Notes on Nova

arnaucube

March 2023

Abstract

Notes taken while reading Nova [1] paper.

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.

Thanks to Levs57, Nalin Bhardwaj and Carlos Pérez for clarifications on the Nova paper.

Contents

1 NIFS

1.1 R1CS modification

R1CS R1CS instance: (A, B, C, io, m, n), where io denotes the public input and output, $A, B, C \in \mathbb{F}^{m \times n}$, with $m \geq |io| + 1$. R1CS is satisfied by a witness $w \in \mathbb{F}^{m-|io|-1}$ such that

$$Az \circ Bz = Cz$$

where $z = (io, 1, w)$.

Want: merge 2 instances of R1CS with the same matrices into a single one. Each instance has $z_i = (W_i,\ x_i)$ (public witness, private values resp.).

traditional R1CS Merged instance with $z = z_1 + rz_2$, for rand r. But, since R1CS is not linear $\longrightarrow$ can not apply.

eg.

$$\begin{aligned} Az \circ Bz &= A(z_1 + rz_2) \circ B(z_1 + rz_2) \\ &= Az_1 \circ Bz_1 + r(Az_1 \circ Bz_2 + Az_2 \circ Bz_1) + r^2(Az_2 \circ Bz_2) \\ &\neq Cz \end{aligned}$$

$\longrightarrow$ introduce error vector $E \in \mathbb{F}^m$, which absorbs the cross-temrs generated by folding.

$\longrightarrow$ introduce scalar u, which absorbs an extra factor of r in $Cz_1 + r^2 Cz_2$ and in $z = (W, x, 1 + r \cdot 1)$.

Relaxed R1CS

$$\begin{aligned} &u = u_1 + ru_2 \\ &E = E_1 + r(Az_1 \circ Bz_2 + Az_2 \circ Bz_1 - u_1 Cz_2 - u_2 Cz_1) + r^2 E_2 \\ &Az \circ Bz = uCz + E, \quad with\ z = (W,\ x,\ u) \end{aligned}$$

where R1CS set $E = 0,\ u = 1$.

$$\begin{aligned} Az \circ Bz &= Az_1 \circ Bz_1 + r(Az_1 \circ Bz_2 + Az_2 \circ Bz_1) + r^2(Az_2 \circ Bz_2) \\ &= (u_1 Cz_1 + E_1) + r(Az_1 \circ Bz_2 + Az_2 \circ Bz_1) + r^2(u_2 Cz_2 + E_2) \\ &= u_1 Cz_1 + \underbrace{E_1 + r(Az_1 \circ Bz_2 + Az_2 \circ Bz_1) + r^2 E_2}_{\text{E}} + r^2 u_2 Cz_2 \\ &= u_1 Cz_1 + r^2 u_2 Cz_2 + E \\ &= (u_1 + ru_2) \cdot C \cdot (z_1 + rz_2) + E \\ &= uCz + E \end{aligned}$$

For R1CS matrices $(A,\ B,\ C)$, the folded witness W is a satisfying witness for the folded instance $(E,\ u,\ x)$.

Problem: not non-trivial, and not zero-knowledge. Solution: use polynomial commitment with hiding, binding, succintness and additively homomorphic properties.

Committed Relaxed R1CS Instance for a Committed Relaxed R1CS $(\overline{E}, u, \overline{W}, x)$, satisfyied by a witness (E, r_E, W, r_W) such that

$$\begin{aligned} &\overline{E} = Com(E, r_E) \\ &\overline{W} = Com(E, r_W) \\ &Az \circ Bz = uCz + E, \quad where\ z = (W, x, u) \end{aligned}$$

1.2 Folding scheme for committed relaxed R1CS

V and P take two *committed relaxed R1CS* instances

$$\varphi_1 = (\overline{E}_1, u_1, \overline{W}_1, x_1)$$
$$\varphi_2 = (\overline{E}_2, u_2, \overline{W}_2, x_2)$$

P additionally takes witnesses to both instances

$$(E_1, r_{E_1}, W_1, r_{W_1})$$
$$(E_2, r_{E_2}, W_2, r_{W_2})$$

Let $Z_1 = (W_1, x_1, u_1)$ and $Z_2 = (W_2, x_2, u_2)$.

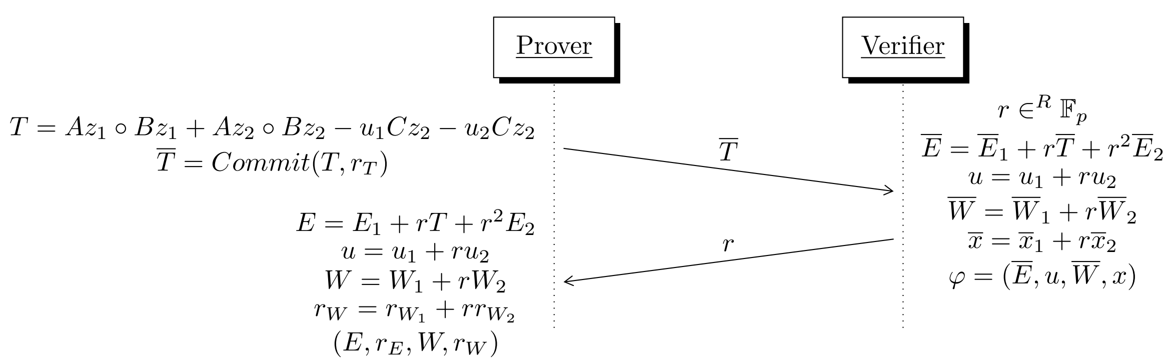
1. P send $\overline{T} = Com(T, r_T)$,
   where $T = Az_1 \circ Bz_1 + Az_2 \circ Bz_2 - u_1Cz_1 - u_2Cz_2$
   and rand $r_T \in \mathbb{F}$
2. V sample random challenge $r \in \mathbb{F}$
3. V, P output the folded instance $\varphi = (\overline{E}, u, \overline{W}, x)$

$$\begin{aligned}
\overline{E} &= \overline{E}_1 + r\overline{T} + r^2\overline{E}_2 \\
u &= u_1 + ru_2 \\
\overline{W} &= \overline{W}_1 + r\overline{W}_2 \\
x &= x_1 + rx_2
\end{aligned}$$

4. P outputs the folded witness (E, r_E, W, r_W)

$$\begin{aligned}
E &= E_1 + rT + r^2E_2 \\
r_E &= r_{E_1} + r \cdot r_T + r^2 r_{E_2} \\
W &= W_1 + rW_2 \\
r_W &= r_{W_1} + r \cdot r_{W_2}
\end{aligned}$$

P will prove that knows the valid witness (E, r_E, W, r_W) for the committed relaxed R1CS without revealing its value.



The previous protocol achieves non-interactivity via Fiat-Shamir transform, obtaining a *Non-Interactive Folding Scheme for Committed Relaxed R1CS*.

Note: the paper later uses u_i, U_i for the two inputed φ_1, φ_2, and later u_{i+1} for the outputed φ. Also, the paper later uses w, W to refer to the witnesses of two folded instances (eg. $\mathsf{w} = (E, r_E, W, r_W)$).

1.3 NIFS

fold witness, $(pk, (u_1, w_1), (u_2, w_2))$:

1. $T = Az_1 \circ Bz_1 + Az_2 \circ Bz_2 - u_1 Cz_2 - u_2 Cz_2$
2. $\overline{T} = Commit(T, r_T)$
3. output the folded witness (E, r_E, W, r_W)

$$\begin{aligned}
E &= E_1 + rT + r^2 E_2 \\
r_E &= r_{E_1} + r \cdot r_T + r^2 r_{E_2} \\
W &= W_1 + rW_2 \\
r_W &= r_{W_1} + r \cdot r_{W_2}
\end{aligned}$$

fold instances $(\varphi_1, \varphi_2) \to \varphi$, $(vk, u_1, u_2, \overline{E}_1, \overline{E}_2, \overline{W}_1, \overline{W}_2, \overline{T})$:
V compute folded instance $\varphi = (\overline{E}, u, \overline{W}, x)$

$$\begin{aligned}
\overline{E} &= \overline{E}_1 + r\overline{T} + r^2 \overline{E}_2 \\
u &= u_1 + ru_2 \\
\overline{W} &= \overline{W}_1 + r\overline{W}_2 \\
x &= x_1 + rx_2
\end{aligned}$$

2 Nova

IVC (Incremental Verifiable Computation) scheme for a non-interactive folding scheme.

2.1 IVC proofs

Allows prover to show $z_n = F^{(n)}(z_0)$, for some count n, initial input z_0, and output z_n.
F: program function (polynomial-time computable)
F': augmented function, invokes F and additionally performs fold-related stuff.

Two committed relaxed R1CS instances:
U_i: represents the correct execution of invocations $1, \ldots, i-1$ of F'
u_i: represents the correct execution of invocations i of F'

Simplified version of F' for intuition F' performs two tasks:

i. execute a step of the incremental computation: instance u_i contains z_i, used to output $z_{i+1} = F(z_i)$

ii. invokes the verifier of the non-interactive folding scheme to fold the task of checking u_i and U_i into the task of checking a single instance U_{i+1}

F' proves that:

1. $\exists((i, z_0, z_i, \mathsf{u}_i, \mathsf{U}_i), \mathsf{U}_{i+1}, \overline{T})$ such that
   i. $\mathsf{u}_i.x = H(vk, i, z_0, z_i, \mathsf{U}_i)$
   ii. $h_{i+1} = H(vk, i+1, z_0, F(z_i), \mathsf{U}_{i+1})$
   iii. $\mathsf{U}_{i+1} = NIFS.V(vk, \mathsf{U}_i, \mathsf{u}_i, \overline{T})$
2. F' outputs h_{i+1}

F' is described as follows:
$\underline{F'(vk, \mathsf{U}_i, \mathsf{u}_i, (i, z_0, z_i), w_i, \overline{T}) \rightarrow x}$:
if $i = 0$, output $H(vk, 1, z_0, F(z_0, w_i), \mathsf{u}_\perp)$
otherwise

1. check $\mathsf{u}_i.x = H(vk, i, z_0, z_i, \mathsf{U}_i)$
2. check $(\mathsf{u}_i.\overline{E}, \mathsf{u}_i.u) = (\mathsf{u}_\perp.\overline{E}, 1)$
3. compute $\mathsf{U}_{i+1} \leftarrow NIFS.V(vk, U, u, \overline{T})$
4. output $H(vk, i+1, z_0, F(z_i, w_i), \mathsf{U}_{i+1})$

IVC Proof iteration $i+1$: prover runs F' and computes u_{i+1}, U_{i+1}, with corresponding witnesses w_{i+1}, W_{i+1}. $(\mathsf{u}_{i+1}, \mathsf{U}_{i+1})$ attest correctness of $i+1$ invocations of F', the IVC proof is $\pi_{i+1} = ((\mathsf{U}_{i+1}, \mathsf{W}_{i+1}), (\mathsf{u}_{i+1}, \mathsf{w}_{i+1}))$.

$\underline{P(pk, (i, z_0, z_i), \mathsf{w}_i, \pi_i) \rightarrow \pi_{i+1}}$:
Parse $\pi_i = ((\mathsf{U}_i, \mathsf{W}_i), (\mathsf{u}_i, \mathsf{w}_i))$, then

1. if $i = 0$: $(\mathsf{U}_{i+1}, \mathsf{W}_{i+1}, \overline{T}) \leftarrow (\mathsf{u}_\perp, \mathsf{w}_\perp, \mathsf{u}_\perp.\overline{E})$
   otherwise: $(\mathsf{U}_{i+1}, \mathsf{W}_{i+1}, \overline{T}) \leftarrow NIFS.P(pk, (\mathsf{U}_i, \mathsf{W}_i), (\mathsf{u}_i, \mathsf{w}_i))$
2. compute $(\mathsf{u}_{i+1}, \mathsf{w}_{i+1}) \leftarrow trace(F', (vk, \mathsf{U}_i, \mathsf{u}_i, (i, z_0, z_i), \mathsf{w}_i, \overline{T}))$
3. output $\pi_{i+1} \leftarrow ((\mathsf{U}_{i+1}, \mathsf{W}_{i+1}), (\mathsf{u}_{i+1}, \mathsf{w}_{i+1}))$

$\underline{V(vk, (i, z_0, z_i), \pi_i) \rightarrow \{0, 1\}}$: if $i = 0$: check that $z_i = z_0$
otherwise, parse $\pi_i = ((\mathsf{U}_i, \mathsf{W}_i), (\mathsf{u}_i, \mathsf{w}_i))$, then

1. check $\mathsf{u}_i.x = H(vk, i, z_0, z_i, \mathsf{U}_i)$

2. check $(\mathsf{u}_i.\overline{E}, \mathsf{u}_i.u) = (\mathsf{u}_\perp.\overline{E}, 1)$
3. check that $\mathsf{W}_i,\ \mathsf{w}_i$ are satisfying witnesses to $\mathsf{U}_i,\ \mathsf{u}_i$ respectively

A zkSNARK of a Valid IVC Proof prover and verifier:

$P(pk, (i, z_0, z_i), \Pi) \rightarrow \pi$:

if $i = 0$, output $\perp$, otherwise:

parse Π as $((\mathsf{U}, \mathsf{W}), (\mathsf{u}, \mathsf{w}))$

1. compute $(\mathsf{U}', \mathsf{W}', \overline{T}) \leftarrow NIFS.P(pk_{NIFS}, (\mathsf{U},\ \mathsf{W}), (\mathsf{u},\ \mathsf{w}))$
2. compute $\pi_{\mathsf{u}'} \leftarrow zkSNARK.P(pk_{zkSNARK}, \mathsf{U}', \mathsf{W}')$
3. output $(\mathsf{U},\ \mathsf{u}, \overline{T}, \pi_{\mathsf{u}'})$

$V(vk, (i, z_0, z_i), \pi) \rightarrow \{0, 1\}$:

if $i = 0$: check that $z_i = z_0$

parse π as $(\mathsf{U}, \mathsf{u}, \overline{T}, \pi_{\mathsf{u}'})$

1. check $\mathsf{u}.x = H(vk_{NIFS}, i, z_0, z_i, \mathsf{U})$
2. check $(\mathsf{u}.\overline{E}, \mathsf{u}.u) = (\mathsf{u}_\perp.\overline{E}, 1)$
3. compute $\mathsf{U}' \leftarrow NIFS.V(vk_{NIFS}, \mathsf{U}, \mathsf{u}, \overline{T})$
4. check $zkSNARK.V(vk_{zkSNARK}, \mathsf{U}', \pi_{\mathsf{u}'}) = 1$

References

[1] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. Cryptology ePrint Archive, Paper 2021/370, 2021. `https://eprint.iacr.org/2021/370`.