Formalising Groth16 in Lean 4

Daniel Rogozin, for Yatima Inc

August 15, 2022*

1 Intro

In this document, we describe the Groth16 soundness formalisation in Lean 4. The text contains the protocol description as well as some comments to its implementation.

Groth16 is a kind of ZK-SNARK protocol. The latter means that:

- It is *zero-knowledge*. In other words, a prover only has a particular piece of information.
- It is *non-interactive* in order to make secret parameters reusable.

Protocols of this kind have the core characteristics such as:

- *Soundness*, i.e., if a statement does not hold, then the prover cannot convince the verifier.
- *Completeness*, i.e., the verifier is convinced whenever a statement is true.
- *Zero-knowledge*, i.e., the only thing is needed is the truth of a statement.

2 Preliminary definitions

We have a fixed finite field F, and $F[X]$ stands for the ring of polynomials over F as usual. The corresponding listing:

```
variable {F : Type u} [field : Field F]
```

In Groth16, we have random values $\alpha, \beta, \gamma, \delta \in F$ that we introduce separately as an inductive data type:

```
inductive Vars : Type
  | alpha : Vars
  | beta : Vars
  | gamma : Vars
  | delta : Vars
```

We also introduce the following parameters:

- $n_{stmt} \in \mathbb{N}$ — the statement size;
- $n_{wit} \in \mathbb{N}$ — the witness size;
- $n_{var} \in \mathbb{N}$ — the number of variables.

In Lean 4, we introduce those parameters as variables in the following way:

```
variable {n_stmt n_wit n_var : Nat}
```

We also define several finite collections of polynomials:

*This document may be updated frequently.

- $u_{stmt} = \{f_i \in F[X] \mid i < n_{stmt}\}$
- $u_{wit} = \{f_i \in F[X] \mid i < n_{wit}\}$
- $v_{stmt} = \{f_i \in F[X] \mid i < n_{stmt}\}$
- $v_{wit} = \{f_i \in F[X] \mid i < n_{wit}\}$
- $w_{stmt} = \{f_i \in F[X] \mid i < n_{stmt}\}$
- $w_{wit} = \{f_i \in F[X] \mid i < n_{wit}\}$

We introduce those collections in Lean 4 as variables as well:

```
variable {u_stmt : Fin n_stmt -> F[X]}
variable {u_wit : Fin n_wit -> F[X]}
variable {v_stmt : Fin n_stmt -> F[X]}
variable {v_wit : Fin n_wit -> F[X]}
variable {w_stmt : Fin n_stmt -> F[X]}
variable {w_wit : Fin n_wit -> F[X]}
```

Let $(r_i)_{i<n_{wit}}$ be a collection of elements of F (that is, each $r_i \in F$) parametrised by elements of n_{wit}. Define a polynomial $t \in F[X]$ as:

$$t = \prod_{i \in n_{wit}} (x - r_i).$$

Crearly, these r_i's are roots of t. The definition in Lean 4:

```
variable (r : Fin n_wit -> F)

def t : F[X] := Pi i in finRange n_wit,
  (x : F[X]) - Polynomial.c (r i)
```

3 Properties of t

The polynomial t has the following properties:

Lemma 1.

1. *$deg(t) = n_{wit}$;*
2. *t is monic, that is, its leading coefficient is equal to 0;*
3. *If $n_{wit} > 0$, then $deg(t) > 0$.*

We formalise these statements as follows (but we drop proofs):

```
lemma nat_degree_t : (t r).natDegree = n_wit
lemma monic_t : Polynomial.Monic (t r)
lemma degree_t_pos (hm : 0 < n_wit) : 0 < (t r).degree
```

Let $\{a_{wit_i} \,|\, i < n_{wit}\}$ and $\{a_{stmt_i} \,|\, i < n_{stmt}\}$ be collections of elements of F. A stamenent witness polynomial pair is a pair of single variable polynomials $(F_{wit_{sv}}, F_{stmt_{sv}})$ such that $F_{wit_{sv}}, F_{stmt_{sv}} \in F[X]$ and

- $F_{wit_{sv}} = \sum_{i<n_{wit}} a_{wit_i} u_{wit_i}(x)$
- $F_{stmt_{sv}} = \sum_{i<n_{stmt}} a_{stmt_i} u_{stmt_i}(x)$

Their Lean 4 counterparts:

```
def V_wit_sv (a_wit : Fin n_wit -> F) : Polynomial F :=
  \sum i in finRange n_wit, a_wit i \bullet u_wit i

def V_stmt_sv (a_stmt : Fin n_stmt -> F) : Polynomial F :=
  \sum i in finRange n_stmt, a_stmt i \bullet u_stmt i
```

Define a polynomial *sat* as:

$$\begin{aligned}sat = (V_{stmt_{sv}} + V_{wit_{sv}})\cdot \\ \cdot ((\sum_{i<n_{stmt}} a_{stmt_i} v_{stmt_i}(x)) + (\sum_{i<n_{wit}} a_{wit_i} v_{wit_i}(x)))- \\ -((\sum_{i<n_{stmt}} a_{stmt_i} w_{stmt_i}(x)) + (\sum_{i<n_{wit}} a_{wit_i} w_{wit_i}(x)))\end{aligned} \quad (1)$$

A pair $(F_{wit_{sv}}, F_{stmt_{sv}})$ satisfies the square span program, if the remainder of division of *sat* by t is equal to 0.

The Lean 4 analogue of the property defined above:

```
def satisfying (a_stmt : Fin n_stmt -> F) (a_wit : Fin n_wit -> F) :=
  (((\sum i in finRange n_stmt, a_stmt i \bullet u_stmt i)
    + \sum i in finRange n_wit, a_wit i \bullet u_wit i)
  *
  ((\sum i in finRange n_stmt, a_stmt i \bullet v_stmt i)
    + \sum i in finRange n_wit, a_wit i \bullet v_wit i)
  -
  ((\sum i in finRange n_stmt, a_stmt i \bullet w_stmt i)
    + \sum i in finRange n_wit, a_wit i \bullet w_wit i) : F[X]) %ₘ (t r) = 0
```

4 Common reference string elements

Assume we interpreted α, β, γ, and δ somehow with elements of F, say crs_α, crs_β, crs_γ, and crs_δ, that is, in Lean 4:

```
def crs_alpha  (f : Vars -> F) : F := f Vars.alpha

def crs_beta (f : Vars -> F) : F := f Vars.beta

def crs_gamma (f : Vars -> F) : F := f Vars.gamma

def crs_delta (f : Vars -> F) : F := f Vars.delta
```

For simplicity, we write this interpretation as a function $f : \{\alpha, \beta, \gamma, \delta\} \to F$ defined by equations:

$$f(a) = crs_a \text{ for } a \in \{\alpha, \beta, \gamma, \delta\}.$$

In addition to those four elements of F we have a collection of degrees for $a \in F$:

$$\{a^i \mid i < n_{var}\}$$

formalised as:

```
def crs_powers_of_x (i : Fin n_var) (a : F) : F := (a)^(i : Nat)
```

We also introduce collections crs_l, crs_m, and crs_n for $a \in F$:

$$crs_l = \frac{((f(\beta)/f(\gamma)) \cdot (u_{stmt_i})(a)) + ((f(\alpha)/f(\gamma)) \cdot (v_{stmt_i})(a)) + w_{stmt_i}(a)}{f(\gamma)} \quad \text{for } i < n_{stmt} \tag{2}$$

$$crs_l = \frac{((f(\beta)/f(\delta)) \cdot (u_{wit_i})(a)) + ((f(\alpha)/f(\delta)) \cdot (v_{wit_i})(a)) + w_{wit_i}(a)}{f(\delta)} \quad \text{for } i < n_{wit} \tag{3}$$

$$crs_l = \frac{a^i \cdot t(a)}{f(\delta)}, \text{for } i < n_{var} \tag{4}$$

Their Lean 4 version:

```
def crs_l (i : Fin n_stmt) (f : Vars -> F) (a : F) : F :=
  ((f Vars.beta / f Vars.gamma) * (u_stmt i).eval (a)
  +
  (f Vars.alpha / f Vars.gamma) * (v_stmt i).eval (a)
  +
  (w_stmt i).eval (a)) / f Vars.gamma.

def crs_m (i : Fin n_wit) (f : Vars -> F) (a : F) : F :=
  ((f Vars.beta / f Vars.delta) * (u_wit i).eval (a)
  +
  (f Vars.alpha / f Vars.delta) * (v_wit i).eval (a)
  +
  (w_wit i).eval (a)) / f Vars.delta

def crs_n (i : Fin (n_var - 1)) (f : Vars -> F) (a : F) : F :=
  ((a)^(i : Nat)) * (t r).eval a / f Vars.delta
```

Assume we have fixed elements of a field A_α, A_β, A_γ, A_δ, B_α, B_β, B_γ, B_δ, C_α, C_β, C_γ, C_δ.

We also have indexed collections $\{A_x \in F \mid x < n_{var}\}$, $\{B_x \in F \mid x < n_{var}\}$, $\{C_x \in F \mid x < n_{var}\}$, $\{A_l \in F \mid l < n_{stmt}\}$, $\{B_l \in F \mid l < n_{stmt}\}$, $\{C_l \in F \mid l < n_{stmt}\}$, $\{A_m \in F \mid m < n_{wit}\}$, $\{B_m \in F \mid m < n_{wit}\}$, $\{C_m \in F \mid m < n_{wit}\}$, $\{A_h \in F \mid h < n_{var-1}\}$, $\{B_h \in F \mid h < n_{var-1}\}$, $\{C_h \in F \mid h < n_{var-1}\}$.

```
variable { A_alpha A_beta A_gamma A_delta : F }
variable { B_alpha B_beta B_gamma B_delta : F }
variable { C_alpha C_beta C_gamma C_delta : F }
variable { A_x B_x C_x : Fin n_var -> F }
variable { A_l B_l C_l : Fin n_stmt -> F }
variable { A_m B_m C_m : Fin n_wit -> F }
variable { A_h B_h C_h : Fin (n_var - 1) -> F }
```

The adversary's proof representation is defined as the following three sums, for $x \in F$:

$$A = A_\alpha \cdot crs_\alpha + A_\beta \cdot crs_\beta + A_\gamma \cdot crs_\gamma + A_\delta \cdot crs_\delta + \\ + \sum_{i<n_{var}} A_{x_i} * x^i + \sum_{i<n_{stmt}} A_{l_i} * crs_l(x) + \\ + \sum_{i<n_{wit}} A_{m_i} * crs_m(x) + \sum_{i<n_{var}-1} A_{h_i} * crs_n(x) \quad (5)$$

$$B = B_\alpha \cdot crs_\alpha + B_\beta \cdot crs_\beta + B_\gamma \cdot crs_\gamma + B_\delta \cdot crs_\delta + \\ + \sum_{i<n_{var}} B_{x_i} * x^i + \sum_{i<n_{stmt}} B_{l_i} * crs_l(x) + \\ + \sum_{i<n_{wit}} B_{m_i} * crs_m(x) + \sum_{i<n_{var}-1} B_{h_i} * crs_n(x) \quad (6)$$

$$C = C_\alpha \cdot crs_\alpha + C_\beta \cdot crs_\beta + C_\gamma \cdot crs_\gamma + C_\delta \cdot crs_\delta + \\ + \sum_{i<n_{var}} C_{x_i} * x^i + \sum_{i<n_{stmt}} C_{l_i} * crs_l(x) + \\ + \sum_{i<n_{wit}} C_{m_i} * crs_m(x) + \sum_{i<n_{var}-1} C_{h_i} * crs_n(x) \quad (7)$$

Here, we provide the Lean 4 version of A only.

```
def A (f : Vars -> F) (x : F) : F :=
  (A_alpha * crs_alpha F f)
  +
  A_beta * crs_beta F f
  +
  A_gamma * crs_gamma F f
  +
  A_delta * crs_delta F f
  +
  \sum i in (finRange n_var), (A_x i) * (crs_powers_of_x F i x)
  +
  \sum i in (finRange n_stmt),
    (A_l i) * (@crs_l F field n_stmt u_stmt v_stmt w_stmt i f x)
  +
  \sum i in (finRange n_wit),
    (A_m i) * (@crs_m F field n_wit u_wit v_wit w_wit i f x)
  +
  \sum i in (finRange (n_var - 1)), (A_h i) * (crs_n F r i f x)
```

A proof is called *verified*, if the following equation holds:

$$A \cdot B = crs_\alpha \cdot crs_\beta + (\sum_{i<n_{stmt}} a_{stmt_i} \cdot crs_{l_i}(x)) \cdot crs_\gamma + C \cdot crs_\delta \quad (8)$$

```
def verified (f : Vars -> F) (x : F) (a_stmt : Fin n_stmt -> F ) : Prop :=
  A f x * B f x =
    (crs_alpha F f * crs_beta F f) +
    ((\sum i in finRange n_stmt, (a_stmt i) * @crs_l i f x) *
      (crs_gamma F f) + C f x * (crs_delta F f))
```

5 Modified common reference string elements

We modify common reference string elements from the previous section as multivariate polynomials.

6 Coefficient lemmas