Technical Assessment

Carlo Brunetta

July 12, 2024

FRI-based polynomial commitment schemes are commonly used for building transparent zkSNARKs. Some variants of such schemes have a wide range of parameters for finding a trade-off between the performance and security of the overall system. For example, a larger proximity parameter of the FRI protocol allows for a more efficient commitment scheme but requires careful consideration when analyzing the system's security level.

We propose to consider two specific proof systems, Redshift [KPV19] and Plonky2 [Plo] (both use the FRI-based commitment scheme), and answer the following research questions.

1. Redshift describes ways to guarantee the uniqueness of setup polynomials (see section 4.3 [KPV19]). Namely, they propose to use extra evaluation points. How many extra points will ensure a 128-bit security level for a 256-bit finite field and code rate $= \frac{1}{8}$?
2. Describe (briefly) how an attacker can exploit the absence of such extra points.
3. Does the Soundness notion capture such types of attacks?
4. Does Plonky2 use extra evaluation points? If yes, how many are used? Why exactly so many? If not, is the system still safe to use? Why?
5. Can the attack described in the first answer be applied to Plonky2?

To prepare your answers, you may need to refer to the source code of these proof systems. We expect you to prepare your answer using LaTeX.

1 Preliminaries

Let $\mathbb{F}[X]$ denote the polynomial ring over the field $\mathbb{F}$ and the set of functions from D to C as C^D. Let $f \in \mathbb{F}^D$ and denote iP^f as the unique polynomial $\mathsf{iP}^f(X) = \sum_{i=0}^{|S|-1} a_i X^i$ with degree $\partial\left(\mathsf{iP}^f\right) < D$ with same graph as f, *i.e.* $\forall_{s \in S}, f(s) = \mathsf{iP}^f(s)$. For a set L, let $\mathsf{Z}_L = \prod_{l_i \in L}(X - l_i)$ be the unique polynomial with degree $\partial\left(\mathsf{Z}_L\right) = |L|$ that vanishes on L.

Define the Reed-Solomon code of rate $\rho \in (0, 1]$ evaluated over D on the field $\mathbb{F}$ as the set of functions,

$$\mathsf{RS}\left[\mathbb{F}, D, \rho\right] = \left\{ f : D \to \mathbb{F} \;\middle|\; \partial\left(f\right) < \rho \cdot |D| \right\}$$

To later make sense, D must be an affine subspace of $\mathbb{F}$ whenever $\mathbb{F} = \mathbb{F}_{2^m}$ while for $\mathbb{F} = \mathbb{F}_p$ for some prime p, D is a multiplicative subgroup of $\mathbb{F}_p$. Define the Hamming distance between two polynomials $f, g : D \to \mathbb{F}$ as $\Delta\left(f, g\right) = |\{a \in D \mid f(a) \neq g(a)\}|$ and the distance between a polynomial and a set of function as $\Delta\left(f, G\right) = \min_{g \in G} \Delta\left(f, g\right)$. The relative Hamming distance is obtained by normalizing the distance with respect to $|D|$.

1.1 Polynomial Commitment Scheme

A polynomial commitment scheme is a scheme that allows a prover to publicly commit to a function f and later provide proof of evaluation correctness of $f(\alpha)$ to the input α provided by a verifier.

Def. 1. *A polynomial commitment scheme Π is a tuple of algorithms:*

- $\mathsf{Setup}(\lambda, \mathbb{F}, d) \to \mathsf{p}$: *given security parameter λ, field $\mathbb{F}$ and maximal degree d, generates the schemes public parameters* p.
- $\mathsf{Commit}(\mathsf{p}, f(X)) \to \mathsf{c}_f$: *given the public parameters and a polynomial $f(X)$, the prover outputs a commitment c_f.*
- $\mathsf{PolyVerify}(\mathsf{p}, f(X), \mathsf{c})$: *given the public parameters, a polynomial $f(X)$ and a commitment c, output if the polynomial is consistent with the commitment.*
- $\mathsf{Open}(\mathsf{p}, f(X), \alpha) \to (y, \pi)$: *given public parameters, a prover is asked by a verifier to provide the evaluation of $f(\alpha)$ for a verifier's chosen value α. The prover provides the evaluation y and a correctness proof π.*
- $\mathsf{Verify}(\mathsf{p}, \mathsf{c}_f, \alpha, y, \pi)$: *given the public parameters, a poylnomial commitment c_f, an evaluation point α, the result of the evaluation y and proof π, the verifier verifies the correctness of the evaluation $y = f(\alpha)$ by accepting or rejecting the proof.*

If the generation can be done without the involvement of a trusted third party (or similar), the commitment scheme is said to be *transparent*. A polynomial commitment scheme should achieve several properties, the most relevant ones are (intuitively described):

- *Polynomial binding*: the commitment should bind the polynomial meaning that once the prover publishes c_f, it is difficult to find another polynomial g of degree $\partial(g) \le d$ with the same commitment.
- *Evaluation binding*: it is hard for the prover to cheat the opening meaning that if the prover opens to an incorrect value $z \neq f(\alpha)$, with high probability the verifier will reject the proof.
- *Polynomial hiding*: the verifier should be unable to obtain too much information on the polynomial from the opening (of course within limitation on the number of evaluations allowed).
- *Succinctness*: the scheme has short proofs and computation times. More specifically, we care only on schemes that communication costs and computation time poly-logarithmic in the polynomial degree $\partial(f)$.

1.2 Fast Reed-Solomon IOPP

Both RedShift and Plonky2 are based on the Fast Reed-Solomon Interactive oracle proofs of proximity [BSBHR18] (FRI) which is a protocol that allows a prover to show that a function f is δ close to a polynomial in $\mathsf{RS}\left[\mathbb{F}, L^{(0)}, \rho\right]$.

There are some subtle differences between the two which provide differences in efficiency and security [Plo, Sec.6.2]: RedShift utilizes FRI as originally described by Ben-Sasson et al. [BSBHR18] despite some improvements are already discussed in the appendix.

Plonky2 is based on DEEP–FRI [BSGKS19] which is an improved version of FRI [BSBHR18] where an additional evaluation point outside the domain is used to increase the soundness guarantees. Additionally, Plonky2 mainly uses batching, *i.e.* several polynomials $\{f_i\}_{i=0}^{|B|-1}$ are linearly combined using a verifier's provided random scalar λ as $h(X) = \sum_{i=0}^{|B|-1} \lambda^i \cdot f_i(X)$ and later the FRI protocol is executed on $h(X)$ which values opening can be computed from opening the values of $f_i(X)$ and computing the correct linear combination using λ. Furthermore, Plonky2 utilizes Merkle cap instead of Merkle root [CY21], *i.e.* a sorted list of Merkle root of the subtrees on a specified stopping level, and *grinding* which is a forced proof-of-work that the prover must execute and improve the soundness guarantees.

Despite the differences between the FRI's protocol used between Plonky2 and RedShift, the rest of the document will discuss the effective differences in how the two are used to obtain a polynomial commitment scheme.

2 RedShift

Proving FRI implies that the code rate ρ is chosen in such a way guarantee the uniqueness of the closest polynomial $p \in \mathsf{RS}[\mathbb{F}, D, \rho]$ such that $\Delta(f, p) < \delta$. RedShift tweaks this idea by considering a list of close polynomials $(p_i)_{i \in I} \subseteq \mathsf{RS}[\mathbb{F}, D, \rho]$ s.t. each polynomial is close, *i.e.* $\Delta(f, p_i) < \delta$. Having such a list can guarantee the protocol to prove the existence of a close polynomial but not necessarily the uniqueness. This is the core of RedShift's *list polynomial commitment* idea.

2.1 Too Many Close Polynomials

However, there is a problem when trying to guarantee the evaluation correctness of the setup polynomial $s(X)$ from which output defines specific constrain which must be correct and verified and not confused with the evaluation of one of the close polynomials $g_i \in \mathsf{RS}[\mathbb{F}, D, \rho]$. Technically, the verifier can compute such constraints (s is known since it's a setup polynomial) however in a non-succinct way since they must evaluate $s(X)$ which is linear in $\partial(s)$.

To avoid this problem, both the prover and verifier can compute the decoding list $L_\delta(s)$ of polynomials δ-relatively close to the setup polynomial $s(X)$ and later find a value ξ such that $s(\xi) \neq g_i(\xi)$ for all $g_i \in L_\delta(s)$. This evaluation point is used to distinguish $s(X)$ from all the other close polynomials.

RedShift's authors suggest two wait to find such a point. The first method computes the decoding list $L_\delta(s)$ which has cost $\mathcal{O}\left(|D|^3\right)$ for $\delta < 1 - (d - N - \mu)$ where D is the function domain, d the maximal degree considered, N number of evaluation points and μ the number of repetitions while, when considering equality, the complexity increases to $\mathcal{O}\left(|D|^{15}\right)$. Once the close polynomials are found, with an overhead linear in $|D|$, it is possible to find such distinguishing point ξ.

The second method is based on random sampling μ points and uses them as distinguishing points. Due to the Schwartz-Zippel lemma, with high-probability these points will indeed separate s from the corresponding list $g \in L_\delta(s)$. This takes $\mathcal{O}(\mu \cdot d)$ which is substantially faster when $d \sim |D|$ however at the cost of a reduced soundness guarantee.

Describe (briefly) how an attacker can exploit the absence of such extra points.

The absence of the distinguishing point(s) provides an insecurity when committing polynomials and requiring them to be uniquely binded (such as setup polynomials). Consider the setup polynomial $s(X)$ and the corresponding decoding list $L_\delta(s)$. Each close polynomial $g \in L_\delta(s)$ is δ close to $s(X)$ thus an adversary can switch the polynomial $s(X)$ with $g(X)$ which has a non-empty decoding list intersection $L_\delta(s) \cap L_\delta(g) \neq \emptyset$. This implies that the adversary forces a wrongful setup polynomial $g(X)$ which might have (in some relevant evaluation point α) a different value $g(\alpha) \neq s(\alpha)$.

For example and intuitively, in the RedShift's instantiation of Algorithm 2 [KPV19], an adversary able to force a wrong output for the setup polynomials $q_j, S_{i_j}, S_{\sigma_j}$ can force a different circuit to be calculated (by modifying the appropriate evaluation of q_j and/or how the index are permuted by S_{i_j}, S_{σ_j}).

Does the Soundness notion capture such types of attacks?

The soundness definition does not capture such an attack because, from a relation point of view, the proof provided for the wrong but still δ-close polynomial $g(X)$ is correct because generated for the intersection of the decoding lists $L_\delta(s) \cap L_\delta(g) \neq \emptyset$ thus allowing the existence of an element in the relation. Intuitively, adding the distinguishing evaluation point adds a constraint in the relation that allows the unique identification of the witness, *i.e.* the relation turns into something like "the witness polynomial is δ-close to the decoding list **and** the distinguishing evaluation is different from *all* the decoded polynomials".

2.2 Concrete Parameters

Redshift describes ways to guarantee the uniqueness of setup polynomials (see section 4.3 [KPV19]). Namely, they propose to use extra evaluation points. How many extra points will ensure a 128-bit security level for a 256-bit finite field and code rate $= \frac{1}{8}$?

To provide concrete parameters for the polynomial commitment scheme, let us focus on the results provided by Thm. 4, Thm. 5, Claim 1, Thm. 6 respectively stating the list polynomial commitment's soundness is the sum of the FRI and the distinguisher's soundness, the distinguishing soundness obtained from the random sampling and how all the pieces come together for the polynomial commitment parameters.

$$\underbrace{\frac{2\log(|D|)}{\eta^3\,|\mathbb{F}|} + (1-\min\{\delta_0,\delta\}+\eta\log(|D|))^{\mu}}_{\text{FRI sound.}} + \underbrace{\left(\frac{d}{|\mathbb{F}|}\cdot(J_{\rho,\nu}-1)\right)^{\mu}}_{\text{Dist. sound.}}$$

We are interested in soundness $< 2^{-128}$, field $\log(|\mathbb{F}|) = 256$ and code rate $\rho = \frac{1}{8}$.

Similarly to the author's argument [KPV19, Sect.6], consider $\nu = |\mathbb{F}|^{-\frac{1}{20}}$ which provides a decoding list size of $\frac{\rho^{-\frac{1}{2}}}{2}|\mathbb{F}|^{\frac{1}{20}} = J_{\rho,\nu}$. Furthermore, $\log(|D|) - 3 = \log(d-N-\mu)$ and $d - N - \mu > 16$ where d is the degree, N the number of extra evaluation points and μ the number of repetitions. For the sake of simplicity, consider $N = 1$ as in a single opening of an evaluation of the committed function meaning $d \sim |D|$. We consider the FRI protocol to be executed for all the $\log(d)$ rounds meaning that the amount of verifier's queries are $l \sim 2\cdot\log(d)$ and with $|D| = 2^{32}$ as the authors discuss. Finally, $\delta_0 = \frac{1-\rho}{2}$ and $\delta \in (0, 1-\sqrt{1-(1-\rho)\cdot(1-\nu)}$ meaning $\delta_0 = \frac{7}{16}$ and $\delta \in\sim (0, \frac{10}{16})$

By putting everything together, we get that,

$$\begin{aligned} \frac{2\log(2^{32})}{2^{-\frac{(256)\cdot 3}{20}}\cdot 2^{256}} + \left(1-\min\{\delta_0,\delta\}+2^{-\frac{256}{20}}\log(2^{32})\right)^{\mu} + \left(\frac{2^{32}}{2^{256}}\cdot\left(\frac{2^{\frac{3}{2}}}{2}2^{\frac{256}{20}}-1\right)\right)^{\mu} &< 2^{-128} \\ \frac{2^6}{2^{217.6}} + \left(1-\frac{7}{16}+2^{-7.8}\right)^{\mu} + \left(\frac{2^{13.3}}{2^{224}}\right)^{\mu} &< 2^{-128} \\ 0.5625^{\mu} + 2^{-210.7\mu} &\lesssim 2^{-128} - 2^{211.6} \\ \mu &\gtrsim \frac{-128}{-0.83} \simeq 154.2 \end{aligned}$$

If the committed polynomial must be binded, this implies $\mu \sim 155$ repetition of the FRI protocol implying μ additional points.

The authors provide a soundness analysis for RedShift instantiated as a Plonk-like system [KPV19, Sec.6]. Despite having similar security parameters, *i.e.* they focus on 80-bit security instead of 128, the field dimensions already provide a high soundness guarantee for the distinguisher used in the list polynomial commitment scheme. However, their FRI soundness estimation is ~ 0.504 for the same ν but $\rho = \frac{1}{16}$ which is in line with the ~ 0.5625 obtained in our case with $\rho = \frac{1}{8}$. The computed amount μ looks coherent with the amount of repetition presented in Table 1 considering that the table is limited to 80-bit of security and $\rho = \frac{1}{16}$.

3 Plonky2

Plonky2 is based on TurboPLONK on which provides an extensive amount of circuit gadget's improvements and specific parameter choices to increase the performance and efficiency of the protocol. As additional differences from RedShift, Plonky2 utilizes the Domain Extending for Eliminating Pretenders FRI (DEEP–FRI) [BSGKS19] for the polynomial commitment scheme and it allows recursion, *i.e.* intuitively the scheme's proving computation can be proved by the same scheme thus allowing to aggregate and compress proofs to a smaller size and with higher computational efficiency.

3.1 Differences

Does Plonky2 use extra evaluation points? If yes, how many are used? Why exactly so many? If not, is the system still safe to use? Why?

Can the attack described in the first answer be applied to Plonky2?

To accommodate recursion, the polynomial distance δ must be larger introducing multiple binding polynomials thus creating the same argument as RedShift. Similarly to RedShift, Plonky2 requires an additional evaluation point used to correctly bind the commitment to a specific polynomial however how binding is guaranteed is different.

Instead of requiring an evaluation point α in which the polynomial $s(X)$ is different from all the other δ-close polynomials $g_i(X)$, Plonky2 requires a ζ value chosen from the considered extension field $\mathbb{F}_p(\phi)$ (*i.e.* $\mathbb{F}_p[X]/(X^2-7)$, a degree 1 polynomial basically) and all the protocols' polynomials are evaluated on such point [Plo, Sec. 7.2, point 6]. This is part of the DEEP methodology where to discriminate between several possible δ-close polynomials, the verifier requests the evaluation $f(\zeta)$ on a random element ζ outside the domain and considers the quotient regarding such evaluation. Intuitively, the only way for the wrong δ-close polynomial g to have the quotient $\frac{g(X)-f(\zeta)}{X-\zeta}$ to be a polynomial by either having $g = f$ (an argument *a-la* Schwartz-Zippel lemma) or g has some peculiar shape which, most likely, will be inconsistent with being δ-close or $g(\zeta) = f(\zeta)$.

Additionally, the additional evaluation is verified with the oracle's commitment thus an adversary providing wrong evaluation will be spotted since the FRI protocol is executed on the quotient polynomial. For Plonky2 the majority of these checks are effectively executed during the FRI's protocol execution, since the prover batch aggregates all the quotient polynomials to prove into a single linear combination using coefficients provided by the verifier.

References

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *45th international colloquium on automata, languages, and programming (icalp 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[BSGKS19] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. Deep-fri: Sampling outside the box improves soundness. Cryptology ePrint Archive, Paper 2019/336, 2019. https://eprint.iacr.org/2019/336. URL: https://eprint.iacr.org/2019/336.

[CY21] Alessandro Chiesa and Eylon Yogev. Subquadratic snargs in the random oracle model. Cryptology ePrint Archive, Paper 2021/281, 2021. https://eprint.iacr.org/2021/281. URL: https://eprint.iacr.org/2021/281.

[KPV19] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. Redshift: Transparent snarks from list polynomial commitments. Cryptology ePrint Archive, Paper 2019/1400, 2019. URL: https://eprint.iacr.org/2019/1400, doi:10.1145/548606.3560657.

[Plo] GitHub - 0xPolygonZero/plonky2 — github.com. https://github.com/0xPolygonZero/plonky2. [Accessed 26-02-2024].