1. Introduction:

- Applications of Zero-knowledge proofs:
- Proving statement on private data;
- Anonymous authorization;
- Anonymous payments;
- Outsourcing computation.

- In any zero-knowledge proof system, a protocol should satisfy three properties:
- Completeness -- if the statement is true, then a prover can convince a verifier.
- Soundness -- a cheating prover can not convince a verifier of a false statement.
- Zero-knowledge -- the interaction only reveals if a statement is true and nothing else.

2. The Medium of a Proof

- Prove that:

$$b = [\boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}, \boxed{?}]$$

Without worrying about the zero-knowledge, non-interactivity, its form and applicability.

Method: reading elements in some arbitrary order and checking if it is truly equal to 1

- Polynomials:
- An advantageous property: If we have two non-equal polynomials of degree at most d, they can intersect at no more than d points.
- It is impossible to find two different non-equal polynomials, which share a consecutive chunk of a curve.
- If we want to find intersections of two polynomials, we need to equate them.
- The result of any such equation for arbitrary degree d polynomials is always another polynomial od degree at most d, since there is no multiplication to produce higher degrees.

- Four steps to verify a statement:
- **Verifier chooses a random value for x and evaluates his polynomial locally.**
- **Verifier gives x to the prover and asks to evaluate the polynomial in question**
- **Prover evaluates his polynomial at x and gives the result to the verifier**
- **Verifier checks if the local result is equal to the prover's result, and if so then the statement is proven with a high confidence**

3.1. Proving Knowledge of a Polynomial:

- A polynomial can be expressed in the form:

$$c_n x^n + \ldots + c_1 x^1 + c_0 x^0$$

3.2. Factorization

- Any polynomial can be factored into linear polynomials, as long as it is solvable.
- P(x)= target(x)·**h(x)** (h(x)=some arbitrary polynomial)
- In other words, there exists some polynomial h(x) which makes t(x) equal to p(x), therefore p(x) contains t(x), consequently p(x) has all roots of t(x), the very thing to be proven.
- Three Steps to check whether p(x) and t(x) are identical

- Verifier samples a random value r, calculates $t = t(r)$ (i.e., evaluates) and gives r to the prover
- Prover calculates $h(x) = \frac{p(x)}{t(x)}$ and evaluates $p(r)$ and $h(r)$; the resulting values p, h are provided to the verifier
- Verifier then checks that $p = t \cdot h$, if so those polynomials are equal, meaning that $p(x)$ has $t(x)$ as a cofactor.

3.3. Obscure Evaluation

3.3.1 Homomorphic Encryption

- It allows to encrypt a value and be able to apply arithmetic operations on such encryption.

3.3.2 Modular Arithmetic

- Instead of having an infinite set of numbers we declare that we select only first n natural numbers, i.e., 0,1,…., n-1 to work with, and if any given integer falls out of this range, we wrap it around.
- The modulo operation will keep it in certain bounds.
- Many different combinations will have the same result.

3.3.3 Strong Homomorphic Encryption

- Let us explicitly state the encryption function: $E(v) = g^v \pmod{n}$, where v is the value we want to encrypt.

3.3.4 Encrypted Polynomial

-

- - Verifier
    - samples a random value s, i.e., secret
    - calculates encryptions of s for all powers i in $0, 1, ..., d$, i.e.: $E(s^i) = g^{s^i}$
    - evaluates unencrypted *target polynomial* with s: $t(s)$
    - encrypted powers of s are provided to the prover: $E(s^0), E(s^1), ..., E(s^d)$
  - Prover
    - calculates polynomial $h(x) = \frac{p(x)}{t(x)}$
    - using encrypted powers $g^{s^0}, g^{s^1}, \ldots, g^{s^d}$ and coefficients $c_0, c_1, \ldots, c_n$ evaluates $E\left(p(s)\right) = g^{p(s)} = \left(g^{s^d}\right)^{c_d} \cdots \left(g^{s^1}\right)^{c_1} \cdot \left(g^{s^0}\right)^{c_0}$ and similarly $E\left(h(s)\right) = g^{h(s)}$
    - the resulting g^p and g^h are provided to the verifier
  - Verifier
    - The last step for the verifier is to checks that $p = t(s) \cdot h$ in encrypted space: $g^p = \left(g^h\right)^{t(s)} \quad \Rightarrow \quad g^p = g^{t(s) \cdot h}$

3.4 Restricting a Polynomial (?)

- ; (i.e., $E\left(s^i\right)^{c_i} = g^{c_i \cdot s^i}$).
- We already restrict a prover in the selection of encrypted powers of s.
- Knowledge-of-Exponent Assumption (or KEA): acting as an arithmetic analogy of checksum, ensuring that the result is exponentiation of the original value.
- Alpha-shift

3.5 Zero-knowledge

$$g^p = \left(g^h\right)^{t(s)} \qquad \text{(polynomial } p(x) \text{ has roots of } t(x))$$

$$(g^p)^\alpha = g^{p'} \qquad \text{(polynomial of a correct form is used)}$$

- In order to extract the knowledge, one first needs to find delta which is considered infeasible and it is a random number

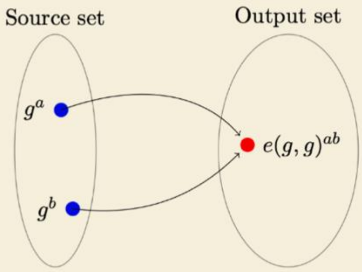
3.6. Non-Interactive

- Nobody else can trust the same proof:
- Collusion (verifier and proover)
- The verifier
- The verifier has to store alpha until all relevant proofs are verified
- We need the secret parameters to reusable, public, trustworthy, and infeasible to abuse.

3.6.1 Multiplication of Encrypted Values

-

Cryptographic pairings (bilinear map) is a mathematical construction, denoted as a function $e(g^*, g^*)$, which given two encrypted inputs (e.g., g^a, g^b) from one set of numbers allows to map them deterministically to their multiplied representation in a different output set of numbers, i.e., $e(g^a, g^b) = e(g, g)^{ab}$:

- 

- <u>It resembles a hash function, which maps all possible input values to an element in the set of possible output values and it is not trivially reversible.</u>
- Both swapped inputs are then multiplied together, such that raw a and b values get multiplied under the same exponent:

$$e(g^a, g^b) = a^{\mathbf{g}} \cdot b^{\mathbf{g}} = (ab)^{\mathbf{g}}$$

3.6.2 Trusted Party Setup

- Common Reference String (CRS):

CRS is divided into two groups (for I in 0, 1, ….., d):

Moreover CRS is divided into two groups (for i in $0, 1, \ldots, d$):

- Proving key[17]: $(g^{s^i}, g^{\alpha s^i})$
- Verification key: $(g^{t(s)}, g^{\alpha})$

3.6.3 Trusting One out of Many (TBC)

- It is necessary to minimize or eliminate the trust.

- - Alice samples her random s_A and α_A and publishes her CRS:
    $\left(g^{s_A^i}, g^{\alpha_A}, g^{\alpha_A s_A^i}\right)$
  - Bob samples his s_B and α_B and augments Alice's encrypted CRS through homomorphic multiplication:
    $\left(\left(g^{s_A^i}\right)^{s_B^i}, \left(g^{\alpha_A}\right)^{\alpha_B}, \left(g^{\alpha_A s_A^i}\right)^{\alpha_B s_B^i}\right) = \left(g^{(s_A s_B)^i}, g^{\alpha_A \alpha_B}, g^{\alpha_A \alpha_B (s_A s_B)^i}\right)$
    and publishes the resulting two-party Alice-Bob CRS:
    $\left(g^{s_{\mathsf{AB}}^i}, g^{\alpha_{\mathsf{AB}}}, g^{\alpha_{\mathsf{AB}}\, s_{\mathsf{AB}}^i}\right)$
  - So does Carol with her s_C and α_C:
    $\left(\left(g^{s_{\mathsf{AB}}^i}\right)^{s_C^i}, \left(g^{\alpha_{\mathsf{AB}}}\right)^{\alpha_C}, \left(g^{\alpha_{\mathsf{AB}}\, s_{\mathsf{AB}}^i}\right)^{\alpha_C s_C^i}\right) = \left(g^{(s_A s_B s_C)^i}, g^{\alpha_A \alpha_B \alpha_C}, g^{\alpha_A \alpha_B \alpha_C (s_A s_B s_C)^i}\right)$
    and publishes Alice-Bob-Carol CRS:
    $\left(g^{s_{\mathsf{ABC}}^i}, g^{\alpha_{\mathsf{ABC}}}, g^{\alpha_{\mathsf{ABC}}\, s_{\mathsf{ABC}}^i}\right)$
- This process can be repeated for as many participants as necessary.

3.7 Succinct Non-Interactive Argument of Knowledge of Polynomial

(TBC Refer to Paper)

- Setup
- Proving
- Verification