An Introduction to ZK SNARKs

Mark Blunden

June 2020

ZK SNARKS are a class of proof, where ZK SNARK stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge". Examples of ZK SNARKS include Bulletproofs, Plonk, and one commonly referred to as Groth16 (denoting the author and year).

This article looks at an early example of a practical ZK SNARK, namely Pinocchio, in an elliptic curve setting. For ease of explanation a simplified version is described. The protocol makes use of elliptic curve parings (described below) is made up of several distinct components, but the focus here is to describe what is being proved and how the proof is performed. Other aspects of the protocol — such as the incorporation of the zero-knowledge aspect — are left to another time.

Note that the aim is for clarity of explanation rather than mathematical precision.

The protocol uses Elliptic Curves, where any two points P_1 and P_2 on an elliptic curve can be added together using a special form of point addition to obtain a third point P_3. That is, $P_3 = P_1 + P_2$ for the special type of addition '+'. Similarly, a point P can be added to itself. Adding a point P to itself n times is denoted as $n * P$, so that, for example, $3 * P$ is the same as $P + P + P$. There exist efficient ways to compute $n * P$. The specific form of encryption used here defines the encryption of a value 'x' as $E(x) = x * U$, where U is a point (specifically, a generator) on the elliptic curve. A property of this form of encryption is that $E(a+b) = (a+b) * U = (a * U) + (b * U) = E(a) + E(b)$.

The Problem Definition

Arithmetic or binary circuits can be converted into a rank-1 constraint system (R1CS). For reasons of simplicity a description of the process for this

is omitted from here, but a good description can be found here.

The result of using a R1CS is that the circuit can be represented in the form

$$A(x) \cdot B(x) - C(x) = H(x) \cdot T(x)$$

where A(x), B(x) and C(x) are linear combinations of $(A_1, \ldots, A_m)$, $(B_1, \ldots, B_m)$ and $(C_1, \ldots, C_m)$ respectively, and $A_1, \cdots, A_m$, $B_1, \cdots, B_m$, $C_1, \cdots, C_m$ are themselves polynomials of degree at most d. In other words, $A(x) = c_1 \cdot A_1 + c_2 \cdot A_2 + \cdots + c_m \cdot A_m$ for some coefficients $c_1, \ldots, c_m$, and $A_i = a_{i,0} + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,d}x^d$ for $i = 1, \ldots, m$. Note that the inclusion of 'i' in the subscript of the coefficients of A_i represents the fact that each A_i is defined by a different set of coefficients. $B(x)$ and $C(x)$ are defined similarly.

Note that A_i is shorthand for $A_i(x)$, and similarly for B_i and C_i. Note also that this linear combination can be represented as $A(x) = (c_1, c_2, \ldots, c_m) \cdot (A_1(x), A_2(x), \ldots, A_m(x))$, where in this instance only '$\cdot$' represents a vector dot product.

There are three important properties of this representation:

1. The linear combinations that define $A(x)$, $B(x)$ and $C(x)$ all use the same coefficients $c_1, \ldots, c_m$.
2. The polynomial sets $(A_1 \ldots, A_m)$, $(B_1, \ldots, B_m)$ and $(C_1, \ldots, C_m)$, along with the polynomial $T(x)$ are known, but the coefficients $c_1, \ldots, c_m$ that define $A(x)$, $B(x)$ and $C(x)$ are known only to the Prover. *Note that in practice some of the coefficients will be known, but are here assumed to be private for reasons of simplicity.*
3. $H(x)$ is dependent on the coefficients $c_1, \ldots, c_m$, and computed by dividing $A(x) \cdot B(x) - C(x)$ by $T(x)$. Consequently, the Prover (knowing the coefficients) can compute $H(x)$, but it is not known to any other party.

Proof Objectives

The objective is to prove knowledge of coefficients $c_1, \ldots, c_m$ such that

$$A(x) \cdot B(x) - C(x) = H(x) \cdot T(x)$$

is true when evaluated at a value t, so that

$$A(t) \cdot B(t) - C(t) = H(t) \cdot T(t)$$

Importantly, the value of t is unknown to the Prover and Verifier (or anyone else — this is a security requirement).

The proof is as follows:

1. Prove that $A(t)$ is a linear combination of $A_1, \ldots, A_m$ (and similarly for $B(t)$ and $C(t)$)
2. Show that $A(t) \cdot B(t) - C(t) = H(t) \cdot T(t)$
3. Prove that all three linear combinations $A(t)$, $B(t)$, $C(t)$ use the same coefficients

This can be achieved using:

- The d-power knowledge of exponent assumption
- Elliptic Curve Pairings

The n-power Knowledge of Exponent Assumption

In essence, this says that *if you know (pairs of) points* $(P_1, Q_1), \ldots, (P_n, Q_n)$ *of an elliptic curve such that* $Q_i = k * P_i$ *for* $i = 1, \ldots, n$ *but* k *is unknown, the only way to generate a new point* (R, S) *such that* $S = k * R$ *is by knowledge of a linear combination of* $P_1, \ldots, P_n$ *that gives* R*.*

That is, given $(P_1, Q_1), \ldots, (P_n, Q_n)$, the provision of (R, S) such that $S = k * R$ implies that (R, S) is a linear combination of $(P_1, Q_1), \ldots, (P_n, Q_n)$, and that this linear combination is known to the entity providing (R, S).

Note that formal definitions typically define the points P_i in a specific way.

To apply this assumption, recall that $A(x)$ (or equivalently, $B(x)$ or $C(x)$) is defined as a linear combination of $(A_1, A_2, \ldots, A_m)$. That is,

$$A(x) = c_1 \cdot A_1(x) + c_2 \cdot A_2(x) + \cdots + c_m \cdot A_m(x)$$

for some coefficients $c_1, \ldots, c_m$ known only to the Prover.

So

$$A(t) = c_1 \cdot A_1(t) + c_2 \cdot A_2(t) + \cdots + c_m \cdot A_m(t)$$

represents $A(x)$ evaluated at 't'.

Recall from earlier that $E(x)$ denotes the Elliptic Curve encryption of a value 'x'. That is, $E(x) = x * U$ for a point U (where U is a generator). So,

$$\begin{aligned} E(A(t)) &= (c_1 \cdot A_1(t) + c_2 \cdot A_2(t) + \cdots + c_m \cdot A_m(t)) * U \\ &= c_1 * (A_1(t) * U) + c_2 * (A_2(t) * U) + \cdots + c_m * (A_m(t) * U) \\ &= c_1 * S_1 + c_2 * S_2 + \cdots + c_m * S_m \end{aligned}$$

where $S_i = A_i(t) * U$, for $i = 1, \ldots, m$. Note that here, '+' denotes the addition of points.

So, given the S_i (which form a part of the setup process), the Prover can compute $E(A(t))$.

Note that the points S_i are equivalent to the points P_i in the knowledge of exponent assumption statement above, and $E(A(t))$ is a linear combination of S_i and so represents R.

Similarly,

$$\begin{aligned} k * E(A(t)) &= k * (c_1 * S_1 + c_2 * S_2 + \cdots + c_m * S_m) \\ &= c_1 * (k * S_1) + c_2 * (k * S_2) + \cdots + c_m * (k * S_m) \\ &= c_1 * cT_1 + c_2 * T_2 + \cdots + c_m * T_m \end{aligned}$$

where $T_i = k * S_i$, for $i = 1, \ldots, m$. Here, the points T_i are equivalent to the points Q_i in the assumption statement above.

Note that $E(A(t))$ and $k * E(A(t))$ can be computed using S_i and T_i respectively provided the coefficients $c_1, \ldots, c_m$ are known, and without knowledge of k.

The value k is chosen as part of the set up parameters, and a different value should be chosen for each of $A(x)$, $B(x)$ and $C(x)$. More formally, for $i = 1, \ldots, n$, we can define the set of values used with $A(t)$ as $(k_A, S_{A,i}, T_{A,i})$, the set of values used with $B(t)$ as $(k_B, S_{B,i}, T_{B,i})$ and the set of values used with $C(t)$ as $(k_C, S_{C,i}, T_{C,i})$.

Since no one including the Prover should know 't' and 'k', it is necessary for a trusted process to generate the T_i and S_i.

So, the n-power knowledge of exponent assumption can be rewritten as the following:

Given points $(T_1, S_1), \ldots, (T_m, S_m)$ of an elliptic curve such that $T_i = k \cdot S_i$ for $i = 1, \ldots, m$, and where k is not known, the provision by the Prover of (R, S) where $R = E(A(t))$ and $S = k * R$ implies that (R, S) is a linear combination of $(T_1, S_1), \ldots, (T_m, S_m)$.

However, this only takes us a part of the way there as given such a point (R, S), it is still necessary to verify that $S = k*R$. For this we need pairings.

Cryptographic Pairings

A pairing is a mapping from a pair of (not necessarily distinct) mathematical sets G_1 and G_2 to a third mathematical set G_T ('T' for 'Target'). That is, it takes a pair of elements (one from G_1 and one from G_2) and maps this pair to an element of G_T. More formally, a pairing is a bilinear mapping and can be written as $e : G_1 \times G_2 \rightarrow G_T$, where e denotes the pairing operation, and G_1, G_2 and G_T are groups.

For pairings commonly used in cryptography, G_1 and G_2 are additive groups (loosely, where members of the group can be added together using a special type of addition) and G_3 is a multiplicative group (where members of G_T can be multiplied together using a special type of addition).

More specifically, G_1 and G_2 are usually subgroups of elliptic curves over a finite field, and the pairing can be written as

$$e(a * U, b * V) = e(U, V)^{ab} = g^{ab}$$

where $U \in G_1$ (that is, the point U is in G_1), $V \in G_2$, $a*U$ is the encryption of 'a' in the elliptic curve group G_1, $b * V$ is the encryption of 'b' in the elliptic curve group G_2, and $g = e(U, V)$ represents an element of G_T. So, the pairing allows the multiplication of two encrypted elliptic curve points, where the result of the multiplication is represented in G_T (by the product 'ab').

For purposes of familiarity, some properties of the pairing function are given below:

$$e(a*U, b*V) = e(a*b*U, V) = e(b*U, a*V) = e(U, a*V)^b = e(U, V)^{ab} = g^{ab}$$

In particular, $e(U, a * V) = e(a * U, V)$. Consequently, if U and W represent the values $E(A(t))$ and $k * E(A(t))$ respectively as computed earlier by the Prover, and provided that $k * V$ is known, it can be verified that $W = k * U$ by verifying $e(U, k * V) = e(W, V)$.

Note that practical pairings of the type used here exist for only some types of elliptic curve, and there is a trade off (dependent on the embedding degree, determined by G_T) in terms of security and efficiency. For this reason the curves are carefully chosen.

Proving that $A(t) \cdot B(t) - C(t) = H(t) \cdot T(t)$

Recall that the objective is to prove that

$$A(t) \cdot B(t) - C(t) = H(t) \cdot T(t)$$

This can be rewritten as

$$A(t) \cdot B(t) = H(t) \cdot T(t) + C(t)$$

which requires multiplications (of $A(t)$, $B(t)$ and also of $H(t)$, $T(t)$) and an addition.

However, when using Elliptic Curve encryption as described previously, there is the problem of how the Prover can compute $E(A(t) \cdot B(t))$ from $E(A(t))$ and $E(B(t))$, for example, without knowledge of t.

But pairings allow encrypted values to be multiplied together. For example, for $U \in G_1$ and $V \in G_2$ (where U and V are typically generators of G_1 and G_2), $E(a) = a * U$ and $E(b) = b * V$, and the product of $E(a)$ and $E(b)$ is

$$e(E(a), E(b)) = e(a * U, b * V) = e(U, V)^{ab}$$

Hence

$$A(t) \cdot B(t) = H(t) \cdot T(t) + C(t)$$

can be rewritten as

$$e(E(A(t)), E(B(t))) = e(E(H(t)), E(T(t))) \cdot e(E(C(t)), V)$$

The correctness of this equation can be verified using only the encrypted values $E(A(t))$, $E(B(t))$, $E(H(t))$, $E(T(t))$ and $E(C(t))$.

However, this does require $E(H(t))$ to be computed by the Prover. Recall from earlier that $H(x)$ is dependent on the coefficients $c_1, \ldots, c_m$. Since these coefficients are known to the Prover, $H(x)$ can be computed by dividing $A(x) \cdot B(x) - C(x)$ by $T(x)$. Consequently, $H(x)$ is a polynomial with coefficients known by the Prover. But since 't' is not known, $H(t)$ cannot be evaluated directly. However, once $H(x)$ is known, $E(H(t))$ can be computed as follows.

Let $H(x) = b_0 + b_1 \cdot x + b_2 \cdot x^2 + \cdots + b_n \cdot x^n$, say.

Then

$$\begin{aligned} E(H(t)) &= (b_0 + b_1 \cdot t + b_2 \cdot t^2 + \cdots + b_n \cdot t^n) * U \\ &= b_0 * U + b_1 * (t * U) + b_2 * (t^2 * U) + \cdots + b_n * (t^n * U) \\ &= b_0 * Y_0 + b_1 * Y_1 + \cdots + b_n * Y_n \end{aligned}$$

where $Y_i = t^i * U$ for $i = 0, \ldots, n$ are provided as part of the trusted setup.

Proving that A(t), B(t), C(t) use the Same Coefficients

It was proven earlier that $A(t)$, $B(t)$ and $C(t)$ are linear combinations of the A_i, B_i and C_i respectively. The objective now is to prove that each linear combination uses the same coefficients $c_1, \ldots, c_m$.

If the coefficients are the same, then for $i = 0, \ldots, m$,

$$r \cdot c_i \cdot (A_i(t) + B_i(t) + C_i(t)) = r \cdot (c_i \cdot A_i(t) + c_i \cdot B_i(t) + c_i \cdot C_i(t))$$

where 'r' is a value used during the trusted setup and is not known to any party.

Summing for $i = 1, \ldots, m$, the right-hand side becomes $r \cdot (A(t) + B(t) + C(t))$.

Therefore,

$$\begin{aligned} E(r \cdot (A(t) + B(t) + C(t))) &= r * (A(t) + B(t) + C(t)) * U \\ &= r * (E(A(t)) + E(B(t)) + E(C(t))) \end{aligned}$$

where $E(A(t))$, $E(B(t))$ and $E(C(t))$ were computed earlier.

Summing for $i = 1, \ldots, m$, the left-hand side gives:

$$c_1 \cdot r \cdot (A_1(t) + B_1(t) + C_1(t)) \; + \; \cdots \; + \; c_m \cdot r \cdot (A_m(t) + B_m(t) + C_m(t)) = Z, say.$$

Therefore, encrypting Z gives

$$\begin{aligned} E(Z) &= c_1 * (r \cdot (A_1(t) + B_1(t) + C_1(t)) * U + \cdots \\ &\quad + \; c_m * (r \cdot (A_m(t) + B_m(t) + C_m(t)) * U \\ &= c_1 \cdot Z_1 \; + \; c_2 \cdot Z_2 \; + \cdots + \; c_m \cdot Z_m \end{aligned}$$

where $Z_i = (r \cdot (A_i(t) + B_i(t) + C_i(t))) * U$ is provided as part of the trusted setup.

Finally, recall from earlier that given a pair (W, U) it can be checked that $W = k * U$ by verifying that $e(U, k * V) = e(W, V)$. Therefore, by setting $W = E(Z)$, $U = E(A(t)) + E(B(t)) + E(C(t))$ and $k = r$, the coefficient check becomes

$$e(E(A(t)) + E(B(t)) + E(C(t)), r * V) = e(E(Z), V)$$