Technical Background:
Gas Cost Analysis for ZK Proof Verification on Ethereum

Vocdoni

Draft v0.1 - February 2026

Abstract

This document provides a compact technical reference for understanding gas costs when verifying zero-knowledge proofs on Ethereum using Groth16 or FFLONK. The costs of precompiles for both BN254 and BLS12-381 are covered. However, verification and deployment costs are analyzed only for SNARK circuits over BN254. This serves as a reference for the implementation decisions of the Vocdoni DKG protocol.

The key takeaways are: (1) Groth16 deployment costs at 420,000 gas are well below FFLONK's almost 4 million gas; (2) Groth16 is more expensive per public input than FFLONK, with costs of 6,650 versus 400 gas, excluding data loading; (3) Groth16 can handle up to 700 public inputs, while FFLONK is limited to ∼38.

Contents

1 Overview

On-chain verification of zero-knowledge proofs requires careful consideration of gas costs, which can impact feasibility and scalability. This document analyzes the two most practical proof systems for Ethereum:

- **Groth16** [1]: The proving system with the smallest proof size, but requires a trusted setup ceremony per circuit.
- **FFLONK** [2]: A variant of Plonk with optimized verification, requiring only a universal trusted setup (powers of tau).

The following table 1 summarizes the results for BN254-based verifiers. Note that per-input gas costs consist of a fixed amount plus data loading costs: either calldata or storage access.

Table 1: Groth16 vs FFLONK Comparison (BN254 verifier)

Property	**Groth16**	**FFLONK**
Fixed gas cost	207,000	200,000
Per-input gas cost	6,650	400
+ calldata (sparse)		∼140
+ calldata (dense)		∼512
+ storage access (cold)		2,100
Proof size	256 bytes	768 bytes
Trusted setup	Per-circuit	Universal (powers of tau)
Contract size (21 inputs)	1,816 bytes	19,471 bytes
Contact deployment cost	∼420,000 gas	∼3,950,000 gas
Contract size scaling	+33 bytes/input	+289 bytes/input
Max. inputs	∼700	∼38
Prover time	Faster	∼3× slower

Although we do cover the costs of BLS12-381 precompiles in section 2.1, the costs analysis only covers the use BN254 (alt_bn128) curve and js-generated Solidity verifiers. Also note that Layer 2 chains may have different slightly gas economics and execution costs than Ethereum mainnet. This is not analyzed in this document.

2 EVM Gas Costs

2.1 Elliptic Curve Operations

Zero-knowledge proof verification on Ethereum relies on elliptic curve operations. The precompiles for BN254 became available in October 2017 (Byzantium; EIP-196 [7] and EIP-197 [8]) with gas costs significantly reduced in December 2019 (Istanbul; EIP-1108 [9]). Regarding the BLS12-381 curve, precompiles were introduced in May 2025 (Pectra; EIP-2537 [11]). Major Layer 2 networks like Arbitrum also began enabling these new precompiles (e.g., in the ArbOS 50 release) to maintain EVM compatibility.

There are some limitations. In BN254, group addition and scalar multiplication are available only in group $\mathbb{G}_1$, but not in $\mathbb{G}_2$. There is also no way to actually compute a pairing for either

curve. It is only possible to verify pairings, as the pairing check precompiles verify an equation of the form:

$$\prod_{i=1}^{k} e(A_i, B_i) = 1_{\mathbb{G}_T}$$

The costs of these precompiles are summarized in tables 2 and 3.

Table 2: EVM Precompile Gas Costs for BN254

Operation	**Group**	**Address**	**Gas**
Point addition ECADD	$\mathbb{G}_1$	0x06	150
Scalar multiplication ECMUL	$\mathbb{G}_1$	0x07	6,000
Pairing check (k pairs)	$\mathbb{G}_1 \times \mathbb{G}_2$	0x08	$45,000 + 34,000\,k$

Table 3: EVM Precompile Gas Costs for BLS12-381

Operation	**Group**	**Address**	**Gas**
Point addition	$\mathbb{G}_1$	0x0b	375
Scalar multiplication (MSM, k=1)	$\mathbb{G}_1$	0x0c	12,000
Point addition	$\mathbb{G}_2$	0x0d	600
Scalar multiplication (MSM, k=1)	$\mathbb{G}_2$	0x0e	22,500
Pairing check (k pairs)	$\mathbb{G}_1 \times \mathbb{G}_2$	0x0f	$37,700 + 32,600\,k$

Note that BLS12-381 does not have a dedicated single-scalar multiplication precompile. Instead there is a multi-scalar multiplication precompile (MSM) which can emulate scalar multiplication by calling it with a single term ($k = 1$). The general formula for the MSM costs is of the form:

$$\lfloor k \cdot \mathsf{multiplication\_cost} \cdot \mathsf{discount\_factor}(k) \rfloor,$$

The discount factor decreases (i.e. the discount increases) with k as shown in table 4. The maximum discount of almost 50% is reached when $k = 128$.

Table 4: BLS12-381 MSM average gas cost per term

k	**G1 factor**	**gas/term**	**G2 factor**	**gas/term**
1	100%	12,000	100%	22,500
2	94.9%	11,388	100%	22,500
3	84.8%	10,176	92.3%	20,767
4	79.7%	9,564	88.4%	19,890
16	67.7%	8,124	71.7%	16,132
64	57.6%	6,912	58.2%	13,095
128	51.9%	6,228	52.4%	11,790

2.2 Data Loading

Calldata Costs Transaction calldata costs are defined by EIP-2028 [10] as 16 gas for non-zero bytes and 4 gas for zero bytes. A public uint256 input that is a small integer will have almost all bytes at zero, its cost will therefore be only a little more than $32 \times 4 = 128$ gas, while a generic BN254 field element of 32 bytes will cost $\approx 32 \times 16 = 512$ gas.

Storage-Sourced Public Inputs Some public inputs may correspond to values already stored on-chain (e.g., commitments, public keys). Reading these these values from storage rather than receiving them as calldata is more expensive because a cold SLOAD costs 2,100 gas (cf. EIP-2929 [12]).

Data loading costs are are summarized in table 5.

Table 5: Data Access Cost per Public Input (32 bytes)

Source	Gas Cost	Notes
"Dense" calldata (field element)	≈ 512	EIP-2028 (16 gas/byte)
"Sparse" calldata (small integer)	≈ 140	If only one non-zero byte
Storage (cold)	2,100	EIP-2929, (first access)
Storage (warm)	100	Subsequent access

3 Groth16 Verification Costs

3.1 Verification Algorithm

A Groth16 proof consists of three group elements: $\pi = (A, B, C)$ where $A, C \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$. Given public inputs $\{x_i\}_{i=1}^{t}$ and verification key elements $\{IC_i\}_{i=0}^{l}$, the verifier:

1. Computes the public input accumulator, which is a *multi-scalar multiplication (MSM)* requiring ℓ scalar multiplications and ℓ point additions:

$$\text{acc} = IC_0 + \sum_{i=1}^{\ell} x_i \cdot IC_i$$

2. Verifies the following pairing equation:

$$e(A, B) \stackrel{?}{=} e(\alpha, \beta) \cdot e(\text{acc}, \gamma) \cdot e(C, \delta)$$

The pairing equation is equivalent to a single pairing check with $k = 4$ pairs:

$$e(-A, B) \cdot e(\alpha, \beta) \cdot e(\text{acc}, \gamma) \cdot e(C, \delta) \stackrel{?}{=} 1_{\mathbb{G}_T}$$

3.2 Groth16 Gas Cost

Based on analysis by Orbiter Research [4], benchmarks by Icer Liang [5] and empirical verification, the verification gas costs are as shown in table 6. As a result, we obtain the following formula for the execution cost (excluding data access):

$$\text{Groth16 Gas (execution only)} \approx 207,000 + 6,650 \times \ell \tag{1}$$

where ℓ is the number of public inputs. Calldata or SLOAD costs will have to be added to this.

3.3 Proof Size

The total proof size is 256 bytes, independently of circuit size. It consists of the proof elements A (64 bytes), B (128 bytes) and C (64 bytes). The size of public inputs is $32\,\ell$ bytes.

Table 6: Groth16 Verification Gas Cost Components (BN254)

Component	Gas Cost	Notes
Fixed costs:		
Pairing check ($k = 4$)	181,000	Dominates fixed cost
Calldata (proof)	∼3,840	2 $\mathbb{G}_1$ + 1 $\mathbb{G}_2$ point
Contract overhead	∼22,000	Function dispatch, memory, etc.
Total fixed costs	**∼207,000**	
Per public input, excluding Calldata or SLOAD costs:		
Field check	61	Verify $x_i < p$
Scalar multiplication	6,000	ecMul precompile
Point addition	150	ecAdd precompile
Memory operations	∼437	Load, store, hashing
Total per input	**∼6,650**	

4 FFLONK Verification Costs

FFLONK (Fast-Fourier PLONK) is an optimized variant of PLONK that reduces the number of pairings to 2 via batching, while using a different polynomial commitment structure.

4.1 Gas Cost Breakdown

We again rely on the Orbiter Research report [4] and Icer Liang's benchamarks [5]. Based on the data in table 7, the execution costs, excluding data access, are:

$$\text{FFLONK Gas (execution only)} \approx 200,000 + 400 \times \ell \tag{2}$$

The number of public inputs is ℓ, and the cost of Calldata or SLOAD costs will have to be added to this.

Table 7: FFLONK Verification Gas Cost Components (BN254)

Component	Gas Cost	Notes
Fixed costs:		
Pairing check ($k = 2$)	113,000	Reduced from Groth16
Polynomial evaluations	∼50,000	Field arithmetic
Challenge computation	∼15,000	Keccak hashing
Calldata (proof)	∼12,000	24 field elements
Contract overhead	∼10,000	
Total fixed costs	**∼200,000**	
Per public input, excluding Calldata or SLOAD costs:		
Lagrange evaluation	∼100	Field multiplications
PI polynomial contribution	∼300	Accumulation
Total per input	**∼400**	

4.2 Proof Size

The proof requires a total of 24 field elements (polynomial commitments and evaluations), requiring 768 bytes. This size is constant. Public inputs add $32\,\ell$ bytes to the total.

4.3 Why FFLONK Has Lower Per-Input Cost

The large difference in per-input cost (∼6,650 vs ∼400 gas) stems from the verification structure. While Groth16 requires computing a MSM with ℓ elements, FFLONK computes public input contributions via:

$$\text{PI}(\xi) = \sum_{i=1}^{\ell} x_i \cdot L_i(\xi)$$

where x_i are the public inputs and $L_i(\xi)$ are Lagrange basis evaluations at the challenge point ξ. This requires only field multiplications and additions which are much cheaper (∼3-8 gas each via mulmod/addmod).

5 Contract Size Constraints

5.1 EIP-170 Limit

Ethereum has a a maximum contract bytecode size of 24,576 bytes, or 24 KB (EIP-170 [6]). Contracts exceeding this limit cannot be deployed. This constraint significantly impacts FFLONK verifiers due to their complex verification logic.

The EIP-170 limit applies to most L2 chains. EVM-equivalent rollups such as Optimism, Base and Arbitrum follow the 24 KB limit. Arbitrum documentation explicitly confirms EIP-170 compliance. Note that zkSync Era does not follow EIP-170 and has a theoretical limit of ∼2 MB, though practical limits are lower due to pubdata constraints (∼100-120 KB for rollups). For maximum portability, targeting the 24 KB limit is recommended.

5.2 Empirical Testing Results

We generated snarkjs [3] verifiers for various public input counts and compiled them with different Solidity optimizer settings. The result are shown in tables 8 and 9. Percentages are relative to the 24,576 byte limit. Red indicates that the bytecode exceeds the EIP-170 limit. The empirical data reveals an asymmetry between the two proof systems.

Table 8: FFLONK Verifier Bytecode Size (deployed)

Inputs	No opt	runs=1	runs=200	runs=999999
1	23,834 (97%)	13,646 (56%)	13,682 (56%)	22,446 (91%)
2	24,349 (99%)	13,991 (57%)	14,027 (57%)	23,076 (94%)
4	25,380 (103%)	14,566 (59%)	14,602 (59%)	24,107 (98%)
8	27,440 (112%)	15,714 (64%)	15,750 (64%)	26,167 (106%)
16	31,576 (128%)	18,026 (73%)	18,062 (73%)	30,303 (123%)
21	34,161 (139%)	19,471 (79%)	19,507 (79%)	32,888 (134%)
32	39,847 (162%)	22,650 (92%)	22,686 (92%)	38,575 (157%)

FFLONK faces a hard constraint on public inputs due to contract size:

Table 9: Groth16 Verifier Bytecode Size (deployed)

Inputs	No opt	runs=1	runs=200	runs=999999
1	1,321 (5%)	1,064 (4%)	1,170 (5%)	1,170 (5%)
4	1,414 (6%)	1,162 (5%)	1,268 (5%)	1,268 (5%)
8	1,538 (6%)	1,281 (5%)	1,387 (6%)	1,387 (6%)
21	1,967 (8%)	1,710 (7%)	1,816 (7%)	1,816 (7%)
32	2,329 (9%)	2,073 (8%)	2,179 (9%)	2,179 (9%)

- With gas-optimized compilation (`runs=999999`): maximum **4 public inputs**
- With size-optimized compilation (`runs=1`): maximum ∼**38 public inputs**

Groth16 faces only a soft constraint (gas costs):

- Even with 100 public inputs, bytecode is only ∼4.4 KB (18% of limit)
- Theoretical maximum before hitting 24 KB: ∼700+ public inputs (plenty for almost all practical applications)

This makes Groth16 the only viable choice for circuits requiring more than ∼38 public inputs, regardless of FFLONK's superior per-input verification efficiency. When FFLONK is for circuits with more than 4 but less than 38 public inputs, use `runs=1` and accept a ∼16k gas penalty.

Table 10: Maximum Public Inputs Before Exceeding 24KB Limit

System	Optimizer Setting	Max Inputs
FFLONK	runs=999999 (gas optimized)	4
FFLONK	runs=1 (size optimized)	∼38
Groth16	Any	∼700+

6 Deployment Costs

The one-time cost of contract deployment is given by the formula:

$$\text{Deployment gas} = 21,000 + 32,000 + 200 \times \text{bytecode\_size}$$

21,000 is the base tx cost, 32,000 is the contract creation cost and 200 is the per-byte deployment cost. Because of its much higher verifier bytecode size, FFLONK has a proportionally higher deployment cost than Groth16, as shown in table 11.

Table 11: Verifier Deployment Costs (21 public inputs, runs=1)

System	Bytecode	Deploy Gas
Groth16	1,816 B	∼420,000
FFLONK	19,471 B	∼3,950,000

7 ABI sizing summary for EC precompiles

The base field of the BN254 has bitlength 254, an element of that field can therefore be represented as one 32 byte word. A point in $\mathbb{G}_1$, which is encoded as 2 base field elements, requires 64 bytes. Elements of $\mathbb{G}_2$, whose base field is the much larger extension field, require twice as much storage. As the base field of BLS12-381 has bitlength 381, its curve points require twice as much storage in terms of 32-byte words. However, the scalar fields of both curves are roughly of the same size, so scalars used for scalar multiplications require only 32 bytes for both curves. Recall that, for a given curve, $\mathbb{G}_1$ and $\mathbb{G}_2$ have the same prime order despite different coordinate fields.

Table 12: Bitlengths and Encoding widths

Item	**BN254**	**BLS12-381**
Scalar field bitlength	≈ 254 bits	≈ 255 bits
Scalar encoding	32 B	32 B
Base field bitlength	≈ 254 bits	≈ 381 bits
G1 encoding	64 B	128 B
Quadratic ext. field	≈ 508 bits	≈ 762 bits
G2 encoding	128 B	256 B

Table 13: EC Precompiles: ABI Sizing Summary

Operation	**Addr**	**In**	**Out**	**ABI sizing rationale**
BN Point add $\mathbb{G}_1$	`0x06`	128	64	In: $2\times\mathbb{G}_1$ (2×64). Out: $1\times\mathbb{G}_1$ (64).
BN Scalar mul $\mathbb{G}_1$	`0x07`	96	64	In: $1\times$Scalar and $1\times\mathbb{G}_1$ ($32 + 64 = 96$). Out: $1\times\mathbb{G}_1$ (64).
BN Pairing (k pairs)	`0x08`	$192 \cdot k$	32	In: $1\times\mathbb{G}_1$ and $1\times\mathbb{G}_2$ ($64 + 128 = 192$) per pair. Out: boolean 0/1 returned as a 32-byte word.
BLS Point add $\mathbb{G}_1$	`0x0b`	256	128	In: $2\times\mathbb{G}_1$ (2×128). Out: $1\times\mathbb{G}_1$ (128).
BLS MSM $\mathbb{G}_1$ (k=1)	`0x0c`	160	128	In: $1\times$Scalar and $1\times\mathbb{G}_1$ ($32 + 128 = 160$). Out: $1\times\mathbb{G}_1$ (128).
BLS MSM $\mathbb{G}_1$	`0x0c`	$160 \cdot k$	128	In: k times 160 bytes.
BLS Point add $\mathbb{G}_2$	`0x0d`	512	256	In: $2\times\mathbb{G}_2$ ($2 \times 256 = 512$). Out: $1\times\mathbb{G}_2$ (256).
BLS MSM $\mathbb{G}_2$ (k=1)	`0x0e`	288	256	In: $1\times$Scalar and $1\times\mathbb{G}_2$ ($32 + 256 = 288$). Out: $1\times\mathbb{G}_2$ (256).
BLS MSM $\mathbb{G}_2$	`0x0e`	$288 \cdot k$	256	In: k times 288 bytes.
BLS Pairing (k pairs)	`0x0f`	$384 \cdot k$	32	In: $1\times\mathbb{G}_1$ and $1\times\mathbb{G}_2$ ($128 + 256 = 384$) per pair. Out: boolean 0/1.

References

[1] J. Groth. *On the Size of Pairing-based Non-interactive Arguments*. EUROCRYPT 2016.

[2] A. Gabizon and Z.J. Williamson. *fflonk: a Fast-Fourier inspired verifier efficient version of PlonK*. Cryptology ePrint Archive, 2021/1167.

[3] iden3. *snarkjs: zkSNARK implementation in JavaScript & WASM*. GitHub. https://github.com/iden3/snarkjs

[4] Orbiter Research (Kiwi). *Maximizing Efficiency in Ethereum ZKPs: Comparing Groth16 and FFLONK Gas Costs*. HackMD, 2024. https://hackmd.io/Fa4A8lVKRM2TwVh1dhUb1Q

[5] Icer Liang. *ZKP System Solidity Gas Estimation Report*. GitHub, 2024. https://github.com/wizicer/zkp-solidity-gas

[6] EIP-170. *Contract code size limit*. https://eips.ethereum.org/EIPS/eip-170

[7] EIP-196. *Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128*. https://eips.ethereum.org/EIPS/eip-196

[8] EIP-197. *Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128*. https://eips.ethereum.org/EIPS/eip-197

[9] EIP-1108. *Reduce alt_bn128 precompile gas costs*. https://eips.ethereum.org/EIPS/eip-1108

[10] EIP-2028. *Transaction data gas cost reduction*. https://eips.ethereum.org/EIPS/eip-2028

[11] EIP-2537. *Precompile for BLS12-381 curve operations*. https://eips.ethereum.org/EIPS/eip-2537

[12] EIP-2929. *Gas cost increases for state access opcodes*. https://eips.ethereum.org/EIPS/eip-2929