

Circuit

From a user perspective, a circuit is like a program, which can be written in a programming language like Go. However, internally a circuit is a constraint system; that is, a list of constraints which have an algebraic form.

For [Groth16](#), a constraint looks like $(\sum_i a_i x_i)(\sum_i b_i y_i) = \sum_i c_i z_i$ ($\sum_i a_i x_i$)($\sum_i b_i y_i$) = $\sum_i c_i z_i$ where a, b, c are constants and x, y, z are variables which depend on the secret inputs known by a prover.

Translating a circuit, written with [gnark API](#) to such a constraint system is called the "arithmetization" of a circuit.

An important point is that every component of a constraint (variables, inputs and constants) live in \mathbb{F}_p , a finite field of characteristic p . To write a circuit which contains a reasonable number of constraints, it is important to work on the field \mathbb{F}_p , so that the field in which the circuits variables live is the same as the field on which the constraint system reasons.

On the other hand, a circuit reasoning on variables which live in \mathbb{F}_r where $r \neq p$

?

\mathbb{F}_p , has a high number of constraints because of the algebraic constraints needed to emulate the arithmetic mod r on a field of characteristic p .

Finally, the number of constraints in a circuit is limited; you cannot write arbitrarily large circuits. For example, using Groth16 on BN254, you cannot exceed ~250M constraints. [Edit this page](#) Last updated on Mar 2, 2023 by aybehrouz
[Previous zk-SNARK](#) [Next Proving schemes and curves](#)