ZK-STARKs explained

Oleg Fomenko
Distributed Lab

July 2024

1 Introduction

ZK-STARK (Zero-Knowledge Scalable Transparent Argument of Knowledge) is a cryptographic proof system that allows one party to prove to another the knowledge of a piece of information without revealing the information itself, while ensuring scalability and transparency. Unlike ZK-SNARKs, STARKs do not require a trusted setup, which enhances their security and trustworthiness. ZK-STARK protocol utilizes advanced mathematical techniques like Fast Reed-Solomon IOP of Proximity and Merkle trees to achieve this. The security of STARK lies on the difficulty of computing the inverse function to the hash function, so we can consider STARK as a quantum-safe protocol.

In this paper, we describe the basic principles of the STARK protocol by building a proof of a simple statement (knowledge of Fibonacci square sequence element) as an example.

1.1 Related work

We recommend to familiarize with the StarkWare documentation [Tea23] that also presents a good explanation of how ZK-STARK protocol works. Our article, in turn, is based on the STARK-101 course from StarkWare which you can check out here: `https://starkware.co/stark-101/`. Also, we recommend to read the Vitalik's series of posts about STARKs: [But17b], [But17c], [But17a] and the newest post about STARKs over Circle filed extansion [But24].

2 Preliminaries

Let's denote by $\mathbb{F}$ the prime-order field of size N. Let's also define $\mathbb{F}^\times$ the multiplicative group of $N-1$ elements and w – any fixed primitive element in $\mathbb{F}^\times$.

In this paper, we consider to describe STARKs using an example statement where we utilize $N = 3*2^{30}+1$, but note, that in real cases you may want to select lager fields, for example $2^{251}+17*2^{192}+1$, that also can be suitable for our example. The only requirement for the field $\mathbb{F}$ is to be able to select a large multiplicative subgroup with order 2^k.

2.1 Merkle trees

Commitment scheme is an approach to encode some data that will be stored hidden until user decides to unhide it. We consider **Merkle tree** as a cryptographic construction that allows us to commit a large amount of data into a single hash value. Structurally, a Merkle tree is a binary tree where each leaf node holds the hash of a data block, while each non-leaf node contains the hash derived from it's child nodes. To reveal any leaf's data, it's value must be presented alongside the Merkle path (also known as the Merkle proof). This combination enables anyone to verify that the hash of this data is indeed committed in the tree.

3 STARK protocol

The Fibonacci square sequence is a sequence of elements defined as follows:

$$a_i = a_{i-1}^2 + a_{i-2}^2$$

To describe how ZK-STARK protocol works we are going to explain how to prove the following statement:

- I know a field element x such that the 1023rd element of the Fibonacci square sequence starting with 1 and x is 2338775057.

The private x in this case equals to 3141592.

3.1 Trace polynomial

We call **trace** a sequence of elements considered as a basis for our proof. This sequence contains private and public values together and follows certain constraints. For our example, we put trace a sequence a of first 1023 elements of the Fibonacci square sequence.

The trace is implied to be an evaluation of some unknown **trace polynomial** of degree $|a| - 1$ (to follow the Unisolvence Theorem). We also call **domain** a sequence of values from $\mathbb{F}$ where we evaluate our polynomials. To interpolate our trace polynomial we select as a domain a multiplicative subgroup of 1024 elements from $\mathbb{F}^{\times}$:

$$G = \{g^i \mid g = w^{3 \cdot 2^{20}} \wedge i \in [0; 1024)\}$$

Next, by using Lagrange interpolation over $(g^i, a_i)_0^{1022}$ points we compute a trace polynomial $f \in \mathbb{F}[x]$. Note, that in practice, you may want to use more efficient algorithm for interpolation, for example – *Fast Fourier Transform (FFT)* [But19].

3.2 Polynomial commitment

To commit our trace polynomial we use the special **evaluation domain** that is larger several times (ρ times) then the polynomial degree. In our case we select a multiplicative subgroup of 8192 elements from $\mathbb{F}^{\times}$:

$$H = \{h^i \mid g = w^{3 \cdot 2^{17}} \wedge i \in [0; 8192)\}$$

Then, we define the evaluation domain as:

$$E = \{w * h_i \mid \forall h_i \in H\}$$

We build a Merkle tree over the values $f(e_i),\ \forall e_i \in E$ and name it's root as a **trace polynomial commitment**. This approach will also be used to commit other polynomials during the protocol walk-through.

3.3 Constraints

The **constraints** are expressed as polynomials evaluated over the trace cells, which are satisfied if and only if the computations are correct. Obviously, our initial statement consists of the following three requirements:

1. The element a_0 is equal to 1;
2. The element a_{1022} is equal to 2338775057;
3. Each element a_{i+2} is equal to $a_{i+1}^2 + a_i^2 \mod N$.

To verify that our committed trace polynomial satisfies all constraints, we can check that it has corresponding roots. In particular, according to the selected interpolation points (g^i, a_i):

1. *The element a_0 is equal to* 1 translated to: $f(x) - 1$ has root at $x = g^0 = 1$;
2. *The element a_{1022} is equal to* 2338775057 translated to: $f(x) - 2338775057$ has root at $x = g^{1022}$;
3. *Each element a_{i+2} is equal to $a_{i+1}^2 + a_i^2$* translated to: $f(g^2x) - f(gx)^2 - f(x)^2$ has roots in $G \setminus \{1021, 1022, 1023\}$

To ensure that the specified polynomials have roots in given values, we can use the following property: if polynomial $f(x) \in \mathbb{F}[x]$ has root in x_0 then the $\frac{f(x)}{x-x_0}$ is also a polynomial in $\mathbb{F}[x]$. So, we define the following polynomial constraints:

$$p_0(x) = \frac{f(x) - 1}{x - 1}$$
$$p_1(x) = \frac{f(x) - 2338775057}{x - g^{1022}}$$
$$p_2(x) = \frac{f(g^2x) - f(gx)^2 - f(x)^2}{\prod_{i=0}^{1020} x - g^i}$$

Unfortunately, the p_2 polynomial still looks inconvenient to work with, so we may want to simplify it. Using the following property we can reduce the denominator of p_2:

$$x^{|G|} - 1 = \prod_{g \in G} (x - g), \forall x \in G$$

This equation works because both sides are polynomials whose roots are exactly the elements of G. Note, that while evaluating our polynomial on lager domain then G we should only ensure that the resulting polynomial still holds the relation $f(g^i) = a_i$, so it is acceptable to use properties that only work over G. So, finally we have:

$$p_2(x) = \frac{(f(g^2x) - f(gx)^2 - f(x)^2)(x - g^{2021})(x - g^{2022})(x - g^{2024})}{x^{2024} - 1}$$

Note, that our constraints follow the obvious requirement: using only the given trace polynomial evaluations for the certain x by the prover ($f(x)$, $f(gx)$ and $f(g^2x)$ in our example) the verifier can compute the constrains polynomials $p_i(x)$ by itself.

3.4 Composition polynomial

To combine all our constraints into a single polynomial, we can follow a commonly used principle by taking a linear combination with the challenges from the verifier. In particular, after receiving trace polynomial commitment from the prover, the verifier selects $\alpha_0, \alpha_1, \alpha_2 \in \mathbb{F}$ and sends to the prover. Then, the prover puts the **composition polynomial** as:

$$CP(x) = \alpha_0 * p_0(x) + \alpha_1 * p_1(x) + \alpha_2 * p_2(x)$$

Prover also commits this polynomial by evaluating on the evaluation domain and building a Merkle tree.

3.5 FRI

In general, our goal is to verify that the committed polynomial $CP(x)$ satisfies all our constrains, by checking it's evaluation at a random point from the evaluation domain that the verifier selects. Anyway, we can face the problem when the malicious prover constructs a lager polynomial that accepts lots of possible roots from our field (even 2^{61} field is still insecure for just checking the evaluation at one point). That is why we have to make sure that committed polynomial degree lies in acceptable range (the upper bound depends on the trace size).

The final stage of STARK protocol is a **Fast Reed-Solomon IOP of Proximity (FRI)**. FRI is a protocol between a prover and a verifier, which establishes that a given evaluation belongs to a polynomial of low-degree – *low* means no more than ρ times less than the length of the evaluation domain size.

The key idea of FRI protocol is to move from a polynomial of degree n to a polynomial of degree $n/2$ until we get a constant value. Let's consider the polynomial $g_0(x)$ of degree $n = 2^t$ and the evaluation domain $E_0 = E$. We suppose to group the odd and the even coefficients of g_0 together into the two

separate polynomials:

$$g_0^e(x^2) = \sum_{i=0}^{n/2}(a_{2i} * x^{2i})$$
$$g_0^o(x^2) = \sum_{i=0}^{n/2}(a_{2i+1} * x^{2i})$$

Then, we define the next-layer of FRI polynomial as $g_1(x^2) = g_0^e(x^2) + \beta g_0^o(x^2)$, where β is a challenge received from verifier. Next, we commit the $g_1(x^2)$ using a next-layer evaluation domain E_1 and continue to repeat the described operations until $g_i(x^{2^i})$ becomes constant.

While we use $E_0 = \{w * h_i \mid i \in [0; 8192)\}$ we define the next-layer domain as $E_1 = \{(w * h_i)^2 \mid i \in [0; 4096)\}$. This allows us to express the $g_1(x^2)$ using only $g_0(x)$ as follows:

$$g_0^e(x^2) = \frac{g_0(x) + g_0(-x)}{2}$$
$$g_0^o(x^2) = \frac{g_0(x) - g_0(-x)}{2x}$$
$$g_1(x^2) = g_0^e(x^2) + \beta g_0^o(x^2)$$

It's important to note that $-x$ must also lie in E for every $x \in E$ to achieve that $g_i(-x)$ can be committed together with $g_i(-x)$. This works because of our choice of a multiplicative coset of size 2^k for integer k as our evaluation domain. The proof of this statement (see – Appendix) can be also applied to each next-layer domain E_i.

3.6 Protocol definition

Finally, to verify all computations prover and verifier perform the following algorithm:

Protocol 1 Zero-Knowledge Scalable Transparent Argument of Knowledge

Input:

- *Private:* $a_1 = 3141592$ – secret element of Fibonacci square sequence
- *Public:* $a_0 = 1$ and $a_{1022} = 2338775057$ – fixed elements of Fibonacci square sequence, $\mathbb{F}$ – field to work with

Protocol:

The prover interpolates $f(x)$ and submits it's commitment to the verifier.

The verifier selects random $\alpha_0, \alpha_1, \alpha_2 \in \mathbb{F}$ and sends to the prover.

The prover builds the composition polynomial $CP(x)$ and submits it's commitment to the verifier.

The verifier selects random $i \in [0; 8192 - 16)$, puts $c = w * h^i$ and sends it to the prover.

The prover responds with the $f(c), f(gc), f(g^2c), CP(c), CP(-c)$ and corresponding Merkle proofs to them. Note, that to claim gx where $x = w * h^i \in E$ from evaluation domain E prover uses power shifting $i + 8$ and $i + 16$ to claim g^2x (see – Appendix).

The verifier checks Merkle proofs and the evaluation of $CP(c)$ by evaluating the constraints polynomials $p_0(c), p_1(c), p_2(c)$.

The prover and the verifier go through the FRI protocol for $g_0(x) = CP(x)$ where the prover commits to the layer-i polynomial $g_i(x)$, the verifier selects a challenge β and queries from the prover $g_i(c), g_i(-c)$ to compute $g_{i+1}(c)$ until $g_i(x), i \in [1; 12)$ becomes constant.

The range $i \in [1; 12)$ is referred to the maximal degree of our composition polynomial $CP(x)$ – in our example $\deg CP(x) \leq 11$.

4 Conclusion

In conclusion, the soundness of the ZK-STARK protocol follows from the impossibility to commit any possible evaluation of the forgery $CP(x)$ over evaluation domain E and simultaneously prove that $CP(x)$ is a low-degree polynomial by the FRI protocol. Since the size of E is ρ times bigger then the maximum allowed polynomial degree (that directly depends on the size of the trace), the attacker either can't construct such a polynomial or can't construct a low-degree polynomial, so a valid low-degree composite polynomial can only be obtained using a valid trace.

References

[But17a] Vitalik Buterin. *STARKs, Part 3: Into the Weeds*. 2017. URL: `https://vitalik.eth.limo/general/2018/07/21/starks_part_3.html`.

[But17b] Vitalik Buterin. *STARKs, Part I: Proofs with Polynomials*. 2017. URL: `https://vitalik.eth.limo/general/2017/11/09/starks_part_1.html`.

[But17c] Vitalik Buterin. *STARKs, Part II: Thank Goodness It's FRI-day*. 2017. URL: `https://vitalik.eth.limo/general/2017/11/22/starks_part_2.html`.

[But19] Vitalik Buterin. *Fast Fourier Transforms*. 2019. URL: `https://vitalik.eth.limo/general/2019/05/12/fft.html`.

[Tea23] StarkWare Team. *ethSTARK Documentation*. 2023. URL: `https://eprint.iacr.org/2021/582.pdf`.

[But24] Vitalik Buterin. *Exploring circle STARKs*. 2024. URL: `https://vitalik.eth.limo/general/2024/07/23/circlestarks.html`.

A Proof of existence of additive inverse element in E

In general, this proof works for any field of order N where we can select a multiplicative subgroup of order 2^k. In our example $N = 3 * 2^{30} + 1$ and $k = 13$.

Any element $x \in E$ is equal to $w * h^i = w * w^{i \cdot 3 \cdot 2^{17}}$ by construction. Without losing generality, we can move to $x = h^i$ for simplicity. Then, $x = h^i = w^{3 \cdot 2^{17} \cdot i}$ and $-x = h^j = w^{3 \cdot 2^{17} \cdot j}$, where $j = i + \frac{|E|}{2} \mod |E|$ (we will show why it works below). Let's also assume for simplicity that $i < j$, then:

$$x + (-x) \equiv w^{3 \cdot 2^{17} \cdot i}(1 + w^{3 \cdot 2^{17} \cdot \frac{|E|}{2}}) \equiv 0 \mod N$$

$$\frac{|E|}{2} = 4096 = 2^{12}$$

$$1 + w^{3 \cdot 2^{17} \cdot 2^{12}} \equiv 0 \mod N$$

$$w^{3 \cdot 2^{29}} \equiv N - 1 \mod N$$

$$w^{3 \cdot 2^{30}} \equiv (N-1)^2 \mod N$$

$$1 \equiv N^2 - 2N + 1 \mod N$$

$$1 \equiv 1 \mod N$$

The equation $w^{3 \cdot 2^{30}} \equiv 1 \mod N$ is obtained from the order property of the primitive element w in the multiplicative group $\mathbb{F}^\times$.

B Proof of shifted element index in E

Any element $x \in E$ is equal to $w * h^i = w * w^{i \cdot 3 \cdot 2^{17}}$ by construction. Then, while $g = w^{3 \cdot 2^{20}}$ to claim gx and g^2x from E we take $w * h^{i+8}$ and $w * h^{i+16}$:

$$gx = w^{3 \cdot 2^{20}} * w * w^{i \cdot 3 \cdot 2^{17}} = w * w^{3 \cdot 2^{17}(i+2^3)} = w * h^{i+8}$$
$$g^2x = w^{3 \cdot 2^{21}} * w * w^{i \cdot 3 \cdot 2^{17}} = w * w^{3 \cdot 2^{17}(i+2^4)} = w * h^{i+16}$$

This approach requires i be less then $|E| - 16$, so the verifier selects i from $[0; 8192 - 16)$ range.