A Further Review of the DL Gadget Of Interest

Brandon Goodell, Rigo Salazar, Freeman Slaughter, Luke Szramowski

Cypher Stack

May 24, 2025

As with any such report, this document may contain errors, and we cannot guarantee the correctness or security of the scheme in question. We also can make no guarantees about the correctness, security, or suitability of any implementations for intended use cases. We note this report has not undergone any formal or peer review.

Change Log

This document may be updated occasionally, especially if security-sensitive results come to light, or if URL references change. We summarize such changes here.

- May 24, 2025. Initial upload to GitHub.

1 Introduction

Eagen presented the barebones sketch of a scheme for demonstrating the correct computation of sums of points in an elliptic curve group in [Eag22], and is based on the theory of divisors (which goes back at least to [DW82]). Eagen's approach lends itself to probabilistically checkable proof schemes, especially for efficient full-chain membership proofs for Monero as described in [Par24a]. Bassa investigated further in [Bas24c], [Bas24a], and [Bas24b]. An implementation Sage by Eagen is at [Eag24]. Eagen's implementation inspired the implementation by Parker in Rust at [Par24a] (and Parker's implementation is described in pseudocode by Parker at [Par24b]). These implementations are variations of the protocol described in [Bas24b], and both pass basic correctness tests. The overall approach was commented upon in [BHLS25] as possibly useful in exponent-VRFs. Cypher Stack also wrote a review of [Bas24c] in [CS].

Great material may come from Eagen's work on divisors and Bassa's follow-ups, but more time is necessary. Production deployment of code based on these approaches is premature. We find the following troubling issues have not fully been addressed, which range from superficial to serious. We describe the approach from a high level in Section 2, elaborating on our complaints along the way.

- The informality of the work in [Eag22] leads to an accumulating cascade of unclear reasoning, leading to unseen complications, weak conclusions, and more.
- Even after Bassa's clarifications in [Bas24c], [Bas24a], and [Bas24b], there still still seems to be some mistakes related to calculus and the application of the Schwartz-Zippel lemma. Specifically, the verification equations may have terms excluded which have no impact on correctness but do impact soundness. These mistakes seem to be restricted to generalizations over higher multiplicities, and they seem to be correctable. Nevertheless, such mistakes would not be caught by typical correctness tests, and fixing them will require a nontrivial amount of work.

- Even after corrections are made, the resulting scheme is (or rather, the schemes described in [Bas24c], [Eag24], and [Par24a] are) highly malleable and with a non-zero soundness error, introducing unnecessary attack surfaces and calling soundness results into question.

2 High Level Description

We briefly explain Eagen's method informally, and along the way explain the difficulties in proving the soundness and correctness of the scheme. We explain it this way to provide the reader contextual scope to our concerns and issues.

Our explanation is based on a verifiable sums-of-algebraic-points-as-a-service thought experiment. In our thought experiment, Alice first presents a trivial service, Bob presents a different service, and so on until we arrive at Eagen's method.

2.0.1 Trivial Solution

For customers who have computational constraints, the service computes sums of points in certain elliptic curve groups $\mathbb{G}$ and provides efficient proofs of correctness. Customers query the service with some $N \geq 1$ distinct non-identity group elements $P_1, \ldots, P_N$ and corresponding integers $m_1, \ldots, m_N$. In response, customers receive a pair (Q, π) where $Q = \sum_i m_i P_i$ and π is a proof. The service uses probabilistically checkable proof systems consisting of pairs of algorithms $\Pi = (\texttt{Prove}, \texttt{Verify})$ to show $Q = \sum_i m_i P_i$, asking their customers to execute `Verify` to check proofs.

We begin with Alice, who produces empty proofs, relying on the customer already having access to the points P_i and multiplicities m_i. Alice expects verifiers to compute $Q' = \sum_i m_i P_i$ themselves and check whether $Q = Q'$. To check Alice's work, Alice's customers need to recreate all the work Alice did. Alice's customers receive their points Q but remain dissatisfied with this "service."

2.0.2 Function Field $K(E)$ and `div`

Bob, on the other hand, notices that $Q = \sum_i m_i P_i$ implies $(\sum_i m_i P_i) + (-Q)$ is a principal divisor. Bob recalls the existence of a surjective map `div` from the (multiplicative subgroup of the) function field to the (additive) group of principal divisors. This map encodes roots and poles of f as a formal sum of points P_i with integer coefficients with $m_i \in \mathbb{Z}$, where the sign indicates multiplicities.

Bob can find a function field element to show his customers and use that as the proof. Bob's customers just need to enumerate the roots and poles of f, and their multiplicities. Bob's service can convince customers if there is a reliable proof of a theorem similar to the following.

Theorem 1. There exists a surjective map `div` carrying elements f to the principal divisor of its roots and poles, weighted by multiplicity.

Even though this theorem is inarguably true, Bob's customers remain dissatisfied, because they enumerate all roots and poles of f, and their multiplicities.

2.0.3 Weil's Reciprocity

Charlene improves Bob's service by exploiting Weil's reciprocity. For any $f, g \in F$, Weil's reciprocity states $f(\texttt{div}g) = g(\texttt{div}f)$. Indeed, Charlene's customers expect f to have some target principal divisor, namely $(\sum_i m_i P_i) + (-Q) - m^* O$. Rather than asking her customers to fully enumerate the roots and poles of f, Charlene increases the stakes by asking her customers to compare $f(\texttt{div}g)$ against $g((\sum_i m_i P_i) + (-Q) - m^* O)$ for every $g \in F^*$. If the customer finds any $g \in F^*$ such that $f(\texttt{div}g) \neq g((\sum_i m_i P_i) + (-Q) - m^* O)$, then certainly $\texttt{div}f \neq (\sum_i m_i P_i) + (-Q) - m^* O$. Charlene's customers are tremendously annoyed, though, because now instead of computing $\sum_i m_i P_i$ directly once, or instead of finding all the roots and poles and their multiplicities for f, they must check every $g \in F^*$.

Charlene's method is valid following Weil's reciprocity, which has a proof, and computations involving Weil's reciprocity are made much more clear using resultants of function field elements.

2.0.4 Schwartz-Zippel lemma

Danielle improves upon Charlene's method with the Schwartz-Zippel lemma. Danielle asks her customers to handle $f(\texttt{div}g) - g(\texttt{div}f)$ as a polynomial evaluated at the coefficients of g and the point-multiplicity pairs in the principal divisor $\texttt{div}g$. Thus, by sampling the coefficients of the rational g, computing $\texttt{div}g$, and checking whether $f(\texttt{div}g) - g(\texttt{div}f) = 0$, Danielle hopes to rely upon the Schwartz-Zippel lemma to conclude $f(\texttt{div}g) - g(\texttt{div}f)$ is the zero polynomial with high probability. Danielle's customers are the first to be pleased. Instead of finding an exhaustive list of roots and poles and their corresponding multiplicities and making sure the list is correct, or instead of having to compare f against every g, customers merely need to sample g and evaluate.

Unfortunately, the usual Schwartz-Zippel lemma is for univariate polynomials. Certainly, f and g are both rational, and the points in $\texttt{div}f$ are fixed by the choice of f and lie on the elliptic curve. Also, g is dependent on the random challenge line. However, it is not clear whether the points in $\texttt{div}g$ are rational functions of the line parameters λ, μ. Thus, $f(\texttt{div}g) - g(\texttt{div}f)$ does not necessarily fulfill the hypotheses of the Schwartz-Zippel lemma. This is touched upon by Bassa in [Bas24c] throughout the narrative, and is claimed to have been rectified in a 4-page discussion. If the following theorem is true, then Weil's reciprocity can be handled as described.

Theorem 2. There exists a polynomial extension of $K[X, Y]$, say $K[\mathcal{X}]$, over which the points in $\texttt{div}g$ are rational functions of the coefficients of g.

As a corollary, if $f(\texttt{div}g) - g(\texttt{div}f)$ can be handled as a rational function, then logarithmic derivatives apply; otherwise, logarithmic derivatives must be extended to whatever set of functions contains $f(\texttt{div}g) - g(\texttt{div}f)$.

2.0.5 Logarithmic Derivatives

Eagen improves Danielle's service by applying the logarithmic derivative to the polynomials on both side of the equality $f(\texttt{div}\widehat{g}) = \widehat{g}(\sum_i m_i + (-Q) - m^*O)$, simplifying the computations by ensuring that the derivative is taken in the direction of the linear g, and by performing more verifiable computations on behalf of the verifier. Eagen uses the derivation "in the direction of" the linear g to ensure the random data presented by g does not introduce undesirable terms.

In a typical college calculus course with real numbers, we use logarithmic derivatives with impunity. We can rely on $\sum_i \frac{f_i'(X)}{f_i(X)} = \sum_j \frac{g_j'(X)}{g_j(X)}$ if and only if $\prod_i f_i(X) = \prod_j g_j(X)$, where these equalities as functions imply their equalities upon common evaluations. The justification is a little more complicated when the evaluations occur at different points; to handle this, we lift $f(X) \in \mathbb{R}[X]$ to $\prod_i f(X_i) \in \mathbb{R}[X_1, \ldots, X_N]$, where each factor has a different indeterminate. We then carefully define a derivation on $\mathbb{R}(X_1, \ldots, X_N)$ so that we can compute and evaluate the logarithmic derivative. Bassa hints at this, but does not discuss it formally, taking the application of the logarithmic derivative to behave as expected due to its homomorphicity.

In a typical college calculus course, we also deal with the *chain rule* obsessively. To ensure that computations are done correctly, all computations must be described in full, and then compared against complicating factors like the chain rule. Some of the expressions in [Eag22] use unstated simplifications which do not generally hold. Some one-line computations performed by Eagen often require several pages of background material, and which are only justified by several more pages of raw computation.

For example, if d is a derivation in the direction of the affine line $Y - \lambda X - \mu$ as described by Eagen, then we require $d(Y - \lambda X - \mu) = 1$, so $dY = \lambda dX + 1$. A derivation on $K(X, Y)$ over K is of the form $a\frac{\partial}{\partial X} + b\frac{\partial}{\partial Y}$ for some polynomials a, b. Thus, $dY = b$ and $dX = a$, so we require $b = \lambda a + 1$ for polynomials a and b. Thus, $b - \lambda a \in K$ must have degree 0, so a and b have the same degree, and selecting a fixes b. However, projective space reveals some additional details. The homogenization of the line is $Y - \lambda X - \mu Z$, so actually we require that $dY = \lambda dX + \mu dZ$. Then if $d = a\frac{\partial}{\partial X} + b\frac{\partial}{\partial Y} + c\frac{\partial}{\partial Z}$, then $d(Y - \lambda X - \mu Z) = 1$

implies $b - \lambda a - \mu c = 1$. Now $b - \lambda a - \mu c \in K$ must have degree 0. In particular, $b - \lambda a$ must have the same degree as c, and selecting a, c fixes b.

These details seem to "wash out" of the calculus computations demonstrated so far, and it is not totally clear whether they were handled appropriately by merely re-working the discussion presented by Bassa, despite that Bassa took great care in these computations.

While justifying the use of logarithmic derivatives over rational functions in a finite field is a nontrivial but simple exercise, it is the method upon which all of Eagen's approach rests. A proof is not just customary, but necessary. Indeed, as discussed above, the element $f(\texttt{div}g) - g(\texttt{div}f)$ must be proven to be rational in $(g, \texttt{div}g)$ or further work must be done. For Eagen's method to be valid requires a proof of a theorem along the following lines.

Theorem 3. Let $f, g \in F^*$, let d be a derivation on $K(X, Y)$ over K restricted to F^*, and let $\texttt{div}g = \sum_i m_i P_i - m^* O$. Define the map θ carrying $f(\texttt{div}g) \mapsto \sum_i \left[\frac{m_i df}{f}\right]_{(X,Y)=P_i}$. Then Weil's reciprocity holds if and only if $\theta(f(\texttt{div}g)) = \theta(g(\texttt{div}f))$ as functions.

3 Rust Implementation

Parker's Rust implementation in [Par24b] is not problematic.

This code appears to correctly recreate Eagen's Sage implementation in [Eag22], which appears to (i) correctly compute a function field element for a principal divisor and (ii) computes the Eagen-style proof for the target principal divisor. Whereas (ii) is a matter of computing some derivatives, (i) is the more interesting component.

For (i), the code works by inputting a divisor, say $\sum_i m_i P_i = O$, and looping through the points. Say $\ell_{i,j}$ is the line interpolating P_i and P_j when $P_i \neq P_j$ (and P_i with $-P_i$ when $P_i = P_j$). The code pairs these points off to pairs (P_i, P_j), computing a running product of the values $\texttt{out} \leftarrow \frac{\ell_{ij}}{\ell_{ii}\ell_{jj}} \times \texttt{out}$ and a running sum of the touched points (counting multiplicities). The code correctly works. Indeed, these points admit an interpolation which is a function field element if and only if the input divisor is principal, say $\sum_i m_i P_i = O$.

For example, say we have distinct non-identity points P, Q, R, S, such that $P + Q + R + S = O$ with no sub-multisets which sum to O. The code finds ℓ_P, ℓ_Q, ℓ_R, and ℓ_S the vertical lines passing through P, Q, and R, respectively. The code starts with the running product $\texttt{out}_{\text{prod}} = 1 \in K(E)$, the function field. In the first step, the code has list of points (P, Q, R, S), so the code pairs (P, Q) and (R, S). The code finds lines ℓ_{PQ} and ℓ_{RS} and computes $\texttt{out}_{\text{prod}} = \frac{\ell_{RS}}{\ell_R \ell_S}$ and $\texttt{out}_{\text{prod}} \leftarrow \frac{\ell_{PQ}}{\ell_P \ell_Q}$. In the second step, the code repeats with the list $(P + Q, R + S)$, finding line $\ell_{P+Q,R+S}$ and computing $\texttt{out}_{\text{prod}} = \frac{\ell_{P+Q,R+S}\ell_{PQ}\ell_{RS}}{\ell_{P+Q}\ell_{R+S}\ell_P\ell_Q\ell_R\ell_S}$. It is straightforward to check that each of P, Q, R, and S are roots of $\texttt{out}$ with multiplicity 1 each. In the case that the list of points has an odd length, the code pairs off points, carrying an odd one over to the next round.

Hence, Parker's implementation appears to faithfully recreate Eagen's implementation, which appears to correctly construct principal divisors from sums of points to zero.

4 Elaborations

4.1 Slow is fast

Rapid timelines for analyzing cryptographic schemes introduce informality and incompleteness. Informality and incompleteness introduce problems due to lack of rigor, and these problems lead to longer timelines for assessment, development, and deployment. It costs time for a reader to look up a definition from another reference. It costs time for a reader to sit down and puzzle out a simple but non-obvious proof. Clarifications, done right the first time, prevent the need for further clarification; it costs time to ask a reviewer to write a clarifying paper, and to rush these analyses is to ensure that clarifications are necessary.

The reason that the cryptography community formalizes security definitions, and then prove schemes secure under those definitions, is to avoid just this class of problem. This is particularly important in a scheme with many moving parts, even if some of those parts are straightforward: any ambiguity in the underlying project expands nonlinearly as the material gets deeper.

Even the status of the DL gadget of interest as a verifiable computation scheme (VCS) or a probabilistically checkable proof (PCP) is not stated in any of the cited documents, so neither are definitions of correctness or soundness stated. Without these fundamental definitions, no proof of soundness is acceptable: against what definition is soundness being checked? Can a reviewer check that a protocol Π is of type T and has soundness if T is not specified and definitions of soundness depend on T?

Assuming the reader is already familiar with basic probability, basic abstract algebra, and basic cryptography concepts like elliptic curves over finite fields and their group laws, most readers would still need detailed explanations on the use of the following concepts in the specific context of Eagen's work.

- The additive group of divisors of an elliptic curve group and its subgroup of principal divisors.
- The Picard group of an elliptic curve $\mathrm{Pic}(E)$, its subgroup of degree-zero divisor classes, $\mathrm{Pic}^0(E)$, the isomorphism $E \cong \mathrm{Pic}^0(E)$, and the distinction between principal divisors and degree-zero divisors.
- The function field $K(E)$ and the surjective group homomorphism $\texttt{div} : K(E)^* \to \mathrm{Pic}^0(E)$, Weil's reciprocity, and resultants.
- The use of calculus and logarithmic derivatives in a function field for an elliptic curve, $K(E)$.
- The use of the Schwartz-Zippel lemma.

To emphasize the scale of the topics being discussed, these topics are complicated enough to justify stand-alone courses in graduate-level education. We do believe that this work is not only worthwhile, but it could benefit from more work.

4.2 Disagreement in Sources

There is a deep disconnect between Eagen's narrative and the subsequent papers. This disconnect is entirely due to the sheer volume of material swept under the rug in Eagen's work.

For example, Eagen states that correctness follows directly from Schwartz-Zippel and describes its application as "straightforward." Bassa, on the other hand, refers to the use of (univariate) Schwartz-Zippel in Eagen's work as "problematic" in [Bas24c], and spends a good deal of effort attempting to use it correctly before resorting to a model of the situation using the elliptic surface $E \times E$. Similarly, Eagen leaves the central calculus work to three paragraphs worth of narrative, and an exercise "left to the reader." Bassa takes 2 to 3 pages to verify only some of these computations in [Bas24a]. These computations rely upon around 2 pages of background material.

To Eagen's credit, an elaboration on the calculus-based material in the case of higher multiplicities is available, with more detailed computations. However, these remain insufficient in the face of the real depth required to check all the work.

4.3 Malleability

Eagen's approach is not exactly malleable. Indeed, in a malleable scheme, an adversary could take a proof π for the statement $y = f(x)$ and massage it into a different proof π', which is valid for showing that $y' = f(x')$, for some other values x', y'. Eagen's approach, on the other hand, allows the same proof π to be used to verify a whole family of (x, y)-pairs.

In this sense, Eagen's approach is an *implication* problem, not a malleability problem. Indeed, a proof π which convinces a reader that some tuple $(Q, \vec{P}, \vec{m}) \in \mathbb{C}^{N+1} \times \mathbb{Z}^m$ satisfies the relation $Q = \sum_i m_i P_i$, in truth, convinces a reader that a set of tuples $\left\{(Q_\ell, \vec{P}_\ell, \vec{m}_\ell)\right\}_\ell$ all satisfy $Q_\ell = \sum_i m_{\ell,i} P_{\ell,i}$ for every ℓ.

This means that proving Eagen's approach to be sound under relation $Q = \sum_i m_i P_i$ is not the correct tactic. At best, such a proof demonstrates soundness only on a subset of proof implications!

So, if f is a function field element attesting to the principality of a divisor corresponding to $D = \sum_i m_i P_i - m^* O$, and this principal divisor has a principal sub-divisor, say $\sum_i m'_i P_i - m'' O$, then the sub-divisor has some g attesting to this fact, and $\frac{f}{g}$ is an element of the function field. This implies that $\sum_i (m_i - m'_i) P_i - (m^* - m'') O$ is also a principal divisor. That is to say, principal divisors factor cleanly out of principal divisors. In this way, it seems that malleability for a proof π attesting to a principal divisor $D = \sum_i m_i P_i - m^* O$ is about *all principal divisors containing D or contained within D* as well as *additional points which happen to vanish on the random challenge line L.*

The possibilities of exploiting a bad proof via malleation are concerning. Indeed, Eagen's approach relies on Weil's reciprocity, a derivation d, and the equality in $\prod_j D(A_j) = \prod_i L(P_i)^{m_i}$. Equivalently, the equality $\sum_j \frac{dD}{D}\big|_{A_j} dA_j = \sum_i \frac{m_i dL}{L}\big|_{P_i} dP_i$, where L is the random challenge line. Note that if a function field element D and point-multiplicity pairs $\{(P_i, m_i)\}$ together constitute a proof which passes this verification equation, and $\{(P'_i, m'_i)\}$ is *any* set of point-multiplicity pairs such that $\prod_i L(P'_i)^{m'_i} = 1$, then D will also pass this verification equation for $\{(P_i, m_i)\} \cup \{(P'_i, m'_i)\}$ for any set A just as well, but only with respect to this same L. Similarly, D will work for $\{(P_i, m_i)\} \setminus \{(P'_i, m'_i)\}$ with respect to this same L. That is to say, the verification equation is independent of points whose images under the random challenge L form a non-trivial decomposition of 1 in K.

So, a semi-honest prover could compute a proof and, under the impression that they are pruning unnecessary data, try to eliminate nontrivial decompositions of 1. Given any one L, these sub-sums are not difficult to find, although when the number of points is even moderately large, the search space expands quite quickly. Unfortunately, each attempt comes with a new L, so this approach will require a collision in L. Or, a malicious prover could try to include additional points once they learn L to fool a verifier into thinking the prover knows the discrete logarithm relationship among the divisor points. Given any L, finding pre-images which generate nontrivial decompositions of 1 is easier than seeking subset sums which decompose 1, so this malicious prover can very rapidly pump out possible proofs looking for such collisions. Either way, the verifier will compute the random challenge L using the new modified data, and so such a trick could only work if the resulting L collides.

4.3.1 Developer watermarking

Therefore, a principal divisor corresponding to a proof is not unique. A given function field element can work to prove the principality of related divisors obtained with subset-sums to O in the divisor, or which decompose 1 nontrivially through the line ℓ.

This means that a malicious developer can insert additional points, or remove certain points, from a statement. This won't be detected if the proof remains valid, and the cryptographic size of $\mathbb{G}$ ensures that stashing a single random element in a divisor is essentially undetectable.

This does not immediately lend itself to soundness attacks, but the possibility is concerning and serious.

4.3.2 A practical attack for the cost of a hash collision

An attacker can produce a valid proof f for a principal divisor, say some $P + Q + R = O$. This proof comes with a random challenge line L. The attacker can find a decomposition of 1 on this line, say $S_1, \ldots, S_N$ and multiplicities $m_1, \ldots, m_N$ such that $\prod_i L(S_i)^{m_i} = 1$. This is not computationally hard. If the attacker can find such a non-trivial decomposition which hashes to the same L, then the prover can pass their proof for $P + Q + R = O$ off as a proof for $P + Q + R + \sum_i m_i S_i \neq O$, and the proof will pass verification. This specific attack does require the random challenge L to collide, which occurs with negligible probability, so the attacker will be looking for a long time.

However, other extant methods are likely available for generating bad proofs. Interestingly, bad proofs are less malleable than good proofs, because of the requirement that the hash L matches.

4.4 Soundness and Proof Implications

If Bassa's proofs are valid (or are correctable) and the soundness error is asymptotically negligible, it may still be the case that the practical soundness error in a given implementation may be unacceptably high. Indeed, all assessments of soundness error put forth so far have been only asymptotic assessments. Computing the literal, practical soundness error of a given implementation is therefore of tremendous importance, is nontrivial, and will depend on implementation, including the choice of elliptic curve over which the statements are being proven.

In the case that soundness error is not negligible, then in our lifetimes, we can expect to see divisors which are not principal and which pass verification. We can call these bad proofs. The existence of bad proofs alone can be manageable in certain use-cases, but in this case, a problem arises: *what are the implications of a bad proof?* Moreover, what do they look like?

As we saw above, the implications of a good proof for the statement $Q = \sum_i m_i P_i$ can be myriad, including a large set of related statements. These related statements are not false - they are simply not the statements that the prover was targeting. If

i) bad proofs are practically likely even though they are asymptotically negligible, and

ii) any given bad proof can be malleated into a proof for an arbitrary, adversarially selected statement,

then this scheme is not suitable for deployment. The former can be protected against with additional challenges, but the latter cannot. Bad proofs are inevitable, so the latter would be catastrophic. A spectrum is suggested by the latter point above by restricting which statements admit malleation. It is a hopeful sign that the hash collision attack described above seems to admit little malleation at all.

Unfortunately, no firm understanding of these bad proofs or their implications is immediately forthcoming. Despite the work in all the references herein, we are not much closer to understanding these problems. As such, because we are cryptographers who aim to only ever "do no harm" Monero, we cannot in good conscience recommend Eagen's approach for practical implementations until a deeper degree of research can put these fears to rest.

5 Conclusion

Bassa did an admirable job introducing rigor to Eagen's approach in [Bas24c], [Bas24a], and [Bas24b], and Parker's Rust implementation appears to correctly recreate the desired approach. However, the work by Bassa is not conclusive, and the informality of the original description is the pervasive cause. Left with only a best-guess for Eagen's thinking, Bassa's works make no claims to be self-contained or stand-alone documents, leading to compounded confusions.

More than a dozen pages of definitions and background facts are prerequisite to a formalization of Eagen's work. This is too much formal background to skip, and bolting formal arguments on top of an informal description leads to further questions about whether the formal arguments actually represent the intended ideas. In particular, it is not clear whether Bassa's protocol description matches Eagen's intended protocol, due to this ambiguity. Due to this, Eagen's Sage implementation, Bassa's described protocol, and Parker's Rust implementation become joint reference points. However, none of these are identical, leading to further ambiguity.

References

[Bas24a] Alp Bassa. On the Use of Logarithmic Derivatives in Eagen's Proof of Sums of Points. `https://repo.getmonero.org/-/project/54/uploads/bfe9f49326a843ef1c9466e30a5d42c8/VAR_Monero_Logarithmic_Derivatives_Final.pdf`, 2024. Accessed: 03-Mar-2025.

[Bas24b] Alp Bassa. Soundness Proof for an Interactive Protocol for the Discrete Logarithm Relation. `https://moneroresearch.info/259`, 2024. Accessed: 02-Apr-2025.

[Bas24c] Alp Bassa. Soundness Proof for Eagen's Proof of Sums of Points. https://repo.getmonero.org/-/project/54/uploads/eb1bf5b4d4855a3480c38abf895bd8e8/Veridise_Divisor_Proofs.pdf, 2024. Accessed: 03-Mar-2025.

[BHLS25] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. Exponent-vrfs and their applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 195–224. Springer, 2025.

[CS] Cypher Stack. A Review of Soundness of Divisors-based Proofs. https://github.com/cypherstack/divisor_deep_dive/tree/main/latex/sum_of_points.

[DW82] Richard Dedekind and Heinrich Weber. Theorie der algebraischen functionen einer veränderlichen. 1882.

[Eag22] Liam Eagen. Zero Knowledge Proofs of Elliptic Curve Inner Products from Principal Divisors and Weil Reciprocity. Cryptology ePrint Archive, Paper 2022/596, 2022. https://eprint.iacr.org/2022/596.

[Eag24] Liam Eagen. Sage implementations of divisors. GitHub, 2024. https://gist.github.com/Liam-Eagen/666d0771f4968adccd6087465b8c5bd4.

[Par24a] Luke Parker. Fcmp++. GitHub, 2024. https://github.com/kayabaNerve/fcmp-plus-plus-paper/blob/develop/fcmp%2B%2B.pdf.

[Par24b] Luke Parker. fcmp-plus-plus repository. GitHub, 2024. https://github.com/kayabaNerve/fcmp-plus-plus/blob/78754718faa21f0a5751fbd30c9495d7f7f5c2b1/crypto/divisors/src/lib.rs.