BLS12-381 vs BN254

From Circom Developer Perspective

Distributed Lab

February 2025

1 Introduction

In March 2025, Ethereum developers will finalize the date for the Pectra hard fork, which will introduce EIP-2537 [1]. This update will bring native support for BLS12-381 curve operations, enhancing the efficiency and accessibility of zkSNARKs and secure multi-party computations.

This document is written to give a quick overview of BLS12-381 vs BN254 curves in Circom so that developers can understand if the migration is justified. We examined the drawbacks and benefits of both curves, comparing their security, ZK witness generation and proving times, and on-chain ZK verification gas costs.

1.1 Life before EIP-2537

Currently, Ethereum uses BN254 precompiles (EIP-196 [2] and EIP-197 [3]) for all ZK cryptographic verification processes.

To solve the discrete logarithm problem (DLP) on BN254, one would theoretically need to perform $O(2^{254})$ operations. However, using Pollard's rho algorithm, which reduces the effort to $O(\sqrt{N})$, the required operations drop to $O(2^{127})$.

In 2015, the development of the exTNFS (extended Tower Number Field Sieve) algorithm further reduced the computational complexity to approximately $O(2^{100})$ operations, effectively lowering the security level to 100 bits.

1.2 Life after EIP-2537

BLS12-381 has an order of approximately 2^{255}, which would theoretically require $O(2^{255})$ operations to solve. However, by using Pollard's rho algorithm, this can be reduced to $O(\sqrt{N})$, resulting in $O(2^{128})$ operations.

The key factor is that BLS12-381 was specifically designed to prevent the use of exTNFS to break it. This is achieved by using a larger prime number compared to BN254 (2^{381} vs 2^{254}), providing the necessary security at the cost of some efficiency.

As a result, the security level of BLS12-381 is slightly below 128 bits.

[1] `https://eips.ethereum.org/EIPS/eip-2537`
[2] `https://eips.ethereum.org/EIPS/eip-196`
[3] `https://eips.ethereum.org/EIPS/eip-197`

1.3 Why is it crucial?

NIST currently accepts the usage of cryptographic systems with 128-bit security for today's applications. However, as we can see, BN254 falls short of meeting these requirements. Therefore, it is advisable to migrate to a more secure curve.

With the introduction of EIP-2537, BLS12-381 emerges as a strong candidate. Let us explore the potential uses of this new curve.

2 Using BLS12-381

BLS12-381 was proposed by Sean Bowe at Zcash in 2017 and began to be actively used with zkSNARKs. In 2022, Circom also added support for this new curve.

2.1 Migration from BN254

As mentioned, BLS12-381 has an order of approximately 2^{255}, which means all operations will be performed using a new modulo. This gives us the advantage of processing larger numbers, using 255 bits instead of 254.

However, ensure your circuits do not depend on BN254-specific constants; for instance, update the Sign [4] and Poseidon Hash [5] circuits.

During circuit compilation, you need to specify the parameter for the new curve:

```
circom <your other params> --prime bls12381
```

Note that you must use new Ptau files for BLS12-381 as well.

2.2 New setup

To bootstrap BLS12-381, we have chosen a simple multiplication circuit as an example. The advantage of using BLS12-381 over BN254 is that it allows us to operate on larger numbers, meaning that the result will overflow later compared to BN254.

```
pragma circom 2.0.0;

template Multiplier(){
   signal input in1;
   signal input in2;
   signal output out;

   out <== in1 * in2;
}

component main = Multiplier();
```

[4] https://github.com/iden3/circomlib/blob/252f8130105a66c8ae8b4a23c7f5662e17458f3f/circuits/sign.circom

[5] https://github.com/iden3/circomlib/blob/252f8130105a66c8ae8b4a23c7f5662e17458f3f/circuits/poseidon.circom

We then proceed through the entire circuit usage process, from compilation to verification:

1. Compile a circuit and generate the necessary files for proving and verification:

   ```
   circom multiplier.circom --r1cs --wasm --sym --c --prime bls12381
   ```

2. Compute the witness for the circuit based on given inputs:

   ```
   node multiplier_js/generate_witness.js multiplier_js/multiplier.wasm
   ↪ input.json witness.wtns
   ```

3. Prepare the Phase 2 Powers of Tau file for Groth16 setup:

   ```
   snarkjs powersoftau prepare phase2 pot_bls12-381.ptau pot.ptau
   ```

4. Perform the trusted setup for the circuit using Groth16:

   ```
   snarkjs groth16 setup multiplier.r1cs pot.ptau circuit.zkey
   ```

5. Extract the verification key from the trusted setup:

   ```
   snarkjs zkey export verificationkey circuit.zkey verification_key.json
   ```

6. Generate a proof using the computed witness:

   ```
   snarkjs groth16 prove circuit.zkey witness.wtns proof.json public.json
   ```

7. Verify the proof using the verification key and public inputs:

   ```
   snarkjs groth16 verify verification_key.json public.json proof.json
   ```

3 Results

3.1 Off-chain times

We have measured the average execution time for each step on the Multiplier circuit. Note that it contains only one constraint and one public signal.

Tool	Operation	BN254 (ms)	BLS12-381 (ms)	Difference
circom	Compilation	4.95	5.01	+1.25%
	Generation of a Witness	43.90	44.57	+1.54%
snarkjs	Groth16 Setup	231.45	347.71	+50.23%
	Export Verification Key	227.61	352.92	+55.05%
	Prove	257.50	350.44	+36.09%
	Verify	233.71	332.03	+42.07%

Table 1: Comparison of Average Execution Times for BN254 and BLS12-381 On Multiplcation

Additionally, we tested the SHA-160 Hash circuit [6], which consists of 52,176 constraints and 160 public signals.

[6] https://github.com/dl-solarity/circom-lib/blob/73ef74c4c68f1c37fb2f6d3277398004fec78c97/circuits/main/hasher/hashBits160.circom

Tool	Operation	BN254 (ms)	BLS12-381 (ms)	Difference
circom	Compilation	1,655.96	1,650.45	-0.33%
	Generation of a Witness	998.35	996.80	-0.16%
snarkjs	Groth16 Setup	1,512.87	1,843.15	+21.83%
	Export Verification Key	241.56	330.35	+36.76%
	Prove	762.75	908.87	+19.16%
	Verify	232.15	315.26	+35.80%

Table 2: Comparison of Average Execution Times for BN254 and BLS12-381 On SHA-160

We observe minimal differences in Circom's performance when using different curves. This is because Circom operates based on the curve's order. Since the orders of BLS12-381 and BN254 are similar, the elapsed times remain nearly identical.

However, zkSNARK operations take significantly longer with BLS12-381 than with BN254. This is because zkSNARKs operate over the entire field, and BLS12-381 uses a larger prime (50% larger than BN254).

Despite the increased computation time, BLS12-381 offers stronger security guarantees, making it a preferable choice for long-term applications where higher security is essential.

3.2 On-chain gas

After the Pectra hard fork, Ethereum dApps can utilize proofs on the BLS12-381 curve. To enable this, we generate a Groth16 verifier using snarkjs. Let us compare the computational cost of verifying proofs for BN254 and BLS12-381.

We have implemented our own Groth16 verifier to support the new curve. The primary difference lies in the representation of BLS12-381 points, which require 381 bits. Consequently, each number must be split into two uint256 values, effectively doubling the number of instructions needed for point operations.

The cost of precompiled operations also changes. For addition and multiplication, it increases by approximately a factor of two. However, the pairing operation is actually cheaper! This reduction in cost is due to better-chosen base curve parameters.

Operation	BN254 (gas)	BLS12-381 (gas)	Difference
Addition	150	375	+150%
Multiplication	6,000	12,000	+100%
Pairing Check (4 pairs)	181,000	168,100	-7%

Table 3: Comparison of Gas Usage for BN254 and BLS12-381

Although we may anticipate that proof verification costs will double, the actual gas cost remains largely unchanged in practice. This is primarily due to the pairing operation, which dominates the overall gas consumption and helps offset the increase in point operation costs.

We can express the gas cost of verification for both BN254 and BLS12-381 as functions of the number of public inputs n, as follows (neglecting minor contract operations):

$$\text{Gas Usage}_{\text{BN254}} = 181,000 + (6,150 \times n)$$

$$\text{Gas Usage}_{\text{BLS12-381}} = 168,100 + (12,375 \times n)$$

n / Public Inputs Count	BN254 (gas)	BLS12-381 (gas)	Difference
1	187,150	180,475	-3.57%
5	211,750	229,975	+8.61%
10	242,500	291,850	+20.35%
50	488,500	786,850	+61.07%

Table 4: Comparison of Gas Usage for BN254 and BLS12-381

4 Conclusion

Despite the increased execution time and gas cost, BLS12-381 is an attractive option to enhance security without significantly affecting the development process.

However, the usage of a new curve will be crucial for long-term scalability and the strengthening of cryptographic guarantees of proofs.