Bulletproofs++ review

Cypher Stack*

March 29, 2024

This report describes the findings of a review of the Bulletproofs++ IACR preprint[1]. It reflects a limited scope and represents a best effort; as with any review, it cannot guarantee correctness or security, or that the preprint is otherwise free of errors. Further, it cannot guarantee that any particular implementation of the Bulletproofs++ protocols is correct, secure, or suitable for intended use cases. The author asserts no warranty and disclaims liability for its use. The author further expresses no endorsement of any kind. This report has not undergone any further formal or peer review.

Contents

*https://cypherstack.com

[1] https://eprint.iacr.org/archive/2022/510/20230717:163509

1 Executive summary

Bulletproofs++ provides a collection of protocols that can be used for range proving and arithmetic circuit satisfiability proving. While parts of the techniques are inspired by those of Bulletproofs[2] and Bulletproofs+[3], the overall design differs significantly due to the use of several novel techniques that must be carefully integrated. Both Bulletproofs and Bulletproofs+ describe general techniques for proving arithmetic circuit satisfaction, but they provide range proving systems using optimizations that do not directly use circuit proofs. In contrast, Bulletproofs++ builds its generic range proofs more directly using circuit proofs, albeit with some modifications.

The preprint is organized in a manner that builds the necessary tools and frameworks incrementally, sometimes providing intermediate or "toy" versions to demonstrate techniques or improve clarity. Protocols are generally expressed directly and indicate the relevant proving relations, both of which are important for analysis.

However, we find that security proofs, which generally show properties like computational witness-extended emulation or special honest-verifier zero knowledge for protocols, and supporting results often lack sufficient clarity that introduces questions to their validity. We also find some notation to be informal or unclear, particularly with respect to aspects like vector or matrix dimensions that are necessary to ensure consistency and certain proof or technique validity. As noted in the findings of this report, we observe two common issues:

- *Unclear notation.* Protocols and proofs often use notation that is nonstandard, inconsistent with other portions of the preprint, or undefined. This makes analysis challenging, and requires that the reader make assumptions about the intent, which may not be correct.

- *Insufficient clarity.* Several proofs aim to define extractors that are used to show computational witness-extended emulation for protocols, and simulators that are used to show special honest-verifier zero knowledge. Perhaps because of the complexity of the protocols, these proofs often make claims about these and other properties that do not provide sufficient explanation of the underlying reasoning, or introduce them only informally. We were able to show that many of these claims are valid, but for some we were unable to do so in the scope available for this review. Lack of assertion of validity for a claim should, however, not be interpreted as an assertion that the claim is invalid.

Aside from issues specific to security proofs, we generally observe a higher degree of protocol complexity compared to other range proving systems like Bulletproofs and Bulletproofs+ used in production systems. While this more complex design provides benefits in terms of computational and communication

[2] `https://eprint.iacr.org/2017/1066`
[3] `https://eprint.iacr.org/2020/735`

efficiency, it is likely to increase engineering risk, and should be taken carefully into account for any considered deployment.

2 Findings

We present findings from the review. Note that some findings comprise errors and issues, but others include explanatory material that may be useful for inclusion in future revisions to aid the reader.

Generally, review was conducted in a "bottom-up" manner. That is, we reviewed results from base protocols initially, as these affect all protocols and results relying on them. We then moved toward higher-level protocols.

Findings are organized according to the relevant sections and results from the preprint. They reflect the portions of the preprint that were subject to review. Notably, the material on multi-asset confidential transactions (in Section 6.5 and elsewhere) was considered out of scope, as were the discussion in Appendix A describing a technique for binary range proofs with small range and the discusison in Appendix B relating to curve-specific scalar multiplication optimization.

We reached out to the preprint authors with initial questions relevant to some of these findings prior to the release of this report, in order to obtain better understanding about them. The preprint authors responded with helpful information and explanation about errors and omissions. We appreciate the authors' time and effort in this regard. However, we note that because this review was not conducted on behalf of the authors, the presence of findings in this report should not imply any endorsement or agreement by them.

2.1 Section 1: introduction

The preprint discusses several important features common to Bulletproofs and Bulletproofs+ that it claims extend to Bulletproofs++. These include batch verification and multiparty computation.

Batch verification allows a verifier to take a collection of arbitrary proofs and, by applying a random weighting of the corresponding verification equations, verify the entire collection at once. This can be done much more efficiently than verifying each proof in the collection separately. The reason for this is twofold. First, verification of a proof in Bulletproofs and Bulletproofs+ can be reduced to checking that a single multiscalar multiplication evaluation vanishes, which can be done efficiently using a variety of techniques. Second, because all proofs in the collection can share many globally-fixed generators, random weighting of each proof's multiscalar multiplication evaluation can share these generators, resulting in a single linear combination evaluation across all proofs that minimizes the computational complexity. The preprint claims that this is possible with Bulletproofs++, and includes a separate short discussion later. However, it does not provide full detail of this in the same manner as, say, Bulletproofs.

Multiparty computation in the context of range proofs typically refers to aggregated range proofs, which are proofs making range assertions for multiple commitments in the same proof. When applied to confidential transaction protocols, this means that a transaction with multiple output commitments requires only a single range proof, which for Bulletproofs and Bulletproofs+ inherits logarithmic size scaling. Multiparty computation allows a set of mutually-untrusting parties, each of which knows the opening to a separate commitment, to produce such an aggregated range proof collaboratively. In Bulletproofs, this is done by taking advantage of certain linearity properties involved in the proving process, and does not leak the openings. In Bulletproofs+, a similar approach is possible, but leaks the value component of each opening; this is due to a subtle but important change to the structure of an internal inner-product argument. The preprint claims that this is possible with Bulletproofs++, and includes a separate short discussion later. However, it does not provide full detail of this, and is informal without proof.

We observe throughout the preprint, including in this section, that left quotation marks do not render. This is likely due to the use of incorrect symbols in the document's source LaTeX.

2.2 Section 2: preliminaries

Notation and conventions for the tensor product of vectors are introduced. These are important, as this is used several times later in the preprint. The notation is introduced using the single example

$$\bigotimes_{i=0}^{n}(1, x_i) = \left(1, x_0, x_1, x_0x_1, x_2, \ldots, \prod_{i=0}^{n} x_i\right)$$

which, while useful for specific use cases, is not particularly enlightening to show the case of vectors of arbitrary length. Specifically, the convention is such that for vectors $\vec{v}$ and $\vec{w} = (w_0, \ldots, w_{n-1})$ (for some $n > 0$) we have the following:

$$\vec{v} \otimes \vec{w} = w_0\vec{v} \| w_1\vec{v} \| \cdots \| w_{n-1}\vec{v}$$

This helps to show why the useful identity

$$\vec{e}_{ab}(\mu) = \vec{e}_a(\mu) \otimes \vec{e}_b(\mu^a) = (1, \mu, \ldots, \mu^{ab-1})$$

holds, which we prove here for completeness.

Proof. We have the following, using the above notation and convention:

$$\begin{aligned}
\vec{e}_a(\mu) \otimes \vec{e}_b(\mu^a) &= (1, \mu, \ldots, \mu^{a-1}) \otimes (1, \mu^a, \ldots, \mu^{(b-1)a}) \\
&= (1, \ldots, \mu^{a-1}) \| \mu^a(1, \ldots, \mu^{a-1}) \| \cdots \| \mu^{(b-1)a}(1, \ldots, \mu^{a-1}) \\
&= (1, \ldots, \mu^{a-1}) \| (\mu^a, \ldots, \mu^{2a-1}) \| \cdots \| (\mu^{ab-a}, \ldots, \mu^{ab-1}) \\
&= (1, \mu, \ldots, \mu^{ab-1}) \\
&= \vec{e}_{ab}(\mu)
\end{aligned}$$

□

A minor typo exists in the discussion of the commitment function Com. It is claimed that the generators $H_0, \ldots, H_7$ are not used in the function, when in fact it is only the generators $H_1, \ldots, H_7$ that are unused.

2.3 Section 3: technical overview

In the technical overview of Section 3, several definitions and corresponding notation are introduced. Specifically, a particular structure related to a multiset is defined; in such a multiset, multiplicities are field elements, and the structure may contain multiple copies of a value with arbitrary multiplicities. Such a structure A is defined to contain pairs of the form (m, v), where m and v are field elements.

When the notation

$$\hat{m}_v = \sum_{(m',v') \in A : v = v'} m'$$

is introduced, the simplification $\hat{m}_v$ has not yet been defined. This should be replaced with $\hat{m}_v(A)$ instead.

Later, when discussing how to apply this concept to range proofs, such a structure is instantiated as $A = \{(-1, v_i) : i\} \cup \{(m_j, t_j) : j\}$. While it can be inferred how standard set notation can be reinterpreted to the requirements of these collections, it would be helpful to the reader to be more specific; this is especially true since the authors then make conclusions about vanishing total multiplicities that rely subtly but importantly on this notation.

2.4 Section 4: norm linear argument

The norm linear argument presented in Section 4 includes statement and witness reductions that are essential for logarithmic scaling. Correctness requires asserting that the reductions are defined properly. Showing this relies on a somewhat non-obvious weighted inner-product identity that is tightly linked to the interleaved vector splitting used in the argument, and which differs from the splitting technique used in earlier work.

We present the identity here for clarity, along with a short proof.

Identity. *Let $\vec{x}$ and $\vec{y}$ be arbitrary vectors of length $2n$ for some $n > 0$ such that their inner product is defined, and let μ be a nonzero scalar. Then the identity*

$$\langle \vec{x}, \vec{y} \rangle_\mu = \mu^{-1} \langle [\vec{x}]_0, [\vec{y}]_0 \rangle_{\mu^2} + \langle [\vec{x}]_1, [\vec{y}]_1 \rangle_{\mu^2}$$

holds.

Proof. Using the weighted inner product and vector-splitting notation, we have

the following:

$$\begin{aligned}
\langle \vec{x}, \vec{y} \rangle_\mu &= \sum_{i=0}^{2n-1} x_i y_i \mu^{i+1} \\
&= \sum_{j=0}^{n-1} \left(x_{2j} y_{2j} \mu^{2j+1} + x_{2j+1} y_{2j+1} \mu^{2j+2} \right) \\
&= \mu^{-1} \sum_{j=0}^{n-1} x_{2j} y_{2j} (\mu^2)^{j+1} + \sum_{j=0}^{n-1} x_{2j+1} y_{2j+1} (\mu^2)^{j+1} \\
&= \mu^{-1} \langle [\vec{x}]_0, [\vec{y}]_0 \rangle_{\mu^2} + \langle [\vec{x}]_1, [\vec{y}]_1 \rangle_{\mu^2}
\end{aligned}$$

□

The weighted norm linear relation requires a statement value $\mu \in \mathbb{F}$, from which a value ρ is derived such that $\rho^2 = \mu$; ρ is used for vector reduction when performing the protocol's recursive step. This means that μ must be a square, which does not hold in general for a finite field. Further, because the reduction also uses ρ^{-1}, we must have $\rho \neq 0$ and therefore $\mu \neq 0$. Neither of these conditions is specified in the relation; they arise only when examining the protocol steps.

Fortunately, in practice the arithmetic circuit protocol described later in the preprint, which uses the weighted norm linear argument as a subprotocol, samples $\rho \in \mathbb{F}$ uniformly at random (as a verifier challenge) and sets $\mu = \rho^2$, so both conditions on μ are satisfied (except if $\rho = 0$ with negligible probability).

Because the definition and security proofs for the protocol are presented independently of the arithmetic circuit protocol, it would be safer and more clear to redefine the weighted linear relation statement in terms of $\rho \neq 0$, from which the protocol can derive μ to use for weighting purposes; this ensures that the protocol is well defined.

In Equation 26, the sum

$$\sum_{i=0}^{k-1} \gamma_i X_i + (\gamma_i^2 - 1) R_i$$

is missing brackets and should read

$$\sum_{i=0}^{k-1} \left[\gamma_i X_i + (\gamma_i^2 - 1) R_i \right]$$

instead.

2.5 Section 5: arithmetic circuits

The arithmetic circuit relations and protocols specify several bounds on input values. While they are directly specified in the circuit relation $\mathcal{R}_{ac}$, their relationships are not, which appears to lead to inconsistencies.

We first note a minor and trivial error, where it is listed that the matrix $W_{l,L}$ consists of the first $N_m - 1$ columns of the matrix W_l; to be properly defined, this submatrix should contain the first N_m columns of its parent.

More notably, the relation provides for a k-vector $\vec{V}$ of group elements, each of which has a corresponding vector opening $\vec{v_i} \in \mathbb{F}^{N_v}$ for $i \in [0, k)$ as part of the witness (along with a separate mask). The vector $\vec{w}_V$ is defined as the concatenation of all such $\vec{v_i}$; this means in particular that $\vec{w}_V \in \mathbb{F}^{kN_v}$. Equation 30 requires that a valid instance of the relation have $\vec{0} = W_l\vec{w} + f_l\vec{w}_V + \vec{a}_l$, which requires that $kN_v = N_l$ in order to be well defined. Similarly, since Equation 31 requires that the instance have $\vec{w}_L \circ \vec{w}_R = W_m\vec{w} + f_m\vec{w}_V + \vec{a}_m$, we must have $kN_v = N_m$ as well. However, in Section 5.2.1, circuit witness commitments C_L, C_R, C_O are defined such that (for $X = L, R, O$)

$$C_X = r_{X,0}G + \langle \vec{r}_{X,1:} \| \vec{l}_X, \vec{H} \rangle + \langle \vec{n}_X, \vec{G} \rangle$$

for the witness tuple $(\vec{r}_X \in \mathbb{F}^8, \vec{l}_X \in \mathbb{F}^{N_l}, \vec{n}_X \in \mathbb{F}^{N_m})$ and generators $G \in \mathbb{G}, \vec{H} \in \mathbb{G}^{N_v+7}, \vec{G} \in \mathbb{G}^{N_m}$. This implies that $N_l = N_v$, which contradicts the above.

The handling of the binary flags $f_l, f_m \in \{0, 1\}$ is unspecified in Equation 29, which defines parameters. However, witness-extended emulation assumes that $f_l f_m \neq 0$.

The statement of Theorem 2 specifies that the arithmetic circuit protocol satisfies the property of perfect honest-verifier zero knowledge. Presumably this should state that it has the property of perfect *special* honest-verifier zero knowledge, as this is the phrasing specified in Definition 5 (and used elsewhere in the literature).

The inner arithmetic circuit protocol listed in Section 5.3 specifies that the polynomial $\hat{f}(T) \in \mathbb{F}^8[T]$, which implies that it is a vector; it is not, and should be an element of $\mathbb{F}[T]$ instead.

The inner arithmetic circuit protocol defines the vector $\vec{s}_r \in \mathbb{F}^8$ using notation that implies each element itself contains a vector. This notation is incorrect, and should use non-bolded notation for each field element entry instead.

The inner arithmetic circuit protocol's final step specifies that the prover $\mathcal{P}$ and verifier $\mathcal{V}$ run the weighted norm linear argument $\langle \mathcal{P}_{nl}, \mathcal{V}_{nl} \rangle = b$. This bit-output notation, while consistent with its original presentation in Section 2.2, is inconsistent with similar notation used elsewhere in the protocol (and other protocols in the preprint).

2.6 Section 6: reciprocal argument

The pole-related function notation $w_{i,P}(X)$ used occasionally in this section is undefined, and is likely intended to be $w_{P,i}(X)$ instead.

The statement of Theorem 3 specifies that the reciprocal form protocol satisfies the property of perfect honest-verifier zero knowledge. Presumably this should state that it has the property of perfect *special* honest-verifier zero knowledge, as this is the phrasing specified in Definition 5 (and used elsewhere in the literature).

The statement of Theorem 4 has the typo "and zero knowledge arguments of knowledge" instead of "are zero knowledge arguments of knowledge".

2.6.1 Lemma 1

The statement of Lemma 1 is technically correct[4] but is too restrictive. The lemma states that if there exist $2|A|$ accepting reciprocal argument protocol transcripts for a multiplicity collection A on distinct challenges, then A vanishes.

The proof assumes the existence of $|V|$ distinct challenges that are not the negative of any item in the value set V corresponding to A, and uses a Cauchy matrix constructed from these values. The proof also notes that up to $|V|$ challenges do not meet this requirement and are invalid, since they cannot be used in the Cauchy matrix construction.

Because all $|V|$ of these invalid challenges could be part of provided transcripts, it is necessary to have an additional $|V|$ challenges to ensure the set of valid challenges is sufficient to build the Cauchy matrix. This means that $2|V|$ transcripts are required.

Since $|V| \leq |A|$ by definition, the existing requirement still holds, but is too restrictive and can be reduced. We note that because the proof of Theorem 3 uses circuit-based encodings of multiplicity sets, the existing statement bound is relevant.

2.7 Section 7: implementation and benchmarks

For the most part, we do not comment on the data in this section, as we did not attempt to reproduce the benchmarks whose results are presented.

However, we do note that the proving and verification benchmarks are presented for four different setups:

- Bulletproofs++ over `secp256k1`, using the authors' C implementation
- Bulletproofs+ over `ristretto255`, using a Rust implementation
- Bulletproofs over `ristretto255`, using a Rust implementation
- Bulletproofs over `secp256k1`, using a C implementation

In general, we caution against comparing the performance of cryptographic libraries in this manner, as this is not always a good proxy for comparing the theoretical efficiency of their corresponding protocols. Different elliptic curve libraries often imply different performance, as do libraries written in different languages and with different implementation goals and techniques.

It is the case that the preprint specifies that its comparisons are to particular implementations, and discusses certain aspects of the timing differences relative to things like expected elliptic curve library performance. However, we consider it likely (and reasonable) that the reader may extrapolate the results to assume they represent the underlying protocols more generally.

[4] The best kind of correct

Further, configuration and build conditions may be important for a fair comparison. As one example, the elliptic curve library used for the `ristretto255`-based libraries supports different arithmetic backends (depending on processor) whose performance variability is significant, and which depend heavily on compiler and build conditions that are unspecified here.

All three range proving systems (Bulletproofs, Bulletproofs+, and Bulletproofs++) support optimizations that can greatly affect their performance in implementations. To ensure a fair comparison, it is important that all libraries be sufficiently optimized (or not optimized) for comparative data to be reasonably applicable. These optimizations might include variable-time verifier operations (since the verifier holds no secrets), arithmetic optimizations, multiscalar multiplication batch weighting, and others. Additionally, it is often the case that prover operations are not the focus of optimizations, since proving is typically done infrequently and not in bulk; this can render prover comparisons across libraries highly variable and fail to represent the complexity of a protocol accurately. It is not clear that the libraries used in the benchmarks represent these in a consistent way.

Typically, multiscalar multiplication evaluation dominates verification in all three protocols (but we stress that important exceptions exist). While the preprint does provide order-of-magnitude estimates for its multiscalar multiplication and scalar verification operations, it does not specify them directly. It would be much more accurate to directly specify these values for all three protocols, providing the reader with a generally implementation-independent comparison of complexity.

Further, batch verification is possible in all three protocols. This is extremely important for transaction protocol use cases where the verifier must handle many proofs. Batch verification generally shifts the computational burden away from multiscalar multiplication evaluation and toward scalar operations, which may produce subtle and unexpected results. While not all existing libraries support batch verification, it would be useful to provide at least theoretical complexity for this as part of comparison data.

2.8 Appendix C: theorems and proofs

2.8.1 Lemma 5

In appendix C, Lemma 5 defines a round extractor that plays a key role in later proofs relating to computational witness-extended emulation. The extractor is intended to produce, under certain conditions, either a nontrivial discrete logarithm relation or a set of vectors with other witness-related properties. However, the lemma contains undefined notation, and neither the lemma's statement nor its proof appear valid as written.

Part of the lemma statement includes transcripts of the form $(\alpha_j \in \mathbb{F}, \vec{w_j} \in \mathbb{F}^k)$. However, the preprint does not define what, if any, additional requirements exist for these transcripts, as the term "transcript" is defined only generically in earlier security definitions. It is likely, inferring based on later context, that

each (distinct) α_j in the transcript tuple corresponds to a verifier challenge, and each $\vec{w_j}$ to a prover witness. However, this is never formalized or explicitly defined.

The statement is incorrect relating to one possible set of vectors that the extractor may be expected to produce. Specifically, the statement indicates that the extractor may produce a set of vectors $\{\vec{x_i}\}_{i=0}^{n-1}$ in $\mathbb{F}^k$ such that the vanishing $\sum_{i=0}^{n-1} f_i(X)\vec{x_i} = \vec{0}$ holds. However, the set of polynomials $\{f_i(X)\}_{i=0}^{n-1}$ in $\mathbb{F}[X]$ is linearly independent by assumption. If the above sum holds, then each polynomial component of the resulting k-vector vanishes in $\mathbb{F}[X]$, so linear independence requires that each $\vec{x_i} = \vec{0}$. Since these vectors must also satisfy $C_i = \langle \vec{x_i}, \vec{G} \rangle$ for all $i \in [0, n)$, this means each $C_i = 0$. While this could *strictly* be valid (the extractor could instead output a discrete logarithm relation), it is unlikely that this is the intent of the statement.

Separately, the proof is invalid as written.

To see why, first note that the authors implicitly treat vectors as column vectors for the purpose of linear algebraic expression evaluation, though they do not appear to state this directly (and such vectors are typically written as row vectors when expressed elementwise). For example, the proof of the preceding Lemma 4 defines an $n \times m$ matrix M and vector $\vec{c} \in \mathbb{F}^n$. It examines the product $M^\top \vec{c}$, which is only well defined if $\vec{c}$ is treated as an $n \times 1$ matrix.

The proof of Lemma 5 defines an $m \times k$ matrix W by setting each row $0 \leq j < m$ to the vector $\vec{w_j} \in \mathbb{F}^k$. (Presumably this assumes each $\vec{w_j}$ is now treated as a row vector for this purpose, which is inconsistent with the above discussion and easily fixed by using $\vec{w_j}^\top$ instead.) It further defines the $n \times m$ matrix M as in Lemma 4.

The proof then claims that $W\vec{G} = M\vec{C}$, where $\vec{G} \in \mathbb{G}^k$ and $\vec{C} \in \mathbb{G}^n$ are vectors of group elements specified in the statement of the lemma. While the left side of the equality is a well-defined $m \times 1$ column vector, the right side is undefined due to a dimension mismatch. The equality does hold if it is replaced by $W\vec{G} = M^\top \vec{C}$.

The matrix product equality is then used with the Moore-Penrose inverse of M as part of the extractor construction. However, there is an error; because $n \leq m$, M has full row rank and therefore its Moore-Penrose inverse is a right inverse (not a left inverse). Fortunately, assuming the use of $M^\top$ as above fixes this, as $M^\top$ has full column rank and therefore its Moore-Penrose inverse is a left inverse and Equation 94 holds using $M^\top$ instead of M.

The proof's claim in its first paragraph about the vanishing of X-related polynomial evaluations is also incorrect, and presented with faulty justification about vanishings.

We were informed by the authors of the intended statement of the lemma, which differs from the preprint. Because of the errors in statement and proof, we provide the corrected statement and fix the proof here.

Lemma (Lemma 5)**.** *For any k group elements $\vec{G} \in \mathbb{G}^k$ and any n linearly-independent polynomials $\vec{f} \in \mathbb{F}[X]^n$ of degree $d < m$, there exists an efficient extractor χ such that for n commitments $\vec{C} \in \mathbb{G}^n$ and $m \geq n$ tuples ($\alpha_j \in$*

$\mathbb{F}, \vec{w_j} \in \mathbb{F}^k)_{j=0}^{m-1}$ *with distinct* α_j *such that* $\langle \vec{w_j}, \vec{G} \rangle = \sum_{i=0}^{n-1} f_i(\alpha_j) C_i$ *for* $j \in [0, k)$*,* χ *outputs either* $\vec{s} \in \mathbb{F}^k$ *such that* $\langle \vec{s}, \vec{G} \rangle$*, or* $\{\vec{x_i} \in \mathbb{F}^k\}_{i=0}^{n-1}$ *such that* $C_i = \langle \vec{x_i}, \vec{G} \rangle$ *for* $i \in [0, n)$ *and* $\vec{w_j} = \sum_{i=0}^{n-1} f_i(\alpha_j) \vec{x_i}$ *for* $j \in [0, m)$*.*

Proof. Define an $n \times m$ matrix M such that $M_{i,j} = f_i(\alpha_j)$. By Lemma 5, we have $\text{rank}(M) = n$. Define an $m \times k$ matrix W such that each row j is the row vector $\vec{w_j}^\top$. From these definitions, it holds by inspection that $W\vec{G} = M^\top \vec{C}$.

Now consider the Moore-Penrose inverse $(M^\top)^+$ of $M^\top$. Since M is of full row rank, $M^\top$ is of full column rank and its Moore-Penrose inverse is a left inverse. This means the following holds:

$$\begin{aligned} M^\top (M^\top)^+ W\vec{G} &= M^\top (M^\top)^+ M^\top \vec{C} \\ &= M^\top \vec{C} \\ &= W\vec{G} \end{aligned}$$

Let $W' = M^\top (M^\top)^+ W$. If $W' = W$, then define $X = (M^\top)^+ W$. This means that

$$\begin{aligned} M^\top X &= M^\top (M^\top)^+ W \\ &= W' \\ &= W \end{aligned}$$

and, since

$$\begin{aligned} M^\top X &= \begin{pmatrix} f_0(\alpha_0) & \cdots & f_{n-1}(\alpha_0) \\ \vdots & \ddots & \vdots \\ f_0(\alpha_{m-1}) & \cdots & f_{n-1}(\alpha_{m-1}) \end{pmatrix} \begin{pmatrix} \vec{x_0} \\ \vdots \\ \vec{x_{n-1}} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=0}^{n-1} f_i(\alpha_0) x_{i,0} & \cdots & \sum_{i=0}^{n-1} f_i(\alpha_0) x_{i,k-1} \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{n-1} f_i(\alpha_{m-1}) x_{i,0} & \cdots & \sum_{i=0}^{n-1} f_i(\alpha_{m-1}) x_{i,k-1} \end{pmatrix} \end{aligned}$$

it follows that the rows $\{\vec{x_i}\}_{i=0}^{n-1}$ of X returned by the extractor satisfy $C_i = \langle \vec{x_i}, \vec{G} \rangle$ for $i \in [0, n)$, and that $\vec{w_j} = \sum_{i=0}^{n-1} f_i(\alpha_j) \vec{x_i}$ for $j \in [0, m)$ as required.

Otherwise, if $W' \neq W$, then we have $W'\vec{G} = W\vec{G}$, so $(W' - W)\vec{G} = \vec{0}$. Since $W' \neq W$, the difference $W' - W$ has a nonzero row vector $\vec{s}$ that is returned by the extractor as the required discrete logarithm relation. □

2.8.2 Theorem 1

In Appendix C.1, the proof of Theorem 1 aims to show that the weighted norm linear argument protocol presented in Section 4.3 is perfectly complete and has computatation witness-extended emulation.

Completeness, while somewhat tedious to show algebraically, holds. However, the proof for computational witness-extended emulation, which uses a

fairly standard special soundness rewinding technique, has unclear validity. The preprint indicates that the rewinding technique is very similar to that of the Bulletproofs inner-product argument, which also uses a recursive round-based design.

The proof recursively fixes a round, which in turn fixes common inputs and the round commitments X and R. It rewinds the transcript to obtain a set $\{\gamma_i\}_{i=0}^3$ of four distinct challenges, and is able to use the subsequent responses as supplied to the verifier in the reduced next round.

The notation used is unfortunately unclear, as it is inconsistent with that of both the Bulletproofs inner-product argument's proof and the preprint's own weighted norm linear argument protocol. In both of those, "non-primed" notation (like μ) indicates a value from the current round of the protocol, and corresponding "primed" notation (like μ') indicates the value to be used in the next round, where extraction is assumed to hold by recursion.

In this proof, the notation does not follow this convention, and is also internally inconsistent. The proof specifies that the extractor obtains a set of values $\{v_i, \vec{l_i}, \vec{n_i}\}_{i=0}^3$ from rewinding such that

$$\begin{aligned} C_i' &= C + \gamma_i X + (\gamma_i^2 - 1)R \\ &= v_i G + \langle \vec{l_i}, [\vec{H}]_0 + \gamma_i [\vec{H}]_1 \rangle + \langle \vec{n_i}, [\vec{G}]_0 + \gamma_i [\vec{G}]_1 \rangle \\ &= v_i G + \langle \vec{l_i}', \vec{H} \rangle + \langle \vec{n_i}', \vec{G} \rangle \end{aligned}$$

holds for each $i \in [0, 4)$. While it is not necessarily clear from inspection, the rewound $\{v_i, \vec{l_i}, \vec{n_i}\}_{i=0}^3$ are those from the subsequent round, not the current round. The values $\{C_i'\}_{i=0}^3$ are as defined, to be used in the subsequent round (they are not separately extracted). The fixed common inputs $\mu, \vec{c}, C, G, \vec{G}, \vec{H}$ are from the current round.

The vectors $\{\vec{l_i}', \vec{n_i}'\}_{i=0}^3$ from the above equality are not extracted from the subsequent round via rewinding (as the notation might imply), but are instead assumed to be defined such that the equality holds. The definitions allow for the use of common generators $\vec{G}$ and $\vec{H}$ across rewindings, which is important for later use of the round extractor; indeed, without these definitions, the generators depend on the challenges and the extractor reasoning would not apply. It is not necessarily obvious how these vectors should be defined, so we comment on this here for clarity.

Suppose that $\vec{G}$ and $\vec{H}$ both have length $2N$ for some $N > 0$; this means that each $\vec{l_i}$ and $\vec{n_i}$ has length N.

If we let $\vec{l_i} = (l_{i,0}, \ldots, l_{i,N-1})$ and define

$$\vec{l_i}' = (l_{i,0}, \gamma_i l_{i,0}, \ldots, l_{i,N-1}, \gamma_i l_{i,N-1}),$$

then we have

$$\begin{aligned}\langle \vec{l_i}, [\vec{H}]_0 + \gamma_i [\vec{H}]_1 \rangle &= \sum_{j=0}^{N-1} l_{i,j}(H_{2j} + \gamma_i H_{2j+1}) \\ &= \sum_{j=0}^{N-1} (l_{i,j} H_{2j} + \gamma_i l_{i,j} H_{2j+1}) \\ &= \langle \vec{l_i}', \vec{H} \rangle\end{aligned}$$

as required. Further, since $[\vec{l_i}']_0 = \vec{l_i}$ and $[\vec{l_i}']_1 = \gamma_i \vec{l_i}$, the required condition $\gamma_i [\vec{l_i}']_0 - [\vec{l_i}']_1 = \vec{0}$ immediately follows.

Before we show the analogous derivation of $\vec{n_i}'$, we first note an error in the equality where it is introduced. The proof indicates that each

$$C_i' = v_i G + \langle \vec{l_i}, [\vec{H}]_0 + \gamma_i [\vec{H}]_1 \rangle + \langle \vec{n_i}, [\vec{G}]_0 + \gamma_i [\vec{G}]_1 \rangle,$$

but the round transition is defined to require

$$C_i' = v_i G + \langle \vec{l_i}, [\vec{H}]_0 + \gamma_i [\vec{H}]_1 \rangle + \langle \vec{n_i}, \rho [\vec{G}]_0 + \gamma_i [\vec{G}]_1 \rangle$$

instead (the proof is missing a factor ρ).

With this correction in place, if we let $\vec{n_i} = (n_{i,0}, \ldots, n_{i,N-1})$ and define

$$\vec{n_i}' = (\rho n_{i,0}, \gamma_i n_{i,0}, \ldots, \rho n_{i,N-1}, \gamma_i n_{i,N-1}),$$

then we have

$$\begin{aligned}\langle \vec{n_i}, \rho [\vec{G}]_0 + \gamma_i [\vec{G}]_1 \rangle &= \sum_{j=0}^{N-1} n_{i,j}(\rho G_{2j} + \gamma_i G_{2j+1}) \\ &= \sum_{j=0}^{N-1} (\rho n_{i,j} G_{2j} + \gamma_i n_{i,j} G_{2j+1}) \\ &= \langle \vec{n_i}', \vec{G} \rangle\end{aligned}$$

as required. The proof implies, but does not specify, the relation between the interleaved halves of each $\vec{n_i}'$. However, since $[\vec{n_i}']_0 = \rho \vec{n_i}$ and $[\vec{n_i}']_1 = \gamma_i \vec{n_i}$, the relation for $\vec{n_i}'$ is $\gamma_i [\vec{n_i}']_0 - \rho [\vec{n_i}']_1 = \vec{0}$.

The proof constructs polynomials $v(T), \vec{l'}(T), \vec{r'}(T)$ and notes that evaluation at each $T = \gamma_i$ gives $v_i, \vec{l_i}', \vec{r_i}'$ respectively. While it is not specified in further detail, presumably this is intended to follow directly by matching generators, and from the assumption that if such polynomial evaluations did not hold, a nontrivial generator discrete logarithm relation would result. It would be helpful if this reasoning were made more clear, as the correctness of the evaluations is necessary for the proof's reasoning.

The proof then uses polynomial vanishing to assume construction of polynomials $\vec{l}(T)$ and $\vec{n}(T)$ from tensor product representations of $\vec{l}'(T)$ and $\vec{n}'(T)$, and then claims that a degree-1 expression of $\vec{l}(T) = \vec{l_0} + \vec{l_1}T$ leads to the following:

$$\vec{l_C} = \vec{l_0} \| \vec{l_1}, \vec{l_X} = \vec{l_1} \| \vec{l_0}, \vec{l_R} = \vec{l_0} \| \vec{0}$$

It is straightforward to use the tensor product representations to derive $\vec{l}(T)$ and $\vec{n}(T)$, and to show the claims that $\vec{l}(\gamma_i) = \vec{l_i}$ and $\vec{n}(\gamma_i) = \vec{n_i}$ from the derivations of $\vec{l_i}'$ and $\vec{n_i}'$ we show above. However, it is not clear how the expression of $\vec{l}(T)$ in terms of coefficients yields $\vec{l_C}, \vec{l_X}, \vec{l_R}$ as claimed; indeed, those values are never defined (and similarly for $\vec{n}(T)$, which is not written out explicitly).

2.8.3 Lemma 6

In Appendix C.2, the proof of Lemma 6 aims to show that a particular distribution of values is uniformly distributed; this is used elsewhere in the preprint.

The language and notation used in the statement and proof of the lemma are informal, particularly relating to descriptions of distributions. For example, the statement describes the "uniform distribution of openings", which presumably assumes the definition of validity. However, the proof discusses the "uniform distribution on $(v, \vec{l}, \vec{n})$" without being specific about whether this assumes such tuples are valid openings or merely contain uniformly-distributed field elements and vectors. Crucially, it cannot be the case that all elements of a valid opening are independently uniformly distributed at random due trivially to the validity definition, which effectively defines v in terms of the fixed elements $\vec{c}, \mu$ and tuple elements $(\vec{l}, \vec{n})$. This is important, as the lemma is used elsewhere in the preprint to imply uniformly-distributed proof elements that must correspond to valid openings to show a zero-knowledge property. One particular approach would be to exclude v from such consideration altogether, as it is fixed when the other aforementioned values are specified.

Further (and related to this), no definition of set membership or vector lengths is provided, which renders the well-definedness and validity of the proof's reasoning unclear. In particular, the reader must determine the lengths of the sub-vectors of $\vec{c}$ and $\vec{l}$. The statement requires that $|\vec{c}| = |\vec{l}|$ to have a well-defined inner product $\langle \vec{c}, \vec{l} \rangle$, so $|\hat{\vec{c}}_r| + |\hat{\vec{c}}_l| = |\vec{r}| + |\vec{l'}|$. However, from the proof we must have $|\hat{\vec{c}}_l| = |\vec{l'}|$ and $|\hat{\vec{c}}_r| - 1 = |\vec{r}|$ for corresponding inner products, which is a contradiction. It is not clear what was intended.

The proof uses the vector notation $\vec{v}$ when the scalar v is intended.

The proof gives a claimed equivalent distribution that assumes $\hat{\vec{c}}_r = 1$, notation which is undefined, unclear, and does not appear to lead to the claim. Further, the lemma statement allows that $\vec{c}$ be *any* fixed vector; if the proof intends that assuming a constant value for any part of $\vec{c}$ does not affect the distribution results due to some kind of restriction, this should be carefully explained with valid notation. Interestingly, the use of the lemma elsewhere in the preprint does involve an input value $\vec{c}$ whose initial term is 1, consistent with such a restriction.

It also suggests that a uniform distribution can be obtained by sampling $\vec{n}$ and $\vec{l'}$, and then sampling $\vec{v}$ and $\vec{r}$ such that a valid opening is obtained. The notation $\vec{v}$ should presumably be v; further, it cannot be sampled independently of $\vec{r}$, as it is fixed once the other values are sampled. It is certainly possible to use rejection sampling on $(v, \vec{r})$ until a valid opening is found; however, it is also possible to sample $\vec{r}$ and merely *define* v to satisfy the required equality; we note that this distinction is likely without difference, but it may be helpful to be more clear in the proof about the relationship between sampled values.

When the proof describes the distribution of values offset by elements of $\vec{s}$ to achieve a uniform distribution, it sets $\vec{r} = \vec{r} + \vec{s}_{1:}$, notation which is undefined in this context. It is likely that $\vec{r} = \vec{r'} + \vec{s}_{1:}$ was intended instead. Further, the requirement that $\vec{s}$ be sampled uniformly at random such that $\langle \hat{\vec{c}}_r, \vec{s} \rangle$ introduces structure that could influence the proof's conclusion of uniform distribution; however, the fact that s_0 only offsets v (which is already fixed by other values) may allow for samplings that do meet the conclusion.

After contacting the authors about some of these inconsistencies and apparent errors, it was implied that an off-by-one error was present, and that the elements s_0 and $\hat{\vec{c}}_{r,0}$ were intended to offset the v component of valid opening tuples; however, detail was not provided at the time.

Given what we suspect this information implies, the statement of the lemma still appears to be incorrect, as offsetting v (in reality, $\hat{v}$) by s_0 would not yield a valid opening.

It is possible that the following corrected statement of the lemma was intended by the authors; in the absence of further details, we state the proposed correction here.

Lemma (Lemma 6)**.** *Let $\lambda_r, \lambda_l, \lambda_n > 0$ be integers, and let $\lambda = \lambda_r + \lambda_l$. For fixed $\vec{c} \in \mathbb{F}^\lambda$ and $\mu \in \mathbb{F}$, we say $(v, \vec{l}, \vec{n}) \in \mathbb{F} \times \mathbb{F}^\lambda \times \mathbb{F}^{\lambda_n}$ is a valid opening if $v = \langle \vec{c}, \vec{l} \rangle + |\vec{n}|_\mu^2$.*

Suppose $\vec{l} = \vec{r} || \vec{l'}$ for $\vec{r} \in \mathbb{F}^{\lambda_r}$ and $\vec{l'} \in \mathbb{F}^{\lambda_l}$, and suppose $\vec{c} = \hat{\vec{c}}_r || \hat{\vec{c}}_l$ for $\hat{\vec{c}}_r \in \mathbb{F}^{\lambda_r}$ and $\hat{\vec{c}}_l \in \mathbb{F}^{\lambda_l}$.

The distribution of valid openings $(v, \vec{l}, \vec{r})$ such that $\vec{l}$ and $\vec{r}$ are uniformly distributed at random is the same as the distribution of valid openings $(\hat{v} - s_0, (\vec{r'} + \vec{s}_{1:}) || \vec{l'}, \vec{n})$ such that $\vec{n}$ and $\vec{l'}$ are sampled uniformly at random, $\hat{v}$ and $\vec{r'}$ are any values such that $(\hat{v}, \vec{r'} || \vec{l'}, \vec{n})$ is a valid opening, $\vec{s}_{1:} \in \mathbb{F}^{\lambda_r}$ is sampled uniformly at random, and $s_0 = -\langle \hat{\vec{c}}_r, \vec{s}_{1:} \rangle$ (so that $\langle 1 || \hat{\vec{c}}_r, \vec{s} \rangle = 0$).

Note that the ordering of steps in this statement, while not specifically enumerated as such, is important. We require that $\vec{c}$ and μ be fixed before choosing $\hat{v}$ and $\vec{r'}$, which must occur before sampling $\vec{s}_{1:}$. Otherwise, it could be the case that $\vec{r'}$ is chosen adaptively in response to $\vec{s}_{1:}$ or the initial fixed parameters.

We now give a proof of the modified lemma.

Proof. Fix any $\vec{c}$ and μ according to the statement of the lemma, and sample $\vec{n}$ and $\vec{l'}$ uniformly at random.

Let $\hat{v}$ and $\vec{r'}$ be values such that $(\hat{v}, \vec{r'}||\vec{l'}, \vec{n})$ is a valid opening. That is:

$$\begin{aligned}\hat{v} &= \langle \vec{c}, \vec{r'}||\vec{l'}\rangle + |\vec{n}|_\mu^2 \\ &= \langle \hat{\vec{c}}_r||\hat{\vec{c}}_l, \vec{r'}||\vec{l'}\rangle + |\vec{n}|_\mu^2 \\ &= \langle \hat{\vec{c}}_r, \vec{r'}\rangle + \langle \hat{\vec{c}}_l, \vec{l'}\rangle + |\vec{n}|_\mu^2\end{aligned}$$

Now sample $\vec{s}_{1:}$ uniformly at random, and set $s_0 = -\langle \hat{\vec{c}}_r, \vec{s}_{1:}\rangle$ as specified. To show that $(\hat{v} - s_0, (\vec{r'} + \vec{s}_{1:})||\vec{l'}, \vec{n})$ is a valid opening, we note that

$$\begin{aligned}\hat{v} - s_0 &= \langle \hat{\vec{c}}_r, \vec{r'}\rangle + \langle \hat{\vec{c}}_l, \vec{l'}\rangle + |\vec{n}|_\mu^2 - s_0 \\ &= \langle \hat{\vec{c}}_r, \vec{r'}\rangle + \langle \hat{\vec{c}}_l, \vec{l'}\rangle + |\vec{n}|_\mu^2 + \langle \hat{\vec{c}}_r, \vec{s}_{1:}\rangle \\ &= \langle \hat{\vec{c}}_r, \vec{r'} + \vec{s}_{1:}\rangle + \langle \hat{\vec{c}}_l, \vec{l'}\rangle + |\vec{n}|_\mu^2 \\ &= \langle \hat{\vec{c}}_r||\hat{\vec{c}}_l, (\vec{r'} + \vec{s}_{1:})||\vec{l'}\rangle + |\vec{n}|_\mu^2 \\ &= \langle \vec{c}, (\vec{r'} + \vec{s}_{1:})||\vec{l'}\rangle + |\vec{n}|_\mu^2\end{aligned}$$

which satisfies the definition.

To show that the vector components of the opening are uniformly distributed at random, we simply observe that $\vec{n}$ and $\vec{l'}$ are sampled uniformly at random by hypothesis, and that $\vec{s}_{1:}$ is sampled uniformly at random after $\vec{r'}$ is fixed. □

It is straightforward to generalize this construction such that the condition $\langle 1||\hat{\vec{c}}_r, \vec{s}\rangle = 0$ is relaxed to allow for any field element to prepend $\hat{\vec{c}}_r$ by defining s_0 accordingly; however, this is not needed for the use of the lemma in the preprint.

2.8.4 Lemma 7

In Appendix C.2, the proof of Lemma 7 asserts a property of uniform distribution relating to matrix multiplication.

For the purpose of this discussion, let M be an $m \times n$ matrix, so the linear transformation in question is from $\mathbb{F}^n \to \mathbb{F}^m$.

The reasoning of the proof is correct; however, it may be useful to provide a more complete explanation of why the reasoning about null space membership implies each preimage set $P_{\vec{b}} \subset \mathbb{F}^n$ must have the same cardinality, as this is important for the conclusion.

Further, it may also be helpful to note that the collection $\{P_{\vec{b}} : \vec{b} \in \mathbb{F}^m\}$ partitions $\mathbb{F}^n$, which makes it clear that the mapping of a vector $\vec{a} \in \mathbb{F}^n$ to a preimage set $P_{\vec{b}}$ is a well-defined operation.

2.8.5 Theorem 2

In Appendix C.2.1, the proof of Theorem 2 aims to show that the arithmetic circuit protocol described in Section 5.3 has the desired properties of perfect completeness, perfect special honest-verifier zero knowledge (SHVZK), and computational witness-extended emulation.

Showing perfect SHVZK requires constructing a simulator that, when given a statement and randomly-sampled verifier challenges, can produce a verifying proof transcript distributed identically to that from a real (non-simulated) valid proof.

We note the important point that the proof of the inner arithmetic circuit subprotocol calls into the weighted norm linear argument, which is not perfect SHVZK. This means that the simulator does not need to simulate the transcript of this argument; given its witness, it simply runs the argument itself. Even though the argument witness is not strictly part of the overall proof transcript, it suffices for the simulator to simulate this witness in a distribution identical to that used by the prover in a real proof.

The proof claims that perfect SHVZK follows by showing that the weighted norm linear argument prover input $(v, \vec{l}, \vec{n}) = (v(\tau), \vec{l}(\tau), \vec{n}(\tau))$ be "distributed uniformly at random among valid openings" as defined in Lemma 6, clarifying this to mean that the opening is distributed uniformly at random.

This is not strictly correct, as v is uniquely determined by $\vec{l}$ and $\vec{n}$ given fixed parameters $\vec{c}$ and μ and cannot be independently sampled. However, despite this (and despite similar language in the statement and proof of Lemma 6), this is easily addressed. All that must be shown instead for these particular witness elements is that $\vec{l}$ and $\vec{n}$ be distributed uniformly at random using the argument of Lemma 6, and that v be chosen accordingly to ensure that the "full" witness $(v, \vec{l}, \vec{n})$ is a valid opening.

It is also important to ensure that the conditions of (the modified) Lemma 6 hold relating to the order in which related values are fixed or otherwise chosen or sampled. Otherwise, it may be possible that they are adaptively chosen such that the intended distribution does not hold, which would render the simulator invalid. Neither the proof nor the statement of Lemma 6 addresses this explicitly.

The proof does not fully provide the correspondence between arithmetic circuit protocol elements and those required by Lemma 6. It does indicate that vectors $\vec{l'}$ and $\vec{n}$ are distributed uniformly at random. However, there is no $\vec{l'}$ vector defined in the arithmetic circuit protocol. It is likely that the authors intended the vector $\hat{\vec{l}}(\tau)$ instead (and $\vec{n}(\tau)$), and that these are intended to correpond to Lemma 6's $\vec{l'}$ and $\vec{n}$ vectors.

The vector $\hat{\vec{l}}(T)$ in the arithemtic circuit protocol (evaluated at $T = \tau$) contains as a summand $T^{-1}\vec{l}_S$, where $\vec{l}_S$ is sampled uniformly at random by the prover, and where τ is subsequently sampled uniformly at random by the verifier. Further, all other summands are defined in advance of these values, so the overall vector is distributed uniformly at random.

Similarly, the vector $\vec{n}(T)$ in the protocol contains $\hat{\vec{n}}(T)$ as a summand. This in turn contains $T^{-1}\vec{n}_S$ as a summand, where $\vec{n}_S$ is sampled uniformly at random by the prover, and where τ is subsequently sampled uniformly at random by the verifier. All other summands are defined in advance of these values as well, so the overall vector is distributed uniformly at random.

The proof specifies that $(\hat{f}_0, \beta^{-1}\hat{\vec{f}}_{1:})$ constitutes the "primitive witness" from Lemma 6. This term is undefined, and is ambiguous since the structure of Lemma 6 does not directly admit a witness. It is likely that this tuple is intended to play the role of $(\hat{v}, \vec{r'})$ in the lemma.

The proof then indicates that it must be shown that the values $(v_b, \vec{r}_b) = (s_0 = r_0 - \hat{f}_0, \vec{s}_{1:} = \vec{r}_{1:} - \hat{\vec{f}}_{1:})$, are distributed uniformly at random. However, the notation $(v_b, \vec{r}_b)$ is undefined, both within the arithmetic circuit protocol and Lemma 6, leaving the reader again to divine the intent. It may be the case that the intent was $(v, \vec{r})$ from the last paragraph in the original Lemma 6, but this does not appear to be consistent with the subsequent definition in terms of s_0 and $\vec{s}_{1:}$, which are presumably intended to correspond to the vector $\vec{s}$ from the lemma.

The proof observes that the vector $\vec{s}$ defined in the protocol (which is to be shown to be comprised of uniformly-distributed elements) is itself a linear combination of the vectors $\vec{r}_L, \vec{r}_R, \vec{r}_O, \vec{r}_V$. To show the desired distribution, it claims to suffice to show that an undefined matrix has rank 7 with image dimension 8. It may be the case that the matrix is intended to be constructed with each of the above vectors as row vectors; however, this is not specifically described. Further, while it is known that random matrices have full rank, it is not clear why this rank argument must result in a random matrix. While it is the case that each of the above vectors is used in the linear construction of $\vec{s}$, they are not the only such vectors: according to a possible interpretation of the incomplete notation, the vector $\vec{r}_s$ may also need to be included, which is not accounted for. Additionally, the proof claims that the construction of the unspecified matrix is such that $\langle \hat{\vec{c}}_r, \vec{s} \rangle = 0$ by Lemma 7. This appears to be an intended consequence of Corollary 1; however, such an application of the corollary requires that $\hat{\vec{c}}_r$ generate the left null space of the matrix, which the proof does not show.

The proof then specifies that the simulator choose the opening $(v, \vec{l}, \vec{n})$ and all commitments aside from C_S uniformly at random, and then define C_S to satisfy Equation 34. These transcript elements are C_L, C_R, C_O. As noted, the opening cannot fully be sampled uniformly at random due to the structure of v; this is a minor oversight. The claim regarding Equation 34 is incorrect, as this equation does not involve C_S. It is likely that the authors intended this to refer to Equation 15, which defines the weighted norm linear argument relation, instead. That is, once the simulator has sampled $\vec{l}, \vec{n}, C_L, C_R, C_O$, it computes v and C_S directly according to the relation, which results in an accepting transcript.

To show computational witness-extended emulation, the proof informally describes the outline of an extractor that, given a suitable tree of transcripts, claims to extract a valid witness. It does so primarily by invoking the round extractor defined in Lemma 5. However, it does not specifically identify the means by which the round extractor is to be instantiated, nor does it assert the validity of the round extractor (which has specific requirements on its inputs). This is unfortunate, as the proof elements used in such a construction are subject

to several layers of definitional indirection that leave the reader to discern the extractor's construction.

The proof indicates that after using the weighted norm linear argument extractor χ_{nl} to extract a weighted norm linear argument witness $(v, \vec{l}, \vec{n})$, it can again apply the round extraction technique from Lemma 5 to obtain openings for $C_S, C_O, C_L, C_R, \hat{V}$. While the proof does not specify details, this is presumably done using the known linear combination

$$C(T) = P(T) + T^{-1}C_S + \delta C_O + TC_L + T^2 C_R + T^3 \hat{V},$$

where $C(T) = v(T)G + \langle \vec{l}(T), \vec{H} \rangle + \langle \vec{n}, \vec{G} \rangle$ via the weighted norm linear argument, and where $P(T)$ is computed by the verifier using $p_s(T)$ and $\vec{p}_n(T)$ defined earlier in the protocol. To apply the round extraction technique from Lemma 5, it must be the case that the witness element inner products be a linear combination of linearly-independent polynomials. This is *nearly* the case for the linear combination comprising $C(T)$ relative to the intended commitments; that is, $C(T)$ directly contains proof elements multiplying polynomials in the linearly-independent set $\{T^{-1}, 1, T, T^2, T^3\}$. However, this alone does not account for $P(T)$, which contains terms of conflicting degree. Strictly, the extractor is able to produce the desired openings only with an offset applied via each corresponding degree component of $P(T)$; it would be useful for the proof to explain this, as it may not be obvious to the reader why this is possible due to the presentation of $C(T)$. (As an example, if the corresponding opening to $P(T)$ were not known to the verifier, this technique would not be possible.)

The proof has a minor typo and related lack of clarity, stating that "polynomial $\hat{f}(T)$ terms for -2 through 6, and $g(T)$ has terms for -2 through possible 10" instead of the more clear and correct "polynomial $\hat{f}$ has terms for degrees -2 through 6, and $g(T)$ has terms for degree -2 through (possibly) 10". Interestingly, the proof differentiates the highest-degree (nonzero) term between the two polynomials, despite this not being necessary for the subsequent vanishing argument to hold.

The proof does not detail the derivation of Equation 96, which relates to the discussion of Equation 34 for producing the necessary connection to circuit satisfiability in the sense of Equations 30 and 31. In Equation 96, the notation $\vec{n}_{\hat{V}}$ used in a specified inner product is undefined. The subsequent discussion of value term challenge "unwinding" does not detail the terms it references.

The proof again invokes the round extractor of Lemma 5, using the structure of $\hat{V}$ as a linear combination of $\{V_i\}$ weighted by powers of λ or μ (depending on the corresponding binary flags f_l, f_m). In the case where exactly one of f_l, f_m is nonzero, the validity of the round extractor argument (implicitly, but not explicitly, defining $\hat{V}$ as a polynomial evaluated at this challenge) seems straightforward. However, the proof is unclear in the case where $f_l f_m = 1$ about the nature of the round extractor, as $\hat{V}$ would be presumably be represented in this case by a multivariate polynomial such that Lemma 5's conditions would not hold.

The remainder of the discussion of the extractor in the proof is informal

and lacks detail about the precise definition and operation of the extractor; this aspect of the proof is critical to establishing witness-extended emulation and showing the protocol is sound.

2.8.6 Theorem 3

In Appendix C.3, the proof of Theorem 3 aims to show that the reciprocal form protocol described in Section 6.3 has the desired properties of completeness, perfect special honest-verifier zero knowledge (SHVZK), and computational witness-extended emulation.

The correctness discussion correctly identifies that the protocol does not have perfect correctness, as the challenge α may be such that $(\alpha + w_{D,i})^{-1}$ is undefined for some i; it informally notes that this occurs only negligibly. This claim does certainly hold in the case where N_P (the number of poles) is negligible in the order of the challenge codomain, but the requirement that this be the case does not appear to be specified outside of the proof. Even though this will hold for any reasonable range proving application that uses the protocol, it need not hold generally. It may be helpful to make this clear, either when the relation $\mathcal{R}_{rf}$ is introduced in Equation 71 or in the statement of Theorem 3.

To show perfect SHVZK, the proof merely notes that the simulator may sample the verifier challenge α uniformly at random and then run the arithmetic circuit protocol simulator from the proof of Theorem 2 on the corresponding circuit.

This is intuitive, especially given the discussion in Section 6.2 that describes the relationship between the reciprocal form and arithmetic circuit protocols. However, it is informal and not strictly correct. Effectively, the reciprocal form protocol separates the corresponding arithmetic circuit protocol's first round into two rounds: instead of running CommitOL and CommitR to obtain C_O, C_L, C_R to send to the verifier in one round, the prover first runs CommitOL to obtain (among other values) C_O, C_L. It sends these values to the verifier, obtains a challenge α, and then computes inputs to CommitR, which it runs to obtain C_R. The prover then sends C_R to the verifier and computes witnesses that it uses as inputs to the inner arithmetic circuit protocol.

Technically, the inner arithmetic circuit protocol invocation duplicates the prover's communication to the verifier: its first step is to send C_O, C_L, C_R. In the reciprocal form protocol, these values have already been sent to the verifier, in part to obtain the initial challenge α. This is mostly a technicality, but does highlight that the simulator for the arithmetic circuit protocol (which uses the inner arithmetic circuit protocol as a subprotocol) does not trivially apply. It also may be useful to modify one or more of the protocol definitions to correct this discrepancy.

In the arithmetic circuit protocol simulator construction from the proof of Theorem 2, it is easy to show that C_O, C_L, C_R may be independently sampled uniformly at random because of the construction of (and inputs to) CommitOL and CommitR. This reasoning does extend to the initial two rounds of the reciprocal form protocol simulator due to its invocations of these subprotocols,

meaning its simulator can in fact sample C_O, C_L, C_R as in the arithmetic circuit protocol simulator.

It may be useful to adjust the proof's discussion of the simulator to note that it is not strictly an invocation of the arithmetic circuit protocol simulator, but that similar reasoning applies to produce an effectively parallel construction.

To show that the protocol has computational witness-extended emulation, the proof invokes the arithmetic circuit protocol extractor χ_{ac} defined in the proof of Theorem 2. Similarly to the above simulator analysis, it is not strictly valid to invoke χ_{ac} as the reciprocal form protocol does not directly invoke the arithmetic circuit protocol in its definition. However, the subsequent analysis assumes a fixed challenge value α in order to extract an arithmetic circuit protocol witness. Because the openings of C_O, C_L, C_R in the arithmetic circuit protocol rely only on rewinding the transcript at the τ challenge point, the analysis should be valid here, as the reciprocal form protocol fixes these commitments before invoking the inner arithmetic circuit protocol where this challenge point occurs.