Baby SNARKs

Ashvni Narayanan, for Yatima Inc

August 1, 2022*

This file describes the implementation of the soundness proof of the Baby SNARKs program. The aim is to explain the code of the proof of soundness in [1]. The mathematics is given in [2], and the code shall be explained in the same notation.

1 Setup

Some notes :

- The `open_locale big_operators` command lets us use the local notation for sums ($\sum$) and products ($\prod$), as defined in the file [3].
- By declaring `universes u`, one assumes that all elements have `Type u`.
- `parameters` is the same as `variables`, and is used to declare variables that have scope in a given `section`. In this case, they are valid throughout the file.
- It would help to `open polynomial` (open the namespace `polynomial`) at the beginning of the file, one then does not need to add the prefix to each lemma that is called from that namespace.

We have as variables `F`, which is a field (although it is mentioned that this is the finite field parameter of the SNARK, the finiteness is nowhere stated or used). We also have the natural number variables `m`, `n_stmt` and `n_wit`. These are m, l and $n - l$ in [2]. n is defined to be the sum of `n_stmt` and `n_wit`.

The collection of polynomials $u_0, u_1, \cdots u_{l-1}$ are defined here. The author defines it in terms of a function `u_stmt`, which takes an element of $\mathbb{Z}/l\mathbb{Z}$ and returns a polynomial with F-coefficients. Note that `fin n_stmt` is nothing but the set of natural numbers up to l, or equivalently, $\mathbb{Z}/l\mathbb{Z}$. `u_wit` is defined similarly to denote the polynomials $u_l, u_{l+1}, \cdots u_{n-1}$. The roots of the polynomial t are defined in the same fashion, with `r i` denoting r_i, for $0 \leq i \leq m - 1$.

The polynomial t is then defined as $t = \prod_{i=0}^{m-1}(X - r_i)$ here. `polynomial.X` denotes X as a polynomial in $F[X]$, and `polynomial.C (r i)` denotes the constant polynomial r_i.

2 Properties of t

The lemma `nat_degree_t` says :

Lemma 1. *The degree of t is m.*

`nat_degree` returns the degree of the polynomial as a natural number. This differs from `polynomial.degree` only when the polynomial is zero. The proof follows simply by noting that the degree of the product of the polynomials $\prod_{i=0}^{m-1}(X - r_i)$ is the sum of the degrees of $X - r_i$ (`nat_degree_prod`), as long as each of these are nonzero (`X_sub_C_ne_zero`).

The lemma `monic_t` then says :

*This document may be updated frequently.

Lemma 2. *The polynomial t is monic.*

The proof follows from the fact that a product of monic polynomials is monic (`monic_prod_of_monic`), and that each $(X - r_i)$ is monic (`monic_X_sub_C`).

The next lemma `degree_t_pos` tells us :

Lemma 3. *If $0 < m$, then the degree of t is positive.*

Note that this lemma uses `degree` instead of `nat_degree`. As a result, we must prove that m is nonzero implies t being nonzero, in which case `nat_degree` and `degree` coincide.

Before getting into the proof, let us first understand the reason for the distinction between `nat_degree` and `degree`. Lean uses the inductive type `option`. Basically, given `A`, `option A` comprises of `none` (the undefined element) and `some a` for all elements `a` of `A`. The function `option.get_or_else a` returns `b` when given `some b` and `a` when given `none`. Given a polynomial `p`, `degree p` returns `some` of the supremum of all numbers n such that X^n has a nonzero coefficient in p. When $p = 0$, this returns the supremum of the empty set, $\perp$, which is the same as `none`. `nat_degree` is then defined to be `(degree p).get_or_else 0` : if `degree p` is $\perp$, it returns 0, and `(degree p)` otherwise.

We first show that it suffices to prove that `degree t = some m`. This follows easily from the fact that `0 < some m` implies `0 < m` (`with_bot.some_lt_some`). The proof is then by induction on `degree t`. If `degree t = none`, then a contradiction is derived, since we then have that `some m = none`, which then implies `m < m`, which is false. In the other case, we have that `degree t = some val` for some value `val`. Then by the definition of `option.get_or_else`, we get that `m = val`, and the proof follows simply from Lemma 1.

3 Some definitions

One of the fundamental concepts used in this proof is that single variable polynomials can also be thought of as multi-variable polynomials. In this section, we give the mechanism to translate between the two, as well as define the polynomials V_w, V_s, B_w, V, H etc, sometimes separately as both single and multivariable polynomials.

Let us first understand the conversion between single and multivariable polynomials. The author defines `vars` to be an inductive type used to index 3-variable polynomials (we shall assume the variables are X, Y and Z throughout). They then define `singlify` to convert 3-variable polynomials to a single variable one : `singlify` replaces the coefficients Y and Z with 1 and leaves X as it is.

On the other side, `X_poly`, `Y_poly` and `Z_poly` are X, Y and Z thought of as elements of $F[X, Y, Z]$.

Any finitely supported function taking values in $\mathbb{N}$ can be thought of as a polynomial. As an example, any finitely supported function m : `vars` $\to \mathbb{N}$ can be thought of as a monomial of the form $X^{m(\texttt{vars.X})}Y^{m(\texttt{vars.Y})}Z^{m(\texttt{vars.Z})}$. Then, given a multivariable polynomial $p(X, Y, Z)$, `mv_polynomial.coeff p m` denotes the coefficient of p at the monomial m.

Given a ring S, a multivariable polynomial $p := \sum_{m\in\sigma} a_\sigma m$ (σ is a set of monomials) with coefficients in R, a function $f : R \to S$, and a function $g : \sigma \to S$, `mv_polynomial.eval_2 f g p :=` $\sum_{m\in\sigma} f(a_\sigma)g(m)$ returns the evaluation of p with respect to f and g. In our case, we get, for $p := \sum_{i,j,k=0}^{1} a_{i,j,k}X^iY^jZ^k$, we get `mv_polynomial.eval_2 polynomial.C singlify p :=` $\sum_{i,j,k=0}^{1} a_{i,j,k}X^i$ as an element of $F[X]$.

We now give the definitions of various single and multivariable polynomials :

- `V_wit_sv` : Given $a_w = (a_l, \cdots, a_{n-1})$, returns $V_w(X) := \sum_{i=l}^{n-1} a_w(i)u_i(X)$ as an element of $F[X]$.
- `V_stmt_sv` : Given $a_s = (a_0, \cdots, a_{l-1})$, returns $V_s(X) := \sum_{i=0}^{l-1} a_s(i)u_i(X)$ as an element of $F[X]$.
- `V_stmt_mv` : Given $a_s = (a_0, \cdots, a_{l-1})$, returns $V_s(X, Y, Z) := V_s(X)$ as an element of $F[X, Y, Z]$.
- `t_mv` : Returns $t(X, Y, Z) := t(X)$ as an element of $F[X, Y, Z]$.
- `crs_powers_of_t` : Given $i \in \{0, \cdots, m-1\}$, returns X^i as an element of $F[X, Y, Z]$.

- `crs_g` : Returns Z as an element of $F[X,Y,Z]$.
- `crs_gb` : Returns ZY as an element of $F[X,Y,Z]$.
- `crs_b_ssps` : Given $i \in \{l, \cdots, n-1\}$, returns $Yu_i(X)$ as an element of $F[X,Y,Z]$.

We also have the variables `b`, `v` and `h` which are functions/strings of length m, $\mathbb{Z}/m\mathbb{Z} \to F$ representing $(b_i)_{i=0}^{m-1}$, $(v_i)_{i=0}^{m-1}$ and $(h_i)_{i=0}^{m-1}$ respectively; `b'`, `v'` and `h'` which are functions/strings of length $n-l$, $\mathbb{Z}/(n-l)\mathbb{Z} \to F$ representing $(b'_i)_{i=l}^{n-l-1}$, $(v'_i)_{i=l}^{n-l-1}$ and $(h'_i)_{i=l}^{n-l-1}$ respectively; and `b_g` `v_g` `h_g` `b_gb` `v_gb` `h_gb`, which are elements of F, representing $b_\gamma, v_\gamma, h_\gamma, b_{\gamma\beta}, v_{\gamma\beta}, h_{\gamma\beta}$ respectively.

We can now define the main polynomials used :

- `B_wit` : Returns $B_w := \sum_{i=0}^{m-1} b_i X^i + b_\gamma Z + b_{\gamma\beta} YZ + \sum_{i=l}^{n-1} b'_i Y u_i(X)$ as an element of $F[X,Y,Z]$
- `V_wit` : Returns $V_w := \sum_{i=0}^{m-1} v_i X^i + v_\gamma Z + v_{\gamma\beta} YZ + \sum_{i=l}^{n-1} v'_i Y u_i(X)$ as an element of $F[X,Y,Z]$
- `H` : Returns $H := \sum_{i=0}^{m-1} h_i X^i + h_\gamma Z + h_{\gamma\beta} YZ + \sum_{i=l}^{n-1} h'_i Y u_i(X)$ as an element of $F[X,Y,Z]$
- `V` : Given $a_s = (a_0, \cdots, a_{l-1})$, returns $V := V_w + V_s$ as an element of $F[X,Y,Z]$

The above information is encapsulated in the following table :

Lean	**Text**	**Description**	**Type**
`X_poly`	X	X	$F[X,Y,Z]$
`Y_poly`	Y	Y	$F[X,Y,Z]$
`Z_poly`	Z	Z	$F[X,Y,Z]$
`V_wit_sv`	$V_w(X)$	$\sum_{i=l}^{n-1} a_w(i) u_i(X)$	$F[X]$
`V_stmt_sv`	$V_s(X)$	$\sum_{i=0}^{l-1} a_s(i) u_i(X)$	$F[X]$
`V_stmt_mv`	$V_s(X)$	$\sum_{i=0}^{l-1} a_s(i) u_i(X)$	$F[X,Y,Z]$
`t_mv`	$t(X)$	$t(X)$	$F[X,Y,Z]$
`crs_powers_of_t i`	X^i	X^i	$F[X,Y,Z]$
`crs_g`	Z	Z	$F[X,Y,Z]$
`crs_gb`	ZY	ZY	$F[X,Y,Z]$
`crs_b_ssps i`	$Yu_i(X)$	$F[X,Y,Z]$	
`b`	$(b_i)_{i=0}^{m-1}$	$(b_i)_{i=0}^{m-1}$	$\mathbb{Z}/m\mathbb{Z} \to F$
`v`	$(v_i)_{i=0}^{m-1}$	$(v_i)_{i=0}^{m-1}$	$\mathbb{Z}/m\mathbb{Z} \to F$
`h`	$(h_i)_{i=0}^{m-1}$	$(h_i)_{i=0}^{m-1}$	$\mathbb{Z}/m\mathbb{Z} \to F$
`b'`	$(b'_i)_{i=l}^{n-l-1}$	$(b'_i)_{i=l}^{n-l-1}$	$\mathbb{Z}/(n-l)\mathbb{Z} \to F$
`v'`	$(v'_i)_{i=l}^{n-l-1}$	$(v'_i)_{i=l}^{n-l-1}$	$\mathbb{Z}/(n-l)\mathbb{Z} \to F$
`h'`	$(h'_i)_{i=l}^{n-l-1}$	$(h'_i)_{i=l}^{n-l-1}$	$\mathbb{Z}/(n-l)\mathbb{Z} \to F$
`b_g`	b_γ	b_γ	F
`v_g`	v_γ	v_γ	F
`h_g`	h_γ	h_γ	F
`b_gb`	$b_{\gamma\beta}$	$b_{\gamma\beta}$	F
`v_gb`	$v_{\gamma\beta}$	$v_{\gamma\beta}$	F
`h_gb`	$h_{\gamma\beta}$	F	
`B_wit`	B_w	$\sum_{i=0}^{m-1} b_i X^i + b_\gamma Z + b_{\gamma\beta} YZ + \sum_{i=l}^{n-1} b'_i Y u_i(X)$	$F[X,Y,Z]$
`V_wit`	V_w	$\sum_{i=0}^{m-1} v_i X^i + v_\gamma Z + v_{\gamma\beta} YZ + \sum_{i=l}^{n-1} v'_i Y u_i(X)$	$F[X,Y,Z]$
`H`	H	$\sum_{i=0}^{m-1} h_i X^i + h_\gamma Z + h_{\gamma\beta} YZ + \sum_{i=l}^{n-1} h'_i Y u_i(X)$	$F[X,Y,Z]$
`V`	V	$V_s + V_w$	$F[X,Y,Z]$

Finally, we say that the pair $(a_i)_{i=0}^{l-1}$ and $(a_i)_{i=l}^{n-1}$ is `satisfying` if

$$\sum_{i=0}^{l-1} a_i u_i(X) + \sum_{i=l}^{n-1} a_i u_i(X) \equiv 1 \texttt{mod}\ t$$

that is, on dividing the above polynomial by t, the remainder obtained is 1. The significance of looking at these sums separately is that the witness information is only available to the prover, not the verifier.

4 Supporting lemmas

In this section we state some lemmas that shall assist us in the proof of the final theorem.

The lemma `my_multivariable_to_single_variable` states that :

Lemma 4. *Given a polynomial p in $F[X]$, we have* `(mv_polynomial.eval_2 polynomial.C singlify (eval_2 mv_polynomial.C X_poly p)) = p`

This gives tells us that converting a single variable polynomial to a multivariable polynomial and reducing it to a single variable polynomial returns the original polynomial. The proof remains the same if `singlify` is replaced with an arbitrary function. Thus we use the lemma `multivariable_to_single_variable`. The proof of this lemma follows simply by unfolding the definitions, using properties of monomials, and the fact that any polynomial p can be written as $\sum_n coeff_p(X^n)X^n$, along with some `simp` lemmas.

The following lemma `eq_helper` is used in `h2_1` :

Lemma 5. *Given natural numbers x and n, $x = j \iff x = j \vee (x = 0 \wedge j = 0)$*

This lemma seems obvious, however, it is quite useful to state beforehand, so it can be used directly in the next lemma. The proof is simple, we split the goal into two statements and get two goals : $x = j \to x = j \vee (x = 0 \wedge j = 0)$ and $x = j \vee (x = 0 \wedge j = 0) \to x = j$. The first implication is trivial. We must split the second implication into 2 cases : $x = j \to x = j$ and $x = 0 \wedge j = 0 \to x = j$. Both implications are trivial.

The next lemma, `h2_1` states that :

Lemma 6. *$\forall 0 \leq i < m$, the coefficient of X^i in B_w (or `B_wit`) is b_i.*

The lemma follows by tracking quotients, unfolding various definitions, removing coercions and applying the lemmas `finsupp.single_eq_single_iff`, `eq_helper` and `fin.eq_iff_veq`. This is done by applying the tactics `simp` and `unfold_coes`. For a full list of lemmas that `simp` uses, one can apply `squeeze_simp`.

Following a similar proof as above, the lemma `h3_1` is proved :

Lemma 7. *The coefficient of Z in B_w (or `B_wit`) is b_γ.*

In fact, a single `simp` proves this, with an addition of `finsupp.single_eq_single_iff`.

The lemma `h4_1` says :

Lemma 8. *Suppose that, $\forall 0 \leq i < m, b_i = 0$. Then, $b_i X^i = 0$. Equivalently, the function defined as $f(i) := b_i \cdot X^i$ is the same as the zero function.*

The lemma is stated in the function form. Here, $\cdot$ represents scalar multiplication of F on $F[X]$. The proof uses the tactic `ext`, which says that functions f and g are equal if and only if $\forall x, f(x) = g(x)$. The conclusion follows from using the hypothesis and applying `zero_smul`.

The lemma `h5_1` says :

Lemma 9. $b_{\gamma\beta} \cdot ZY = Y(b_{\gamma\beta} \cdot Z)$

The lemma uses the fact `mv_polynomial.smul_eq_C_mul`, which says that scalar multiplication of a polynomial by a constant in F is the same as multiplication of the polynomial by the constant polynomial, that is $b \cdot p(X) = b(X) * p(X)$, where $b \in F$ and a polynomial $p(X) \in F[X]$. The tactic `ring` then finishes the proof by using associativity and commutativity of multiplication. One can check what `ring` does by looking at `show_term`{`ring`}.

The lemma `h6_2` says :

Lemma 10. *The coefficient of Z^2 in $Ht + 1$ is 0.*

The coefficient of Z^2 in Ht+1 is precisely the coefficient of Z^2 in Ht, which is the same as $\sum_{i=0}^{2} coeff_H(Z^i)coeff_t(Z^{2-i})$. We know that $coeff_t(Z^i)$ is 0 for every i, which concludes the proof.

The lemma `h6_3` says :

Lemma 11. *Given $(a_i)_{i=0}^{l-1}$, the coefficient of Z^2 in $(b_{\gamma\beta} \cdot Z + \sum_{i=0}^{l-1} a_i u_i(X) + \sum_{j=l}^{n-1} b'_i u_i(X))^2$ is $b^2_{\gamma\beta}$.*

The mathematical proof follows by looking at the coefficients. The code relies on first computing the power. This is done by looking at $z^2 = z * z$. One then uses `mv_polynomial.coeff_mul` to write out the coefficients in terms of sums over the `antidiagonal`, which is the same as using the binomial theorem. Given a finitely supported function s taking values in the natural numbers, `antidiagonal s` is the set $\{(m, n)|m + n = s\}$. Given an element s of type S, the function `finsupp.single s n` should be interpreted as taking value n at s and 0 at all other elements of `vars`. Then, it is easy to see that the lemma `single_2_antidiagonal` follows :

Lemma 12. *Given an element s of a random type S,*
`antidiagonal (finsupp.single s 2) =` $\{$ `(finsupp.single s 0, finsupp.single s 2), (finsupp.single s 1, finsupp.single s 1), (finsupp.single s 2, finsupp.single s 0),` $\}$

We then use `finset.sum_insert` (writing the sum over a union of sets as an addition of the sums over each of the sets), `finsupp.single_eq_single_iff` (`finsupp.single a c = finsupp.single b d` $\iff$ $a = b \wedge c = d \vee c = d = 0$) and `finset.sum_singleton` (evaluating sums over singleton sets), along with some `simp` lemmas.

5 Main theorem

Let us first make a definition. We define the `extractor` to be the function $b' : \mathbb{Z}/(n-l)\mathbb{Z} \to F$.

We are now ready to prove that the Baby SNARK protocol satisfies the property of knowledge soundness. This means that, if an adversary produces polynomials $B(X,Y,Z), V(X,Y,Z), H(X,Y,Z)$ which satisfy

$$B_w = Y V_w \tag{1}$$

$$Ht = V^2 - 1 \tag{2}$$

then one can extract a suitable `satisfying` witness that the adversary must have knowledge of, which is precisely b'.

We now look at Case 1 in the paper [2], that is, when the above equations hold for all X, Y, Z. This is called `case_1` :

Theorem 1. *Given $(a_i)_{i=0}^{l-1}$, $m > 0$, and B_w, V_w, H satisfy the equations (1) and (2) then a_s and the `extractor` b' are `satisfying`.*

The proof is done combining 12 sub-lemmas.

5.1 Sub-lemmas

The lemma `h1` states that :

Lemma 13. *Given a monomial m not having a Y-term (thought of as a finitely supported function from `vars` to $\mathbb{N}$), the coefficient of m in B_w is 0.*

This follows directly from (1) and `mul_var_no_constant`, which says that given a monomial m with no s-term and a multivariable polynomial a, the coefficient of m in $a * s$ is 0.

The lemma h2 states that :

Lemma 14. $\forall 0 \leq i \leq m-1$*,* $b_i = 0$*.*

Given $0 \leq i \leq m-1$, using 4, we must prove that the coefficient of X^i in B_w is 0. One then uses 1, `mul_var_no_constant` and `finsupp.single_apply` ((`finsupp.single a b) a'` is b if `a = a'`, otherwise it is `0`), along with some `simp` lemmas.

The lemma h3 states that :

Lemma 15. $b_\gamma = 0$

First, one uses 4, which reduces to showing that the coefficient of Z in B_w is 0. The proof then follows using 1, `mul_var_no_constant` and `finsupp.single_apply`, along with some `simp` lemmas.

The lemma h4 states that :

Lemma 16. $B_w = b_{\gamma\beta} ZY + \sum_{i=l}^{n-1} b'_i Y u_i(X)$

It suffices to prove that $\sum_{i=0}^{m-1} X^i = b_\gamma Z = 0$. The former claim follows from 8 and 14, and the latter from 15, along with some `simp` lemmas.

The lemma h5 states that :

Lemma 17. $V_w = b_{\gamma\beta} Z + \sum_{i=l}^{n-1} b'_i u_i(X)$

We first multiply both sides by Y using `left_cancel_X_mul vars.Y`. The result then follows by using 1, 5.1 and 4, along with some `simp` lemmas.

The lemma h6 states that :

Lemma 18. $V(a_i)_{i=0}^{l} = b_{\gamma\beta} Z + \sum_{i=0}^{l-1} a_i u_i(X) + \sum_{i=l}^{n-1} b'_i u_i(X)$

This follows easily from the definition of V, and using 17 along with some `simp` lemmas.

The lemma h7 states that :

Lemma 19. $b_{\gamma\beta} = 0$

Consider 2, and expand on it using 18. The statement `h6_1` then says that the coefficient of Z^2 in both sides of the equation must be the same. Finally, using 10 and 11, one obtains $b_{\gamma\beta}^2 = 0$, from which the result follows easily, since the field F has no zero divisors.

The lemma h8 states that :

Lemma 20. $V(a_i)_{i=0}^{l} = \sum_{i=0}^{l-1} a_i u_i(X) + \sum_{i=l}^{n-1} b'_i u_i(X)$

The lemma follows easily using 18, 19 and `simp`.

The lemma h10 states that :

Lemma 21. $(Ht + 1) \equiv (V(a_i)_{i=0}^{l})^2 (mod\ t)$

Here, `singlify` has been used to think of them as single variable polynomials in terms of X. This follows directly from 2.

The lemma h12 states that :

Lemma 22. *The multivariable constant polynomial 1 is the same as the multivariable polynomial $\sum_n (coeff_1(X^n))X^n$, where, given a polynomial $p \in F[X]$, $coeff_p(X^n)$ denotes the coefficient of X^n in p.*

This lemma is dependent on the definition of `polynomial.eval_2`. The definition of this is very complicated, however, we shall remark that the proof follows directly from the lemma `polynomial.eval_2_C`.

The lemma h13 states that :

Lemma 23. $(Ht + 1) \equiv H(mod\ t)$ *and* $(Ht + 1) \equiv 1(mod\ t)$*, where these polynomials are thought of as single variable polynomials.*

We first apply `polynomial.div_mod_by_monic_unique`, which says that, given polynomials f, g, q, r, with g monic, such that $f = gq + r$ and $deg(r) < deg(g)$, then, $f \div_m g = q$ and $f(\text{mod } g) \equiv r$. Here, $\div_m$ means the quotient when f is divided by g. We must then prove that t is monic (follows from 2), and that the degree of the constant polynomial 1 is larger than $deg(t)$ (follows from 3), along with some `simp` lemmas.

We are now ready to tackle the proof of the theorem.

5.2 Proof of main theorem

We must prove that, given $(a_i)_{i=0}^{l-1}$,

$$\sum_{i=0}^{l-1} a_i u_i(X) + \sum_{i=l}^{n-1} b'_i u_i(X) \equiv 1 \texttt{mod } t$$

First, we define `h9` to be 20, seen as an element of $F[X]$. We then use the additivity properties of the `eval_2` map, and the lemma 4 on `h9`. Using `h9`, it now suffices to prove that
mv_polynomial.eval_2 polynomial.C singlify (V a_stmt)$^2 \equiv 1 (\text{mod } t)$.

We then use 21 and 4 to change the statement to
(mv_polynomial.eval_2 polynomial.C singlify H * t + mv_polynomial.eval_2 polynomial.C singlify (mv_polynomial.C 1)) $\equiv 1 (\text{mod } t)$.

Finally, the result follows from applying 22,23 and 4 along with some `simp` lemmas. Yay!

References

[1] Bolton Bailey. Knowledge soundness of baby snarks. `https://github.com/BoltonBailey/formal-snarks-project/blob/master/src/snarks/babysnark/knowledge_soundness.lean`, 2021.

[2] Ye Zhang Andrew Miller and Sanket Kanjalkar. Baby snark (do do dodo dodo). `https://github.com/initc3/babySNARK/blob/master/babysnark.pdf`, 2020.

[3] Lean 3. `https://github.com/leanprover-community/mathlib/blob/master/src/algebra/big_operators/basic.lean`.