ECCFROG522PP: An Enhanced 522-bit Weierstrass Elliptic Curve

Víctor Duarte Melo and William J. Buchanan

Blockpass ID Lab, Edinburgh Napier University, Edinburgh

Abstract. Whilst many key exchange and digital signature systems still rely on NIST P-256 (secp256r1) and secp256k1, offering around 128-bit security, there is an increasing demand for transparent and reproducible curves at the 256-bit security level. Standard higher-security options include NIST P-521, Curve448, and Brainpool-P512. This paper presents ECCFROG522PP ("Presunto Powered"), a 522-bit prime-field elliptic curve that delivers security in the same classical ∼260-bit ballpark as NIST P-521, but with a fundamentally different design philosophy. All of the curve parameters are deterministically derived from a fixed public seed via BLAKE3, with zero hidden choices. The curve has prime order (cofactor = 1), a verified twist with a proven ∼505-bit prime factor, safe embedding degree (≥14), and passes anti-MOV checks up to $k \leq 200$ and CM discriminant sanity up to 100k. Unlike prior opaque or ad-hoc constructions, ECCFROG522PP is fully reproducible: anyone can regenerate and verify it byte-for-byte using the published scripts. The intent is not to outperform NIST P-521 in raw speed, but to maximise trust, verifiability, and long-term auditability in a practical curve of equivalent security level.

1 Introduction

Around 1985, Neal I. Koblitz [1] and Victor Miller [2] independently invented ECC. Their idea came from preliminary work conducted by H.W Lenstra Jr on using elliptic curves to crack the RSA method. His paper was entitled *Factoring Integers using Elliptic Curves* [3]. Although they created ideas between 1985 and 1987, the main methods of ECC did not take off until 2005. While NIST P-256 has become popular in TLS communications and secp256k1 is widely used for blockchain applications, in certain circumstances, there is a need for enhanced security, such as with NIST P-521, Curve448, and Brainpool-P512. This paper introduces the ECCFROG522PP curve, which improves transparency and reproducibility compared with the NIST P-521 curve.

2 Basics of Elliptic Curve Cryptography

Overall, there are three main curves used in production. With an Edwards curve, we have the form:

$$x^2 + y^2 = 1 + d \cdot x^2 y^2 \pmod{p} \tag{1}$$

and is named after Harold Edwards. An example of this curve is Ed25519. For a Montgomery curve, we have the form:

$$By^2 = x^3 + Ax^2 + x \pmod{p} \tag{2}$$

and named after Peter Montgomery. With the Weierstrass curve, we have the form:

$$y^2 = x^3 + ax + b \pmod{p} \tag{3}$$

and which is used with Bitcoin. This elliptic curve is defined with p, a, b, g_x, g_y, and n, and where (g_x, g_y) is a base point on our curve, and n is the order of the curve. The curve used in Bitcoin and Ethereum is secp256k1, and which has the form of:

$$y^2 = x^3 + 7 \pmod{p} \tag{4}$$

and where $p = 2^{256} - 2^{32} - 977$. We can also use the NIST P-256 (secp256r1) curve or the NIST-defined P-521 curve.

3   ECCFROG522PP: An Enhanced 522-bit Weierstrass Elliptic Curve

ECCFROG522PP is "Presunto Powered", and is a deterministically generated 522-bit elliptic curve with full and scripted reproducibility.

3.1   Motivation and Scope

Modern deployments of elliptic-curve cryptography benefit from curves whose origins are inspectable and reproducible. ECCFROG522PP was designed to satisfy two constraints simultaneously: (i) **strong, conventional ECC security properties**; and (ii) **public, deterministic generation** with minimal degrees of freedom, so independent parties can rebuild the curve byte-for-byte from scripts. The curve is integrated into an open-source file-encryption tool ("HippoFrog") HippoFrog Repository, but all cryptographic parameters, derivation rules, and verification steps stand on their own and are published.

3.2 High-Level Properties (Published)

Nothing up our sleeves Daniel J. Bernstein identifies secp256k1 and NIST P-256 (secp256r1) as unsafe [4]. Overall, at the current time, it is likely to be safe, but we cannot guarantee that. This is the reason that it cannot be marked as safe. The great debate on the security of NIST P256 focuses on how the parameters were generated, and which uses a seed value of 0xc49d360886e70493 6a6678e1139d26b7819f7e90. At the core of the debate is that the curve was first defined by Jerry Solinas and who at one time worked for the NSA [5]. Selecting such a large random value makes it difficult to check the security of the curve, and where the seed value could have been discovered to have a backdoor.

Overall, in the generation of the elliptic curve parameters, we can have randomly generated ones or specially selected ones. Curve 25519 has ones which are specially selected for performance and security, and the Brainpool ones are randomly generated. The method for generating random curves is well documented and includes the seed value used. This approach is the *nothing up my sleeves*, and the random seed can be shown to have some randomness from a random source. Unfortunately, we do not have the original proof of the random value for P256, as its source has been lost.

3.3 Curve co-factor

In cryptographic systems, we typically implement within a group with a prime order of l, and which defines the number of elements in a group. The problem we have, is that many of our existing elliptic curves do not have a prime number order, and there is typically a cofactor. This means that the order is $h.l$, and where h is typically 4 or 8. Overall, this can cause vulnerabilities, and which are fixed within the implementation of the method. This means that we cannot formally prove the methods. Unfortunately, though, prime-order curves provide a correct proof, but they are slower than the ones with cofactors. The solution to this was proposed by Mike Hamburg in his Decaf proposal [6]. With this, we can convert from a curve with a small cofactor into one which has a prime order, and with no loss of performance.

With Curve 25519, for example, we have a co-factor of 8. This means that the total number of points is $8.p'$ and where p' is a prime of:

$$2^{252} + 27742317777372353535851937790883648493 \tag{5}$$

Overall, we use a prime number of $2^{255} - 19$ (and which has 255 bits), and but the prime order value has 252 bits - due to the bits of the co-factor (as the value of eight has three bits).

With Curve 25519, we thus have an order which has a cofactor of eight. This can lead to a range of weaknesses, such as for octal spending. Sometimes this can be overcome with some bit twiddling, but the best way is to create a lightweight group that integrates on top of Curve 25519. To overcome this, we can

use Ristretto255 [6], and which has an order which is a prime number. The core usage is with Ed25519, and which provides digital signatures using Curve 25519. The order of Ristretto255 is $2^{252} + 27742317777372353535851937790883648493$, and which is the same as the subgroup prime order for Curve 25519. The security of Ristretto255 is thus the same as Curve 25519.

3.4 Curve properties

To overcome the issues related to the lack of transparency in the developed curve, we define the properties with:

- **Field**: prime p of **522 bits** (decimal and hex given in the spec).
- **Model**: short Weierstrass $y^2 = x^3 - 9x + b$ over $\mathbb{F}_p$, with $a = -9$.
- **Group order** N: **521-bit prime**; **cofactor = 1**.
- **Base point** $G = (G_x, G_y)$: generated deterministically; $\mathrm{ord}(G) = N$.
- **Twist**: order computed; **largest proven prime divisor** $\sim$**505 bits**.
- **Sanity**: anti-MOV check up to $k \leq 200$; CM discriminant passes small square-free sanity up to 100k.
- **Security framing**: in the **same ballpark as NIST P-521** ($\sim$**260-bit classical**); **no claim of being faster** than P-521.

3.5 Deterministic Generation via BLAKE3

Public Seed and Indices All parameters are derived from the **public seed**

`ECCFrog522PP|v1`

and **published indices** for the search:

- **b-index** $i = 1,294,798$
- **base-point index** $j = 0$

The "$j = 0$" outcome is simply where the deterministic search first encountered a valid base point of full order; although rare, this is entirely consistent with a blind, fixed-rule search.

4 ECCFROG522PP parameters (concise narrative)

ECCFROG522PP is defined over a 522-bit prime field with modulus
$p = 68647976601306097149819007990813932172694353001433054093944634591855431833976560521225596406614545549772963113914808580371219879997166438125740282911150580399$

It uses the short Weierstrass form $y^2 = x^3 - 9x + b$ with $a = -9$ and a deterministically derived coefficient
$b = 66113913618419585086045246993774479113899949001297542130776831122509641950938825109341549233710118205542545725598961368239935656330069556661974287606199119$

(found at index $i = 1{,}294{,}798$ under the published BLAKE3 pipeline).

The group order is prime,

$N = $ 68647976601306097149819007990813932172694353001433054093944634591855431833976547078399309980690724371788986343232184197382451179107260804349074955412511 56283

so the cofactor equals 1.

The Frobenius trace is

$t = $ 134428262864259238211779839767706826243829887687008899056337766653274986 3901757

yielding CM discriminant

$D = $ −256520948548522009231824897555627094008137834109075384238812591282940638274983686660617754444935299793675172689772006399405032312306051338446315069327125 45107

which passes small square-free checks up to 100k.

No embedding-degree vulnerabilities were found for $k \leq 200$, and the lower bound on the embedding degree is 14.

The quadratic twist has order

$\#E'(F_p) = $ 686479766013060971498190079908139321726943530014330540939446345918554318339765739640518828325383667277569398845974329633599885808870720719024056 1040978959797

with a largest proven prime factor of 505 bits, supporting standard invalid-curve/twist-resistance under subgroup validation.

A deterministic base point G derived from the same seed with index $j = 0$ has full order N, with coordinates

$G_x = $ 11483659870055913964623536371313631260976767098619949198405802655079012131788815900015100098140592301158799072401266653548293144687306675149107389798128134

and

$G_y = $ 30386944574284420243881321173706779431273439385121134630343186387096004511363257470251386108023914919140912764811056993539192024949028106865930301722863 95020

All parameters are fully reproducible from the public seed "ECCFROG522PP—v1" and the published indices. Independent verification with SageMath scripts confirms the prime order, twist factorisation, anti-MOV and CM checks. Security analysis places the curve in the same ∼260-bit classical security ballpark as NIST P-521, but with the advantage of a transparent and fully deterministic generation process.

Canonical b-Derivation Rule The **published reproduction script** specifies the exact canonical rule for computing b from the seed and index i:

$$b = (\text{BLAKE3}(\text{seed} \,|\, b \,|\, i) \bmod (p - 3)) + 2 \quad \in [2, p - 2].$$

This forces b into a safe range and ensures non-triviality of the curve search. The script then **selects the first** b that is (i) in the required range, (ii) non-singular, and (iii) meets the published prime/non-singularity criteria. The specific published result is the decimal/hex value listed in the specification.

Deterministic Base-Point Search The **base-point x-coordinate** candidate is derived as

$$G_x = \text{BLAKE3}(\text{seed} \,|\, G \,|\, j) \bmod p,$$

with the published script taking $j = 0$ and then checking whether the resulting point $G = (G_x, G_y)$ on E has **full order** N. In ECCFROG522PP, the **first** candidate tested ($j = 0$) already has $\text{ord}(G) = N$. This outcome is documented and verified by script.

Digest Lengths and Determinism The reproduction script uses **BLAKE3 with a 64-byte digest** for both b and G_x candidate derivations, truncating modulo the appropriate field value (or $p - 3$ for b). The seed, indices, and all constants are **published** and embedded in the script. Running the script reproduces the **exact** public parameters and all sanity checks.

4.1 Published Parameters (Selected)

- **Prime** p (522 bits): as printed in decimal and hex in the spec (hex begins with `0x2000...377`).
- **Curve**: $y^2 = x^3 - 9x + b$ over $\mathbb{F}_p$.
- b **(decimal & hex)**: published; produced by the BLAKE3 pipeline at $i = 1{,}294{,}798$.
- j-**invariant** (hex): published for independent verification.
- **Base point** $G = (G_x, G_y)$ with $j = 0$ and $\text{ord}(G) = N$.
- **Order** N: 521-bit prime; **cofactor = 1**.

All of the above are reproduced and verified by the included script and also restated in the long-form "Full Specification."

4.2 Security Checks and Sanity

Prime Order and Cofactor The group order N is **prime (521 bits)**, so the **cofactor is 1**. This eliminates small-subgroup concerns and simplifies validation and protocols (e.g., subgroup confinement checks reduce to $[N]Q = \mathcal{O}$). The script optionally proves the order using **PARI/ECPP**.

Anti-MOV and CM Discriminant Sanity

- **Anti-MOV**: the published check confirms **no** $k \leq 200$ with $p^k \equiv 1 \pmod{N}$.
- **CM discriminant** $D = t^2 - 4p$: sanity check for **small square factors up to 100k** passes.

These checks are implemented and reported by the reproduction script.

Twist Security The twist order is computed, and a **largest proven prime factor** of about **505 bits** divides the twist order, supporting resistance to invalid-curve/twist attacks in standard validation settings. The exact decimal value is published.

Embedding Degree (Lower Bound) The **embedding-degree lower bound is 14**, providing the expected safety margin against MOV-type reductions at the published sizes.

4.3 Reproducibility and Build Artefacts

Reproduction Script (SageMath + BLAKE3) A **stand-alone Sage-Math script** re-derives b, reconstructs E, re-derives G_x from j, verifies the j-invariant, recomputes the order (PARI/ECPP option), checks the twist factor, anti-MOV up to the published bound, CM discriminant sanity, and validates that $\text{ord}(G) = N$. It also emits human-readable reports and CSV facts. **Running the script is the normative procedure** to verify the entire publication.

System and Timing (for the Original Search) The original parameter search ran on **Linux 6.14**, **Ryzen 9 5950X (32 logical cores)**, with ~**130 GB RAM**, using **32 processes** and a batch/chunk configuration. The published runtime is about **216k seconds (~60 h)**. These details aid replication but are not required to validate correctness from the seed and indices.

4.4 Practical Integration (HippoFrog)

ECCFROG522PP ships in an **open-source** file-encryption tool. The implementation uses **ephemeral ECDH → HKDF-SHA-256 → AES-256-GCM**, with a deterministic, endian-safe on-disk format and an explicit **parameter-hash binding** (SHA-256 of the curve parameters) in each header for robust context binding. The repository includes code layout, CLI usage, a deterministic header structure, and a **MIT license**. (These engineering aspects are orthogonal to the curve design and are summarised here for completeness.)

4.5 Performance Positioning (No Speed Claims vs P-521)

ECCFROG522PP is **not claimed to be faster** than **NIST P-521**. The publication focuses on **reproducibility, transparency, and security sanity**, not micro-optimisations. A Sage-based **benchmark harness** is provided to compare scalar multiplication and ECDH against common curves (secp256k1, P-256, P-384, P-521, ECCFROG522PP), but this framework is for **measurement**, not for making speed claims here. Users should assume **performance at or below** P-521 unless and until they produce controlled, reproducible measurements in their own environment.

4.6 Validation Guidance (Step-by-Step)

1. **Obtain seed and indices**: `Seed = "ECCFrog522PP|v1"`, `b_index = 1,294,798`, `G_index = 0`.
2. **Re-derive** b via the canonical rule $b = (\mathrm{BLAKE3}(\mathrm{seed}|b|i) \bmod (p-3)) + 2$; confirm it matches the published decimal/hex.
3. **Construct** E over $\mathbb{F}_p$ with $a = -9$ and the published b. Verify the **published j-invariant**.
4. **Re-derive** G_x with $j = 0$ using BLAKE3; recover G on E and validate $\mathrm{ord}(G) = N$.
5. **Recompute** N (PARI/ECPP option available) and confirm **cofactor = 1**.
6. **Run sanity checks**: anti-MOV (to $k \leq 200$), CM discriminant small-square-free up to 100k, and twist large prime factor divisibility.

All of these steps are automated in the published reproducibility script.

4.7 Security Framing and Use

By construction and checks, ECCFROG522PP targets the **same classical security ballpark as P-521** ($\approx$260-bit) while providing a **deterministic, audit-friendly origin**. The curve's **cofactor-1** group, **twist factorization evidence**, and **anti-MOV/CM sanity** make it a conventional, defensible choice for protocols that already accommodate P-521-class security levels. The design intent is **trust minimization** through a simple, fully documented pipeline.

4.8 Limitations and Non-Goals

- **No speed claim vs P-521**: the project emphasizes **correctness, openness, and determinism**; users should not expect throughput advantages over a heavily engineered P-521 stack.
- **Scripted bounds**: anti-MOV bound and small-square-free CM checks are exactly as published; they can be extended by independent reviewers if desired.

4.9 Availability and Licensing

- **Curve data & scripts**: published with explicit seed/indices and complete SageMath code to regenerate and verify the curve and to emit machine-parsable facts.
- **HippoFrog tool** (using ECCFROG522PP): open-source, **MIT licensed**, with documented header format and parameter binding.

5 Conclusion

ECCFROG522PP demonstrates that a practical, P-521-class curve can be delivered with **zero ambiguity about its origin**. By deriving **every critical constant from a public seed via BLAKE3** and by providing **turn-key scripts** that re-create and re-check the curve end-to-end, the publication invites verification rather than requiring trust. This work complements the ecosystem of standardized curves with a design whose **primary value proposition is transparency and reproducibility**, not speed.

6 Appendix A — Selected Published Values (for quick reference)

- **Seed**: `ECCFrog522PP|v1`
- **Indices**: `b_index = 1,294,798`; `G_index = 0`
- **Field size**: 522-bit prime p (decimal/hex as published)
- **Curve**: $y^2 = x^3 - 9x + b$
- **Order**: 521-bit prime N, **cofactor 1**
- **Twist**: large proven prime factor $\approx$505 bits
- **Sanity**: anti-MOV to $k \leq 200$; CM discriminant small-square-free up to 100k
- **Reproduction**: SageMath + BLAKE3 script provided (derivation, checks, reports)
- **Integration**: HippoFrog tool; **MIT license**

7 Appendix B — Benchmark Harness (for independent measurements)

A public Sage-based harness is provided to time scalar multiplication (variable-/fixed base) and ECDH across several curves (secp256k1, P-256, P-384, P-521, ECCFROG522PP). It is suitable for **your own**, reproducible measurements; it does **not** imply any speed advantage of ECCFROG522PP over P-521 in this publication.

7.1 Artifacts and Reproduction

- **Full specification** (parameters, checks, timing environment).
- **Reproduction & verification script** (BLAKE3 seed/indices $\rightarrow$ parameters; ECPP/paricard; twist, anti-MOV, CM, reports).
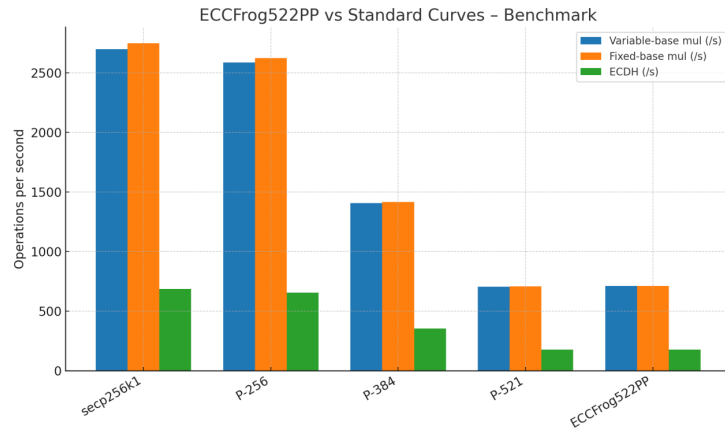- **Open-source tool integration** (format, parameter binding, license) HippoFrog Repository .

Fig. 1. Benchmark

References

1. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.
2. V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.
3. H. W. Lenstra Jr, "Factoring integers with elliptic curves," *Annals of mathematics*, pp. 649–673, 1987.
4. D. J. Bernstein, "SafeCurves: Introduction — safecurves.cr.yp.to," https://safecurves.cr.yp.to/, [Accessed 04-09-2025].
5. DiSSECT, "DiSSECT — dissect.crocs.fi.muni.cz," https://dissect.crocs.fi.muni.cz/standards/nist, [Accessed 04-09-2025].
6. M. Hamburg, "Decaf: Eliminating cofactors through point compression," in *Annual Cryptology Conference*. Springer, 2015, pp. 705–723.