GroLup: Plookup for R1CS

by Alexandre Belling and Azam Soleimanian; Consensys R&D

Given a binary relation R(x, w)

, SNARKs allow proving knowledge of a witness w

such that the relation R

is satisfied with respect to a public input x

. In particular, the verifier needs less time to verify the proof, generated by SNARK, rather than to re-do all the computations.

Groth16 and PLONK are among the most famous and practical general-purpose SNARKs. Still, since addition is free in Groth16, it can be more efficient (for general but fixed computations).

Plookup is a SNARK scheme for a special argument, where the prover should convince the verifier that she knows the polynomial f

such that the values of f

are in the table T

.

If a general SNARK scheme can be combined with a Plookup argument, one can move a huge amount of online computation to the offline phase (preprocessing). As a more concrete example consider the case that the prover needs to prove several witnesses are respecting a given range. Combining general SNARK with Plookup would remove part of the circuit that is used for range check, and delegate the task to the Plookup.

In PlonKup, the authors presented a technique to combine PLONK with Plookup. Yet combining Groth16 with Plookup has remained an open question.

The technique in PlonKup relies on adding the lookup-gates, that can increase the degree of polynomials involved in the PLONK proof. Here we present a different technique that particularly aims for Groth16, but can also be used for Plonk (without increasing the degree of Plonk polynomials). There are many variants of Plookup argument (Caulk, Caulk+, etc), one can combine Groth16 or PLONK with any of these variants, as far as they work on the same curves.

A CP-Link approach

The technique we present here is distilled fromLegoSnark, where two different argument systems can work independently, but yet connected.

Let \vec u

be a subvector of witness \vec w

, that its entries u_i

are restricted to be inside of a given table T

. On the other hand, assume that the PLookup argument proves that values of polynomial f(X)

are inside T

. So far these two argument systems are independent (apart from using the same table T

, which is publicly known).

If one is sure that f(X)

is indeed interpolating to u

, then it is guaranteed that the witness \vec u

of Groth is in the table T

. For this, we use the idea of CP-link, first presented inLegoSnark. CP-link is a general approach that can connect two different proof systems (that share the same (sub)witness) together.

To use CP-link over two proof systems, they require to fulfill a special property that is called Commit-Carrying proof. Intuitively, this means the proof generated by the proof system should contain a commitment to the witness (for us to $\vec u$). Groth16 can be modified to fulfill this property, where the commitment part is an extractable MSM from the proof. In LegoSnark they also show how to bootstrap Groth16 to a commit-carrying SNARK (CC-SNARK). On the other hand, Plookup is already CC-SNARK, since the commitment to $f(X)$ is already available.

What remains is the CP-Link, in LegoSnark they use a SNARK for linear subspace to build the required CP-Link. Here we recap their CP-Link scheme:

- We know that cc-SNARK version of Groth16 already contains a commitment inside its proof. Let c be such a commitment to $\vec u$.
- During the Plookup setup, we commit to f by the Lagrange bases such that in this representation the coefficients are the evaluation value of f (supposed to be $\vec u$). Call this commitment c'. This requires that the CRS of Plookup be based on Lagrange bases, rather than powers of a random τ.

Therefore, we have two commitments to $\vec u$ by different keys ;one w.r.t CRS of Groth16, the other w.r.t CRS of Plookup.

- Then we give a proof for the fact that c and c' are commitments to the same values (this is precisely what we call CP-link and connect Groth16 to Plookup), the proof is as follows: * Let $\vec pk\in G_1$ be the CRS of Groth16 used to build c, and $\vec pk'\in G_1$ be the one for Plookup used in c'. Namely (for $<. , .>$ stands for MSM); $c=<\vec u,\vec pk>, \qquad c'=<\vec u,\vec pk'>$
- We consider a matrix M such that the first row of M is $\vec pk$ and the second row is $\vec pk'$. Therefore, the claim is $M\cdot \vec u=(c,c')^T$.
- The new setup includes M, $P=k^T\cdot M\in G_1$ for a random vector $k\in \mathbb{F^2}$, $C=(g_2^{ak_1},g_2^{ak_2})$ and $A=g_2^{a}$

.

- The prover sends the proof $\pi= P\cdot \vec u$

.

The verifier checks that $e(\pi, A)=e(c,C_1)\cdot e(c',C_2)$

- Let $\vec pk\in G_1$

be the CRS of Groth16 used to build c

, and $\vec pk'\in G_1$

be the one for Plookup used in c'

. Namely (for $<. , .>$

stands for MSM); $c=<\vec u,\vec pk>, \qquad c'=<\vec u,\vec pk'>$

- We consider a matrix M

such that the first row of M

is $\vec pk$

and the second row is $\vec pk'$

. Therefore, the claim is $M\cdot \vec u=(c,c')^T$

.

- The new setup includes M

, $P=k^T\cdot M\in G_1$

for a random vector $k\in \mathbb{F^2}$

, $C=(g_2^{ak_1},g_2^{ak_2})$

and $A=g_2^{a}$

.

- The prover sends the proof $\pi= P\cdot \vec u$

.

The verifier checks that $e(\pi, A)=e(c,C_1)\cdot e(c',C_2)$

Applications; field emulation in Groth16

Assume that the Groth circuit is based on a prime field p

and the prover needs to prove a relation R

based on prime-field p'

such that $p'<p$

. We know that if each number in the relation R

is smaller than p'

then proving $R(w,x)$

based on p

results in a proof based on p'

. This means the bottleneck for the field emulation is proving that each number involved in R

is indeed less than p'

.Thus, we use the Plookup approach to prove the inequalities (i.e., all the numbers are from the range $[0,p'-1]$).

The number of rows should be small enough to allow searching over the rows of the table. Thus, if p' is large we require a slicing-strategy. Where the table of T includes the slices.

Let t be the integer such that $2^t<p'<2^{t+1}$. We then focus on 2^t-1 which due to its structure provides an efficient way for comparisons over slices that we present soon.

The idea is that: if a witness u is smaller than 2^t-1 we do the range proof through the slice-wise Plookup. And if the witness is bigger than 2^t-1, then in the circuit we show that $p'-u=a$, and then by Plookup we argue that a belongs to the table (again slice-wise).

So we apply a slicing strategy where we cut the witness into several slices (done in the circuit), and then show each slice belongs to the table. For example, if 2^t-1 is 64 bits we have 4 slices of 16 bits, and the table contains the range $[0,\sum_{i\in [16]}2^i]$ as its rows.