Notes on Spartan

arnaucube

April 2023

Abstract

Notes taken while reading about Spartan [1].

Usually while reading papers I take handwritten notes, this document contains some of them re-written to $LaTeX$.

The notes are not complete, don't include all the steps neither all the proofs.

Contents

1 R1CS into Sum-Check protocol

Def 1.1. R1CS $\exists w \in \mathbb{F}^{m-|io|-1}$ such that $(A \cdot z) \circ (B \cdot z) = (C \cdot z)$, where $z = (io, 1, w)$.

Thm 4.1 $\forall$ R1CS instance $x = (\mathbb{F}, A, B, C, io, m, n)$, $\exists$ a degree-3 log m-variate polynomial G such that $\sum_{x \in \{0,1\}^{logm}} G(x) = 0$ iff $\exists$ a witness w such that $Sat_{R1CS}(x, w) = 1$.

We can view matrices $A, B, C \in \mathbb{F}^{m \times m}$ as functions $\{0,1\}^s \times \{0,1\}^s \rightarrow \mathbb{F}$ ($s = \lceil \log m \rceil$). For a given witness w to x, let $z = (io, 1, w)$. View z as a function $\{0,1\}^s \rightarrow \mathbb{F}$, so any entry in z can be accessed with a s-bit identifier.

$$F_{io}(x) = \left(\sum_{y \in \{0,1\}^s} A(x,y) \cdot Z(y) \right) \cdot \left(\sum_{y \in \{0,1\}^s} B(x,y) \cdot Z(y) \right) - \sum_{y \in \{0,1\}^s} C(x,y) \cdot Z(y)$$

Lemma 4.1. $\forall x \in \{0,1\}^s,\ F_{io}(x) = 0$ iff $Sat_{R1CS}(x, w) = 1$.

$F_{io}(\cdot)$ is a function, not a polynomial, so it can not be used in the Sum-check protocol.

$F_{io}(x)$ function is converted to a polynomial by using its polynomial extension $\widetilde{F}_{io}(x) : \mathbb{F}^s \to \mathbb{F}$,

$$\widetilde{F}_{io}(x) = \left(\sum_{y \in \{0,1\}^s} \widetilde{A}(x,y) \cdot \widetilde{Z}(y) \right) \cdot \left(\sum_{y \in \{0,1\}^s} \widetilde{B}(x,y) \cdot \widetilde{Z}(y) \right) - \sum_{y \in \{0,1\}^s} \widetilde{C}(x,y) \cdot \widetilde{Z}(y)$$

Lemma 4.2. $\forall x \in \{0,1\}^s,\ \widetilde{F}_{io}(x) = 0$ iff $Sat_{R1CS}(x,w) = 1$.

(proof: $\forall x \in \{0,1\}^s,\ \widetilde{F}_{io}(x) = F_{io}(x)$, so, result follows from Lemma 4.1.)

So, for this, V will need to check that $\widetilde{F}_{io}$ vanishes over the boolean hypercube ($\widetilde{F}_{io}(x) = 0\ \forall x \in \{0,1\}^s$).

Recall that $\widetilde{F}_{io}(\cdot)$ is a low-degree multivariate polynomial over $\mathbb{F}$ in s variables. Thus, checking that $\widetilde{F}_{io}$ vanishes over the boolean hypercube is equivalent to checking that $\widetilde{F}_io = 0$.

Thus, V can check $\sum_{x \in \{0,1\}^s} \widetilde{F}_{io}(x) = 0$ using the Sum-check protocol (through SZ lemma, V can check if for a random value it equals to 0, and be convinced that applies to all the points whp.).

But: as $\widetilde{F}_{io}(x)$ is not multilinear, so $\sum_{x \in \{0,1\}^s} \widetilde{F}_{io}(x) = 0 \not\Leftrightarrow F_{io}(x) = 0\ \forall x \in \{0,1\}^s$. Bcs: the 2^s terms in the sum might cancel each other even when the individual terms are not zero.

Solution: combine $\widetilde{F}_{io}(x)$ with $\widetilde{eq}(t,x)$ to get $Q_{io}(t,x)$ which will be the unique multilinear polynomial, and then check that it is a zero-polynomial

$$Q_{io}(t) = \sum_{x \in \{0,1\}^s} \widetilde{F}_{io}(x) \cdot \widetilde{eq}(t,x)$$

where $\widetilde{eq}(t,x) = \prod_{i=1}^s (t_i \cdot x_i + (1-t_i) \cdot (1-x_i))$, which is the MLE of $eq(x,e) = \{1 \text{ if } x = e,\ 0 \text{ otherwise}\}$.

Basically $Q_{io}(\cdot)$ is a multivariate (the unique multilinear) polynomial such that

$$Q_{io}(t) = \widetilde{F}_{io}(t)\ \forall t \in \{0,1\}^s$$

thus, $Q_{io}(\cdot)$ is a zero-polynomial iff $\widetilde{F}_{io}(x) = 0\ \forall x \in \{0,1\}^s$. $\iff$ iff $\widetilde{F}_{io}(\cdot)$ encodes a witness w such that $Sat_{R1CS}(x,w) = 1$.

$\widetilde{F}_{io}(x)$ has degree 2 in each variable, and $\widetilde{eq}(t,x)$ has degree 1 in each variable, so $Q_{io}(t)$ has degree 3 in each variable.

To check that $Q_{io}(\cdot)$ is a zero-polynomial: check $Q_{io}(\tau) = 0,\ \tau \in^R \mathbb{F}^s$ (Schwartz-Zippel-DeMillo–Lipton lemma) through the sum-check protocol.

This would mean that the R1CS instance is satisfied.

Recap

We have that $Sat_{R1CS}(x,w) = 1$ iff $F_{io}(x) = 0$.

To be able to use sum-check, we use its polynomial extension $\widetilde{F}_{io}(x)$, using sum-check to prove that $\widetilde{F}_{io}(x) = 0 \; \forall x \in \{0,1\}^s$, which means that $Sat_{R1CS}(x,\ w) = 1$.

To prevent potential canceling terms, we combine $\widetilde{F}_{io}(x)$ with $\widetilde{eq}(t,x)$, obtaining $G_{io,\tau}(x) = \widetilde{F}_{io}(x) \cdot \widetilde{eq}(t,x)$.

Thus $Q_{io}(t) = \sum_{x \in \{0,1\}^s} \widetilde{F}_{io}(x) \cdot \widetilde{eq}(t,x)$, and then we prove that $Q_{io}(\tau) = 0$, for $\tau \in^R \mathbb{F}^s$.

2 NIZKs with succinct proofs for R1CS

From Thm 4.1: to check R1CS instance $(\mathbb{F}, A, B, C, io, m, n)$ V can check if $\sum_{x \in \{0,1\}^s} G_{io,\tau}(x) = 0$, which through sum-check protocol can be reduced to $e_x = G_{io,\tau}(r_x)$, where $r_x \in \mathbb{F}^s$.

Recall: $G_{io,\tau}(x) = \widetilde{F}_{io}(x) \cdot \widetilde{eq}(\tau, x)$.

Evaluating $\widetilde{eq}(\tau, r_x)$ takes $O(log\ m)$, but to evaluate $\widetilde{F}_{io}(r_x)$, V needs to evaluate

$$\widetilde{A}(r_x, y), \widetilde{B}(r_x, y), \widetilde{C}(r_x, y), \widetilde{Z}(y),\ \forall y \in \{0,1\}^s$$

which requires 3 sum-check instances ($\left(\sum_{y \in \{0,1\}^s} \widetilde{A}(x,y) \cdot \widetilde{Z}(y)\right)$, $\left(\sum_{y \in \{0,1\}^s} \widetilde{B}(x,y) \cdot \widetilde{Z}(y)\right)$, $\left(\sum_{y \in \{0,1\}^s} \widetilde{C}(x,y) \cdot \widetilde{Z}(y)\right)$), one for each summation in $\widetilde{F}_{io}(x)$.

But note that evaluations of $\widetilde{Z}(y) \; \forall y \in \{0,1\}^s$ are already known as $(io, 1, w)$.

Solution: combination of 3 protocols:

- Sum-check protocol
- randomized mini protocol
- polynomial commitment scheme

Basically to do a random linear combination of the 3 summations to end up doing just a single sum-check.

Observation: let $\widetilde{F}_{io}(r_x) = \overline{A}(r_x) \cdot \overline{B}(r_x) - \overline{C}(r_x)$, where

$$\overline{A}(r_x) = \sum_{y \in \{0,1\}} \widetilde{A}(r_x, y) \cdot \widetilde{Z}(y), \quad \overline{B}(r_x) = \sum_{y \in \{0,1\}} \widetilde{B}(r_x, y) \cdot \widetilde{Z}(y)$$

$$\overline{C}(r_x) = \sum_{y \in \{0,1\}} \widetilde{C}(r_x, y) \cdot \widetilde{Z}(y)$$

Prover makes 3 separate claims: $\overline{A}(r_x) = v_A$, $\overline{B}(r_x) = v_B$, $\overline{C}(r_x) = v_C$, then V evaluates:

$$G_{io,\tau}(r_x) = (v_A \cdot v_B - v_C) \cdot \widetilde{eq}(r_x, \tau)$$

which equals to

$$= \left(\overline{A}(r_x) \cdot \overline{B}(r_x) - \overline{C}(r_x)\right) \cdot \widetilde{eq}(r_x, \tau) =$$

$$\left(\left(\sum_{y\in\{0,1\}} \widetilde{A}(r_x, y) \cdot \widetilde{Z}(y)\right) \cdot \left(\sum_{y\in\{0,1\}} \widetilde{B}(r_x, y) \cdot \widetilde{Z}(y)\right) - \sum_{y\in\{0,1\}} \widetilde{C}(r_x, y) \cdot \widetilde{Z}(y)\right) \cdot \widetilde{eq}(r_x, \tau)$$

This would be 3 sum-check protocol instances (3 claims: $\overline{A}(r_x) = v_A$, $\overline{B}(r_x) = v_B$, $\overline{C}(r_x) = v_C$).

Instead, combine 3 claims into a single claim:

- V samples $r_A, r_B, r_C \in^R \mathbb{F}$, and computes $c = r_A v_A + r_B v_B + r_C v_C$.
- V, P use sum-check protocol to check:

  $$r_A \cdot \overline{A}(r_x) + r_B \cdot \overline{B}(r_x) + r_C \cdot \overline{C}(r_x) == c$$

  Let

  $$\begin{aligned} L(r_x) &= r_A \cdot \overline{A}(r_x) + r_B \cdot \overline{B}(r_x) + r_C \cdot \overline{C}(r_x) \\ &= \sum_{y\in\{0,1\}^s} \left(r_A \cdot \widetilde{A}(r_x, y) \cdot \widetilde{Z}(y) + r_B \cdot \widetilde{B}(r_x, y) \cdot \widetilde{Z}(y) + r_C \cdot \widetilde{C}(r_x, y) \cdot \widetilde{Z}(y)\right) \\ &= \sum_{y\in\{0,1\}^s} M_{r_x}(y) \end{aligned}$$

  $M_{r_x}(y)$ is a s-variate polynomial with deg ≤ 2 in each variable ($\Longleftrightarrow \mu = s,\ l = 2,\ T = c$).

$$\begin{aligned} M_{r_x}(r_y) &= r_A \cdot \widetilde{A}(r_x, r_y) \cdot \widetilde{Z}(r_y) + r_B \cdot \widetilde{B}(r_x, r_y) \cdot \widetilde{Z}(r_y) + r_C \cdot \widetilde{C}(r_x, r_y) \cdot \widetilde{Z}(r_y) \\ &= (r_A \cdot \widetilde{A}(r_x, r_y) + r_B \cdot \widetilde{B}(r_x, r_y) + r_C \cdot \widetilde{C}(r_x, r_y)) \cdot \widetilde{Z}(r_y) \end{aligned}$$

only one term in $M_{r_x}(r_y)$ depends on prover's witness: $\widetilde{Z}(r_y)$, the other terms can be computed locally by V in $O(n)$ time (Section 6 of the paper for sub-linear in n).

Instead of evaluating $\widetilde{Z}(r_y)$ in $O(|w|)$ communications, P sends a commitment to $\widetilde{w}(\cdot)$ (= MLE of the witness w) to V before the first instance of the sum-check protocol.

Recap

To check the R1CS instance, V can check $\sum_{x\in\{0,1\}^s} G_{io,\tau}(x) = 0$, which through the sum-check is reduced to $e_x = G_{io,\tau}(r_x)$, for $r_x \in \mathbb{F}^s$.

Evaluating $G_{io,\tau}(x)$ ($G_{io,\tau}(x) = \widetilde{F}_{io}(x) \cdot \widetilde{eq}(\tau, x)$) is not cheap. Evaluating $\widetilde{eq}(\tau, r_x)$ takes $O(log\ m)$, but to evaluate $\widetilde{F}_{io}(r_x)$, V needs to evaluate $\widetilde{A}, \widetilde{B}, \widetilde{C}, \widetilde{Z},\ \forall y \in \{0,1\}^s$

P makes 3 separate claims: $\overline{A}(r_x) = v_A,\ \overline{B}(r_x) = v_B,\ \overline{C}(r_x) = v_C$, so V can evaluate $G_{io,\tau}(r_x) = (v_A \cdot v_B - v_C) \cdot \widetilde{eq}(r_x, \tau)$

The previous claims are combined into a single claim (random linear combination) to use only a single sum-check protocol:

> P: $c = r_A v_A + r_B v_B + r_C v_C$, for $r_A, r_B, r_C \in^R \mathbb{F}$
>
> V, P: sum-check $r_A \cdot \overline{A}(r_x) + r_B \cdot \overline{B}(r_x) + r_C \cdot \overline{C}(r_x) == c$

$c = L(r_x) = \sum_{y \in \{0,1\}^s} M_{r_x}(y)$, where $M_{r_x}(y)$ is a s-variate polynomial with deg ≤ 2 in each variable ($\iff \mu = s,\ l = 2,\ T = c$). Only $\widetilde{Z}(r_y)$ depends on P's witness, the other terms can be computed locally by V.

Instead of evaluating $\widetilde{Z}(r_y)$ in $O(|w|)$ communications, P uses a commitment to $\widetilde{w}(\cdot)$ (= MLE of the witness w).

2.1 Full protocol

(Recall: Sum-Check params: μ: n vars, n rounds, l: degree in each variable upper bound, T: claimed result.)

- $pp \leftarrow Setup(1^\lambda)$: invoke $pp \leftarrow PC.Setup(1^\lambda, logm)$; output pp
- $b \leftarrow < P(w), V(r) > (\mathbb{F}, A, B, C, io, m, n)$:
    1. P: $(C, S) \leftarrow PC.Commit(pp, \widetilde{w})$ and send C to V
    2. V: send $\tau \in^R \mathbb{F}^{log\ m}$ to P
    3. let $T_1 = 0,\ \mu_1 = log\ m,\ l_1 = 3$
    4. V: set $r_x \in^R \mathbb{F}^{\mu_1}$
    5. Sum-check 1. $e_x \leftarrow < P_{SC}(G_{io,\tau}), V_{SC}(r_x) > (\mu_1, l_1, T_1)$
    6. P: compute $v_A = \overline{A}(r_x),\ v_B = \overline{B}(r_x),\ v_C = \overline{C}(r_x)$, send (v_A, v_B, v_C) to V
    7. V: abort with $b = 0$ if $e_x \neq (v_A \cdot v_B - v_C) \cdot \widetilde{eq}(r_x, \tau)$
    8. V: send $r_A, r_B, r_C \in^R \mathbb{F}$ to P
    9. let $T_2 = r_A \cdot v_A + r_B \cdot v_B + r_C \cdot v_C,\ \mu_2 = log\ m,\ l_2 = 2$
    10. V: set $r_y \in^R \mathbb{F}^{\mu_2}$
    11. Sum-check 2. $e_y \leftarrow < P_{SC}(M_{r_x}), V_{SC}(r_y) > (\mu_2, l_2, T_2)$
    12. P: $v \leftarrow \widetilde{w}(r_y[1..])$, send v to V
    13. $b_e \leftarrow < P_{PC.Eval}(\widetilde{w}, S), V_{PC.Eval}(r) > (pp, C, r_y, v, \mu_2)$
    14. V: abort with $b = 0$ if $b_e == 0$
    15. V: $v_z \leftarrow (1 - r_y[0]) \cdot \widetilde{w}(r_y[1..]) + r_y[0]\widetilde{(io, 1)}(r_y[1..])$
    16. V: $v_1 \leftarrow \widetilde{A}(r_x, r_y),\ v_2 \leftarrow \widetilde{B}(r_x, r_y),\ v_3 \leftarrow \widetilde{C}(r_x, r_y)$

17. V: abort with $b = 0$ if $e_y \neq (r_A v_1 + r_B v_2 + r_C v_3) \cdot v_z$
18. V: output $b = 1$

Section 6 of the paper, describes how in step 16, instead of evaluating $\widetilde{A}$, $\widetilde{B}$, $\widetilde{C}$ at r_x, r_y with $O(n)$ costs, P commits to $\widetilde{A}$, $\widetilde{B}$, $\widetilde{C}$ and later provides proofs of openings.

In a practical implementation those commits to $\widetilde{A}$, $\widetilde{B}$, $\widetilde{C}$ could be done in a preprocessing step.

WIP: covered until sec.6

References

[1] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. Cryptology ePrint Archive, Paper 2019/550, 2019. `https://eprint.iacr.org/2019/550`.