WARPfold: Wrongfield ARithmetic for Protostar folding

Lev Soukhanov *

February 2024

Abstract

Inspired by range-check trick from recent Latticefold paper [4] we construct elliptic-curve based IVC capable of simulating non-native arithmetic efficiently.

We explain the general principle (which can be applied to both Protostar and Hypernova), and describe the Wrongfield ARithmetic for Protostar folding in details.

Our construction supports circuits over mutilple non-native fields simultaneously and allows interfacing between them using range-checked elements.

WARPfold can be used to warp between different proof systems and construct folding schemes over curves not admitting a dual partner (such as BLS12-381).

1 Introduction

Simulation of non-native arithmetic in proof systems is, typically, hard, yet most direct recursive constructions for elliptic curve based IVCs require it. One way of avoiding it is using a pair of dual curves, i.e. finding such a pair of elliptic curves that $E_1/\mathbb{F}_p$, $|E_1| = q$, $E_2/\mathbb{F}_q$, $|E_2| = p$.

This line of thoughts starts from [3], which uses direct SNARK recursion over a pair of pairing-friendly elliptic curves, but most examples today are either non pairing-friendly, or only use a single pairing-friendly curve with non pairing-friendly partner.

These IVC schemes, both recursion and folding, such as Halo2 [5], Nova [10] and its descendants - Hypernova [9] and Protostar [6], while technically

*Privacy Scaling Exporations, Ethereum Foundation. email:0xdeadfae@gmail.com

can be run on non-native arithmetic simulation, typically require a cycle of curves to be practical. The second curve circuit can be made relatively lightweight, as shown in Cyclefold [8].

This situation has some downsides. Most notably, the decider for the dual-curve IVC scheme is forced to be linear time at least in the size of one of the circuits (assuming other one has pairings). Naively wrapping it into the "main curve" proof system seems to incur simulation of multi-scalar multiplication of the size of the second circuit.

Another downside is the fact that there are currently no practically feasible approaches to running IVC over curves *not* allowing a dual partner. Notable such curves include the BLS family [1], gaining popularity in recent zero knowledge proof systems as it is less vulnerable to index calculus attacks than half-cycle friendly BN family [2].

Our source of inspiration is the recent Latticefold paper [4]. It uses Ajtai commitments to construct a post-quantum folding scheme, and, as these commitments are only binding for small vector values, does some range-check tricks to ensure they stay bounded during the folding process.

We observe that these tricks can be adapted back to the elliptic curve based world. We also make few modifications - namely, we do not "refresh" values on each folding, and instead ensure that they do not overflow by restricting the folding topology. This, sadly, means that tree-like folding is unavailable, but drastically reduces the amount of commitments being manipulated in a single recursive verifier. Another, rather trivial, modification is that we do not insist that our range-check is sumcheck-based, and instead use any lookup available in the base folding scheme.

Specifically, we describe our construction based on Protostar folding scheme; but it seems to be clear that others can be adapted in the same fashion.

Acknowledgement. Author would like to thank Yar Rebenko, Amir Ismailov, Benedikt Bunz, Liam Eagen, Arnaucube, Carlos Perez, Dohoon Kim, Soowon Jeong and Wanseob Lim for discussions on non-native arithmetic and folding schemes.

2 Preliminaries

We recall the definition of the Protostar folding scheme, with a slight optimization, resembling Protogalaxy [7]. This optimization is completely optional, and it trades 2 elliptic curve multiplications for some hashing work, which we believe is generally desirable.

We start by defining the circuit primitive.

Definition 1. *n-**round degree** d **homogeneous algebraic circuit** over field $\mathbb{F}$ consists of following data:*

1. *Vector spaces over $\mathbb{F}$,*

$$(P_0, ...P_{n-1}), (W_0, ..., W_{n-1})$$

P_i for $i > 0$ called challenge spaces, P_0 public witness, W_i - private witness. We will also call $Z_i = P_i \oplus W_i$ full witness, and conventionally denote its elements with letter z, for example $z_i = (p_i, w_i)$

2. *The constraint space C.*

3. *Degree d homogeneous **constraint mapping**,*

$$F : \bigoplus_i Z_i \to C$$

Patricular coordinates $c_i = F_i(z)$ are called "constraints", but no additional assumptions are put on them by Protostar system. Efficient decider might require some particular additional requirements on these constraints, such as adhering to R1CS or some other arithmetization.

We also always assume that P_0 contains a special variable u which is always set to 1. This allows us to express non-homogeneous circuits as homogeneous ones, and is called "relaxation factor".

Such algebraic circuit naturally corresponds to a public coin protocol - namely, given p_0, prover sends w_0, gets a uniformly random challenge response p_1, responds with w_1, and, after $2n - 1$ rounds of communication, verifier checks the constraint equations $F(z) = 0$.

In original Protostar terminology, this protocol is assumed to be a special-sound interactive protocol for some NP-relation R_{NP}. Then, Fiat-Shamir heuristic suggests the following definition.

Assume that we have a linearly homomorphic commitment scheme $\text{comm}_i : W_i \to G$ for some cryptographic group G.

Definition 2. ***Committed witness** to the multi-round circuit (Z, C, F) with public parameters p_0 is the witness $z \in Z$ satisfying the following properties:*

$$\forall i > 1 : \quad p_i = \text{Hash}(p_{i-1}, \text{comm}_{i-1}(w_{i-1}))$$

$$F(z) = 0$$

Committed instance *is a collection* $(p_0, ..., p_{n-1}), (g_0, ..., g_{n-1})$*, satisfying*

$$p_i = \text{Hash}(p_{i-1}, g_{i-1})$$

We refer to original Protostar paper [6] for the details and assumptions of the reduction of R_{NP} to the knowledge of committed witness of Fiat-Shamir transform of a protocol. In what follows, we will reason in terms of committed witnesses and associated relation $R^R_{\text{cm}}(z, g)$ saying that witness z corresponds to a committed instance g.

Next, we define our version of Protostar transform. Original transform turns an n-round protocol into $n+1$-round protocol, replacing $|C|$ constraints of degree d with $2\sqrt{|C|}$ constraints of degree 2, and a single equation of degree $d + 2$.

We typically use a slight modification, which instead replaces equations with a single equation of degree $d + \lceil \log |C| \rceil$, strongly resembling the layout from Protogalaxy [7]. All our constructions work the same in the original Protostar too.

Definition 3. ***Protostar-transform*** *(log-version) of a multiround circuit* $F : Z \to C$ *is defined as follows:*

A new round is appended to the witness space $\tilde{Z} = Z \oplus Z_n$*, without private variables at all* $W_n = 0$*, and with public variables* $P_n = (\gamma_0, ..., \gamma_{\lceil \log |C| \rceil - 1})$*.*

Denote $\ell = \lceil \log |C| \rceil$

The constraint space is replaced by a single one-dimensional constraint, with equation:

$$\tilde{F}(\tilde{z}) = \sum_{\substack{j \in \{0..|C|-1\} \\ b_0 + 2b_1 + 4b_2 + ... = j \\ \forall i: \quad b_i \in \{0,1\}}} F_j(z) \gamma_0^{b_0} \gamma_1^{b_1} ... \gamma_{\ell-1}^{b_{\ell-1}} u^{(d + \ell - \sum\limits_{0 \le i < \ell} b_i)} = 0$$

Note 1. *We have defined this transform for a homogeneous circuit. In case the circuit was non-homogeneous, the sum*

$$\sum_{\substack{j\in\{0..|C|-1\}\\ b_0+2b_1+4b_2+...=j\\ \forall i:\quad b_i\in\{0,1\}}} F_j(z)\gamma_0^{b_0}\gamma_1^{b_1}...\gamma_{\ell-1}^{b_{\ell-1}}$$

can be considered, and then homogenized. This also works, and can sometimes lead to smaller degree of the resulting circuit.

In what follows, we assume that we have our field larger than security parameter λ - because discrete logarithm problem is $\sqrt{|\mathbb{F}|}$ hard, its bitsize must be assumed to be at least 2λ. Challenges that we generate will be, typically, not uniform, but distributed in the segment $\{0..2^\lambda - 1\}$.

Lemma 1. *The protocol of the original circuit has a special-sound reduction to its protostar-transform even in a restricted case where verifier sends the following message in the last round: $\gamma_i = t^{2^i}$, for a single challenge $t < 2^\lambda$.*

This is, of course, very similar to what happens with original Protostar paper, Lemma 3. Indeed:

Proof. For a random challenge t, and $u = 1$ (recall that we always require that from our public inputs), the equation reduces to

$$\sum_{j\in 0..|C|-1} t^j F_j(z)$$

Given access to adversary's code, run it up to the last challenge to obtain a witness. In order to ensure that the witness is correct, we can fork the adversary $|C|$ times, and then feed it different values of t. Then, using Lagrange interpolation, we recover that all $F_i(z)$ indeed were 0. (alternative point of view is saying that unless all $F_i(z) = 0$, the adversary would not pass a random challenge t due to Schwartz-Zippel lemma, except with soundness error $\frac{|C|}{|2^\lambda|}$) □

Now, we can define the following relations:

Definition 4. ***Strict Protostar instance*** *is the committed instance to the Protostar-transform of a circuit, with last challenge being computed as*

$$t = \text{Hash}(p_{n-1}, g_{n-1})$$

$$\gamma_i = t^{2^i}$$

Note that it has no g_n, because the last round lacks private witness (and so commitment is trivial).

Strict protostar witness *to this instance is a witness, according to Definition 2.*

Definition 5. ***Relaxed Protostar instance*** *is the same as strict committed instance, but without any requirements on challenges being obtained as hashes, without requirement $u = 1$, and an additional value e, called "error term".*

Relaxed Protostar witness *is decommitment of all commitments g_i involved, satisfying the relaxed equation*

$$\tilde{F}(\tilde{z}) = e$$

Typically, folding schemes are presented as a protocol reducing knowledge of witness to some strict instance $\mathcal{U}$ and some relaxed instance $\mathcal{U}'$ to the knowledge of a single relaxed instance. However, there is also a reduction between from two relaxed instances to a single one.

WARPfold, sadly, does not support it - it can only fold together strict and relaxed instances.

Protocol 1. *Folding protocol starts with two instances, strict instance $\mathcal{U}$ and relaxed instance $\mathcal{U}'$. Prover claims that it knows corresponding witnesses w, w'*

1. *Prover sends a $d+\ell-1$-degree univariate polynomial $e(t) = \tilde{F}(\tilde{z'}+t\tilde{z})$.*
2. *Verifier checks that $e(0) = e'$. (Note that the fact that this polynomial has degree $d + \ell - 1$ stems from condition $e = 0$ - that $\mathcal{U}$ is strict. For two relaxed instances verifier would also check that highest coefficient $\lim_{t\to\infty} e(t)/t^{d+\ell} = e$).*
3. *Verifier samples challenge c, and the new instance is set as follows:*

$$g_i'' = g_i' + cg_i$$

$$p_i'' = p_i' + cp_i$$

$$e'' = e(c)$$

Proof. The soundness of this reduction is proven in essentially the same way as in original paper. We only provide sketch here, referring to the original paper for details.

Provided that we have an adversary that outputs w'', our goal is to extract w, w' back. This is done by forking it just before the challenge c arrives, and, having witnesses w_α to the commitment $g' + c_\alpha g$ and w_β to the commitments $g' + c_\beta g$, construct $w = (w_\beta - w_\alpha)/(c_\beta - c_\alpha)$ and $w' = w_\alpha - c_\alpha w$.

Now, the claim is that reconstructed witnesses will satisfy $F(z) = 0$, $F(z') = e'$. Indeed, if any of those relations are broken, then "true" polynomial $e(t)$ was different from the one sent by the prover. Then, by Schwartz-Zippel lemma for all c-s but negligible $\frac{d+\ell}{2^\lambda}$ part the decommitment $w' + cw$ of $g' + cg$ does not satisfy F.

This leaves us with dichotomy - either adversary fails to output satisfying instance in all but negligible amount of cases, or it can output *different* decommitment - which is a commitment break. □

3 WARPfold

Now, we are ready to discuss modifications necessary to support wrongfield arithmetic.

Consider a collection of fields $(\mathbb{F}; \mathbb{F}_1, ..., \mathbb{F}_s)$, first one called "rightfield", and others "wrongfields",

Fields all must be prime, and the order of wrongfields is not bounded from above, but is bounded from below as $\Omega(2^\lambda)$.

Suppose we are also given limb base b - the size of a primitive range-check over $\mathbb{F}$.

Note 2. *It is unclear what way of range-checking elements is desirable; in Latticefold [4] it is done by directly computing indicator polynomial, so b is relatively small. This does impact commitment cost, albeit insignificantly. For decider, however, this option is worse than the second one.*

Other option is using a lookup argument, which is available in Protostar by default. It is unclear which of these approaches is better (as lookup argument has some overhead in terms of additional, non-small witness elements).

Definition 6. *We define $(b, (\mathbb{F}; \mathbb{F}_1, ..., \mathbb{F}_s))$-multicircuit to be a collection of circuits over these fields, with additional native representation of wrongfield elements - i.e. for every wrongfield witness element there is, actually, its representation in base b over rightfield, with all limbs being range-checked.*

We denote parts of the circuit living over different fields the same way as in Protostar, with index superscript such as $F^{(i)}$ to signify that this system of equations lives over a wrongfield, and superscript $F^{\cdot}$ to signify that it lives in the rightfield. We denote s-th limb of the wrongfield witness as $L_s(w^{(i)})$.

For simplicity, we assume that all challenges in our protocols are automatically sent to all fields (as they are already range-checked and of size 2^λ).

Note 3. *Reader might ask, what is the purpose of this construction if every wrongfield element is represented as a collection of limbs. The answer to this is that committing to limbs themselves is, typically, not a problem - it is the simulation of modular reductions and arithmetic operations that gets us in trouble with non-native arithmetic. The point of WARPfold is to avoid this until the decider.*

Definition 7. ***Strict WARPfold instance*** *corresponding to the multicircuit is the normal Protostar instance of the main circuit, but with additional wrongfield public inputs $\gamma_i^{(j)}$, obtained from the same 2^λ-sized last challenge.*

Strict WARPfold witness *is a witness to the rightfield part of the multicircuit, additionally satisfying wrongfield constraints (recall that wrongfield part of the witness is encoded in limbs of the rightfield part).*

Definition 8. ***Relaxed WARPfold instance*** *is the same as strict WARPfold instance, but with additional error terms $e \in \mathbb{F}, (..., e^{(i)} \in \mathbb{F}_i, ...)$, and a* ***degradation counter*** *N. Each time the folding occurs, this parameter will be increased by 1, and there is a maximal size of N such that folding after it is achieved is impossible.*

N must be small enough to ensure limbs do not overflow (which happens at $N = \frac{|\mathbb{F}|}{b\lambda}$), but in practice it is useful for decider to set up maximal N even smaller - around $2^{\lambda/3}$.

We also set base case for $N = 0$ to be trivial relaxed solution $\tilde{z} = 0$.

Relaxed WARPfold witness *is a witness to corresponding rightfield instance, which additionally satisfies the following: for every limb $L_s(w^{(i)})$, it is no larger than $bN2^\lambda$, and satisfies relaxed wrongfield equations*

$$\tilde{F}^{(i)}(\tilde{z}) = e^{(i)}$$

where the wrongfield witness is reconstructed from limbs using modular reduction: $\sum_k b^s L_s(w^{(i)})(in\ \mathbb{F}_i)$.

Folding is defined unsurprisingly (and requires sending $e^{\cdot}(t), (..., e^{(i)}(t), ...)$), however, the direct approach at constructing extractor, of course, fails: the

extracted relaxed witness is not guaranteed to satisfy any range requirements, and without range requirements limbs might indeed overflow, breaking the non-native arithmetic part.

Therefore, we instead construct an extractor for a *sequence* of foldings, starting from non-relaxed ones. Similar to situation in regular folding schemes, this extractor runs in time exponential in sequence length. This is believed to not be an actual issue, and there is a paper analyzing Nova in Algebraic Group Model [11], which could, potentially, be adapted to this case. We do not conduct such analysis.

Theorem 1. *Consider the adversary which, on the setup phase, samples sequence of strict WARPfold instances* $\mathcal{U}_0, \mathcal{U}_1, ..., \mathcal{U}_{n-1}$*, and then engages in a sequence of folding protocols:*

$$\mathcal{U}'_{i+1} = \mathit{Fold}(\mathcal{U}'_i, \mathcal{U}_i)$$

where $\mathcal{U}_0 = 0$*, and with non-negligible probability outputs the witness to* $\mathcal{U}'_n$*.*

We extract from this adversary the witnesses for all $\mathcal{U}_i$*'s.*

Proof. First, we prove by induction the following statement: any adversary that provides a witness to $\mathcal{U}'_n$ satisfying rightfield constraints does, in fact, output witness that satisfies $L_s(w^{(i)}) < bN2^\lambda$ condition.

This is clear, because rightfield part of the circuit can always be extracted (using normal Protostar extractor). Then, we observe that strictf witnesses actually do satisfy the range-checks, and the statement is trivial - as relaxed witness limbs are obtained by accumulating in N elements of size $< b$, multiplied by folding challenge $< 2^\lambda$

Then, as no overflows occur in the limbs, we can be sure that extracted wrongfield witness indeed satisfies

$$(w^{(i)}_{n-1})' + cw^{(i)}_{n-1} = (w^{(i)}_n)' (\text{in } \mathbb{F}_i)$$

Now, we can use the same argument as in normal Protostar, but for wrongfield. Indeed, unless both $F^{(i)}(w^{(i)}_{n-1}) = 0$ and $F^{(i)}((w^{(i)}_{n-1})') = (e^{(i)}_{n-1})'$, the communicated univariate polynomial over $\mathbb{F}_i$ was incorrect.

Then, by Schwartz-Zippel lemma for all but negligible part of challenges c the adversary would not output a valid witness, which contradicts the assumption. □

4 IVC

Any folding scheme has a corresponding IVC. In our opinion, the original Nova paper [10] is the best source for this part.

However, as our construction is not, technically, a folding scheme (as it lacks standard extractor and only supports finite amount of foldings), we take some time to discuss the implications of these changes for IVC.

Naturally, we can only construct bounded IVC (supporting finite amount of steps). This is not a problem in practice, both because the supported amount of steps is very large, and because the computation can be continued by calling a decider or flushing limbs in some other way.

However, there are more significant differences.

Let us briefly recall how standard IVC construction works. Given a step-circuit S, implementing some (potentially, non-deterministic) function, we construct a new wrapping circuit with the following functionality (some values will be denoted by index k on the k-th step of the IVC - the actual circuit is the same for each k, this is added for clarity).

1. It has a single nontrivial public output x.
2. It has private input $\mathcal{U}_k$, which is a strict instance with the structure of the circuit itself. It validates the strictness (in Nova, that means just $\mathcal{U}_k.u = 1, \mathcal{U}_k.e = 0$, in Protostar this also means validating that challenges were produced correctly).
3. $\mathcal{U}_k.x = \text{Hash}(q_0, q_k, k, \mathcal{U}'_k)$, where $\mathcal{U}'_k$ is a relaxed instance with the same structure.
4. It runs the (Fiat-Shamir, non-interactive) version of folding protocol verifier on $\mathcal{U}_k, \mathcal{U}'_k$, and outputs $\mathcal{U}'_{k+1}$.
5. It sets $x = \text{Hash}(q_0, q_{k+1} = S(q_k), k+1, \mathcal{U}'_{k+1})$
6. For case $k = 0$ it validates some additional conditions, namely $q_k = q_0$, $\mathcal{U}'_0 = 0$.
7. For case $k = -1$ it accepts any dummy inputs into folding, and instead outputs $\mathcal{U}'_{k+1} = \mathcal{U}'_0 = 0$, and, similarly, sets input $q_{k+1} = q_0$.

Now, to run the IVC, prover performs the following recursive construction: given witnesses w_k, w'_k to instances $\mathcal{U}_k, \mathcal{U}'_k$, they perform (non-interactive) folding to obtain w'_{k+1}. Then, they are now capable of calculating the witness to the IVC circuit, and its execution trace becomes w_{k+1}.

Let us first consider rather boring case, that we will call **weak** - in that case, all wrongfield arithmetic is contained in the implementation of step circuit S, and the rest of the construction is implemented in the rightfield

arithmetic. This case is not entirely impractical - for example, we could want to emulate some external field, but it doesn't help us with recursion itself.

4.1 Weak case

Assume the **weak** case - the IVC circuit is built entirely over the rightfield $\mathbb{F}$, and all wrongfield arithmetic is contained in implementation of S.

Then, we do not modify the construction at all (and only check that program counter is less than $N_{\max}$). Indeed

Theorem 2. *Assume (heuristically) that there exists an extractor for standard non-interactive rightfield Protostar scheme (in fact, its small modification which also hashes additional wrongfield-related prover messages without doing any checks).*

Assume that we also have access to non-interactive version of the extractor from Theorem 1.

Then, for an adversary outputting valid instance-witness data $(\mathcal{U}_k, \mathcal{U}'_k)$, (w_k, w'_k)*, it is true that there is an extractor returning the sequence of valid witnesses* $(\mathcal{U}_j, \mathcal{U}'_j), (w_j, w'_j)$ *for* $j < k$*.*

Proof. First, let us call the extractor for rightfield Protostar to obtain the sequence of $\mathcal{U}_j, \mathcal{U}'_j$ for $j < k$. Note that it is always doable, because the recursive circuit itself is defined over the rightfield. We are now in the exactly the same situation as in Theorem 1, and can apply non-interactive version of its extractor to validate that our witnesses satisfy wrongfield constraints too. □

This is already useful, but this still leaves us with large recursive overhead. However, the same argument clearly does not work in case where there are wrongfield constraints in the definition of recursive circuit itself.

We believe this approach is also unsound, but couldn't come up with falsifying example. We leave it as a conjecture.

Conjecture 1. *Naive construction which uses wrongfield arithmetic in IVC circuit is unsound.*

4.2 Reinforcement

Using wrongfield arithmetic for the IVC circuit is highly desirable (as a lot of operations there are defined over the *base* field of the circuit), so we need to do some adjustment.

We start by discussing a primitive allowing us to copy chunks of data between different instances cheaply.

Definition 9. ***Multi-commitment circuit*** *is a circuit as in Definition 1, but with commitment target groups potentially being cartesian powers of G, such as $G \times G \times ... \times G$. All definitions naturally trivially generalize.*

We will use this to split our rounds into pieces - i.e. we will have a normal circuit with rounds split into subsets, each of these committed separately.

This can also be simulated without introducing new multi-commitment formalism at all, by just splitting the round into few subrounds without any challenges.

We will refer to these subsets of the circuits as **exposed subsets**. They can be used to efficiently copy data between different instances - by declaring that commitments to them coincide we can ensure that large batches of data are correctly copied. This is useful in standard folding schemes for inter-step communication, and originally was introduced by author in a forum post [12].

Collection of tricks related to usage of the exposed subsets later became known as Moon-Moon, and it currently seems to largely be unpublished folklore. Related tricks were rediscovered independently in the context of ZK - virtual machines and are known as "memory continuations".

We, however, use them in a new context - we are copying a subset of data from *relaxed* instance to a subset of *strict* one.

Definition 10. *$(b; (\mathbb{F}; \mathbb{F}', ...))$-multicircuit with reinforced field $\mathbb{F}'$ is a multicircuit with the following additional structure:*

1. *There are two new exposed subsets of the same size, using the same commitment key. Their witness spaces are called W_{limbs} and W_{check}.*
2. *Subset W_{limbs} hosts (large) limbs of $\mathbb{F}'$ elements of bitsize μ. Note that earlier we kept elements in small limbs, but here we insist that we group them and represent their linear combinations as large limbs. We require that $N_{max} < |\mathbb{F}|/(2^{\mu+\lambda})$ to prevent these limbs from overflowing, and suggest that realistically $\mu \sim \frac{2}{3}\lambda$ seems like a good choice, as it allows us to do at least $2^{\frac{\lambda}{3}}$ foldings, which, for 256-bit base field and $\lambda = 128$ gives $> 2^{41}$ sequential foldings, more than enough for any realistic application.*
3. *Subset W_{check} hosts values that are range-checked to be $< 2^{\mu+\lambda}(N_{max} - 1)$, and no other requirements are applied in circuit.*

Now, we are ready to describe our modifications to IVC circuit.

Definition 11. *Reinforced IVC circuit is standard IVC circuit with two additional checks:*

1. $k < N_{max} - 1$.
2. $\mathcal{U}_k.g_{check} = \mathcal{U}'_k.g_{limbs}$

The recursive construction rule then becomes the following: given $(\mathcal{U}_k, \mathcal{U}'_k)$, (w_k, w'_k), we perform non-interactive folding to obtain $\mathcal{U}'_{k+1}, w'_{k+1}$. Then, we use execution trace of the circuit to populate default part of w_{k+1}, and copy $(w_{k+1})_{\text{limbs}}$ into $(w_{k+1})_{\text{check}}$ to populate the rest.

Definition 12. *Reinforced IVC instance-witness pair is data* $((\mathcal{U}_k, \mathcal{U}'_k), (w_k, w'_k))$, *satisfying the following requirements:*

1. *Standard:* w_k, w'_k *are satisfying witnesses to* $\mathcal{U}_k, \mathcal{U}'_k$.
2. *Standard:* $\mathcal{U}_k.x = \text{Hash}(q_0, q_k, k, \mathcal{U}'_k)$
3. *New:* $\mathcal{U}_k.g_{check} = \mathcal{U}'_k.g_{limbs}$

Theorem 3. *Assume there is a non-interactive extractor for Protostar. Then, there is a non-interactive extractor for the reinforced IVC (extracting* w_{k-1}, w'_{k-1}*)*

Proof. Given an adversary outputting valid $((\mathcal{U}_k, \mathcal{U}'_k), (w_k, w'_k))$, first, we will apply Protostar extractor to obtain (w_{k-1}, w'_{k-1}) - as non-interactive folding for $\mathcal{U}'_{k-1}, \mathcal{U}_{k-1}$ is contained in the execution trace w_k.

Then, we observe that rightfield part of w_{k-1} satisfies all the equations (this is guaranteed by the extractor), and so values in $(w_{k-1})_{\text{check}}$ are, indeed, properly range-checked to be less than $2^{\mu+\lambda}(N_{\max} - 1)$. On the other hand, they must be equal to $(w'_{k-1})_{\text{limbs}}$, unless our extractor has computed a commitment break.

Hence, there is no overflow in limbs, and so $w_k = w'_{k-1} + cw_{k-1}$ (in $\mathbb{F}'$). Hence, Protostar extractor also guarantees that w'_{k-1}, w_{k-1} satisfy reinforced wrongfield equations. □

5 Cost analysis

From the standpoint of the commitment cost, non-reinforced wrongfield elements cost is comparable to rightfield, but with range-check overhead.

This overhead can be made small by choosing small l (as committing to a large amount of limbs costs roughly the same as to one large element).

Alternative choice is using some lookup argument - this decreases the witness size, and, thus, decider work, but, a bit paradoxically, increases commitment cost. Standard Protostar lookup likely has around $2\times$ overhead for rangechecks, though, practical evaluations are required to elaborate this.

Reinforced wrongfield elements cost roughly 2.5 times more: for suggested value $\mu = \frac{2\lambda}{3}$ this means that means that instead of μ-sized rangecheck of a limb we additionally do a rangecheck of size $\mu + \lambda = 2.5\mu$.

Costs for the decider involve actually simulating the non-native arithmetic of the circuit, and so are highly specific on the exact nature of used constraints. However, we believe that decider overhead will not be significantly larger than Cyclefold, as Cyclefold decider requires computing a commitment to the secondary circuit, which has size ranging from ~ 400 (author's high degree circuit from [13]), to roughly 1700 for naive R1CS implementation, which is to be traded against 4 non-native in-circuit scalar multiplications.

We also point out that this scheme works in contexts where cyclefold is not available at all (such as BLS family of curves), and allows to simulate fields completely external to the proof system.

References

[1] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. Cryptology ePrint Archive, Paper 2002/088, 2002. https://eprint.iacr.org/2002/088.

[2] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Paper 2005/133, 2005. https://eprint.iacr.org/2005/133.

[3] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. Cryptology ePrint Archive, Paper 2014/595, 2014. https://eprint.iacr.org/2014/595.

[4] Dan Boneh and Binyi Chen. Latticefold: A lattice-based folding scheme and its applications to succinct proof systems. Cryptology ePrint Archive, Paper 2024/257, 2024. https://eprint.iacr.org/2024/257.

[5] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Paper 2019/1021, 2019. https://eprint.iacr.org/2019/1021.

[6] Benedikt Bünz and Binyi Chen. Protostar: Generic efficient accumulation/folding for special sound protocols. Cryptology ePrint Archive, Paper 2023/620, 2023. https://eprint.iacr.org/2023/620.

[7] Liam Eagen and Ariel Gabizon. Protogalaxy: Efficient protostar-style folding of multiple instances. Cryptology ePrint Archive, Paper 2023/1106, 2023. https://eprint.iacr.org/2023/1106.

[8] Abhiram Kothapalli and Srinath Setty. Cyclefold: Folding-scheme-based recursive arguments over a cycle of elliptic curves. Cryptology ePrint Archive, Paper 2023/1192, 2023. https://eprint.iacr.org/2023/1192.

[9] Abhiram Kothapalli and Srinath Setty. Hypernova: Recursive arguments for customizable constraint systems. Cryptology ePrint Archive, Paper 2023/573, 2023. https://eprint.iacr.org/2023/573.

[10] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. Cryptology ePrint Archive, Paper 2021/370, 2021. https://eprint.iacr.org/2021/370.

[11] Hyeonbum Lee and Jae Hong Seo. On the security of nova recursive proof system. Cryptology ePrint Archive, Paper 2024/232, 2024. https://eprint.iacr.org/2024/232.

[12] Lev Soukhanov. Folding endgame. zkresear.ch forum, 2023. https://zkresear.ch/t/folding-endgame/106.

[13] Lev Soukhanov. Reverie: an end-to-end accumulation scheme from cyclefold. Cryptology ePrint Archive, Paper 2023/1888, 2023. https://eprint.iacr.org/2023/1888.