar{\mathsf{w}}_1^{(2)} \leftarrow \mathsf{Commit}(\mathrm{pp}_W^{(2)}, w_1^{(2)})$.
  - Define $\mathbb{U}_1^{(1)} := \mathbbm{u}_1^{(1)}$ and $\mathbb{W}_1^{(1)} := \mathbbm{w}_1^{(1)}$.
  - Define $x_0 := \mathbbm{u}_1^{(1)}.x_1$ and compute $x_1 := \mathsf{H}_2\big(\mathsf{vk}, 1^{(2)}, z_0^{(2)}, z_1^{(2)} := \mathsf{F}_2(z_0^{(2)}, \mathsf{aux}_0^{(2)}), \mathbb{U}_1^{(1)}\big)$.
  - Assign $\mathbbm{u}_1^{(2)} := \big(\bar{0}^{(2)}, 1^{(2)}, \bar{\mathsf{w}}_1^{(2)}, (x_0, x_1)\big)$ and $\mathbbm{w}_1^{(2)} := \big(\vec{0}^{(2)}, w_1^{(2)}\big)$.
- **Output Prover State:** Output IVC Proof for step 1

  $$\pi_1 := \big((\mathbbm{u}_1^{(2)}, \mathbbm{w}_1^{(2)}),\ \ (\mathbb{U}_1^{(1)}, \mathbb{W}_1^{(1)}),\ \ (\mathbb{U}_1^{(2)}, \mathbb{W}_1^{(2)})\big)$$

  along with new evaluations $z_1^{(1)} := \mathsf{F}_1(z_0^{(1)}, \mathsf{aux}_0^{(1)})$ and $z_1^{(2)} := \mathsf{F}_2(z_0^{(2)}, \mathsf{aux}_0^{(2)})$. These outputs are sufficient to execute another step of the Nova prover for iteration 1.

5.3.3 The Non-Base Case

The Nova Prover $\mathcal{P}$ takes in as input:

- IVC public parameters pp.
- Constraint Systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$.
- An index $i \in \mathbb{N}$, where $i \geq 1$.
- Starting values $z_0^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_0^{(2)} \in \mathbb{F}_2^{a_2}$.
- Evaluations $z_i^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_i^{(2)} \in \mathbb{F}_2^{a_2}$.
- Auxiliary inputs $\mathsf{aux}_i^{(1)} \in \mathbb{F}_1^{b_1}$ and $\mathsf{aux}_i^{(2)} \in \mathbb{F}_2^{b_2}$.
- An IVC Proof for Iteration i $\quad \pi_i := \big((\mathbbm{u}_i^{(2)}, \mathbbm{w}_i^{(2)}),\ (\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)}),\ (\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})\big)$.

The prover proceeds as follows (see also Figure 2):

1. **Fold Prior Pairs for $\mathsf{R1CS}^{(2)}$:** Fold the committed pairs $(\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)})$ and $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ for $\mathsf{R1CS}^{(2)}$.

$$\mathsf{Fold}_{\mathcal{P}}\big(\mathsf{pk}, (\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)}), (\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})\big) \to \big(\bar{\mathsf{T}}_i^{(2)}, (\mathbb{U}_{i+1}^{(2)}, \mathbb{W}_{i+1}^{(2)})\big)$$

Obtain a folding proof $\bar{\mathsf{T}}_i^{(2)}$ and new committed relaxed instance-witness pair $(\mathbb{U}_{i+1}^{(2)}, \mathbb{W}_{i+1}^{(2)})$.

2. **Compute New Pair for $\mathsf{R1CS}^{(1)}$:** Compute the new committed pair $(\mathbb{u}_{i+1}^{(1)}, \mathbb{w}_{i+1}^{(1)})$ for $\mathsf{R1CS}^{(1)}$ as follows:
   - Define $\hat{w}_{i+1}^{(1)} := (\mathsf{vk}, i^{(1)}, z_0^{(1)}, z_i^{(1)}, \mathsf{aux}_i^{(1)}, \mathbb{U}_i^{(2)}, \mathbb{u}_i^{(2)}, \bar{\mathsf{T}}_i^{(2)})$ as the relation witness for $\mathcal{R}_1$. Then, compute the extended witness $w_{i+1}^{(1)}$ by performing the computation on $\hat{w}_{i+1}^{(1)}$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(1)}$.
   - Commit to the extended witness $\bar{\mathsf{w}}_{i+1}^{(1)} \leftarrow \mathsf{Commit}(\mathrm{pp}_W^{(1)}, w_{i+1}^{(1)})$.
   - Define $x_0 := \mathbb{u}_i^{(2)}.x_1$ and $x_1 := \mathsf{H}_1\big(\mathsf{vk}, (i+1)^{(1)}, z_0^{(1)}, z_{i+1}^{(1)} := \mathsf{F}_1(z_i^{(1)}, \mathsf{aux}_i^{(1)}), \mathbb{U}_{i+1}^{(2)}\big)$.
   - Assign $\mathbb{u}_{i+1}^{(1)} := \big(\bar{0}^{(1)}, 1^{(1)}, \bar{\mathsf{w}}_{i+1}^{(1)}, (x_0, x_1)\big)$ and $\mathbb{w}_{i+1}^{(1)} := \big(\vec{0}^{(1)}, w_{i+1}^{(1)}\big)$.

3. **Fold Pairs for $\mathsf{R1CS}^{(1)}$:** Fold the newly computed pair $(\mathbb{u}_{i+1}^{(1)}, \mathbb{w}_{i+1}^{(1)})$ with the committed pair $(\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})$ for $\mathsf{R1CS}^{(1)}$.

$$\mathsf{Fold}_{\mathcal{P}}\big(\mathsf{pk}, (\mathbb{u}_{i+1}^{(1)}, \mathbb{w}_{i+1}^{(1)}), (\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})\big) \to \big(\bar{\mathsf{T}}_i^{(1)}, (\mathbb{U}_{i+1}^{(1)}, \mathbb{W}_{i+1}^{(1)})\big)$$

Obtain a folding proof $\bar{\mathsf{T}}_i^{(1)}$ and new committed relaxed instance-witness pair $(\mathbb{U}_{i+1}^{(1)}, \mathbb{W}_{i+1}^{(1)})$.

4. **Compute New Pair for $\mathsf{R1CS}^{(2)}$:** Compute the new committed pair $(\mathbb{u}_{i+1}^{(2)}, \mathbb{w}_{i+1}^{(2)})$ for $\mathsf{R1CS}^{(2)}$ as follows:
   - Define $\hat{w}_{i+1}^{(2)} := (\mathsf{vk}, i^{(2)}, z_0^{(2)}, z_i^{(2)}, \mathsf{aux}_i^{(2)}, \mathbb{U}_i^{(1)}, \mathbb{u}_{i+1}^{(1)}, \bar{\mathsf{T}}_i^{(1)})$ as the relation witness for $\mathcal{R}_2$. Then, compute the extended witness $w_{i+1}^{(2)}$ by performing the computation on $\hat{w}_{i+1}^{(2)}$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(2)}$.
   - Commit to the extended witness $\bar{\mathsf{w}}_{i+1}^{(2)} \leftarrow \mathsf{Commit}(\mathrm{pp}_W^{(2)}, w_{i+1}^{(2)})$.
   - Define $x_0 := \mathbb{u}_{i+1}^{(1)}.x_1$ and $x_1 := \mathsf{H}_2\big(\mathsf{vk}, (i+1)^{(2)}, z_0^{(2)}, z_{i+1}^{(2)} := \mathsf{F}_2(z_i^{(2)}, \mathsf{aux}_i^{(2)}), \mathbb{U}_{i+1}^{(1)}\big)$.
   - Assign $\mathbb{u}_{i+1}^{(2)} := \big(\bar{0}^{(2)}, 1^{(2)}, \bar{\mathsf{w}}_{i+1}^{(2)}, (x_0, x_1)\big)$ and $\mathbb{w}_{i+1}^{(2)} := \big(\vec{0}^{(2)}, w_{i+1}^{(2)}\big)$.

5. **Output Prover State:** Output IVC Proof for step $i+1$

$$\pi_{i+1} := \big((\mathbb{u}_{i+1}^{(2)}, \mathbb{w}_{i+1}^{(2)}),\ (\mathbb{U}_{i+1}^{(1)}, \mathbb{W}_{i+1}^{(1)}),\ (\mathbb{U}_{i+1}^{(2)}, \mathbb{W}_{i+1}^{(2)})\big)$$

along with new evaluations $z_{i+1}^{(1)} := \mathsf{F}_1(z_i^{(1)}, \mathsf{aux}_i^{(1)})$ and $z_{i+1}^{(2)} := \mathsf{F}_2(z_i^{(2)}, \mathsf{aux}_i^{(2)})$. These outputs are sufficient to execute another step of the Nova prover for iteration $i+1$.

This completes our description of the prover.

6 Proof of security

▶ **Theorem 9.** *If the non-interactive folding scheme is knowledge sound (Definition 7) and the hash function is collision resistant (Definition 8), then our modified Nova IVC scheme is knowledge sound (Definition 1).*

The proof of Theorem 9 can be found in the full version of our paper (eprint.iacr.org/2023/969).

7 The Original Nova Vulnerability

In this section, we describe the prior implementation of the Nova Verifier and the vulnerability in detail. At the end, we provide a proof of concept attack against the Minroot VDF [10] Nova verifier.

7.1 The Prior (Vulnerable) Nova Verifier

Before our patch added on 05/18/2023, the prior (vulnerable) 2-cycle Nova IVC Verifier $\mathcal{V}$ took in as input:

- Constraint Systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$.
- An index $i \in \mathbb{N}$.
- Starting values $z_0^{(1)} \in \mathbb{F}_1,\ z_0^{(2)} \in \mathbb{F}_2$.
- Claimed evaluations $z_i^{(1)} \in \mathbb{F}_1,\ z_i^{(2)} \in \mathbb{F}_2$
- An IVC Proof for iteration i $\pi_i := \big((\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)}),\ (\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)}),\ (\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)}),\ (\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})\big)$.

▶ Remark 10. The prior IVC proof π_i contained an additional instance-witness pair $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$. This pair is no longer included in our modified verifier Section 5.2. As we explain in Section 7.2, the inclusion of these elements (along with misplaced checks) lead to the vulnerability.

The verifier performs an initial procedure:

1. Given functions F_1 and F_2, deterministically generate augmented R1CS constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ which implement relations $\mathcal{R}_1$ and $\mathcal{R}_2$ from Figures 1a and 1b
2. Compute the folding verification key

$$(\,\cdot\,, \mathsf{vk}) \leftarrow \mathsf{Fold}_{\mathcal{K}}\big(\mathsf{pp},\ \mathsf{R1CS} := (\mathsf{R1CS}^{(1)}, \mathsf{R1CS}^{(2)})\big)$$

The verifier accepts if the following conditions are met:

1. The index i must be greater than 0.
2. $\mathbb{u}_i^{(1)}.x_1 = \mathsf{H}_1\left(\mathsf{vk}, i^{(1)}, z_0^{(1)}, z_i^{(1)}, \mathbb{U}_i^{(2)}\right)$
3. $\mathbb{u}_i^{(2)}.x_1 = \mathsf{H}_2\left(\mathsf{vk}, i^{(2)}, z_0^{(2)}, z_i^{(2)}, \mathbb{U}_i^{(1)}\right)$
4. Pair $(\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})$ satisfies $\mathsf{R1CS}^{(1)}$.
5. Pairs $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ satisfies $\mathsf{R1CS}^{(2)}$.
6. Pair $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$ strictly satisfies $\mathsf{R1CS}^{(1)}$.
7. Pair $(\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)})$ strictly satisfies $\mathsf{R1CS}^{(2)}$.

7.2 The Vulnerability

In this section we first break down the implications of the verifier checks. Then, we explore a vulnerability with the approach. Finally, we describe a process to forge convincing IVC proofs in two stages.

Informally, for $i > 2$, the security argument for Nova IVC proceeds as follows:

- The verifier checks that $\mathbb{u}_i^{(1)}.x_1 = \mathsf{H}_1\left(\mathsf{vk}, i^{(1)}, z_0^{(1)}, z_i^{(1)}, \mathbb{U}_i^{(2)}\right)$. This ensures that $\mathbb{u}_i^{(1)}.x_1$ is derived from the inputs $z_i^{(1)}$ and $\mathbb{U}_i^{(2)}$ that are provided to the verifier.
- The verifier checks that the pair $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$ satisfies $\mathsf{R1CS}^{(1)}$ which implements the relation $\mathcal{R}_1$. This implies two things:

- First, $\mathbb{U}_i^{(2)}$ is the result of folding the instances $\mathbb{u}_{i-1}^{(2)}$ and $\mathbb{U}_{i-1}^{(2)}$ specified in $\mathbb{w}_i^{(1)}$,
- Second, $\mathbb{u}_{i-1}^{(2)}.x_0 = \mathsf{H}_1\left(\mathsf{vk},\ (i-1)^{(1)},\ z_0^{(1)},\ z_{i-1}^{(1)},\ \mathbb{U}_{i-1}^{(2)}\right)$ where $z_i^{(1)} = \mathsf{F}_1(z_{i-1}^{(1)},\ \mathsf{aux}_{i-1}^{(1)})$ for some element $\mathsf{aux}_{i-1}^{(1)}$.

- The verifier checks that $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ satisfies $\mathsf{R1CS}^{(2)}$, which implements the relation $\mathcal{R}_2$. Then by knowledge soundness of the folding scheme, one can extract valid witnesses $\mathbb{w}_{i-1}^{(2)}$ for $\mathbb{u}_{i-1}^{(2)}$ and $\mathbb{W}_{i-1}^{(2)}$ for $\mathbb{U}_{i-1}^{(2)}$ with respect to $\mathsf{R1CS}^{(2)}$.
- Now, since $\mathbb{w}_{i-1}^{(2)}$ is a valid witness for $\mathbb{u}_{i-1}^{(2)}$, there are instances $\mathbb{u}_{i-1}^{(1)}$ and $\mathbb{U}_{i-2}^{(1)}$ specified in $\mathbb{w}_{i-1}^{(2)}$. By definition of $\mathcal{R}_2$, the instance $\mathbb{u}_{i-1}^{(1)}$ must satisfy $\mathbb{u}_{i-1}^{(1)}.x_1 = \mathbb{u}_{i-1}^{(2)}.x_0 = \mathsf{H}_1\left(\mathsf{vk}, (i-1)^{(1)}, z_0^{(1)}, z_{i-1}^{(1)}, \mathbb{U}_{i-1}^{(2)}\right)$.

We would now like to conclude that both $\mathbb{u}_{i-1}^{(1)}$ and $\mathbb{U}_{i-2}^{(1)}$ are satisfiable for $\mathsf{R1CS}^{(1)}$. However, none of the verifier checks or invariants induced by the relations imply that either $\mathbb{u}_{i-1}^{(1)}$ or $\mathbb{U}_{i-2}^{(1)}$ are satisfiable with respect to $\mathsf{R1CS}^{(1)}$. To see why, observe that $\mathcal{R}_2$ verifies that $\mathbb{u}_{i-1}^{(1)}$ and $\mathbb{U}_{i-2}^{(1)}$ fold into some $\mathbb{U}_{i-1}^{(1)}$. Then this $\mathbb{U}_{i-1}^{(1)}$ is hashed into $\mathbb{u}_{i-1}^{(2)}.x_1$, which gets copied to $\mathbb{u}_i^{(1)}.x_0$. The verifier is given an instance $\mathbb{U}_i^{(1)}$ that it expects to be the result of folding $\mathbb{u}_i^{(1)}$ and $\mathbb{U}_{i-1}^{(1)}$, but this need not be the case. In fact, $\mathbb{U}_i^{(1)}$ can be the result of folding entirely different $\mathbb{u}^{(1)}$ and $\mathbb{U}^{(1)}$.

Our attack exploits this by running the honest Nova prover for two stages. The first stage generates a satisfiable instance $\mathbb{u}_{i-1}^{(2)}$ with x_0 containing our own adversarially chosen values of $(i-1)^{(1)}$ and $z_{i-1}^{(1)}$. Then, the second stage generates pairs $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$, $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ by running the honest prover again with $\mathbb{U}_\perp^{(2)}$, $\mathbb{u}_{i-1}^{(2)}$ as relational witness inputs. The attack proceeds symmetrically to generate pairs $(\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)})$, $(\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})$.

7.3 Attack Procedure

Our adversary $\mathcal{A}$ takes in as input:

- Constraint Systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$.
- An attack index $i > 2 \in \mathbb{N}$.
- Arbitrary starting values $z_0^{(1)} \in \mathbb{F}_1$ and $z_0^{(2)} \in \mathbb{F}_2$.
- Arbitrary claimed evaluations $z_i^{(1)} \in \mathbb{F}_1$ and $z_i^{(2)} \in \mathbb{F}_2$.
- Preimages $(z_{i-1}^{(1)},\ \mathsf{aux}_{i-1}^{(1)}) \in \mathbb{F}_1$ and $(z_{i-1}^{(2)},\ \mathsf{aux}_{i-1}^{(2)}) \in \mathbb{F}_2$ such that $z_i^{(1)} = \mathsf{F}_1(z_{i-1}^{(1)},\ \mathsf{aux}_{i-1}^{(1)})$ and $z_i^{(2)} = \mathsf{F}_2(z_{i-1}^{(2)},\ \mathsf{aux}_{i-1}^{(2)})$.

$\mathcal{A}$ will produce a false but convincing IVC proof π_i that the elements $z_i^{(1)} = \mathsf{F}_1^{(i)}(z_0^{(1)},\ \cdot)$ and $z_i^{(2)} = \mathsf{F}_2^{(i)}(z_0^{(2)},\ \cdot)$ are produced by iteratively applying the non-deterministic functions $\mathsf{F}_1,\ \mathsf{F}_2$ i-times on $z_0^{(1)},\ z_0^{(2)}$ for some collection of auxillary values $\{\mathsf{aux}_j^{(1)},\ \mathsf{aux}_j^{(2)}\}_{0 \le j < i}$.

Stage One. $\mathcal{A}$ will imitate an honest Nova Prover to produce a satisfying pair $(\mathbb{u}_{i-1}^{(2)}, \mathbb{w}_{i-1}^{(2)})$ for $\mathsf{R1CS}^{(2)}$, but with adversarial inputs.

1. **Produce Adversarial Instance:** We will produce an adversarial $\mathbb{u}_{i-1}^{(1)}$ by performing the following steps:
   a. Compute $x_0 := \mathsf{H}_2\left(\mathsf{vk}, (i-2)^{(2)}, z_0^{(2)}, z_{i-2}^{(2)}, \mathbb{U}_\perp^{(1)}\right)$, where $z_{i-2}^{(2)}$ can be set to anything, such as $\vec{0}^{(2)}$.
   b. Compute $x_1 := \mathsf{H}_1\left(\mathsf{vk}, (i-1)^{(1)}, z_0^{(1)}, z_{i-1}^{(1)}, \mathbb{U}_\perp^{(2)}\right)$.

c. Commit to the extended witness $\bar{\mathsf{w}}^{(1)}_{i-1} \leftarrow \mathsf{Commit}(\mathsf{pp}^{(1)}_W, w^{(1)}_{i-1})$, where the extended witness $w^{(1)}_{i-1}$ can be set to anything, such as $\vec{0}^{(2)}$.

d. Assign $\mathbb{u}^{(1)}_{i-1} := \left(\bar{0}^{(1)}, 1^{(1)}, \bar{\mathsf{w}}^{(1)}_{i-1}, (x_0, x_1)\right)$ and $\mathbb{w}^{(1)}_{i-1} := \left(\vec{0}^{(1)}, w^{(1)}_{i-1}\right)$.

2. **Fold Pair for $\mathsf{R1CS}^{(1)}$:** Fold the newly computed pair $(\mathbb{u}^{(1)}_{i-1}, \mathbb{w}^{(1)}_{i-1})$ with the trivially satisfiable pair $(\mathbb{U}^{(1)}_\perp, \mathbb{W}^{(1)}_\perp)$ for $\mathsf{R1CS}^{(1)}$.

$$\mathsf{Fold}_{\mathcal{P}}\left(\mathsf{pk}, (\mathbb{u}^{(1)}_{i-1}, \mathbb{w}^{(1)}_{i-1}), (\mathbb{U}^{(1)}_\perp, \mathbb{W}^{(1)}_\perp)\right) \rightarrow \left(\bar{\mathsf{T}}^{(1)}_{i-2}, (\mathbb{U}^{(1)}_{i-1}, \mathbb{W}^{(1)}_{i-1})\right)$$

Obtain a folding proof $\bar{\mathsf{T}}^{(1)}_{i-2}$ and new committed relaxed instance-witness pair $(\mathbb{U}^{(1)}_{i-1}, \mathbb{W}^{(1)}_{i-1})$.

3. **Compute New Pair for $\mathsf{R1CS}^{(2)}$:** Compute the new committed pair $(\mathbb{u}^{(2)}_{i-1}, \mathbb{w}^{(2)}_{i-1})$ for $\mathsf{R1CS}^{(2)}$ as follows:
   - Define $\hat{w}^{(2)}_{i-1} := (\mathsf{vk}, (i-2)^{(2)}, z^{(2)}_0, z^{(2)}_{i-2}, \mathsf{aux}^{(2)}_{i-2}, \mathbb{U}^{(1)}_\perp, \mathbb{u}^{(1)}_{i-1}, \bar{\mathsf{T}}^{(1)}_{i-2})$ as the relation witness for $\mathcal{R}_2$, where $\mathsf{aux}^{(2)}_{i-2}$ can be set to anything, such as $\vec{0}^{(2)}$. Then, compute the extended witness $w^{(2)}_{i-1}$ by performing the computation on $\hat{w}^{(2)}_{i-1}$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(2)}$.
   - Commit to the extended witness $\bar{\mathsf{w}}^{(2)}_{i-1} \leftarrow \mathsf{Commit}(\mathsf{pp}^{(2)}_W, w^{(2)}_{i-1})$.
   - Define $x_0 = \mathbb{u}^{(1)}_{i-1}.x_1$ and compute $x_1 = \mathsf{H}_2\left(\mathsf{vk}, (i-1)^{(2)}, z^{(2)}_0, \mathsf{F}^{(2)}_2(z^{(2)}_{i-2}, \mathsf{aux}^{(2)}_{i-2}), \mathbb{U}^{(1)}_{i-1}\right)$.
   - Assign $\mathbb{u}^{(2)}_{i-1} := \left(\bar{0}^{(2)}, 1^{(2)}, \bar{\mathsf{w}}^{(2)}_{i-1}, (x_0, x_1)\right)$ and $\mathbb{w}^{(2)}_{i-1} := \left(\vec{0}^{(2)}, w^{(2)}_{i-1}\right)$.

This new committed instance-witness pair $(\mathbb{u}^{(2)}_{i-1}, \mathbb{w}^{(2)}_{i-1})$ is valid, because the computation performed above explicitly satisfies the constraints of $\mathsf{R1CS}^{(2)}$. Furthermore, $\mathbb{u}^{(2)}_{i-1}.x_0 = \mathbb{u}^{(1)}_{i-1}.x_1$, which is maliciously set to $\mathsf{H}_1\left(\mathsf{vk}, (i-1)^{(1)}, z^{(1)}_0, z^{(1)}_{i-1}, \mathbb{U}^{(2)}_\perp\right)$.

Stage Two. $\mathcal{A}$ will imitate an honest Nova Prover for $\mathsf{R1CS}^{(1)}$, but with witness input derived in stage one.

1. **Fold Pair for $\mathsf{R1CS}^{(2)}$:** Fold the newly computed pair $(\mathbb{u}^{(2)}_{i-1}, \mathbb{w}^{(2)}_{i-1})$ with the trivially satisfiable pair $(\mathbb{U}^{(2)}_\perp, \mathbb{W}^{(2)}_\perp)$ for $\mathsf{R1CS}^{(2)}$.

$$\mathsf{Fold}_{\mathcal{P}}\left(\mathsf{pk}, (\mathbb{u}^{(2)}_{i-1}, \mathbb{w}^{(2)}_{i-1}), (\mathbb{U}^{(2)}_\perp, \mathbb{W}^{(2)}_\perp)\right) \rightarrow \left(\bar{\mathsf{T}}^{(2)}_{i-1}, (\mathbb{U}^{(2)}_{i}, \mathbb{W}^{(2)}_{i})\right)$$

Obtain a folding proof $\bar{\mathsf{T}}^{(2)}_{i-1}$ and new committed relaxed instance-witness pair $(\mathbb{U}^{(2)}_i, \mathbb{W}^{(2)}_i)$. Note that since both pairs are satisfiable, this new pair $(\mathbb{U}^{(2)}_i, \mathbb{W}^{(2)}_i)$ is also satisfiable.

2. **Compute New Pair for $\mathsf{R1CS}^{(1)}$:** Compute the new committed pair $(\mathbb{u}^{(1)}_i, \mathbb{w}^{(1)}_i)$ for $\mathsf{R1CS}^{(1)}$ as follows:
   - Define $\hat{w}^{(1)}_i := (\mathsf{vk}, (i-1)^{(1)}, z^{(1)}_0, z^{(1)}_{i-1}, \mathsf{aux}^{(1)}_{i-1}, \mathbb{U}^{(2)}_\perp, \mathbb{u}^{(2)}_{i-1}, \bar{\mathsf{T}}^{(2)}_{i-1})$ as the relation witness for $\mathcal{R}_1$. Then, compute the extended witness $w^{(1)}_i$ by performing the computation on $\hat{w}^{(1)}_i$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(1)}$.
   - Commit to the extended witness $\bar{\mathsf{w}}^{(1)}_i \leftarrow \mathsf{Commit}(\mathsf{pp}^{(1)}_W, w^{(1)}_i)$.
   - Define $x_0 = \mathbb{u}^{(2)}_{i-1}.x_1$ and compute $x_1 = \mathsf{H}_1\left(\mathsf{vk}, i^{(1)}, z^{(1)}_0, z^{(1)}_i := \mathsf{F}_1(z^{(1)}_{i-1}, \mathsf{aux}^{(1)}_{i-1}), \mathbb{U}^{(2)}_i\right)$.
   - Assign $\mathbb{u}^{(1)}_i := \left(\bar{0}^{(1)}, 1^{(1)}, \bar{\mathsf{w}}^{(1)}_i, (x_0, x_1)\right)$ and $\mathbb{w}^{(1)}_i := \left(\vec{0}^{(1)}, w^{(1)}_i\right)$.

This new committed instance-witness pair $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$ is satisfiable, because the computation performed above explicitly satisfies the constraints of $\mathsf{R1CS}^{(1)}$. Furthermore, $\mathbb{u}_i^{(1)}.x_1 = \mathsf{H}_1\left(\mathsf{vk}, i^{(1)}, z_0^{(1)}, z_i^{(1)}, \mathbb{U}_i^{(2)}\right)$. To recap, after these two stages we obtained pairs $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$ and $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ such that the following hold:

- $\mathbb{u}_i^{(1)}.x_1 = \mathsf{H}_1\left(\mathsf{vk}, i^{(1)}, z_0^{(1)}, z_i^{(1)}, \mathbb{U}_i^{(2)}\right)$
- $(\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)})$ satisfies $\mathsf{R1CS}^{(1)}$.
- $(\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})$ satisfies $\mathsf{R1CS}^{(2)}$.

Symmetry. Since the relations expressed by the augmented constraint systems $\mathsf{R1CS}^{(1)}$ and $\mathsf{R1CS}^{(2)}$ are symmetric (Section 4) when $i > 2$. We can repeat both stages above symmetrically to produce pairs $(\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)})$ and $(\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})$ such that the following hold:

- $\mathbb{u}_i^{(2)}.x_1 = \mathsf{H}_2\left(\mathsf{vk}, i^{(2)}, z_0^{(2)}, z_i^{(2)}, \mathbb{U}_i^{(1)}\right)$
- $(\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)})$ satisfies $\mathsf{R1CS}^{(2)}$.
- $(\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)})$ satisfies $\mathsf{R1CS}^{(1)}$.

Finally, adversary $\mathcal{A}$ outputs an IVC proof $\pi_i := \left((\mathbb{u}_i^{(1)}, \mathbb{w}_i^{(1)}),\ (\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)}),\ (\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)}),\ (\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})\right)$. By construction, π_i is a convincing IVC proof (i.e. all the verifier checks pass).

7.3.1 Proof of Concept Attack Against the Minroot Verifier

We implement our attack against the prior 2-cycle Nova Proof System generically for any choice of F_1 and F_2 and parameters specified in Section 7.3. Our implementation can be found in our repo MercysJest/NovaBreakingTheCycleAttack, which is a direct fork of the original microsoft/Nova repo. To demonstrate the attack, we can compute a convincing Nova proof for the correct evaluation of 2^{75} rounds of the Minroot VDF in only 116 milliseconds on a Macbook. The demonstration code can be found in examples/vuln.rs.

```
=========================================================================
Demonstrating exploit against Nova-based VDF with MinRoot delay function
=========================================================================
Producing public parameters...
PublicParams::setup, took 2.9136875s
...
Each IVC Step Performs 4096 iterations of Minroot.
Generating fake proof of 9223372036854775808 IVC Steps.
In total, faking 37778931862957161709568 Minroot iterations.
Generating fake proof took 115.872416ms
Verifying a RecursiveSNARK...
RecursiveSNARK::verify: true, took 27.8225ms
Generating a CompressedSNARK using Spartan with IPA-PC...
CompressedSNARK::prove: true, took 1.5859465s
CompressedSNARK::len 9713 bytes
Verifying a CompressedSNARK...
CompressedSNARK::verify: true, took 55.04425ms
```

8 Malleability of Nova's IVC proofs

In this section, we show that the 2-cycle Nova IVC proofs, described in Section 5 are malleable. This attack readily generalizes to the original (single chain) Nova construction [12]. We later discuss how to prevent this malleability attack by making use of either an additional ctx element in the verification key vk or use of a simulation-extractable zkSNARK (e.g., Spartan) for IVC proof compression.

Suppose an adversary is given a valid Nova IVC proof π_i with respect to the following parameters

- IVC public parameters pp,
- a description of functions $\mathsf{F}_1 : \mathbb{F}_1^{a_1} \times \mathbb{F}_1^{b_1} \to \mathbb{F}_1^{a_1}$ and $\mathsf{F}_2 : \mathbb{F}_2^{a_2} \times \mathbb{F}_2^{b_2} \to \mathbb{F}_2^{a_2}$,
- an index $i \in \mathbb{N}$,
- starting values $z_0^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_0^{(2)} \in \mathbb{F}_2^{a_2}$, and
- claimed evaluations $z_i^{(1)} \in \mathbb{F}_1^{a_1}$ and $z_i^{(2)} \in \mathbb{F}_2^{a_2}$.

We show in Section 8.1 that the adversary can construct a proof π_{prime} for the same iteration i, but for some $z_{\mathsf{prime}}^{(2)}$ different from $z_i^{(2)}$. In particular, running the IVC verifier with arguments

$$\big(\mathsf{pp},\ (\mathsf{F}_1, \mathsf{F}_2),\ i,\ (z_0^{(1)}, z_0^{(2)}),\ (z_i^{(1)}, z_{\mathsf{prime}}^{(2)}),\ \pi_{\mathsf{prime}}\ \big)$$

causes the verifier to accept. We stress that our adversary does not need to know the auxiliary values $(\mathsf{aux}_0^{(2)}, \mathsf{aux}_1^{(2)}, \ldots, \mathsf{aux}_{i-1}^{(2)})$ used to compute $z_i^{(2)}$. By choosing an alternate final auxiliary value $\mathsf{aux}_{\mathsf{prime}}^{(2)} \neq \mathsf{aux}_{i-1}^{(2)}$, our adversary can construct a proof π_{prime} for an alternate value $z_{\mathsf{prime}}^{(2)}$ for i iterations, without knowledge of the first $i-1$ auxiliary values. We discuss two ways to mitigate this issue in Section 8.2 below.

Why does this matter? A malleable proof system [5] can lead to a real world security vulnerability. Suppose Alice uses her secret auxiliary values to compute $z_i^{(2)}$ and this $z_i^{(2)}$ encodes her payment address. She sends the $z_i^{(2)}$ and the proof to a payment contract. An attacker could intercept her message and maul $z_i^{(2)}$ to a $z_{\mathsf{prime}}^{(2)}$ which encodes the attackers payment address instead, along with a valid proof π_{prime}. The payment contract will then send the funds to the attacker instead of Alice. Concretely, if Tornado Cash had used a proof system that were malleable on statements, it would have been possible to steal funds.

8.1 The Malleability Attack

We present a malleability attack on the last step of the IVC chain. Recall that the Nova IVC proof π_i contains the following elements

$$\pi_i := \big((\mathbb{u}_i^{(2)}, \mathbb{w}_i^{(2)}),\ (\mathbb{U}_i^{(1)}, \mathbb{W}_i^{(1)}),\ (\mathbb{U}_i^{(2)}, \mathbb{W}_i^{(2)})\big).$$

The malleability attack proceeds as follows:

1. **Parse witness:** In Section 6, we argued that we can parse the witness $\mathbb{w}_i^{(2)}$ to obtain relational witness

$$\hat{w}_i^{(2)} = \big(\mathsf{vk},\ (i-1)^{(2)},\ z_0^{(2)}, z_{i-1}^{(2)},\ \mathsf{aux}_{i-1}^{(2)},\ \mathbb{U}_{i-1}^{(1)}, \mathbb{u}_i^{(1)},\ \bar{\mathsf{T}}_{i-1}^{(1)}\big)$$

for which we know

$$z_i^{(2)} = \mathsf{F}_2\big(z_{i-1}^{(2)}, \mathsf{aux}_{i-1}^{(2)}\big)$$
$$\mathbb{U}_i^{(1)} = \mathsf{Fold}_{\mathcal{V}}\big(\mathsf{vk},\ \mathbb{U}_{i-1}^{(1)},\ \mathbb{u}_i^{(1)},\ \bar{\mathsf{T}}_{i-1}^{(1)}\big)$$

Thus, we parse $\mathbb{w}_i^{(2)}$ to obtain $\hat{w}_i^{(2)}$

2. **Find a different auxiliary value:** Using $z_{i-1}^{(2)}$, choose some $\mathsf{aux}_{\mathsf{prime}}^{(2)}$ such that

$$z_{\mathsf{prime}}^{(2)} := \mathsf{F}_2\big(z_{i-1}^{(2)},\ \mathsf{aux}_{\mathsf{prime}}^{(2)}\big) \quad \neq \quad \mathsf{F}_2\big(z_{i-1}^{(2)},\ \mathsf{aux}_{i-1}^{(2)}\big) = z_i^{(2)}$$

We assume that finding such an $\mathsf{aux}_{\mathsf{prime}}^{(2)}$ is efficient for F_2.

3. **Compute a new pair for $\mathsf{R1CS}^{(2)}$:** Compute the pair $(\mathbb{u}^{(2)}_{\mathsf{prime}}, \mathbb{w}^{(2)}_{\mathsf{prime}})$ for $\mathsf{R1CS}^{(2)}$ as follows:
   - Define $\hat{w}^{(2)}_{\mathsf{prime}} := (\mathsf{vk}, (i{-}1)^{(2)}, z^{(2)}_0, z^{(2)}_{i-1}, \mathsf{aux}^{(2)}_{\mathsf{prime}}, \mathbb{U}^{(1)}_{i-1}, \mathbb{u}^{(1)}_i, \bar{\mathsf{T}}^{(1)}_{i-i})$ as the relation witness for $\mathcal{R}_2$. Then, compute the extended witness $w^{(2)}_{\mathsf{prime}}$ by performing the computation on $\hat{w}^{(2)}_{\mathsf{prime}}$ required to satisfy the constraints expressed in $\mathsf{R1CS}^{(2)}$.
   - Commit to the extended witness $\bar{\mathsf{w}}^{(2)}_{\mathsf{prime}} \leftarrow \mathsf{Commit}(\mathrm{pp}^{(2)}_W, w^{(2)}_{\mathsf{prime}})$.
   - Define $x_0 := \mathbb{u}^{(1)}_i.x_1$ and $x_1 := \mathsf{H}_2\big(\mathsf{vk}, i^{(2)}, z^{(2)}_0, z^{(2)}_{\mathsf{prime}} := \mathsf{F}_2(z^{(2)}_{i-1}, \mathsf{aux}^{(2)}_{\mathsf{prime}}), \mathbb{U}^{(1)}_i\big)$.
   - Assign $\mathbb{u}^{(2)}_{\mathsf{prime}} := \big(\bar{0}^{(2)}, 1^{(2)}, \bar{\mathsf{w}}^{(2)}_{\mathsf{prime}}, (x_0, x_1)\big)$ and $\mathbb{w}^{(2)}_{\mathsf{prime}} := \big(\vec{0}^{(2)}, w^{(2)}_{\mathsf{prime}}\big)$.
4. **Output mauled proof:** Output $\pi_{\mathsf{prime}} := \big((\mathbb{u}^{(2)}_{\mathsf{prime}}, \mathbb{w}^{(2)}_{\mathsf{prime}}),\ (\mathbb{U}^{(1)}_i, \mathbb{W}^{(1)}_i),\ (\mathbb{U}^{(2)}_i, \mathbb{W}^{(2)}_i)\big)$.

By construction, the proof π_{prime} is convincing.

▶ Remark 11 (Generalizing the Malleability Attack). In more general terms, given an IVC proof that contains information about a valid pre-image z_{i-1} to z_i Our malleability attack re-executes the last step of the IVC prover with a different choice of the final auxiliary value aux_{i-1}. In particular, our attack readily generalizes to the original Nova construction [12].

8.2 Preventing This Malleability Attack

There are several strategies that defeat this specific malleability attack.

Incorporating Context Elements. The first approach is to expand the verification key vk to include a context string ctx which includes the IVC verifier context

$$(\,\cdot\,, \mathsf{vk}) \leftarrow \mathsf{Fold}_{\mathcal{K}}\big(\mathsf{pp},\ \mathsf{R1CS} := (\mathsf{R1CS}^{(1)}, \mathsf{R1CS}^{(2)}),\ \mathsf{ctx} := (i, z^{(1)}_i, z^{(2)}_i)\big)$$

This inductively binds the proof to a particular choice of $(i, z^{(1)}_i, z^{(2)}_i)$. However, this breaks the incremental property (namely, completeness after iteration i as described in Definition 1) of the proof since the prover cannot use this proof to produce another valid proof for an iteration $j > i$. Additionally, the IVC prover must compute the evaluations $(z^{(1)}_i, z^{(2)}_i)$ before generating the IVC proof.

Compression. A different defense is to use a compressed IVC proof π'_i, namely

$$\pi'_i := \big(\mathbb{u}^{(2)}_i,\ \mathbb{U}^{(2)}_i,\ \mathbb{U}^{(1)}_i,\ \bar{\mathsf{T}}^{(2)}_i,\ \pi_{\mathsf{sat}}\big) \tag{2}$$

where π_{sat} is a SNARK proof for the relation

$$\mathcal{R}_{\mathsf{sat}} := \left\{ \big(\mathbb{U}^{(2)}_{i+1},\ \mathbb{U}^{(1)}_i\ ;\ \mathbb{W}^{(2)}_{i+1},\ \mathbb{W}^{(1)}_i\big) \quad : \quad \begin{array}{c} (\mathbb{U}^{(1)}_i, \mathbb{W}^{(1)}_i) \text{ satisfies } \mathsf{R1CS}^{(1)} \\ \wedge\ (\mathbb{U}^{(2)}_{i+1}, \mathbb{W}^{(2)}_{i+1}) \text{ satisfies } \mathsf{R1CS}^{(2)} \end{array} \right\} \tag{3}$$

Here the SNARK must be zero knowledge so that π_{sat} contains no information about the underlying witnesses. Similarly, the witness commitment $\bar{\mathsf{w}}^{(2)}_i$ in $\mathbb{u}^{(2)}_i$ must be a hiding commitment. Furthermore, the SNARK may also need to be simulation extractable [9]. The Spartan SNARK [18] is simulation-extractable [3]. Unfortunately, applying the SNARK compression step would remove the efficient incremental property of the IVC since we can no longer run the native Nova IVC prover for subsequent iterations. Nova [12, 14] uses Spartan produce compressed IVC proofs π'_i (2).

References

1 Benedikt Bünz and Binyi Chen. ProtoStar: Generic efficient accumulation/folding for special sound protocols. Cryptology ePrint Archive, Paper 2023/620, 2023. URL: https://eprint.iacr.org/2023/620.

2 Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 681–710, Virtual Event, August 16–20 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84242-0_24.

3 Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 531–562, Lyon, France, April 23–27 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30617-4_18.

4 Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak fiat-shamir attacks on modern proof systems. Cryptology ePrint Archive, Paper 2023/691, 2023. URL: https://eprint.iacr.org/2023/691.

5 Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598, Santa Barbara, CA, USA, August 19–23 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_33.

6 Morris Dworkin. SHA-3 standard: Permutation-based hash and extendable-output functions, 2015-08-04 2015. doi:10.6028/NIST.FIPS.202.

7 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. doi:10.1007/3-540-47721-7_12.

8 Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 519–535. USENIX Association, August 11–13 2021.

9 Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459, Shanghai, China, December 3–7 2006. Springer, Heidelberg, Germany. doi:10.1007/11935230_29.

10 Dmitry Khovratovich, Mary Maller, and Pratyush Ranjan Tiwari. MinRoot: Candidate sequential function for Ethereum VDF. Cryptology ePrint Archive, Paper 2022/1626, 2022. URL: https://eprint.iacr.org/2022/1626.

11 Abhiram Kothapalli and Srinath Setty. HyperNova: Recursive arguments for customizable constraint systems. Cryptology ePrint Archive, Paper 2023/573, 2023. URL: https://eprint.iacr.org/2023/573.

12 Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 359–388, Santa Barbara, CA, USA, August 15–18 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15985-5_13.

13 Nicholas Mohnblatt. Sangria: A folding scheme for PLONK, 2023. link.

14 Nova Contributors. Nova implementation, 2022. URL: https://github.com/Microsoft/Nova.

15 oskarth. Towards a nova-based ZK virtual machine. https://zkresear.ch/t/towards-a-nova-based-zk-vm/105, 2023.

16 Pasta Contributors. Pasta curves, 2020. URL: https://github.com/zcash/pasta_curves.

17 Carla Ràfols and Alexandros Zacharakis. Folding schemes with selective verification. Cryptology ePrint Archive, Report 2022/1576, 2022. URL: https://eprint.iacr.org/2022/1576.

18 Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737, Santa Barbara, CA, USA, August 17–21 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56877-1_25.

19 Srinath Setty. Nova pull request 167, 2023. URL: https://github.com/Microsoft/Nova/pull/167.

20 Supernational. Open VDF: Accelerating the nova snark-based vdf. https://medium.com/supranational/open-vdf-accelerating-the-nova-snark-based-vdf-2d00737029bd, 2023.

21 Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18, San Francisco, CA, USA, March 19–21 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-78524-8_1.