Solidity Review of secp256r1

Arash Afshar Yehuda Lindell Samuel Ranellucci

Feb 6, 2024

Summary

The paper by Renaud Dubois offers a gas-efficient way of computing operations on sec256r1 curve which can be used to speed up ECDSA signature verification in smart contracts. These optimizations have been implemented in the solidity code that can be accessed here. Our goal was to provide a cryptographic review of the paper and the implementation.
To perform this review, in addition to the above two documents, we referenced the NIST standard definitions of the curves and have annotated the code with details that can help better understand the operations performed. We have also worked with the Base team to unit test and fuzz test the important parts of the code.

In short, we found some problems with code and some typos in the paper which have been fixed already. We also emphasize that although the main flow of the code through ecdsa_verify function is good after the fixes, special care should be taken when using the rest of the functions for the following reasons:

- Some of the functions in the code have non-standard behavior (e.g., returning zero instead of returning an error). As a result, special care should be taken when using those functions in “other” smart contracts
- The code uses the point (0,0) as the point at infinity which could result in bugs if one is not careful
- Some of the functions do not check all edge cases (e.g., that something is not zero) since the checks have already been carried out in the signature verification flow. This makes the overall code valid, but means that sub-functions cannot be pulled out and used independently.

We note that the scope of the review was limited to the functions defined in the provided solidity file. We assumed that the functions that were not defined in the file (such as `addmod, mulmod, etc`) are working correctly.

Artifacts and resources

Following is the list of relevant artifacts used or produced during this review:
Produced artifacts:

- Unit test and fuzz test plan: secp256r1-ecdsa-verify-solidity-review-testing-plan.pdf
-

- List of findings: Findings section in this document

Other artifacts:

- Original paper: Speeding up elliptic computations for Ethereum Account Abstraction
- NIST digital signature standard (also FIPS-186-4)
- NIST Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters
- Projective Coordinates Optimization

Findings

ID	Severity	Location	Description
1	Low	Paper	Missing constants in ecdblneg as compared to this implementation.
2	Low	Paper	In ecnegaddN, R2 is never defined.
3	Low	FC_nModInv	On input 0, the code returns 0 which is wrong since 0 does not have a multiplicative inverse
4	Low	FC_pModInv	On input 0, the code returns 0 which is wrong since 0 does not have a multiplicative inverse
5	Low	ecZZ_mulmuladd_S_asm	If the result is point at infinity, the function returns X=0 (for example when u and v are both zero) There exists a point on the curve that meets the following two conditions. • It is not the point at infinity. • The x-coordinate of this point is zero.

8	High	ecAff_isOnCurve	Potential bug in the condition. The code checks the following ● x == 0 && y == 0 ● x == p || y == p ○ This should be x==p && y == p
9	Medium	ecZZ_mulmuladd_S_asm	Infinite loop on (-gx, -gy, 1, 1)
10	Info	ecZZ_AddN	It is documented that this function does not work when its two inputs are the same and therefore point doubling should be used in its place. Although this was documented, it was not very obvious and we wasted some time troubleshooting why some tests were failing. So leaving this comment here to bring visibility into it.