Quasi-perfect (de)compression of elliptic curve points in the highly 2-adic scenario

Dimitri Koshelev[1⋆]

University of Lleida, Department of Mathematics, Catalonia, Spain
dimitri.koshelev@gmail.com

Abstract. This short note is devoted to a significant enhancement of [8] by resolving satisfactorily the problem formulated at the end of that article. More precisely, a new laconic, secure, and efficient (de)compression method is provided for points of any elliptic curve over any highly 2-adic finite field of large characteristic. Such fields are ubiquitous in modern elliptic curve cryptography, whereas they severely slow down the conventional x-coordinate (de)compression technique. In comparison with the main method from the cited work, the new one requires neither complicated mathematical formulas nor conditions on the curve. Thereby, the current work is universal and much more implementation-friendly, which justifies its existence, despite the absence of interesting mathematics behind it.

Keywords: (D)DoS attacks · elliptic curve cryptography · highly 2-adic finite fields · Müller's square-root algorithm · point (de)compression

1 Main

With the reader permission, the full introduction on *(de)compression of elliptic curve points* (in the *highly* 2*-adic* setting) is omitted to avoid repetitions with the fresh comprehensive article [8]. All the necessary details on the topic can be found in that article and especially in its introduction. Nevertheless, let's survey very briefly the state of affaires.

As usual, we are given an elliptic curve $E\colon y^2 = f(x) := x^3 + ax + b$ over a finite field $\mathbb{F}_q$ of large characteristic. In the past, *ECC (elliptic curve cryptography)* used to be mostly employed when the 2-adicity $\nu := \nu_2(q-1)$ is small or even $\nu \leqslant 2$. Therefore, the classical x-coordinate (de)compression method [4, Appendix D.2.1] was an ideal solution for compact point representation. Indeed, the follow-up decompression stage extracts $y = \sqrt{f(x)} \in \mathbb{F}_q$, which is readily done by *Tonelli–Shanks' algorithm* [12] or just by one exponentiation in $\mathbb{F}_q$. However, the base fields for many modern elliptic curves (see, e.g., [1]) are often highly 2-adic (i.e., $\nu \gg 2$) owing to advanced *zk-SNARK (zero knowledge succinct non-interactive argument of knowledge)* applications. As is known, Tonelli–Shanks' algorithm becomes very slow over such fields, since it requires $O(\ell + \nu^2)$ multiplications in $\mathbb{F}_q$, where $\ell := \lceil \log_2(q) \rceil$.

⋆ https://www.linkedin.com/in/dimitri-koshelev

Alternatively, there is *Müller's algorithm* [11] (look also at [9]) whose cost is close to that of a general field exponentiation (notably for the large ν), namely $\approx 2\ell - \nu$ multiplications in $\mathbb{F}_q$. Unfortunately, this algorithm does not function in constant time, which implies that in some rare situations it "freezes" for a quite long time. In a nutshell, [8, Section 3.1] explains how to leverage the given effect to mount a *(D)DoS (distributed denial-of-service) attack* on a decompressor based on the naive usage of Müller's algorithm. By the way, on the Internet page [10] there is a short discussion about the analogous issue in the hash-to-curve setting, although it can be seemingly fixed by means of a resilient hash function. In addition, [8, Section 3.2] proposes an efficient countermeasure (following a similar idea as in [7]), namely novel (de)compression for which Müller's algorithm turns out to be completely deterministic. Nonetheless, the invented technique is not universal: It is relevant if and only if the order of the group $E(\mathbb{F}_q)$ is even. Thus, a lot of useful prime-order curves (including NIST P-224 [4, Section 3.2.1.2] and MNT 2-cycles [5]) remain uncovered.

Recall that Müller's algorithm of finding $\sqrt{f(x)}$ needs an additional value $u \in \mathbb{F}_q^*$ such that $g(x,u) := u^2 - f(x)$ is a quadratic non-residue in $\mathbb{F}_q$. Searching for such u makes the execution variable-time and so long-time for a "poisoned" value of x. In order to treat this trouble, it is reasonable to iterate u prior to sending x. Fortunately, the non-constant-time behavior of a compressor does not pose any threat. In other words, it is impossible to prepare a (D)DoS attack as earlier, since the compression stage (unlike the decompression one) is independent of any data received from a public channel. In particular, a generator for the next values of u must not be cryptographically strong (as one of the solutions from [8, Table 1]). Thereby, there is no additional risk of relying mistakenly on a predictable generator. In fact, one can simply increment $u := u + 1$, starting with a certain fixed value $u_0 \in \mathbb{F}_q^*$, as it is frequently done for hashing to elliptic curves without secret inputs [3, Section 3.2] or for generating a transparent point basis [2, Section 5.1], [6] with a view towards multi-scalar multiplication.

Hereafter, $u = u_0 + i$ for $i \in \mathbb{N}_{<2^m} := [0, 2^m) \cap \mathbb{N}$, where $m \in \mathbb{N}$ is an auxiliary parameter. According to [11, Theorem 3.5], the probability of being a (non-)square for $g(x,u)$ amounts to $\approx 1/2$ for the random u and for any x such that $f(x) \neq 0$. Hence, the new compression fails with probability $\approx 1/2^{2^m}$ for the general x. At the same time, it is necessary to transmit/store not only this coordinate but also the index i, not to mention one bit for the sign $\pm$ of the coordinate y. To sum up, $L := \ell + m + 1$ bits are required for compressing a point from $E(\mathbb{F}_q) \setminus E[2]$. We thus get a kind of trade-off. It is satisfactory in the sense that m is relatively negligible compared to ℓ of cryptographic size, i.e., $\ell \approx L$. For instance, already for $m = 7$, we achieve the standard 128-bit security level, that is, the probability of solving the discrete logarithm problem on E is not less. On the other hand, the most popular types of computer networks/memory operate with bytes rather than bits. So, the case $m + 1 = 8$ ideally fits them whenever $8 \mid \ell$, which is usually true in practice (e.g., for NIST P-224). In this respect, $\ell + 1$ bits do not constitute a more optimal representation than L bits at least if we are not talking about point packets.

The compression and decompression under consideration are formalized in Algorithms 1, 2, respectively. As is customary, $\mathrm{sign} : \mathbb{F}_q^* \to \mathbb{F}_2$ is an arbitrary cheap function such that $\mathrm{sign}(y) \neq \mathrm{sign}(-y)$. Besides, $\left(\frac{\cdot}{q}\right)$ stands for the Legendre symbol in $\mathbb{F}_q$ and $M(x,u)$ does for Müller's algorithm of extracting the square root of $f(x)$ with the help of u such that $\left(\frac{g(x,u)}{q}\right) = -1$. The bit complexity of determining the Legendre symbol is known to be only (sub-)quadratic in ℓ due to the *(binary) Euclidean algorithm* [13, Section 12.3]. Consequently, the compressor succeeds in finding promptly the desired u with overwhelming probability (unless the parameter $m \approx 0$). To be more precise, the given entity computes on average two $\left(\frac{\cdot}{q}\right)$ as the total overhead, while the decompressor calls one $M(\cdot,\cdot)$ apart from exactly two $\left(\frac{\cdot}{q}\right)$. Finally, the maximum four 2-torsion $\mathbb{F}_q$-points on E can be separately processed by adding to L one more bit[1]. We do not elaborate on this, since small-order points do not occur in ECC protocols by trivial security reasons.

Algorithm 1: New point compression

Data: A point $(x,y) \in E(\mathbb{F}_q) \setminus E[2]$;
Result: The triple $\mathrm{com}(x,y) \in \mathbb{F}_q \times \mathbb{N}_{<2^m} \times \mathbb{F}_2$;
begin
  $f := y^2$;
  $u := u_0$;
  for $i := 0$ **to** $2^m - 1$ **do**
    $g := u^2 - f$;
    if $\left(\frac{g}{q}\right) \in \{-1, 0\}$ **then**
      $\beta := \mathrm{sign}(y)$;
      return (x, i, β).
    end
    $u := u + 1$;
  end
  return fail.
end

Acknowledgements. This research is a result of the strategic project "Advances in post-quantum cryptography applied to the development of a coupon system" (C039/24), resulting from an agreement between the Spanish National Cybersecurity Institute (INCIBE) and University of Lleida. This initiative is carried out in the scope of the funds from the Recovery, Transformation and Resilience Plan, funded by the European Union (Next Generation). The paper is also part of the R&D+i project PID2021-124613OB-I00 funded by MICIU/AEI/10.13039/501100011033 and FEDER, EU.

[1] In reality, L bits are sufficient, because $L > \ell$ or, equivalently, $2^L \gg \#E(\mathbb{F}_q)$ and thereby the points of $E(\mathbb{F}_q)[2]$ can be easily codified by any quartet (duet or one) of the unoccupied L-bit strings.

Algorithm 2: New point decompression

Data: A triple $(x, i, \beta) \in \mathbb{F}_q \times \mathbb{N}_{<2^m} \times \mathbb{F}_2$;
Result: The point $(x, y) \in E(\mathbb{F}_q) \setminus E[2]$ such that $\mathrm{com}(x, y) = (x, i, \beta)$;
begin
  $f := x^3 + ax + b$;
  $u := u_0 + i$;
  $g := u^2 - f$;
  if $\left(\frac{f}{q}\right) \in \{-1, 0\}$ **or** $\left(\frac{g}{q}\right) = 1$ **then**
    return fail.
  end
  if $g = 0$ **then**
    $y := u$;
  else
    $y := M(x, u)$;
  end
  if $\mathrm{sign}(y) \neq \beta$ **then**
    $y := -y$;
  end
  return (x, y).
end

References

1. Aranha, D.F., El Housni, Y., Guillevic, A.: A survey of elliptic curves for proof systems. Designs, Codes and Cryptography **91**(11), 3333–3378 (2023), https://doi.org/10.1007/s10623-022-01135-y
2. Baylina, J., Bellés, M.: 4-bit window Pedersen hash on the Baby Jubjub elliptic curve (2019), https://iden3-docs.readthedocs.io/en/latest/_downloads/4b929e0f96aef77b75bb5cfc0f832151/Pedersen-Hash.pdf
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**(4), 297–319 (2004), https://doi.org/10.1007/s00145-004-0314-9
4. Chen, L., Moody, D., Regenscheid, A., Robinson, A., Randall, K.: Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters (NIST Special Publication 800-186) (2023), https://csrc.nist.gov/publications/detail/sp/800-186/final
5. Guillevic, A.: Pairing-friendly curves (2021), https://members.loria.fr/AGuillevic/pairing-friendly-curves
6. Koshelev, D.: Generation of two "independent" points on an elliptic curve of j-invariant $\neq 0, 1728$ (2023), https://eprint.iacr.org/2023/785
7. Koshelev, D.: Hashing to elliptic curves through Cipolla–Lehmer–Müller's square root algorithm. Journal of Cryptology **37**(2), article 11 (2024), https://doi.org/10.1007/s00145-024-09490-w
8. Koshelev, D.: Point (de)compression for elliptic curves over highly 2-adic finite fields. Advances in Mathematics of Communications **19**(5), 1539–1559 (2025), https://doi.org/10.3934/amc.2025008
9. Lambert, R.J.: Method to calculate square roots for elliptic curve cryptography (2013), https://patents.google.com/patent/US9148282B2/en, United States patent No. 9148282B2

10. Liang, C.C.: Non-constant time hash to point attack vector (2020), https://github.com/thehubbleproject/hubble-contracts/issues/171
11. Müller, S.: On the computation of square roots in finite fields. Designs, Codes and Cryptography **31**(3), 301–312 (2004), https://doi.org/10.1023/B:DESI.0000015890.44831.e2
12. Shanks, D.: Five number-theoretic algorithms. In: Thomas, R.S.D., Williams, H.C. (eds.) Proceedings of the Second Manitoba Conference on Numerical Mathematics. Congressus Numerantium, vol. 7, pp. 51–70. Utilitas Mathematica Publishing Inc., Winnipeg (1973)
13. Shoup, V.: A computational introduction to number theory and algebra. Cambridge University Press, Cambridge, 2 edn. (2008), https://doi.org/10.1017/CBO9780511814549