For proof (A,B,C)

, verification key $(\alpha, \beta, \gamma, \delta, I_n)$

and input data x_i

, the groth16 proof is

$(X, \gamma) + (C, \delta) + (\alpha, \beta) - (A, B) = O$

, where O

is zero point and $X = I_0 + \sum\limits_{i=1}^{n} I_i x_i$

.

For proof batching we can use approach, presented at[Fast verification of multiple BLS signatures] by [@vbuterin] and use set of batcing multipliers r_i

.

$(\sum X_i r_i, \gamma) + (\sum C_i r_i, \delta) + (\sum r_i \alpha, \beta) - \sum (r_i A_i, B_i) = O.\ \ \ \ \ \ \ \ \ \ \ (1)$

Here is $3+N$

pairing operations instead of $4N$

.

If the attacker knows $\{r_i\}$

, the proof may be forged by simple way for any two or more batched proofs:

$C_i := 0$

,

$(A_0, B_0) := (\frac{1}{r_0} \sum X_i r_i, \gamma)$

,

$(A_1, B_1) := ((1+\frac{r_0}{r_1})\alpha, \beta)$

.

If we substitute these expressions into (1), we get proof for any public inputs. The forged proof may be computed by miner, somebody, who knows expected r_i

or directly onchain if the source of r_i

is available for the attacker's contract.

We can determine r_i

as hash of input data. Similar approach is using in zkSTARKS to select branches of Merkle tree for the proof:

$s := H(\{A_i, B_i, C_i, X_i\})$

,

$r_0 := 1,\ r_i = H(s, i)$

.

In such case the batching is not so vulnerable.