BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **25**, No 1
Sofia • 2025 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2025-0001

ZK-STARK: Mathematical Foundations and Applications in Blockchain Supply Chain Privacy

Madhuri S. Arade, Nitin N. Pise

Department of Computer Engineering & Technology, Dr. Vishwanath Karad, MIT World Peace University, Pune, India
E-mails: arademadhuri15@gmail.com nitin.pise@mitwpu.edu.in

***Abstract*:** *Privacy is one of the major security concerns. The zero-knowledge proof enables the transmission of data from the sender to the receiver without disclosing the actual content of the data. The proposed work uses the ZK-STARK* (*Zero-Knowledge Scalable Transparent ARgument of Knowledge*) *Algorithm for transaction privacy in the organic jaggery supply chain. The paper emphasizes a detailed mathematical model, involving two key participants: the prover* (*food processor*) *and the verifier* (*distributor*). *The prover calculates the polynomial for the problem, its composition polynomial, and provides its Merkle proof to the verifier. The verifier conducts queries to confirm and validate the accuracy of the information. Using the fast reed-solomon interactive oracle proofs protocol, the proof is validated. It measures performance as proof generation and verification time, proof size, and throughput. Plans involve increasing the domain size of this algorithm, varying the polynomial interpolation, and evaluating its performance measures by integrating it into Blockchain.*

***Keywords*:** *Privacy preservation, Zero Knowledge Proof* (*ZKP*), *Zero-Knowledge Scalable Transparent ARgument of Knowledge* (*ZK-STARK*), *Polynomial interpolation, Blockchain.*

1. Introduction

Privacy preservation plays a vital role in security, especially during data transactions between a sender and receiver, when multiple parties are often involved. For example, in blockchain applications, data exchanges involve several entities, raising concerns about maintaining privacy between the sender (prover) and receiver (verifier). To safeguard privacy, various cryptographic algorithms are used, with one of the popular protocols being zero-knowledge proof. This protocol allows data to be transmitted from the sender to the receiver without revealing the content of the data itself.

This paper focuses on Zero Knowledge Proof (ZKP) based ZK-STARK((Zero-Knowledge Scalable Transparent ARgument of Knowledge) protocol (B e r e n t s e n, J e r e m i a s and R e m o [1], B e n-S a s s o n et al. [2], B e n-S a s s o n et al. [3]) with all its detail mathematical expressions, evaluation, proving and verification steps. ZK-STARK is popular for its scalability and transparency, which enable efficient handling of large transactions, such as in blockchain applications. Its scalability supports high transaction volumes, while its transparency requires no trusted setup, which enhances reliability. Since ZK-STARKs do not need a trusted setup, they do not depend on any pre-existing information to generate keys or data for the algorithm, which increases their resistance to quantum attacks. Additionally, using the SHA-256 hash function to generate Merkle proofs, ZK-STARKs provide further protection against quantum threats. These features make ZK-STARK more effective than other zero-knowledge-proof techniques.

This paper is structured as follows: Section 2 discusses related work, while Section 3 details the proposed work. Section 4 covers the experiments and results, Section 5 presents the conclusion and Section 6 outlines the future work.

2. Literature survey

L i u, B., et al. [4] have constructed a system that integrates a Trusted Execution Environment (TEE) and a blockchain to execute the auction and protect sensitive inputs privately. The auction mechanism allows buyers and sellers to dynamically submit their bids/prices to exchange resources until convergence. This system addresses the privacy issue for the double auction by utilizing the TEE to protect and verify sensitive inputs.

Y a n, D e n g and S u n [5] have proposed a protocol aimed at providing strong security guarantees in scenarios where concurrent and non-malleable zero-knowledge proofs are required. The concept of non-malleability ensures that the information revealed in one instance does not compromise the security of other cases.

J a y a b a l a n, J e y a n t h i [6] discuss various aspects of blockchain-based healthcare systems, focusing on data security, access control, and scalability. Asymmetric encryption and digital signatures provide data security for shared EHR data. Attribute-based encryption and signatures provide fine-grained access control, and coarse-grained access control mechanisms ensure data security and privacy.

S u m a t h i et al. [7] propose a blockchain-based system allowing patients to securely control their Personal Medical Records (PMR) and share them with doctors. Huffman code data compression resolves the scalability issue, and the Advanced Encryption Standard (AES) achieves security.

As the name suggests, the ZKP protocol allows for the verification of data without possessing any knowledge of the secret data. The transaction involves two main entities: the prover and the verifier. The prover demonstrates to the verifier that the data being sent is correct without disclosing any additional information about the data.

There are three criteria that a ZKP [8] must satisfy.

1. Completeness. If the statement is true, an honest verifier will be convinced of its truth by the prover.

2. Soundness. If the statement is false, the prover cannot prove or convince the verifier of its truth.

3. Zero Knowledge. If the statement is true, the verifier will not gain any information other than the fact itself.

Yinjie G. et al. [8] present a paper on comparative analysis of zero-knowledge proof algorithms such as ZK-SNARK, ZK-STARK, MPC-based Algorithm, and BulletProof in different aspects such as trusted setup, proof length and complexity, application, and quantum threat. ZK-STARK doesn't require a trusted setup and provides security against post-quantum attacks.

Frimpong et al. [9] proposed a blockchain-based system that ensures user privacy through decentralized storage and control of user data. The system incorporates a Graph Convolutional Network (GCN) for detecting malicious nodes, ensuring the integrity of the network.

Chi, Lu and Guan [10] present a novel approach to enhancing privacy in blockchain applications through a new scheme called Blockchain Designated Verifier Proof (BDVP). This scheme builds upon the existing concepts of Zero-Knowledge Proof (ZKP) and Designated Verifier Proof (DVP) to address privacy concerns that arise when verifiers can inadvertently disclose information about provers.

Zhou, Lu, et al. [11] provide a critical evaluation of how ZKP technology can be leveraged within blockchain frameworks to create more secure and privacy-focused identity-sharing systems, addressing the limitations of traditional identity management approaches.

Feng et al. [12] proposed a blockchain-based data privacy protection and sharing scheme that addresses several issues in data sharing processes, especially within the context of the Industrial Internet of Things (IIoT) using zero-knowledge proof and proxy re-encryption algorithms.

Radeva and Popchev [13] aim to design a blockchain-enabled supply chain model specifically for smart crop production. It focuses on tasks such as analyzing the blockchain ecosystem, defining a supply chain model, designing a blockchain infrastructure, and detailing blockchain information channels with smart contracts.

Research gap. Current literature addresses privacy-preserving mechanisms using various cryptographic techniques and zero-knowledge proofs like ZK-SNARK. However, these techniques require trusted setups and are ineffective against quantum attacks. The proposed methodology addresses this gap by using the ZK-STARK Algorithm.

3. Proposed work

The primary emphasis of this research revolves around the mathematical implementation of the ZK-STARK algorithm (Berentsen, Jeremias and Remo [1], Barbara [14]) within a blockchain-oriented supply chain framework. It thoroughly outlines the step-by-step mathematical modeling process specific to the case study: the production of jaggery and its verification as an organic product. The approach ensures the validation of ingredients and methodologies to the verifier without disclosing the actual contents.

This research proposes to use the ZK-STARK zero-knowledge proof algorithm to uphold transaction privacy between the prover and verifier. In this specific scenario, the prover represents the food processor, while the verifier embodies the distributor.

Here’s how it works.

Organic jaggery production. The food processor starts by making organic jaggery from sugarcane, by using only organic components such as lime powder and lady’s finger and with an organic process.

Secure transaction. Then enclose the transaction details (like product information and organic certification) within the ZK-STARK protocol. This protects the actual details while still allowing verification.

Verification. The distributor receives the transaction and uses ZK-STARK to verify its validity without needing to see the specific contents. This ensures the jaggery is indeed organic without disclosing sensitive information.

Successful transaction. Once the distributor confirms the organic status through ZK-STARK, the transaction is completed successfully.

This ZK-STARK zero-knowledge proof involves many steps, the major three steps are as follows:

1. By converting this real-time example into a mathematical expression.
2. Arithmetization of the problem.
3. Low degree testing.

The detailed flow for the ZK-STARK algorithm is explained in Fig. 1.

ZK-STARK Algorithm

- The first prover will define the problem and its Computational Integrity (CI) Statement and fix the finite field to work within it.
- Once the problem is defined, polynomial $f(x)$ is created using Lagrange polynomial interpolation with a finite field defined.
- The constraint $c(x)$ is calculated using CI statement and $f(x)$.
- The composition polynomial $p(x)$ is formed by using the constraint and polynomial which will be used to create Merkle proof.
- For sending data to the verifier, the Merkle proof calculated for polynomial $f(x)$ as MT_f and composition polynomial $p(x)$ as MT_p.
- Low-degree testing will be performed with step number 7, 8, and 9 sequentially in Fig. 1.
- This proof will be sent to the verifier.
- The verifier queries for random number z, and then the prover runs FRI (Fast Reed soloman Interactive) Oracle protocol for low-degree testing.
- If the polynomial results in the low degree returning constant after $N - 2$ iterations, and both prover and verifier get the same constant, then PROOF IS CORRECT.

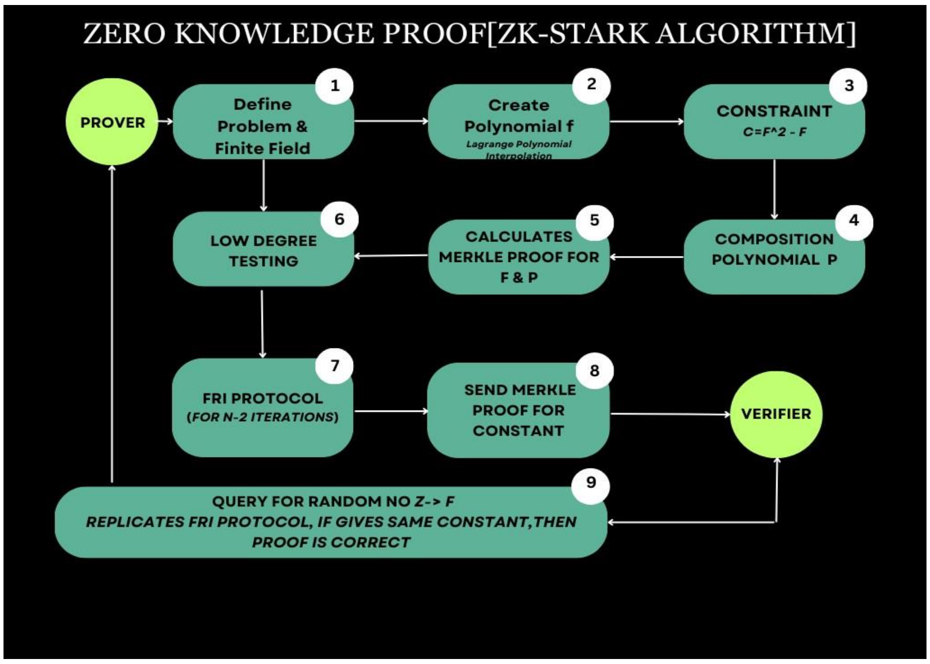
ZERO KNOWLEDGE PROOF[ZK-STARK ALGORITHM]
PROVER
1 Define Problem & Finite Field
2 Create Polynomial f Lagrange Polynomial Interpolation
3 CONSTRAINT C=F^2 - F
4 COMPOSITION POLYNOMIAL P
5 CALCULATES MERKLE PROOF FOR F & P
6 LOW DEGREE TESTING
7 FRI PROTOCOL (FOR N-2 ITERATIONS)
8 SEND MERKLE PROOF FOR CONSTANT
VERIFIER
9 QUERY FOR RANDOM NO Z-> F REPLICATES FRI PROTOCOL, IF GIVES SAME CONSTANT,THEN PROOF IS CORRECT

Fig. 1. ZK-STARK detail stepwise algorithm

3.1. Working of ZK-STARK algorithm: use case organic jaggery supply chain

The next part of the paper will discuss the detailed mathematical modeling of ZK-STARK algorithm for the use case as the prover has to prove that the jaggery produced is organic without revealing the content of the data transaction to the verifier.

3.1.1. Define the problem – CI Statement

CI Statement: The prover has a sequence J of N integers, all of which are 1 or 0 (boolean).

The statement is true if all elements $J_i \in J$ are either 1 or 0.

Where:

J is for jaggery produced;

N is for ingredients used for processing in the production of jaggery.

Initially, let’s establish a CI Statement that requires validation. CI denotes the accuracy of a specific computation’s result.

The prover must demonstrate the organic nature of the produced jaggery. If the prover is unable to substantiate this claim, the jaggery is deemed inorganic.

Thus, the CI Statement can be formulated as follows.

The goal is to prove that this statement is true with a sufficiently high probability without revealing *J* (i.e., elements processed for jaggery production), i.e., without revealing which element of *J* is 1 or any other. So, *J* is called the trace.

Assume that organic ingredients such as (lime powder, and ladyfinger) are set to 1.

Ingredients chemicals such as hydro sulphite powder and phosphoric acid are set to 0. (That is, it is not present. If present then it should be set to other than 0 (–1, 2)),

$$J_i(J_i - 1) = 0,$$
$$J_i^2 - J_i \quad = 0,$$

N1 – Lime Powder (Organic ingradient),
N2 – Lady Finger Powder (Organic ingradient),
N3 – Hydro sulphite powder (Chemicals ingradient),
N4 – Phosphoric acid (Chemicals ingradient).

N1 represents the prover and the verifier should agree on it if the statement is correct. For the organic jaggery supply chain, if the jaggery produced is organic, then only Equation (1) bellow is true.

The prover and verifier must agree on the conditions if the statement is true.

(1) $$J_i(J_i - 1) = 0, J_i^2 - J_i = 0,$$

where $i = 0, 1, 2, \dots, N - 1$.

Trace
In this use case example, fix *N* = 4, such that:
• J_{true} = [1, 1, 0, 0] satisfies the CI Statement,
• J_{false} = [1, 1, 2, 2] does not satisfy the CI Statement;
N1, N2 are organic ingredients,
N3, N4 are inorganic ingredients.

- Finite field

Before generating the proof, both the prover and verifier need to establish their working field. In cases involving finite fields, modular arithmetic is commonly employed. Visualizing it is easiest through a clock analogy. Consider operating within a prime field where *M* equals 17 in arithmetic modulo. This field comprises the set {0, 1, ..., 16}. If an operation yields a value larger than 16, it progresses in a clockwise manner, looping back to 0 to maintain the field's boundaries,

F= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].

In a finite field containing 17 elements, determining the multiplicative inverse of each non-zero element involves using modular arithmetic and the Extended Euclidean Algorithm.

Let's compute:

The multiplicative inverse of 1 remains as 1 because any number multiplied by 1 equals itself. For the other non-zero elements (from 2 to 16), their inverses can be found as follows:

[1, 9, 6, 13, 7, 3, 5, 15, 2, 12, 14, 10, 4, 11, 8, 16].

These are the multiplicative inverses [16] for the non-zero elements in the field of 17 elements. Note that the inverse of 0 does not exist in a field since division by zero is undefined.

The finite field then is

$$F = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].$$

For this set of values, multiplicative inverses are

$$[1, 9, 6, 13, 7, 3, 5, 15, 2, 12, 14, 10, 4, 11, 8, 16].$$

3.1.2. Arithmetization of the problem – by transforming into polynomial

- Evaluating trace into polynomial

The trace is the evaluation of some polynomial f, i.e., for some $X \in F$,

$$f(X_i) = J_i\ , i = 0, 1, \ldots, N - 1.$$

This states that the outputs of the polynomial f relate to the trace. To choose X, define a subgroup G of size N of the multiplicative group $F/\{0\}$, where $F/\{0\}$ is the finite field without the zero. Furthermore, define a generator g of G.

Since $N = 4$, the size of the subgroup $G \in F$ is 4. Find a $g \in F/\{0\}$ for which the powers of g uniquely yield the elements of G.

In this example, $g = 4$ and $G = \{1; 4; 13; 16\}$.

After finding a finite set for working with polynomials, next is the arithmetization. This requires to transform the problem into an algebraic problem using Lagrange polynomial interpolation [17].

Given $J = J_{\text{true}} = [1; 1; 0; 0]$, the prover needs to find a polynomial

$$f\colon G \to F,$$
$$f(g_0) = f(1) = J_0 = 1,$$
$$f(g_1) = f(4) = J_1 = 1,$$
$$f(g_2) = f(13) = J_2 = 0,$$
$$f(g_3) = f(16) = J_3 = 0.$$

Applying the Lagrange interpolation for the given set of points: (1, 1), (4, 1), (13, 0), and (16, 0).

The Lagrange interpolation formula for four points is given in the equation

(2) $$p(x) = \sum_{i=0}^{3} y_i\ .\ l_i(x),$$

where $p(x)$ is the Lagrange polynomial, y_i are the y-coordinates, and $l_i(x)$ are the Lagrange basis polynomials.

The Lagrange basis polynomial for each point is given by the equation:

(3) $$l_i(x) = \prod_{j=0}^{3}\ (x - x_i)\,/(x_i - x_j),\ j \neq i.$$

After solving Equation (1) by using the formula of Equation (2) and Equation (3), the polynomial $f(x)$ associated with the trace [1, 1, 0, 0] is presented in the equation

(4) $$f(x) = -3x^3 - 5x - 8.$$

This equation represents the polynomial $f(x)$ generated for the organic jaggery supply chain. Using this polynomial, the constraint polynomial is created, and a Merkle proof is prepared to be sent to the verifier.

- Working with polynomials on a larger domain.

The next step is to extend polynomial f to a larger domain. To do that, evaluate polynomial f not only in G but on a larger domain L, where $G \in L = F/\{0\}$ such that $f: L \rightarrow F$.

In this example, choose this larger domain to be equal to the multiplicative group of the finite field, i.e., $L = F/\{0\}$.

The multiplicative for the Finite field of 17 elements as the next equation is including all elements except 0,

(5) $$F = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}.$$

For this equation, set values, and multiplicative inverses are calculated in the equation

(6) $$[1, 9, 6, 13, 7, 3, 5, 15, 2, 12, 14, 10, 4, 11, 8, 16].$$

For this equation, large domains including all points the scatter plot graph in Fig. 2 denotes points in the space,

$$x = [1, 9, 6, 13, 7, 3, 5, 15, 2, 12, 14, 10, 4, 11, 8, 16],$$

$$y = [-16, -2240, -686, -6664, -1072, -104, -408, -10208, -42, -5252, -8310, -3058, -220, -4056, -1584, -12376].$$

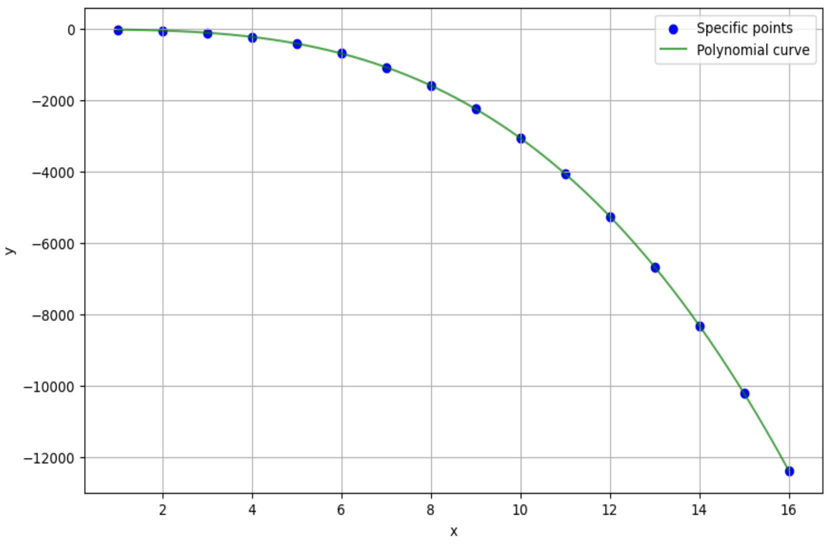

Fig. 2. Scatter plot for polynomial $-3x^3 - 5x - 8$

- Calculate constraint for polynomial

Use the constraint to trace the polynomial containing 1 or 0 if the CI Statement is true.

$$j_i^2 - j_i, \text{ where } i = 0, 1, \ldots, N-1,$$

and define this expression as the constraint polynomial $c: L \rightarrow F$.

The constraint polynomial makes it computationally infeasible for a malicious prover to cheat. If the prover submits an invalid computation, the constraint polynomial will fail to evaluate to zero at specific points, revealing inconsistencies. Verifiers use techniques like low-degree testing to ensure that the constraint polynomial follows the expected rules, for more detail on the role of $c(x)$ (see papers [1, 2]),

(7) $$c(x) = f(x)^2 - f(x) = 0.$$

By putting Equation (4) $f(x) = -3x^3 - 5x - 8$ in the Equation (7) constraint *c*(*x*), it gives,

$$c(x) = (-3x^3 - 5x - 8\,)^2 - (-3x^3 - 5x - 8\,).$$

In the next equation, *c*(*x*) represents the constraint polynomial, which is useful in verifying the accuracy of data present in the supply chain,

(8) $$\mathrm{c}(x) = -8x^6 - 4x^4 + 8x^2 + 4.$$

- Create the composition polynomial

The prover transforms *c*(*x*) using the properties of roots of polynomials to get another composition polynomial *p*(*x*) to add an extra layer of security. By transforming *c*(*x*) into a composition polynomial *p*(*x*), the prover ensures that the verifier does not directly interact with *c*(*x*) itself. Instead, the verifier interacts with a higher-level abstraction that encodes multiple checks. This reduces the risk of revealing information about *c*(*x*). The composition polynomial's additional layer ensures that any deviation by the prover (such as a fake *c*(*x*)) is caught with high probability, making the system both sound and secure. For more details see the paper (K a t e, Z a v e r u c h a and G o l d b e r g [18]).

The next equation presents a formula for the composition polynomial *p*(*x*) using constraint polynomial *c*(*x*),

(9) $$p(x) = \frac{c(x)}{\prod_{i=0}^{\mathrm{N}-1}(x-g^i)\quad \forall x \in F \text{ and } x \notin \{g^0, g^1, g^2, \ldots,\ g^{N-1}\}}.$$

Using this equation, after simplifying, get *p*(*x*) as in the equation

(10) $$p(x) = \frac{c(x)}{u(x)} = (f(x)^2 - f(x))/(x^N - 1\,),$$

where $\forall x \in F$ and $x \notin \{g^0, g^1, g^2, \ldots, g^{N-1}\}$.

By using Equation (8) for *c*(*x*), the Equation (10) solved for *p*(*x*):

$$p(x) = \frac{-8x^6 - 4x^4 + 8x^2 + 4}{x^4 - 1}.$$

After solving, Equation (11) gives a composition polynomial, which is used to generate Merkle proof using the SHA-256 hash function. for more detail of role of *p*(*x*) (see papers [1, 2]):

(11) $$p(x) = -8x^2 - 4.$$

For this equation, the upper bound of the degree on *p*(*x*), i.e.,

$$\deg(p) = N - 2 = 2.$$

- Commitments using Merkle proof

After calculating *p*(*x*) and *f*(*x*), the next step is to calculate the Merkle tree (B e c k e r [19], Wikipedia [20]) root proof one for *p*(*x*) and *f*(*x*) for sending to the verifier.

This algorithm construct Merkle tree and create Merkle root proof MT_f for $f(x) = -3x^3 - 5x - 8$ and construct Merkle tree and create Merkle root proof MT_p, $p(x) = -8x^2 - 4$.

After creating the Merkle tree and calculating the Merkle proof, the hash function SHA-256, the Merkle proof calculated for each node, and then the Merkle root proof. It gives Merkle root which will be used to send the verifier.

Prover sends Merkle root of MT_f to verifier:

601e615a178c275f7df2be8bdb2b4d84deafb2dc45956b2e12217a550ad15a7b.

The leaves of MT_f are: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Prover sends Merkle root of MT_p to verifier:

c14764c1e39b84fc72f9a00cadbd71a8c601cf3b8942afead0fda678fe831e6c.

The leaves of MT_p are: [5, 15, 9, 4, 0, 14, 12, 11, 11, 12, 14, 0, 4, 9, 15, 5].

After calculating the Merkle trees, the prover sends the Merkle roots of the two trees to the verifier. For the organic jaggery supply chain, the food processor calculates the Merkle proof using a polynomial and sends this proof to the distributor. This eliminates the problem that the prover might respond with random values because the verifier can check whether these values are the commitment (i.e., the Merkle roots).

3.1.3. Low-degree testing

- Querying

After receiving Merkle root values of *p*(*x*) from the prover(food processor), the next task is to verify these values of *p*(*x*) by the verifier(distributor), by verifying it is of low degree. For this need to ensure that the low-degree polynomial *p*(*x*) relates to polynomial constraint *c*(*x*).

The equation of *c*(*x*) can be written as given in the next equation

$$p(x)u(x) = c(x),$$

$$p(x)(x^N - 1) = f(x)^2 - f(x). \tag{12}$$

The verifier sends a random *x*-value and checks for the *p*(*x*) and *f*(*x*). If the prover is honest then the Equation (12) holds for all possible values of *x*.

Query 1:

Verifier queries: *z* = 3.

Prover sends: *f*(3) = 15 and *p*(3) = 9, the corresponding Merkle Paths for *f*(*z*):

(i) Verifier now checks using equation (12),

$$p(z)(z^N - 1) = f(z)^2 - f(z),$$

LHS yields: 6; RHS yields: 6. Equation correct, proof continues.

(ii) Verifier now checks whether received *p*(*z*) and *f*(*z*) correspond to Merkle root received.

PROOF FAILS: *f*(*z*) DOES NOT correspond to commitment.

The proposed work checked whether the polynomial is correct or not, additionally, the ZK-STARK algorithm needs to check whether the polynomial is of a low degree. If it is of low degree then the data received is correct.

As the malicious prover can not cheat, what if the *p*(*x*) and *f*(*x*) are of a high degree and satisfy the next equation? Berentsen, Jeremias and Remo [1] have given proof that if both *p*(*x*) and *f*(*x*) are high degrees, then they can satisfy next equation, and malicious data sent by the prover will be treated as correct. So, it is important to check the low-degree testing.

$$\underbrace{p(x)(x^N - 1)}_{c_1(x)} = \underbrace{f(x)^2 - f(x)}_{c_2(x)}. \tag{13}$$

Low-degree testing ensures security by verifying the degree of the polynomial. The formula used to determine the degree is $d{=}N{-}2$, where N represents the number of elements (or "ingredients") in the given context. If the value of N is incorrect, the resulting degree will also be incorrect. This calculated degree, $d{=}N{-}2$, is then used to validate consistency over $N{-}2$ iterations. During the verification process, the verifier provides a random value to check the claim. This random value is substituted into the polynomial, and the function is evaluated across $N{-}2$ iterations to determine whether it consistently produces a constant result. If a constant value is returned, the proof is valid; otherwise, it indicates an error or fraud. For instance, if the N value is incorrect or the polynomial generated is fake, the low-degree test will fail to produce a constant output, thus exposing the inconsistency. This process ensures the accuracy and originality of the information, playing a critical role in maintaining the security of the ZK-STARK algorithm. Low-degree testing is therefore a fundamental step in verifying the correctness of the proof and preserving the integrity of the protocol.

For this research, a degree can be at most as per the next equation. The degree of polynomial is less than equal to $N – 2$. N value is 4,

(14) $$\deg(p) \leq N - 2 = d.$$

The verifier's ability to confirm, after querying certain steps, ultimately arriving at a constant involves proving that the polynomial possesses a low degree d through the application of the FRI protocol (Fast Reed-Solomon Interactive Oracle Proof of Proximity) [21, 22]. FRI utilizes specific mathematical calculations and iterations to validate that the polynomial confirms the desired low-degree criteria, leading to identifiable constants.

The correlation between the number of iterations and the polynomial's degree is important. If the count of iterations doesn't surpass the degree d of the mathematical steps and consistently leads to the same constant for both the prover and verifier, it strongly suggests that the polynomial possesses a low degree, resulting in a SUCCESSFUL PROOF.

Here, as from the Equation (14), as N=4 ($N \rightarrow$ Number of ingredients),

$$\text{Degree } d = N – 2 = 4 – 2 = 2.$$

So there will be two iterations, after the second iteration constant value will be gained. Let's first calculate $p_1(x)$ and $p_2(x)$ and the constant value for prover,

$$p(x) = -8x^2 - 4,$$

$p(x)$ only includes an even part, no odd part in the Equation (11) so,

$$g_0(x) = -8x - 4,$$
$$h_0(x) = 0.$$

Assume, $\alpha_0 = 10$,

$$p_1(x) = g_0(x) + \alpha_0 h_0(x) = -8x - 4 + 10 \times 0 = -8x - 4.$$

For the second iteration,

$$p_1(x) = -8x - 4,$$
$$p_1(x) = -4 - x \times 8.$$

Here, $g_1(x) = -4$ and $h_1(x) = -8$.

Assume the verifier sends $\alpha_1 = 15$ the prover will calculate,

$$p_2(x) = g_1(x) + \alpha_1 h_1(x) = -4 + 15 \times (-8) = -124 = -124 \bmod 17 =$$
$$= -5 = -5 \text{ equals to } 12 = 12.$$

Prover sends 12 and it's Merkle proof to the verifier.

Merkle root is

['c7fddb6397f7a194b8d6c0ae8f47a1fc666e27a5cfec1fdb6779850899543bf7']

The verifier now calculates for some queries and results into constant.

Assume that the verifier queries for $z = 12$.

The prover then sends $p(12) = -8x^2 - 4$,

$$p(12) = -8(12)^2 - 4 = -1156 \bmod 17 = 0,$$

(As (-1156 mod 17), the division result is 68, the remainder is 0).

Now calculate for $z = -12$,

$$p(-12) \to p(5),$$

$$p(5) = -8(5)^2 - 4 = -204 \bmod 17 = 0,$$

(-204 mod 17, as the division result is 12, the remainder is 0).

The prover then sends $p(12) = 0$ & $p(-12)= p(5) = 0$ and the verifier solves

$$0 = g_0(z^2) + 12h_0(z^2),$$

$$0 = g_0(z^2) - 12h_0(z^2),$$

which gives $g_0(z^2) = 0$ & $h_0(z^2) = 0$.

Now calculates,

$$p_1(z^2) = g_0(z^2) + \alpha_0 h_0(z^2) = 0 + 10 \times 0 = 0.$$

The verifier checks whether calculated p_1=0 corresponds to commitment in the Merkle tree. Prover sends authentification path for leaf number 3, where $z = 8$ and p_1(8)=0 this is

['c47401d026e2cde77e3a3621fcefd9efa516c6d6518761a030fff79d58132f19',
'7247174a760a3a6b31b66052cceaa81a35220be4fadbd847028e1764ad30c100',
'e629fa6598d732768f7c726b4b621285f9c3b85303900aa912017db7617d8bdb'],

and uses this to calculate the Merkle Root p_1=0 corresponds to commitment, proof can continue.

Round 2 of 2

The prover returns

$p(12^2) = 0$ and $p(-12^2) = 9$, $0 = g_1(z^4) + 8h_1(z^4)$, $9 = g_1(z^4) - 8h_1(z^4)$, which gives

$$g(z^4) = \frac{9}{2} = 4 \bmod 17 = 13,$$

$$h(z^4) = \frac{9}{16} \bmod 17 = 9.$$

The verifier finds $g(z^4) = 13$ and $h(z^4) = 9$, which yields $g_1(z^4) = 9$ and $h_1(z^4) = 1$. Again the verifier then calculates

$$p_2(z^4) = g_1(z^4) + \alpha_1 h_1(z^4) = 13 + 15 \times 9 = 148 \bmod 17 = 12,$$

and then calculates p_2 = $g(z^4)$ + alpha_2 $h(z^4) = 13 + 15 \times 9 = 12$.

Verifier finds constant 12 which equals 12 received from prover PROOF SUCCESSFUL!

This proves that food processor data is organic jaggery only and verified by the distributor without revealing the enclosed content.

4. Experiments and result discussion

For the experiment, this research used Python language to implement mathematical modeling of the ZK-STARK algorithm on the Google Colab platform with a laptop configuration of 8 GB RAM, and a core i5 processor. The depth of mathematical processing starts from converting the real-time problem into Lagrange interpolation, calculating constraint and compositional polynomial with Merkle SHA 256 proof that will be sent to the verifier and verification by the verifier to the proof is implemented [23, 24]. Benefits of ZK-STARK over ZK-SNARK is that it doesn't require the trusted setup and defending against quantum attack (Fernández-Caramès and Fraga-Lamas [25]).

Different parameters are considered for assessing the performance of the algorithm.

Proof generation time. The time it takes to create a ZK-STARK proof, measured in seconds or milliseconds.

Proof verification time. The time it takes to verify a ZK-STARK proof, is typically much faster than generation.

Proof size. The size of the generated proof in bytes, affects storage and transmission overhead.

Throughput. It can refer to the number of operations completed in a specific period.

The performance of the ZK-STARK algorithm experiment with the given parameters is detailed in Table 1. Fig. 3 graph illustrates the ZK-STARK performance as presented in Table 1. Fig. 4 is a graph depicting the throughput data from Table 1. Table 1 provides data on various parameters, including proof generation time, proof verification time, proof size, and throughput, tested for domain sizes of 16 and 256.

Upon analyzing the parameters of Table 1, it's observed that the proof generation time and proof verification time are almost identical, showing little variation. The throughput also remains consistent across both domain sizes. This proves that the ZK-STARK algorithm operates efficiently even when the domain size is increased from 16 to 256.

Proof Size is fixed, making ZK-STARKs scalable regardless of the computational size. Constant proof size and slight throughput improvements with increasing domain size demonstrate ZK-STARK's scalability for larger computations.

The ZK-STARK algorithm is scalable, but in the real-world scenario for example in the case of blockchain supply, as data size increases, the Merkle hash proof size increases and that may create overhead on the system. The solution is to compress proof size or keep the computation of ZK-STARK off-chain and the verification part can be on on-chain.

Table 1. ZK-STARK algorithm performance parameters measures

Parameters for performance	Value (Domain size 16)	Domain size 256
Proof generation time	0.0477	0.0244
Proof verification time	0.0277	0.0535
Proof size	0.512	0.512
Throughput(Op/Second)	12.41	12.85

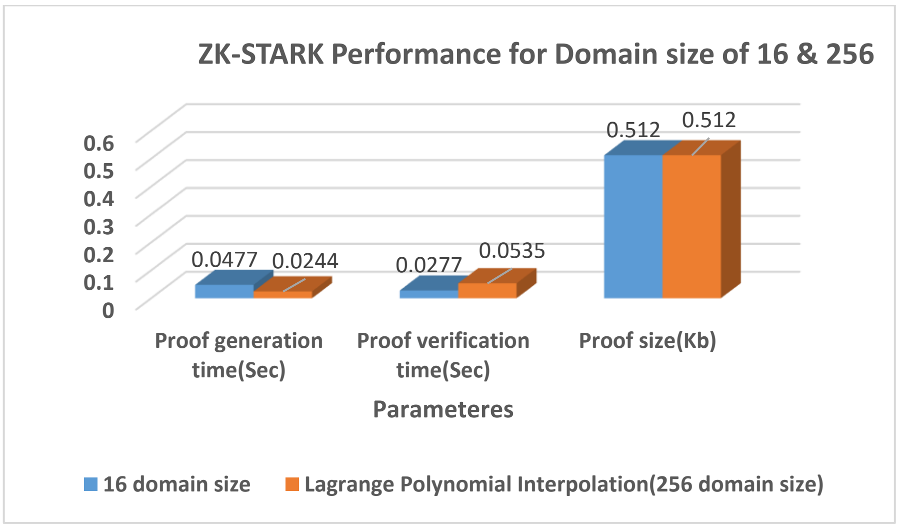

Fig. 3. ZK-STARK Algorithm performance parameter measures

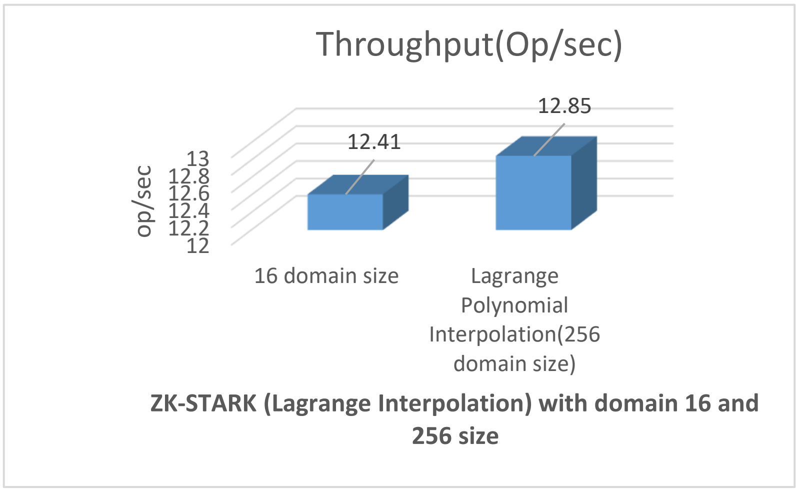

Fig. 4. ZK-STARK Algorithm – Throughput (op/second)

5. Conclusion

This paper concentrated on employing the ZK-STARK zero-knowledge proof algorithm for in-depth mathematical problem-solving within the context of the organic jaggery production supply chain. The system operates with two key roles: the prover and verifier. The prover's task involves concealing real-time information regarding the technique used in organic jaggery production and presenting proofs to the verifier without disclosing the content. This research paper outlines the mathematical procedures involved in transmitting data from the prover using Merkle proofs, which the verifier subsequently authenticated through specific queries. This research thoroughly illustrates these mathematical procedural steps. Additionally, it evaluated the performance metrics of the ZK-STARK algorithm, including proof generation time, verification time, proof size, and throughput. Notably, a major advantage of this algorithm is its elimination of the need for a trusted setup, unlike ZK-SNARK. Leveraging ZK-STARK ensures privacy in applications like blockchain-based supply chains and holds potential for various applications within blockchain networks.

6. Future work

This paper implemented the ZK-STARK algorithm with all its mathematical proofs as an independent entity, the next objective involves integrating this algorithm into the blockchain food supply chain. This integration will facilitate a comparative analysis of security performance between the enhanced blockchain supply chain with ZK-STARK and the standard blockchain model. In the future, the current finite domain used in the first step of the ZK-STARK algorithm can be expanded to more than a size of 256. This expansion involves working with larger datasets to evaluate the performance. There is scope to explore alterations in certain mathematical procedures, such as experimenting with different polynomial interpolation methods other than Lagrange Interpolation. Furthermore, variations in the employment of Merkle proofs using diverse SHA types are under consideration. The future research goal here is to use the ZK-STARK algorithm, aiming for a lighter version making it more suitable for different types of blockchain supply chains beyond food such as in pharmaceutical or electronics, and testing its performance.

References

1. B e r e n t s e n, A., L. J e r e m i a s, N. R e m o. A Walk-through of a Simple ZK-STARK Proof. 21.12.2022. **https://ssrn.com/abstract=4308637 or http://dx.doi.org/10.2139/ssrn.4308637**
2. B e n-S a s s o n, E., et al. Scalable, Transparent, and Post-Quantum Secure Computational Integrity. – Cryptology ePrint Archive, 2018.
3. B e n-S a s s o n, E., I. B e n t o v, Y. H o r e s h, M. R i a b z e v. Scalable Zero Knowledge with No Trusted Setup. – In: A. Boldyreva, D. Micciancio, Eds. Advances in Cryptology – CRYPTO 2019. CRYPTO 2019. – Lecture Notes in Computer Science, Vol. **11694**, 2019, Springer, Cham. DOI: 10.1007/978-3-030-26954-8_23.
4. L i u, B., S. X i e, Y. Y a n g, et al. Privacy-Preserving Divisible Double Auction with a Hybridized TEE-Blockchain System. – Cybersecurity, Vol. **4**, 2021, No 37. DOI: 10.1186/s42400-021-00100-x.
5. Y a n, Z., Y. D e n g, Y. S u n. Concurrent Non-Malleable Zero-Knowledge and Simultaneous Resettable Non-Malleable Zero-Knowledge in Constant Rounds. – Cybersecur. Vol. **1**, 2018, No 12. DOI: 10.1186/s42400-018-0014-7.
6. J a y a b a l a n, J., N. J e y a n t h i. A Review of State-of-Art Blockchain Schemes for Electronic Health Records Management. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 1, pp. 35-63.
7. S u m a t h i, M., S. R a j a, V. N a t a r a j a n, R. M u r u g e s a n. A Decentralized Medical Network for Maintaining Patient Records Using Blockchain Technology. – Cybernetics and Information Technologies, Vol. **22**, 2022, No 4, pp. 129-141.
8. G o n g, Y., Y. J i n, Y. L i, Z. L i u, Z. Z h u. Analysis and Comparison of the Main Zero-Knowledge Proof Scheme. – 2022 International Conference on Big Data, Information and Computer Network (BDICN'22), Sanya, China, 2022, pp. 366-372. DOI: 10.1109/BDICN55575.2022.00074.

9. F r i m p o n g, S., A. M. H a n, E. K. B o a h e n, R. N. A. S o s u, I. H a n s o n, O. L a r b i-S i a w, I. B. S e n k y i r e. RecGuard: An Efficient Privacy Preservation Blockchain-Based System for Online Social Network Users. – Blockchain: Research and Applications, Vol. **4**, 2023, Issue 1, 100111. ISSN: 2096-7209. DOI: 10.1016/j.bcra.2022.100111.

10. C h i, P. -W., Y. -H. L u, A. G u a n. A Privacy-Preserving Zero-Knowledge Proof for Blockchain. – In: IEEE Access, Vol. **11**, 2023, pp. 85108-85117. DOI: 10.1109/ACCESS.2023.3302691.

11. Z h o u, L u, A. D i r o, A. S a i n i, S. K a i s a r, P. C. H i e p. Leveraging Zero-Knowledge Proofs for Blockchain-Based Identity Sharing: A Survey of Advancements, Challenges, and Opportunities. – Journal of Information Security and Applications, Vol. **80**, 2024, 103678. ISSN 2214-2126, DOI: 10.1016/j.jisa.2023.103678.

12. F e n g, T., P. Y a n g, C. L i u, J. F a n g, R. M a. Blockchain Data Privacy Protection and Sharing Scheme Based on Zero-Knowledge Proof. – Wireless Communications and Mobile Computing, 2022, 1040662, 2022, 11 p. DOI: 10.1155/2022/1040662.

13. R a d e v a, I., I. P o p c h e v. Blockchain-Enabled Supply-Chain in Crop Production Framework. – Cybernetics and Information Technologies, Vol. **22**, 2022, No 1, pp. 151-170.

14. B a r b a r a, M. A. Proof of All: Verifiable Computation in a Nutshell. – arXiv Preprint arXiv:1908.02327, 2019, 14.

15. N a k a m o t o, S. Bitcoin: A Peer-to-Peer Electronic Cash System. – Decentralized Business Review, 2008, 21260.

16. An article on ‘Modular Multiplicative Inverse’.
https://www.extendedeuclideanalgorithm.com/multiplicative_inverse.php#owmultinv

17. M. Abramowitz, Stegun, Irene Ann, Eds. (1983) June 1964. Chapter 25, eqn 25.2.3. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. – Applied Mathematics Series, Vol. **55** (9th reprint with additional corrections of 10th original printing with corrections (December 1972), First Ed.). Washington DC.; New York: United States Department of Commerce, National Bureau of Standards; Dover Publications. 878 p. ISBN: 978-0-486-61272-0. LCCN: 64-60036. MR: 0167642. LCCN: 65-12253.

18. K a t e, A., G. M. Z a v e r u c h a, I. G o l d b e r g. Constant-Size Commitments to Polynomials and Their Applications. – In: M. Abe, Ed. Advances in Cryptology – ASIACRYPT 2010. ASIACRYPT 2010. Lecture Notes in Computer Science, Vol. **6477**. 2010. Berlin, Heidelberg, Springer. DOI: 10.1007/978-3-642-17373-8_11.

19. B e c k e r, G. Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis (PDF). Ruhr-Universität Bochum. 18. 7. 2008, p. 16. Archived from the original (PDF) on 22.12.2014. Retrieved 20.11.2013.

20. Merkle Tree. Available online on Wikipedia.
https://en.wikipedia.org/wiki/Merkle_tree

21. B e n-S a s s o n, E., I. B e n t o v, Y. H o r e s h, M. R i a b z e v. Licensed under Creative Commons License CC-BY 45th International Colloquium on Automata, Languages, and Programming (ICALP’2018), 2018. – In: I. Chatzigiannakis, C. Kaklamanis, D. Marx, D. Sannella, Eds. Article No 14, pp. 1-14.

22. B e n-S a s s o n, E., I. B e n t o v, Y. H o r e s h, M. R i a b z e v. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. Electron. Colloquium Comput. Complex., TR17. 2017.

23. B e n-S a s s o n, E. StarkWare – Productizing zk-STARKs to Provide Blockchain Scalability and Privacy.
https://epicenter.tv/episodes/310/

24. B e n-S a s s o n, E. Scaling Computation on Blockchains with ZK-STARK Invited Talk by Eli Ben-Sasson.
https://www.youtube.com/watch?v=8vduDYBu8uQ&list=PLeeS-3Ml-rpoVMNQkUrFDSfaTuUMxVtjy

25. F e r n á n d e z-C a r a m è s, T. M., P. F r a g a-L a m a s. Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks. – In: IEEE Access, Vol. **8**, 2020, pp. 21091-21116. DOI: 10.1109/ACCESS.2020.2968985.

Received: 28.09.2024, First Revision: 07.10.2024, Second Revision: 20.11.2024, Accepted: 25.11.2024