MyZKP: Introduction to Zero Knowledge Protocol

Hideaki Takahashi
Columbia University
ht2673@columbia.edu

THIS MANUSCRIPT IS WIP. IT IS LIKELY TO CONTAIN MANY WRONG INFORMATION

Contents

1 Basics of Number Theory

Let X be a set in this section.

1.1 Computation Rule and Properties

Definition 1: Binary Operation

A mapping $\circ : X \times X \to X$ is a binary operation on X if for any pair of elements (x_1, x_2) in X, $x_1 \circ x_2$ is also in X.

Example: Addition (+) on the set of integers is a binary operation. For example, $5 + 3 = 8$, and both $5, 3, 8$ are integers, staying within the set of integers.

Definition 2: Associative Property

A binary operation $\circ$ is associative if $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in X$.

Example: Multiplication of real numbers is associative: $(2\times3)\times4 = 2\times(3\times4) = 24$. In a modular context, we also have addition modulo n being associative. For example, for $n = 5$, $(2+3) \bmod 5 + 4 \bmod 5 = 2 + (3 \bmod 5 + 4) \bmod 5 = 4$.

Definition 3: Commutative Property

A binary operation $\circ$ is commutative if $a \circ b = b \circ a$ for all $a, b \in X$.

Example: Addition modulo n is also commutative. For $n = 7$, $5 + 3 \bmod 7 = 3 + 5 \bmod 7 = 1$.

1.2 Semigroup, Group, Ring

Definition 4: Semigroup

A pair $(H, \circ)$, where H is a non-empty set and $\circ$ is an associative binary operation on H, is called a semigroup.

Example: The set of positive integers under multiplication modulo n forms a semigroup. For instance, with $n = 6$, the elements $\{1, 2, 3, 4, 5\}$ under multiplication modulo 6 form a semigroup, since multiplication modulo 6 is associative.

Definition 5: Abelian Semigroup

A semigroup whose operation is commutative is called an abelian semigroup.

Example: The set of natural numbers under addition modulo n forms an abelian semigroup. For $n = 7$, addition modulo 7 is both associative and commutative, so it is an abelian semigroup.

Definition 6: Identity Element

An element $e \in H$ is an identity element of H if it satisfies $e \circ a = a \circ e = a$ for any $a \in H$.

Example: 0 is the identity element for addition modulo n. For example, $0 + a \bmod 5 = a + 0 \bmod 5 = a$. Similarly, 1 is the identity element for multiplication modulo n. For example, $1 \times a \bmod 7 = a \times 1 \bmod 7 = a$.

Definition 7: Monoid

A semigroup with an identity element is called a monoid.

Example: The set of non-negative integers under addition modulo n forms a monoid. For $n = 5$, the set $\{0, 1, 2, 3, 4\}$ under addition modulo 5 forms a monoid with 0 as the identity element.

Definition 8: Inverse

For an element $a \in H$, an element $b \in H$ is an inverse of a if $a \circ b = b \circ a = e$, where e is the identity element.

Example: In modulo n arithmetic (addition), the inverse of an element exists if it can cancel itself out to yield the identity element. In the set of integers modulo 7, the inverse of 3 is 5, because $3 \times 5 \bmod 7 = 1$, where 1 is the identity element for multiplication.

Definition 9: Group

A monoid in which every element has an inverse is called a group.

Example: The set of integers modulo a prime p under multiplication forms a group (Can you prove it?). For instance, in $\mathbb{Z}/5\mathbb{Z}$, every non-zero element $\{1 + 5\mathbb{Z}, 2 + 5\mathbb{Z}, 3 + 5\mathbb{Z}, 4 + 5\mathbb{Z}\}$ has an inverse, making it a group.

Definition 10: Order of a Group

The order of a group is the number of elements in the group.

Example: The group of integers modulo 4 under addition has order 4, because the set of elements is $\{0, 1, 2, 3\}$.

Definition 11: Ring

A triple $(R, +, \cdot)$ is a ring if $(R, +)$ is an abelian group, $(R, \cdot)$ is a semigroup, and the distributive property holds: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ and $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$ for all $x, y, z \in R$.

Example: The set of integers with usual addition and multiplication modulo n forms a ring. For example, in $\mathbb{Z}/6\mathbb{Z}$, addition and multiplication modulo 6 form a ring.

Definition 12: Commutative Ring

A ring is called a commutative ring if its multiplication operation is commutative.

The set of real numbers under usual addition and multiplication forms a commutative ring.

Definition 13: Field

A commutative ring with a multiplicative identity element where every non-zero element has a multiplicative inverse is called a field.

The set of rational numbers under usual addition and multiplication forms a field.

Definition 14: Residue Class

The residue class of a modulo m, denoted as $a + m\mathbb{Z}$, is the set $\{b : b \equiv a \pmod{m}\}$.

Example: For $m = 3$, the residue class of 2 is $2 + 3\mathbb{Z} = \{\ldots, -4, -1, 2, 5, 8, \ldots\}$.

Definition 15: Inverse of Residue Class

We denote the set of all residue classes modulo m as $\mathbb{Z}/m\mathbb{Z}$. We say that $a + m\mathbb{Z}$ is invertible in $\mathbb{Z}/m\mathbb{Z}$ if and only if there exists a solution for $ax \equiv 1 \pmod{m}$.

Theorem 1

$a + m\mathbb{Z}$ is invertible in $\mathbb{Z}/m\mathbb{Z}$ if and only if $\gcd(a, m) = 1$.

Proof. TBD □

Example: In $\mathbb{Z}/5\mathbb{Z}$, $3 + 5\mathbb{Z}$ is invertible because $\gcd(3, 5) = 1$ ($3 \cdot 2 \equiv 1 \bmod 5$). However, in $\mathbb{Z}/6\mathbb{Z}$, $3 + 6\mathbb{Z}$ is not invertible because $\gcd(3, 6) = 3 \neq 1$ ($3 \cdot 1 \equiv 3 \bmod 6$, $3 \cdot 2 \equiv 0 \bmod 6$, $3 \cdot 3 \equiv 9 \bmod 6$, $3 \cdot 4 \equiv 0 \bmod 6$ ...).

Definition 16: Residue Class Ring

$(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ is a commutative ring where $1 + m\mathbb{Z}$ is the multiplicative identity element. This ring is called the residue class ring modulo m.

$\mathbb{Z}/4\mathbb{Z} = \{0 + 4\mathbb{Z}, 1 + 4\mathbb{Z}, 2 + 4\mathbb{Z}, 3 + 4\mathbb{Z}\}$ is a residue class ring modulo 4. We omit

Definition 17: Primitive Residue Class

A residue class $a + m\mathbb{Z}$ is called primitive if $\gcd(a, m) = 1$.

Example: In $\mathbb{Z}/6\mathbb{Z}$, the primitive residue classes are $1 + 6\mathbb{Z}$ and $5 + 6\mathbb{Z}$.

Theorem 2

A residue ring $\mathbb{Z}/m\mathbb{Z}$ is a field if and only if m is a prime number.

Proof. TBD □

Example: For $m = 5$, $\mathbb{Z}/5\mathbb{Z} = \{0 + 5\mathbb{Z}, 1 + 5\mathbb{Z}, 2 + 5\mathbb{Z}, 3 + 5\mathbb{Z}, 4 + 5\mathbb{Z}\}$ forms a field because 5 is a prime number, and every non-zero element has a multiplicative inverse. For example, $3 \times 2 \bmod 5 = 1$, so 2 is the inverse of 3 modulo 5.

However, for $m = 6$, $\mathbb{Z}/6\mathbb{Z} = \{0 + 6\mathbb{Z}, 1 + 6\mathbb{Z}, 2 + 6\mathbb{Z}, 3 + 6\mathbb{Z}, 4 + 6\mathbb{Z}, 5 + 6\mathbb{Z}\}$ does not form a field because 6 is not prime, and not all elements have inverses. For instance, there is no inverse for 2, as $\gcd(2, 6) \neq 1$.

Definition 18: Primitive Residue Class Group

The group of all primitive residue classes modulo m is called the primitive residue class group, denoted by $(\mathbb{Z}/m\mathbb{Z})^\times$.

Example: For $m = 8$, the set of all primitive residue classes is $(\mathbb{Z}/8\mathbb{Z})^\times = \{1 + 8\mathbb{Z}, 3 + 8\mathbb{Z}, 5 + 8\mathbb{Z}, 7 + 8\mathbb{Z}\}$. These are the integers less than 8 that are coprime to 8 (i.e., $\gcd(a, 8) = 1$).

Contrast this with $m = 9$. The primitive residue class group is $(\mathbb{Z}/9\mathbb{Z})^\times = \{1 + 9\mathbb{Z}, 2 + 9\mathbb{Z}, 4 + 9\mathbb{Z}, 5 + 9\mathbb{Z}, 7 + 9\mathbb{Z}, 8 + 9\mathbb{Z}\}$, as these are the integers less than 9 that are coprime to 9.

Definition 19: Euler's Totient Function

Euler's totient function $\phi(m)$ is equal to the order of the primitive residue class group modulo m, which is the number of integers less than m and coprime to m.

Example: For $m = 12$, $\phi(12) = 4$ because there are 4 integers less than 12 that are coprime to 12: $\{1, 5, 7, 11\}$.

For $m = 10$, $\phi(10) = 4$, as there are also 4 integers less than 10 that are coprime to 10: $\{1, 3, 7, 9\}$.

Definition 20: Order of an element within a group

Order of $g \in G$ is the minimum number of natural number e satisfying $g^e = 1$. We denote it as $\text{order}_g g$ or order g

Example: In $(\mathbb{Z}/7\mathbb{Z})^\times$, the element 3 has order 6 because $3^6 \bmod 7 = 1$. In other words, $3 \times 3 \times 3 \times 3 \times 3 \times 3 \bmod 7 = 1$, and 6 is the smallest such exponent.

Definition 21: Subgroup

The subset $U \subseteq G$ is a subgroup of G if U itself is a group by the operation of G.

Example: Consider $(\mathbb{Z}/8\mathbb{Z})^\times = \{1 + 8\mathbb{Z}, 3 + 8\mathbb{Z}, 5 + 8\mathbb{Z}, 7 + 8\mathbb{Z}\}$ under multiplication modulo 8. The subset $\{1 + 8\mathbb{Z}, 7 + 8\mathbb{Z}\}$ forms a subgroup because it satisfies the group properties: closed under multiplication, contains the identity element (1), and every element has an inverse ($7 \times 7 \equiv 1 \bmod 8$).

Definition 22: Subgroup generated by g

The set $g^k : k \in \mathbb{Z}$, for some element $g \in G$, forms a subgroup of G and is called the subgroup generated by g, denoted by $\langle g \rangle$.

Example: Consider the group $(\mathbb{Z}/7\mathbb{Z})^\times = \{1 + 7\mathbb{Z}, 2 + 7\mathbb{Z}, 3 + 7\mathbb{Z}, 4 + 7\mathbb{Z}, 5 + 7\mathbb{Z}, 6 + 7\mathbb{Z}\}$ under multiplication modulo 7. If we take $g = 3$, then $\langle 3 + 7\mathbb{Z} \rangle = \{3^1 + 7\mathbb{Z}, 3^2 + 7\mathbb{Z}, 3^3 + 7\mathbb{Z}, 3^4 + 7\mathbb{Z}, 3^5 + 7\mathbb{Z}, 3^6 + 7\mathbb{Z}\} \bmod 7 =$

$\{3+7\mathbb{Z}, 2+7\mathbb{Z}, 6+7\mathbb{Z}, 4+7\mathbb{Z}, 5+7\mathbb{Z}, 1+7\mathbb{Z}\}$, which forms a subgroup generated by 3. This subgroup contains all elements of $(\mathbb{Z}/7\mathbb{Z})^\times$, making 3 a generator of the entire group.

If g has a finite order e, we have that $\langle g\rangle = g^k : 0 \le k \le e$, meaning e is the order of $\langle g\rangle$.

Definition 23: Cyclic Group

A group G is called a cyclic group if there exists an element $g \in G$ such that $G = \langle g\rangle$. In this case, g is called a generator of G.

Example: The group $(\mathbb{Z}/6\mathbb{Z})^\times = \{1+6\mathbb{Z}, 5+6\mathbb{Z}\}$ under multiplication modulo 6 is a cyclic group. In this case, both 1 and 5 are generators of the group because $\langle 5+6\mathbb{Z}\rangle = \{(5^1 \bmod 6)+6\mathbb{Z} = 5+6\mathbb{Z}, (5^2 \bmod 6)+6\mathbb{Z} = 1+6\mathbb{Z}\}$. Since 5 generates all the elements of the group, G is cyclic.

Theorem 3

If G is a finite cyclic group, it has $\phi(|G|)$ generators, and each generator has order $|G|$.

Proof. TBD □

Example: Consider the group $(\mathbb{Z}/8\mathbb{Z})^\times = \{1+8\mathbb{Z}, 3+8\mathbb{Z}, 5+8\mathbb{Z}, 7+8\mathbb{Z}\}$. This group is cyclic, and $\phi(8) = 4$. The generators of this group are $\{1+8\mathbb{Z}, 3+8\mathbb{Z}, 5+8\mathbb{Z}, 7+8\mathbb{Z}\}$, each of which generates the entire group when raised to successive powers modulo 8. Each generator has the same order, which is $|G| = 4$.

Theorem 4

If G is a finite cyclic group, the order of any subgroup of G divides the order of G.

Proof. TBD □

Example: Consider the cyclic group $(\mathbb{Z}/6\mathbb{Z})^\times = \{1+6\mathbb{Z}, 5+6\mathbb{Z}\}$ under multiplication modulo 6. If we take the subgroup $\langle 5+6\mathbb{Z}\rangle = \{1+6\mathbb{Z}, 5+6\mathbb{Z}\}$, this is a subgroup of order 2, and 2 divides the order of the original group, which is 6. This theorem generalizes this property: for any subgroup of a cyclic group, its order divides the order of the group.

Theorem 5: Fermat's Little Theorem

If $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$.

Proof. TBD □

Example: Take $a = 2$ and $m = 5$. Since $\gcd(2, 5) = 1$, Fermat's Little Theorem tells us that $2^{\phi(5)} = 2^4 \equiv 1 \bmod 5$. Indeed, $2^4 = 16$ and $16 \bmod 5 = 1$.

This theorem suggests that $a^{\phi(m)-1} + m\mathbb{Z}$ is the inverse residue class of $a + m\mathbb{Z}$.

Theorem 6

The order of any element in a group divides the order of the group.

Proof. TBD □

Example: In the group $(\mathbb{Z}/7\mathbb{Z})^\times$, consider the element $3+7\mathbb{Z}$. The order of $3+7\mathbb{Z}$ is 6, as $3^6 \equiv 1 \bmod 7$. The order of the group itself is also 6, and indeed, the order of the element divides the order of the group.

Theorem 7: Generalization of Fermat's Little Theorem

For any element $g \in G$, we have $g^{|G|} = 1$.

Proof. TBD □

Example: In the group $(\mathbb{Z}/7\mathbb{Z})^\times$, for any element g, such as $g = 3 + 7\mathbb{Z}$, we have $3^6 \equiv 1 \bmod 7$. This holds for any $g \in (\mathbb{Z}/7\mathbb{Z})^\times$ because the order of the group is 6. Thus, $g^{|G|} = 1$ is satisfied.

Definition 24: Pairing

Let G_1 and G_2 be cyclic groups under addition, both of prime order p, with generators P and Q respectively:

$$G_1 = \{0, P, 2P, ..., (p-1)P\} \tag{1}$$
$$G_2 = \{0, Q, 2Q, ..., (p-1)Q\} \tag{2}$$

Let G_T be a cyclic group under multiplication, also of order p. A pairing is a map $e : G_1 \times G_2 \to G_T$ that satisfies the following bilinear property:

$$e(aP, bQ) = e(P, Q)^{ab} \tag{3}$$

for all $a, b \in \mathbb{Z}_p$.

Imagine G_1 represents length, G_2 represents width, and G_T represents area. The pairing function e is like calculating the area: If you double the length and triple the width, the area becomes six times larger: $e(2P, 3Q) = e(P, Q)^6$

1.3 Polynomials

Theorem 8: Lagrange Interpolation

A n-degre polynomial $P(x)$ that goes through different $n+1$ points $\{(x_1, y_1), (x_2, y_2), \cdots (x_{n+1}, y_{n+1})\}$ is uniquely represented as follows:

$$P(x) = \sum_{i=1}^{n+1} y_i \frac{f_i(x)}{f_i(x_i)} \tag{4}$$

, where $f_i(x) = \prod_{k \neq i}(x - x_k)$

Proof. TBD □

For example, the quadratic polynomial that goes through $\{(1, 0), (2, 3), (3, 8)\}$ is as follows:

$$P(x) = 0\frac{(x-2)(x-3)}{(1-2)(1-3)} + 3\frac{(x-1)(x-3)}{(2-1)(2-3)} + 8\frac{(x-1)(x-2)}{(3-1)(3-2)} = x^2 - 1$$

Note that Lagrange interpolation finds the lowest degree of interpolating polynomial for the given vector.

Proposition 1: Homomorphisms of Lagrange Interpolation

Let $L(v)$ and $L(w)$ be the polynomial resulting from Lagrange Interpolation on the output (y) vector v and w for the same inputs (x). Then, the following properties hold:

- Additivity: $L(v + w) = L(v) + L(w)$ for any vectors v and w
- Scalar multiplication: $L(\gamma v) = \gamma L(v)$ for any scalar γ and vector v

Proof. Let $v = (v_1, \ldots, v_n)$ and $w = (w_1, \ldots, w_n)$ be vectors, and $x_1, \ldots, x_n$ be the interpolation points.

$$L(v + w) = \sum_{i=1}^{n}(v_i + w_i)\prod_{j\neq i}\frac{x - x_j}{x_i - x_j} = \sum_{i=1}^{n} v_i \prod_{j\neq i}\frac{x - x_j}{x_i - x_j} + \sum_{i=1}^{n} w_i \prod_{j\neq i}\frac{x - x_j}{x_i - x_j} = L(v) + L(w)$$

$$L(\gamma v) = \sum_{i=1}^{n}(\gamma v_i)\prod_{j\neq i}\frac{x - x_j}{x_i - x_j} = \gamma\sum_{i=1}^{n} v_i \prod_{j\neq i}\frac{x - x_j}{x_i - x_j} = \gamma L(v)$$

□

Lemma 1: Schwartz - Zippel Lemma

Let $\mathbb{F}$ be a field and $P : F^m \to \mathbb{F}$ and $Q : \mathbb{F}^m \to \mathbb{F}$ be two distinct multivariate polynomials of total degree at most n. For any finite subset $\mathbb{S} \subseteq \mathbb{F}$, we have:

$$Pr_{u\sim\mathbb{S}^m}[P(u) = Q(u)] \leq \frac{n}{|\mathbb{S}|} \tag{5}$$

where u is drawn uniformly at random from $\mathbb{S}^m$.

Proof. TBD □

This lemma states that if $\mathbb{S}$ is sufficiently large and n is relatively small, the probability that the two different polynomials return the same value for a randomly chosen input is negligibly small. In other words, if we observe $P(u) = Q(u)$ for a random input u, we can conclude with high probability that P and Q are identical polynomials.

1.4 Elliptic curve

1.5 Assumptions

Assumption 1: Discrete Logarithm Problem

Let G be a finite cyclic group of order n, with γ as its generator and 1 as the identity element. For any element $\alpha \in G$, there is currently no known efficient (polynomial-time) algorithm to compute the smallest non-negative integer x such that $\alpha = \gamma^x$.

The Discrete Logarithm Problem can be thought of as a one-way function. It's easy to compute g^x given g and x, but it's computationally difficult to find x given g and g^x.

Assumption 2: Knowledge of Exponent Assumption

Let G be a cyclic group of prime order q with generator $g \in G$. For any probabilistic polynomial-time algorithm $\mathcal{A}$ that outputs:

$$\mathcal{A}(g, g^x) = (h, h') \quad s.t. \quad h' = h^x \tag{6}$$

, there exists an efficient extractor $\mathcal{E}$ such that:

$$\mathcal{E}(\mathcal{A}, g, g^x) = y \quad s.t. \quad h = g^y \tag{7}$$

This assumption states that if $\mathcal{A}$ can compute the pair (g^y, g^{xy}) from (g, g^x), then $\mathcal{A}$ must "know" the value y, in the sense that $\mathcal{E}$ can extract y from $\mathcal{A}$'s internal state. The Knowledge of Exponent Assumption is useful for constructing verifiable exponential calculation algorithms. Consider a scenario where Alice has a secret value a, and Bob has a secret value b. Bob wants to obtain g^{ab}. This can be achieved through the following protocol:

Protocol 1: Verifiable Exponential Calculation Algorithm

1. Bob sends $(g, g' = g^b)$ to Alice
2. Alice sends $(h = g^a, h' = g'^a)$ to Bob
3. Bob checks $h^b = h'$.

Thanks to the Discrete Logarithm Assumption and the Knowledge of Exponent Assumption, the following properties hold:

- Alice cannot derive b from (g, g').
- Bob cannot derive a from (h, h').
- Alice knows a value a such that $h = g^a$. In other words, h is the power of g.

1.6 Useful Algorithms

2 Basic of zk-SNARKs

We will explore the design of zk-SNARKs by incrementally developing zero-knowledge proof protocols.

Overview

2.1 Arithmetization

The ultimate goal of Zero-Knowledge Proofs (ZKP) is to allow the prover to demonstrate their knowledge to the verifier without revealing any additional information. This knowledge typically involves solving complex problems, such as finding a secret input value that corresponds to a given public hash. ZKP protocols usually convert these statements into polynomial constraints. This process is often called *arithmetization*.

To make the protocol flexible, we need to encode this knowledge in a specific format, and one common approach is using Boolean circuits. It's well-known that problems in P (those solvable in polynomial time) and NP (those where the solution can be verified in polynomial time) can be represented as Boolean circuits. This means adopting Boolean circuits allows us to handle both P and NP problems.

However, Boolean circuits are often large and inefficient. Even a simple operation, like adding two 256-bit integers, can require hundreds of Boolean operators. In contrast, arithmetic circuits—essentially systems of equations involving addition, multiplication, and equality—offer a much more compact representation. Additionally, any Boolean circuit can be converted into an arithmetic circuit. For instance, the Boolean expression $z = x \wedge y$ can be represented as $x(x-1) = 0$, $y(y-1) = 0$, and $z = xy$ in an arithmetic circuit. Furthermore, as we'll see in this section, converting arithmetic circuits into polynomial constraints allows for much faster evaluation.

2.1.1 Rank-1 Constraint System (R1CS)

There are many formats to represent arithmetic circuits, and one of the most popular ones is R1CS (Rank-1 Constraint System), which represents arithmetic circuits as a set of equality constraints, each involving only one multiplication. In an arithmetic circuit, we call the concrete values assigned to the variables within the constraints witness. We first provide the formal definition of R1CS as follows:

Definition 25: R1CS

An R1CS structure $\mathcal{S}$ consists of:

- Size bounds $m, n, N, \ell \in \mathbb{N}$ where $n > \ell$
- Three matrices $O, L, R \in \mathbb{F}^{m \times n}$ with at most $N = \Omega(\max(m, n))$ non-zero entries in total

An R1CS instance includes a public input $p \in \mathbb{F}^{\ell}$, while an R1CS witness is a vector $w \in \mathbb{F}^{n-\ell-1}$. A structure-instance tuple (S, p) is satisfied by a witness w if:

$$(L \cdot a) \circ (R \cdot a) - O \cdot a = \mathbf{0} \tag{8}$$

where $a = (1, w, p) \in \mathbb{F}^n$, $\cdot$ denotes matrix-vector multiplication, and $\circ$ is the Hadamard product.

The intuitive interpretation of each matrix is as follows:

- L: Encodes the left input of each gate
- R: Encodes the right input of each gate
- O: Encodes the output of each gate
- The leading 1 in the witness vector allows for constant terms

Single Multiplication Let's consider a simple example where we want to prove $z = x \cdot y$, with $z = 3690$, $x = 82$, and $y = 45$.

- **Witness vector**: $(1, z, x, y) = (1, 3690, 82, 45)$
- **Number of witnesses**: $m = 4$
- **Number of constraints**: $n = 1$

The R1CS constraint for $z = x \cdot y$ is satisfied when:

$$\begin{aligned}
&([0 \quad 0 \quad 1 \quad 0] \cdot a) \circ ([0 \quad 0 \quad 0 \quad 1] \cdot a) - [0 \quad 1 \quad 0 \quad 0] \cdot a \\
=&([0 \quad 0 \quad 1 \quad 0] \cdot \begin{bmatrix} 1 \\ 3690 \\ 82 \\ 45 \end{bmatrix}) \circ ([0 \quad 0 \quad 0 \quad 1] \cdot \begin{bmatrix} 1 \\ 3690 \\ 82 \\ 45 \end{bmatrix}) - [0 \quad 1 \quad 0 \quad 0] \cdot \begin{bmatrix} 1 \\ 3690 \\ 82 \\ 45 \end{bmatrix} \\
=&82 \cdot 45 - 3690 \\
=&3690 - 3690 \\
=&0
\end{aligned}$$

This example demonstrates how R1CS encodes a simple multiplication constraint:

- $L = [0 \quad 0 \quad 1 \quad 0]$ selects x (left input)
- $R = [0 \quad 0 \quad 0 \quad 1]$ selects y (right input)
- $O = [0 \quad 1 \quad 0 \quad 0]$ selects z (output)

Multiple Constraints Let's examine a more complex example: $r = a \cdot b \cdot c \cdot d$. Since R1CS requires that each constraint contain only one multiplication, we need to break this down into multiple constraints:

$$\begin{aligned}
v_1 &= a \cdot b \\
v_2 &= c \cdot d \\
r &= v_1 \cdot v_2
\end{aligned}$$

Note that alternative representations are possible, such as $v_1 = ab, v_2 = v_1 c, r = v_2 d$. In this example, we use 7 variables $(r, a, b, c, d, v_1, v_2)$, so the dimension of the witness vector will be $m = 8$ (including the constant 1). We

have three constraints, so $n = 3$. To construct the matrices L, R, and O, we can interpret the constraints as linear combinations:

$$\begin{aligned} v_1 &= (0 \cdot 1 + 0 \cdot r + 1 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d + 0 \cdot v_1 + 0 \cdot v_2) \cdot b \\ v_2 &= (0 \cdot 1 + 0 \cdot r + 0 \cdot a + 0 \cdot b + 1 \cdot c + 0 \cdot d + 0 \cdot v_1 + 0 \cdot v_2) \cdot d \\ r &= (0 \cdot 1 + 0 \cdot r + 0 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d + 1 \cdot v_1 + 0 \cdot v_2) \cdot v_2 \end{aligned}$$

Thus, we can construct L, R, and O as follows:

$$L = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$O = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Where the columns in each matrix correspond to $(1, r, a, b, c, d, v_1, v_2)$.

Addition with a Constant Let's examine the case $w = a \cdot b + 3$. We can represent this as $-3 + w = a \cdot b$. For the witness vector $(1, w, a, b)$, we have:

$$\begin{aligned} L &= [0 \quad 0 \quad 1 \quad 0] \\ R &= [0 \quad 0 \quad 0 \quad 1] \\ O &= [-3 \quad 1 \quad 0 \quad 0] \end{aligned}$$

Note that the constant 3 appears in the O matrix with a negative sign, effectively moving it to the left side of the equation

Multiplication with a Constant Now, let's consider $w = 3a^2 + b$. The requirement of "one multiplication per constraint" doesn't apply to multiplication with a constant, as we can treat it as repeated addition.

$$\begin{aligned} L &= [0 \quad 0 \quad 3 \quad 0] \\ R &= [0 \quad 0 \quad 1 \quad 0] \\ O &= [0 \quad 1 \quad 0 \quad -1] \end{aligned}$$

2.1.2 Quadratic Arithmetic Program (QAC)

If the prover wants to prove that it knows a witness w, it means that the prover knows the vector a that satisfies $(L \cdot a) \circ (R \cdot a) = O \cdot a$. However, evaluating this equivalence requires $\Omega(n)$, where n is the number of rows. To make it faster, if we can convert this comparison of matrices to the comparison of polynomials, we can leverage Schwartz-Zippel Lemma, which states that we need $\Omega(1)$ evaluation to check whether two polynomials are identical.

Suppose we just want to test a simple equivalence: $Av = Bu$, where:

$$A = \begin{bmatrix} 6 & 3 \\ 4 & 7 \end{bmatrix}, B = \begin{bmatrix} 3 & 9 \\ 12 & 6 \end{bmatrix}, v = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, u = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \tag{9}$$

.

Then, the equivalence check can be represented as :

$$\begin{bmatrix}6\\4\end{bmatrix} \cdot 2 + \begin{bmatrix}3\\7\end{bmatrix} \cdot 4 = \begin{bmatrix}3\\12\end{bmatrix} \cdot 2 + \begin{bmatrix}9\\6\end{bmatrix} \cdot 2 \tag{10}$$

Then, the above matrix-vector equality check is equivalent to the following polynomial equality check (recall that Lagrangean Interpolation meets the scalar multiplication and addition homomorphism):

$$2 \cdot L([(1,6),(2,4)]) + 4 \cdot L([(1,3),(2,7)]) = 2 \cdot L([(1,3),(2,12)]) + 2 \cdot L([(1,9),(2,6)]) \tag{11}$$

, where L means the Lagrange Interpolation. In $\mod 17$, we have that $L([(1,6),(2,4)]) = 15x+8$, $L([(1,3),(2,7)]) = 4x+16$, $L([(1,3),(2,12)]) = 9x+11$, $L([(1,9),(2,6)]) = 14x+12$. Recall the Scwartz-Zipple Lemma! That lemma says that we need only one evaluation on a random point to check the equivalacne of polynomials.

However, holomorphic multiplication between two Lagrange Interpolation does hot hold, so that we do not have $\ell(x) \cdot r(x) = o(x)$, where $\ell(x)$, $r(x)$, and $o(x)$ are the polynomials resulted from the lagrange interpolation on $L \cdot a$, $R \cdot a$, and $O \cdot a$. It is easy to see that all of $\ell(x)$, $r(x)$, and $o(x)$ are $n-1$ degrees at most, while $\ell(x) \cdot r(x)$ is $2n-2$ degrees at most. To adjust this this difference, let me introduce the n degrees polynomial: $t(x) = \Pi_{i=1}^{n}(x-i)$. Then, we can rewrite the $\ell(x) \cdot r(x)$ as follows:

$$\ell(x) \cdot r(x) = o(x) + h(x) \cdot t(x) \tag{12}$$

, where $h(x) = \frac{\ell(x) \cdot r(x) - o(x)}{t(x)}$.

2.2 Protocol Ideas

Goal The Prover $\mathcal{A}$ aims to convince a Verifier $\mathcal{B}$ that $\mathcal{A}$ knows a specific polynomial. Let's denote this polynomial of degree n with coefficients in a finite field as:

$$P(x) = c_0 + c_1 x + c_2 x^2 + \cdots c_n x^n \tag{13}$$

Assume $P(x)$ has n roots, $a_1, a_2, \ldots, a_n \in \mathbb{F}$, such that $P(x) = (x-a_1)(x-a_2)\cdots(x-a_n)$. The Verifier $\mathcal{B}$ knows $d < n$ roots of $P(x)$, namely $a_1, a_2, \ldots, a_d$. Let $T(x) = (x-a_1)(x-a_2)\cdots(x-a_d)$. Note that the Prover also knows $T(x)$.

The Prover's objective is to convince the Verifier that $\mathcal{A}$ knows a polynomial $H(x) = \frac{P(x)}{T(x)}$.

Naive Approach The simplest approach to prove that $\mathcal{A}$ knows $H(x)$ is as follows:

Protocol 2: Naive Approach

1. $\mathcal{B}$ sends all possible values in $\mathbb{F}$ to $\mathcal{A}$.
2. $\mathcal{A}$ computes and sends all possible outputs of $H(x)$ and $P(x)$.
3. $\mathcal{B}$ checks whether $H(a)T(a) = P(a)$ holds for any a in $\mathbb{F}$.

This protocol is highly inefficient, requiring $\mathcal{O}(|\mathbb{F}|)$ evaluations and communications.

+ Schwartz-Zippel Lemma Instead of evaluating polynomials at all values in $\mathbb{F}$, we can leverage the Schwartz-Zippel Lemma: if $H(s) = \frac{P(s)}{T(s)}$ or equivalently $H(s)T(s) = P(s)$ for a random element s, we can conclude that $H(x) = \frac{P(x)}{T(x)}$ with high probability. Thus, the Prover $\mathcal{A}$ only needs to send evaluations of $P(s)$ and $H(s)$ for a random input s received from $\mathcal{B}$

Protocol 3: + Schwartz-Zippel Lemma

1. $\mathcal{B}$ draws random s from $\mathbb{F}$ and sends it to $\mathcal{A}$.
2. $\mathcal{A}$ computes $h = H(s)$ and $p = P(s)$ and send them to $\mathcal{B}$.
3. $\mathcal{B}$ checks whether $p = th$, where t denotes $T(s)$.

This protocol is efficient, requiring only a constant number of evaluations and communications. However, it is vulnerable to a malicious prover who could send an arbitrary value h' and the corresponding p' such that $p' = h't$.

+ Discrete Logarithm Assumption To address this vulnerability, the Verifier must hide the randomly chosen input s from the Prover. This can be achieved using the discrete logarithm assumption: it is computationally hard to determine s from α, where $\alpha = g^s \mod p$. Thus, it's safe for the Verifier to send α, as the Prover cannot easily derive s from it.

An interesting property of polynomial exponentiation is:

$$g^{P(x)} = g^{c_0 + c_1 x + c_2 x^2 + \cdots c_n x^n} = g^{c_0} (g^x)^{c_1} (g^{x^2})^{c_2} \cdots (g^{x^n})^{c_n} \tag{14}$$

Instead of sending s, the Verifier can send g, $\alpha = g^s$, and its powers: $\alpha, \alpha^2, \ldots, \alpha^n$. The Prover can still evaluate $g^p = g^{P(s)}$ using these powers of α:

$$g^p = g^{P(s)} = g^{c_0} \alpha^{c_1} (\alpha^2)^{c_2} \cdots (\alpha^n)^{c_n} \tag{15}$$

Similarly, the Prover can evaluate $g^h = g^{H(s)}$. The Verifier can then check $p = ht \iff g^p = (g^h)^t$.

Protocol 4: + Discrete Logarithm Assumption

1. $\mathcal{B}$ randomly draw s from $\mathbb{F}$.
2. $\mathcal{B}$ computes and sends $\{\alpha, \alpha^2, ..., \alpha^n\}$, where $\alpha = g^s$.
3. $\mathcal{A}$ computes and sends $u = g^p$ and $v = g^h$.
4. $\mathcal{B}$ checks whether $u = v^t$.

This approach prevents the Prover from obtaining s or $t = T(s)$, making it impossible to send fake (h', p') such that $p' = h't$.

However, this protocol still has a flaw. Since the Prover can compute $g^t = \alpha^{c_1} (\alpha^2)^{c_2} \cdots (\alpha^d)^{c_d}$, they could send fake values $((g^t)^z, g^z)$ instead of (g^p, g^h) for an arbitrary value z. The verifier's check would still pass, and they could not detect this deception.

+ Knowledge of Exponent Assumption To address the vulnerability where the verifier $\mathcal{B}$ cannot distinguish if $v(= g^h)$ from the prover is a power of $\alpha = g^s$, we can employ the Knowledge of Exponent Assumption. This approach involves the following steps:

1. $\mathcal{B}$ sends both α and $\alpha' = \alpha^r$ for a new random value r.
2. The prover returns $a = (\alpha^i)^{c_i}$ and $a' = (\alpha'^i)^{c_i}$ for $i = 1, ..., n$.
3. $\mathcal{B}$ can conclude that a is a power of α if $a^r = a'$.

Based on this assumption, we can design an improved protocol:

Protocol 5: + Knowledge of Exponent Assumption

1. $\mathcal{B}$ randomly selects s and r from field $\mathbb{F}$.
2. $\mathcal{B}$ computes and sends $\{\alpha, \alpha^2, ..., \alpha^n\}$ and $\{\alpha', \alpha'^2, ..., \alpha'^n\}$, where $\alpha = g^s$ and $\alpha' = \alpha^r = g^{sr}$.
3. $\mathcal{A}$ computes and sends $u = g^p$, $v = g^h$, and $w = g^{p'}$, where $g^{p'} = g^{P(sr)}$.
4. $\mathcal{B}$ checks whether $u^r = w$.
5. $\mathcal{B}$ checks whether $u = v^t$.

The prover can compute $g^{p'} = g^{P(sr)} = \alpha'^{c_1} (\alpha'^2)^{c_2} \cdots (\alpha'^n)^{c_n})$ using powers of α'. This protocol now satisfies the properties of a SNARK: completeness, soundness, and efficiency.

+ Zero Knowledge To transform the above protocol into a zk-SNARK, we need to ensure that the verifier cannot learn anything about $P(x)$ from the prover's information. This is achieved by having the prover obfuscate all information with a random secret value δ:

Protocol 6: + Zero Knowledge

1. $\mathcal{B}$ randomly selects s and r from field $\mathbb{F}$.
2. $\mathcal{B}$ computes and sends $\{\alpha, \alpha^2, ..., \alpha^n\}$ and $\{\alpha', \alpha'^2, ..., \alpha'^n\}$, where $\alpha = g^s$ and $\alpha' = \alpha^r = g^{sr}$.
3. $\mathcal{A}$ randomly selects δ from field $\mathbb{F}$.
4. $\mathcal{A}$ computes and sends $u' = (g^p)^\delta$, $v' = (g^h)^\delta$, and $w' = (g^{p'})^\delta$.
5. $\mathcal{B}$ checks whether $u'^r = w'$.
6. $\mathcal{B}$ checks whether $u' = v'^t$.

By introducing the random value δ, the verifier can no longer learn anything about p, h, or w, thus achieving zero knowledge.

+ Non-interactivity The previously described protocol requires each verifier to generate unique random values, which becomes inefficient when a prover needs to demonstrate knowledge to multiple verifiers. To address this, we aim to eliminate the interaction between the prover and verifier. One effective solution is the use of a trusted setup.

Protocol 7: + Non-Interactive: Trusted Setup

1. **Secret Seed**: A trusted third party generates the random values s and r
2. **Proof Key**: Provided to the prover
   (a) $\{\alpha, \alpha^2, ..., \alpha^n\}$, where $\alpha = g^s$
   (b) $\{\alpha', \alpha'^2, ..., \alpha'^n\}$, where $\alpha' = g^{sr}$
3. **Verification Key**: Distributed to verifiers
   (a) g^r
   (b) $g^{T(s)}$
4. After distribution, the original s and r values are securely destroyed.

Then, the non-interactive protocol consists of two main parts: proof generation and verification.

Protocol 8: + Non-Interactive: Proof

1. $\mathcal{A}$ receives the proof key
2. $\mathcal{A}$ randomly selects δ from field $\mathbb{F}$.
3. $\mathcal{A}$ broadcast the proof $\pi = (u' = (g^p)^\delta, v' = (g^h)^\delta, w' = (g^{p'})^\delta)$

Since r is not shared and already destroyed, the verifier $\mathcal{B}$ cannot calculate u'^r to check $u'^r = w'$. Instead, the verifier can use a paring with bilinear mapping; $u'^r = w'$ is equivalent to $e(u' = (g^p)^\delta, g^r) = e(w' = (g^{p'})^\delta, g)$.

Protocol 9: + Non-Interactive: Verification

1. $\mathcal{B}$ receives the verification key.
2. $\mathcal{B}$ receives the proof π.
3. $\mathcal{B}$ checks whether $e(u', g^r) = e(w', g)$.
4. $\mathcal{B}$ checks whether $u' = v'^t$.

3 Basic of zk-STARKs

4 Circom: Domestic Specific Language for ZKP

References