Introduction into ZK-STARK protocol

January 23, 2025

Distributed Lab

zkdl-camp.github.io

github.com/ZKDL-Camp

Plan

Introduction

What is STARK?

ZK-STARK – Zero-Knowledge Scalable Transparent Argument of Knowledge.

- *scalable* implies that the proving time grows at most quasilinearly (linear up to the logarithmic factor) relative to the witness-checking process. Additionally, the verification is limited to a polylogarithmic growth concerning same process.
- *transparent* means there is no requirement for a trusted setup.

STARK is a SNARK?

Non-interactive STARK = transparent SNARK. All existing protocols in production are non-interactive.

STARK-friendly fields

Two-adicity fields

Definition

We call two-adicity fields, the fields where we can select the multiplicative subgroup of order 2^k.

For the multiplicative group generator $w \in \mathbb{F}_N^\times$, the generator of the two-adicity subgroup will be $w^{\frac{N-1}{2^k}}$.

Example fields:

- Goldilocks field: $N = 2^{64} - 2^{32} + 1$
- Mersenne31 field: $N = 2^{31} - 1$
- StarkNet field: $N = 2^{251} + 17 \cdot 2^{192} + 1$

h – two-adicity group H generator.

$$h = w^{\frac{N-1}{|H|}}$$

$$\forall x \in H, x = h^i = w^{\frac{N-1}{|H|} \cdot i}$$

$$-x = h^j = w^{\frac{N-1}{|H|} \cdot j}$$

Then, the i and j values obtain the following property:

$$j = i + \frac{|H|}{2} \mod |H|$$

Witness and commitments

Trace

Definition

We call **trace** a sequence of elements from $\mathbb{F}$ that represents our witness.

Definition

We call **domain** a two-adicity subgroup $G \in \mathbb{F}$ where we evaluate our polynomials.

Example

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

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

We gonna evaluate this sequence under the prime modulus $N = 3 \cdot 2^{30} + 1$. Then, we can prove for example 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).

Example

In our example, we put trace a sequence *a* of first 1023 elements of the Fibonacci square sequence over $\mathbb{F}_N$, where $N = 3 \cdot 2^{30} + 1$.

$$1, 1, 2, 5, 29, \ldots$$

To interpolate our trace polynomial we select as a domain a two-adicity subgroup of 2^{10} elements from $\mathbb{F}^\times$ with generator $g = 5^{\frac{3 \cdot 2^{30}}{2^{10}}}$ (here 5 stands for the primitive element in $\mathbb{F}_N^\times$):

$$G = \{g^i \mid g = 5^{3 \cdot 2^{20}} \wedge i \in [0; 1024)\}$$

Using any interpolation scheme over $(g^i, a_i)_0^{|a|-1}$ points we compute a trace polynomial $f \in \mathbb{F}[x]$.

Definition

We call **evaluation domain** a two-adicity coset $E = wH \in \mathbb{F}$, where $H \in \mathbb{F}$ is a two-adicity subgroup, that is larger ρ times (some small constant) then the domain.

Example

In our case we select a two-adicity subgroup of 2^{13} elements from $\mathbb{F}^\times$ ($\rho = 8$):

$$H = \{h^i \mid h = 5^{3 \cdot 2^{17}} \wedge i \in [0; 8192)\}$$

Then, we define the evaluation domain as:

$$E = \{5 \cdot h_i \mid \forall h_i \in H\}$$

Commitment

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

Constraints

The **constraints** in STARK protocol are expressed as polynomials evaluated over the trace cells, which are satisfied if and only if the computations are correct.

Example

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 $a_{i+1}^2 + a_i^2 \mod N$.

The relation $r(a_i, a_j) = 0$ can be rewritten as $r(f(g^i), f(g^j)) = 0$.

Example

For our Fibonacci trace we have the following constraints to be checked over the interpolated polynomial:

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 $G \setminus \{g^{1021}, g^{1022}, g^{1023}\}$

Note, that the verifier should be able to compute the constraints polynomials $p_i(x)$ using only the given trace polynomial evaluations for the certain x.

Composition polynomial

$$CP(x) = \sum \alpha_i \cdot p_i(x)$$

Example

The Fibonacci composition polynomial looks like as follows:

$$CP(x) = \alpha_0 p_0(x) + \alpha_1 p_1(x) + \alpha_2 p_2(x) =$$
$$\alpha_0 \frac{f(x) - 1}{x - 1} + \alpha_1 \frac{f(x) - 2338775057}{x - g^{1022}} +$$
$$\alpha_2 \frac{(f(g^2 x) - f(gx)^2 - f(x)^2)(x - g^{2021})(x - g^{2022})(x - g^{2024})}{x^{1024} - 1}$$

FRI

FRI - Fast Reed-Solomon IOP of Proximity

$$z_0(x) = \sum a_i \cdot x^i$$

$$z_0^o(x^2) = \sum_{i=0}^{n/2} (a_{2i+1} \cdot x^{2i})$$

$$z_0^e(x^2) = \sum_{i=0}^{n/2} (a_{2i} \cdot x^{2i})$$

Or, in more comfortable form:

$$z_0^e(x^2) = \frac{z_0(x) + z_0(-x)}{2}$$

$$z_0^o(x^2) = \frac{z_0(x) - z_0(-x)}{2x}$$

Next layer

$$z_1(x^2) = z_0^e(x^2) + \beta z_0^o(x^2)$$
$$E_1 = \{(w \cdot h_i)^2 \mid i \in [0; \frac{|E_0|}{2})\}$$

Protocol definition

The prover and the verifier run the interactive version of the ZK-STARK protocol. Both know the statement to be proved, that is defined by the constraint polynomials and the field $\mathbb{F}$ to work over. Prover also knows the witness to be able to generate the trace.

Preparation:

- ✓ The prover interpolates trace polynomial $f(x)$ and submits it's commitment to the verifier.
- ✓ The verifier selects challenges 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.

FRI:

- ✓ The verifier selects random $i \in [0; |E|)$, puts $c = w \cdot h^i$ and sends it to the prover.
- ✓ The prover responds with the $CP(c), CP(-c)$ and all $f(x)$ required to check CP evaluation with corresponding Merkle proofs to them.
- ✓ The verifier checks Merkle proofs and the evaluation of $CP(c)$ by evaluating the constraints polynomials $p_i(c)$.
- ✓ The prover and the verifier go through the FRI protocol for $z_0(x) = CP(x)$ where the prover commits to the layer-i polynomial $z_i(x)$, the verifier selects a challenge β and queries from the prover $z_i(c), z_i(-c)$ to compute $z_{i+1}(c)$ until $z_i(x), i \leq log_2(\deg CP(x))$ becomes constant.

Security

- Blowup factor ρ
- Proof-of-work bits δ
- NUmber of queries s

$$\lambda \geq min\{\delta + log_2(\rho) \cdot s, log_2(|F|)\} - 1$$

Example

If the protocol is deployed over 256-bit field and the domain ratio is $\rho = 3$, to achieve the 128 bit security we can for example execute 33 FRI query and evaluate 29 proof-of-work bits:

$$min\{29 + 3 \cdot 33, 256\} = 128$$