Optimized ZK verification of ECDSA signature with partially outsourced computations

Denys Riabtsev, Mykhailo Khotian

Distributed Lab

1 Introduction

The Elliptic Curve Digital Signature Algorithm (ECDSA) is one of the most widely adopted signature schemes, particularly in blockchain systems. However, when it comes to zero-knowledge proof systems, verifying ECDSA signatures presents significant computational challenges.

Current implementations of ECDSA signature verification in zero-knowledge circuits face several limitations. The primary challenge lies in the computational complexity of the verification process, which requires numerous elliptic curve operations and modular arithmetic calculations. These operations translate into a large number of constraints when expressed in arithmetic circuits, making the proof generation process computationally intensive and time-consuming.

We propose to ousource part of computations to a verification party, while still preserving user's privacy.

2 Problem overview

Let PK be a public key, (r, s) - signature, m - message, G - base point (generator of EC group), R - point, where $R.x == r$. Then to verify an ECDSA signature, the following equation must hold:

$$[m \cdot s^{-1}]G + [r \cdot s^{-1}]PK = R \tag{1}$$

This equation consist of two scalar multiplications:

1. $[m \cdot s^{-1}]G$ - scalar multiplication of base point;
2. $[r \cdot s^{-1}]PK$ - scalar multiplication of public key.

By using precompute tables and windowed double-and-add algorithm we can significantly optimize scalar multiplication of *base point*. As the base point is known, the precomputation table can be generated during circuit compilation time, significantly reducing the number of constraints in the circuit.

The problem rises with scalar multiplication of *public key*. As it is not known beforehand, table should be calcutated in-circuit, which adds additional computational complexity.

Let $\mathbb{A}$ be number of constraints for point addition, $\mathbb{D}$ - for point double; F - field size, W - window size for windowed double and add. Then the complexity of *base point* scalar multiplication is

$$(\frac{F}{W} - 1) \cdot \mathbb{A};$$

for *public key* scalar multiplication is

$$(F-W)\cdot\mathbb{D}+(\frac{F-W}{W})\cdot\mathbb{A}+(2^W-1)\cdot\mathbb{A}+\mathbb{D}$$

,

where $(2^W-1)\cdot\mathbb{A}+\mathbb{D}$ is required to generate a precompute table.

The complexity of scalar multiplication of public key is significantly higher than base point multiplication.

With window size $W=4$ and field size $F=256$, the number of constraints for base point multiplication is $63\cdot\mathbb{A}$, while for public key multiplication it is $252\cdot\mathbb{D}+63\cdot\mathbb{A}+15\cdot\mathbb{A}+\mathbb{D}$. Additionally, the window size for base point can be increased, resulting in lowering the number of constraints, but increasing off-circuit precomputation load.

Table 1: Number of constraints for scalar multiplication operations in secp256r1

Operation	**Constraints**
Base point multiplication	122,383
Public key multiplication	1,428,821

This shows that public key multiplication requires significantly more constraints, especially considering that point doubling ($\mathbb{D}$) is typically more expensive than point addition ($\mathbb{A}$).

3 Proposed optimization

Let $\mathcal{B}$ be random value selected uniformly from range $[0, 2^c)$, where c is security parameter; $\mathbb{B}=[\mathcal{B}]PK$

By multiplying both sides of the equation by $\mathcal{B}$, we obtain:

$$[m\cdot s^{-1}\cdot\mathcal{B}]G+[r\cdot s^{-1}\cdot\mathcal{B}]PK=[\mathcal{B}]R \tag{2}$$

$$[m\cdot s^{-1}\cdot\mathcal{B}]G+[r\cdot s^{-1}]\mathbb{B}=[\mathcal{B}]R \tag{3}$$

There are