The Last Challenge Attack: Exploiting a Vulnerable Implementation of the Fiat-Shamir Transform in a KZG-based SNARK

Oana Ciobotaru[1], Maxim Peter[1], and Vesselin Velichkov[1]

[1]OpenZeppelin

April 17, 2024

Abstract

The Fiat-Shamir transform [1] is a well-known and widely employed technique for converting sound public-coin interactive protocols into sound non-interactive protocols. Even though the transformation itself is relatively clear and simple, some implementations choose to deviate from the specifications, for example for performance reasons. In this short note, we present a vulnerability arising from such a deviation in a KZG-based PLONK verifier implementation. This deviation stemmed from the incorrect computation of the last challenge of the PLONK protocol [2], where the KZG batching proof challenge was computed before, and, hence, independently from the KZG evaluation proofs. More generally, such a vulnerability may affect any KZG [3] implementation where one uses batched KZG proof evaluations on at least two distinct evaluation points. We call an attack enabled by such a deviation a *Last Challenge Attack*. For concreteness, we show that when a PLONK verifier implementation presents such a deviation, a malicious PLONK prover can mount a *Last Challenge Attack* to construct verifiable proofs of false statements. The described vulnerability was initially discovered as part of an audit, and has been responsibly disclosed to the developers and fixed. A proof of concept of the vulnerability, in which a proof is forged for an arbitrary public input, is made available.

In addition to the above, in this work we also provide a security proof of the knowledge-soundness of the batched KZG scheme with evaluations on at least two distinct values.

1 Introduction

The Fiat-Shamir (FS) transform [1] is a well-known and widely employed technique for converting certain *sound public-coin interactive protocols* into functionally equivalent *sound non-interactive protocols*. The security of the FS transform is proven, in general, in the random oracle model [4, 5] or, in special cases, relies on the *correlation intractability* of the hash function used [6], [7].

In practice, the FS transform is instantiated with a hash function and the resulting algorithm is a *heuristic*: for the resulting non-interactive protocols, at the moment, there are no known general proofs of soundness based on the computational intractability of some well-studied mathematical problem (e.g., discrete logarithm problem). However, the protocols obtained using the FS transform are believed to be secure in practice.

Intuitively, in standardised implementations of the FS transform, a public-coin interactive protocol is transformed into a non-interactive protocol by replacing the random challenges at each interactive round with the hash of the concatenation of all the public parameters of the instance, the public input of the instance and all the messages exchanged between the interactive parties, up to the current round.

Even though the transformation itself is relatively clear and simple, there are various reasons why implementations of the FS transform deviate from standard specifications, ranging from misunderstandings to trying to optimise for efficiency of the implementation. In such cases, the honest parties are at increased risk of being amenable to various types of potentially disastrous attacks.

In the following, we describe a new type of attack targeting incorrect implementations of the FS transform, called the *Last Challenge Attack*. This attack arises in implementations of the KZG polynomial commitment scheme verifier [3] in which the FS transform is applied to batched KZG evaluation proofs for at least two distinct evaluation points, but the challenge used to batch the evaluation proofs is incorrectly computed. In particular, the *Last Challenge Attack* is likely to affect implementations in which the proof batching challenge is incorrectly computed prior to and, thus, independently of at least two individual KZG evaluation proofs. For concreteness, we show an example of a *Last Challenge Attack* found in a KZG-based PLONK [2] verifier implementation which a malicious PLONK prover could exploit in order to construct verifiable proofs of false statements.

The vulnerability exploited by the *Last Challenge Attack* and described in this paper was initially discovered as part of an audit requested by Linea [1]. This audit was focused on their PLONK verifier implementation. The vulnerability has been responsibly disclosed to the developers and fixed [2].

1.1 Related Work

The Fiat-Shamir transform, a technique using a random oracle to obtain non-interactive protocols, was introduced in [1]. The first formal foundations for the random oracle model were published in [4]. The first security proof for a Fiat-Shamir transform in the random oracle model is part of [5]. In particular, the authors show how to transform a constant-round interactive zero-knowledge proof into a secure non-interactive signature scheme in the random oracle model. In practice, the random oracle model is replaced by a hash function.

So far, no general proofs of security are known for the resulting non-interactive protocols without the random oracle model. Recently, the idea of proving the security of the Fiat-Shamir transform based on standard cryptographic assumptions, even though applied to restricted types or sets of protocols, has gained more attention from the research community [8, 9, 10, 11, 12]. This line of work is centred around the notion of *correlation intractability* which was first introduced in [6] and [7]. At a high level and in the context of replacing the random oracle model with a concrete hash function implementation, correlation intractability for a hash function formalises the notion that *it should be infeasible to find inputs to the hash function that stand in some rare relationship with their corresponding outputs*, assuming this is also an infeasible property for the random oracle (see [7]). In some concrete cases, when the random oracle is instantiated with a hash function satisfying the above property, one is able to prove the soundness of the FS transform in the standard model based on standard security assumptions.

Finally, the research work that is closest to ours, describes attacks found in implementations of the Fiat-Shamir transform where the verifier deviates from the prescribed specifications in some way. One prominent such type of attack (see e.g., [13], [14] or [15]) stems from omitting all or part of the public inputs or the public parameters of the protocol when computing the Fiat-Shamir transform. In contrast, the attack presented in this work is possible due to the omission of a non-input part of the full transcript when computing the Fiat-Shamir transform.

1.2 Structure

The rest of the paper is organised as follows: in Section 2, we provide the necessary background, including the definition of polynomial commitment schemes. In Section 3, we prove knowledge-soundness of the KZG polynomial commitment scheme [3] applied to batch evaluation proofs for at least two distinct evaluation points. In Section 4, we describe the *Last Challenge Attack* applied to the general KZG scheme and, also, to one of its concrete and widely spread use cases, namely the KZG-based PLONK protocol. In Section 5, we conclude.

[1] `https://linea.build/`

[2] `https://github.com/Consensys/gnark/security/advisories/GHSA-7p92-x423-vwj6`

2 Preliminaries

We denote by λ the security parameter; for simplicity, we sometimes may omit explicitly mentioning λ. All algorithms used in this work run in polynomial time in λ or, simply, $\mathsf{poly}(\lambda)$. For short, we call such algorithms *efficient* or *polynomially bounded*. We denote by $\mathbb{F}$ a *finite field*. We consider only finite fields such that $|\mathbb{F}| \in O(2^\lambda)$, where by $|S|$ we understand the cardinality of some set S. We denote by $\mathbb{F}_{<d}[X]$ the set of univariate polynomials of degree less than d over $\mathbb{F}$. By $[t]$, where t is a natural number, we denote the set $\{1, \ldots, t\}$. By $\mathsf{negl}(\lambda)$, we understand the set of all negligible functions with λ as parameter.

We denote by $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ groups of some prime size q such that e is a secure, efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We denote by g_1, g_2, g_T some generators of $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, respectively, chosen uniformly at random such that $e(g_1, g_2) = g_T$. We use additive notation for $\mathbb{G}_1$ and $\mathbb{G}_2$ and multiplicative notation for $\mathbb{G}_T$. We also use the notation $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$.

Our security proof makes use of the *algebraic group model* (AGM) [16]. Given a cyclic group $\mathbb{G}$ of prime order p, by *algebraic adversary* with respect to $\mathbb{G}$ we understand an algorithm A such that whenever A outputs $h \in \mathbb{G}$, A also outputs a vector of field elements $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}_p^m$ such that $h = \sum_{i=1}^{m} \alpha_i \cdot L_i$, where $(L_1, \ldots, L_m)$ are all the group elements given to A during its execution so far.

2.1 Polynomial Commitment Schemes

We remind the reader the definition of the *polynomial commitment scheme* as per Section 3 from [2].

Definition 1 (Polynomial Commitment Scheme)**.** *Let P_{PCS} and V_{PCS} respectively be a prover and a verifier. A d-polynomial commitment scheme consists of*

- a generation algorithm $\mathsf{gen}(d)$ *that outputs a structured reference string* srs.
- a commitment algorithm $\mathsf{com}(f, \mathsf{srs})$ *that given a polynomial $f(X) \in \mathbb{F}_{<d}[X]$ outputs a commitment* cm *to $f(X)$.*
- a public-coin opening algorithm open *between P_{PCS} and V_{PCS}. P_{PCS} is given*

  $$f_1(X), \ldots, f_t(X) \in \mathbb{F}_{<d}[X]$$

  and P_{PCS} and V_{PCS} are both given integer $t = \mathsf{poly}(\lambda)$, $\mathsf{cm}_1, \ldots, \mathsf{cm}_\mathsf{t}$ (the alleged commitments to $f_1(X), \ldots, f_t(X)$), $z_1, \ldots, z_t \in \mathbb{F}$ (the opening or evaluation points) and $s_1, \ldots, s_t \in \mathbb{F}$ (the alleged correct evaluations $f_1(z_1), \ldots, f_t(z_t)$). At the end of the protocol V_{PCS} outputs either acc *or* rej

such that completeness *and* knowledge soundness in the algebraic group model *hold, where these properties are defined as follows:*

Completeness: *Fix integer t, scalars $z_1, \ldots, z_t \in \mathbb{F}$ and polynomials $f_1(X), \ldots, f_t(X) \in \mathbb{F}_{<d}[X]$. Let $\mathsf{cm_i} = \mathsf{com}(f_i, \mathsf{srs})$ for all $i \in [t]$. Then if* open *is run correctly with values $t, \{\mathsf{cm_i}, z_i, s_i = f_i(z_i)\}_{i \in [t]}$, V_{PCS} outputs* acc *with probability 1.*

Knowledge soundness in the algebraic group model: *There exists an efficient extractor E such that for any algebraic adversary A the probability of A winning the following game is* $\mathsf{negl}(\lambda)$ *over the randomness of A and* gen.

1. *Given* srs*, A outputs t,* $\mathsf{cm}_1, \ldots, \mathsf{cm}_\mathsf{t}$.
2. *E, given access to the messages of A during the previous step, outputs*

   $$f_1(X), \ldots, f_t(X) \in \mathbb{F}_{<d}[X].$$

3. *A outputs $z_1, \ldots z_t \in \mathbb{F}$, $s_1, \ldots, s_t \in \mathbb{F}$.*
4. *A takes the part of P_{PCS} in the protocol* open *with inputs t,* $\mathsf{cm}_1, \ldots, \mathsf{cm}_\mathsf{t}$*, $z_1, \ldots, z_t$, $s_1, \ldots, s_t$.*
5. *A wins if both*
   - *V_{PCS} outputs* acc *at the end of the protocol and*

- *For some* $i \in [t]$, $s_i \neq f_i(z_i)$.

Note that step 2 in Definition 1 is well-defined because, in the AGM model, for every group element $\mathsf{cm_i}$ that the adversary A outputs, the same adversary outputs as well a vector of scalars $(\alpha_{i,1}, \ldots, \alpha_{i,m})$ such that $\mathsf{cm_i} = \sum_{j=1}^{m} \alpha_{i,j} \cdot L_j$, where $(L_1, \ldots, L_m)$ are all the group elements given to A in its execution so far.[3] Once provided with these vectors of scalars $\{(\alpha_{i,1}, \ldots, \alpha_{i,m})\}_{i=1}^{t}$, E is able to obtain via Lagrange interpolation the corresponding polynomials $f_1(X), \ldots, f_t(X)$ mentioned in step 2 of the knowledge-soundness definition.

3 Security Considerations

In this section we detail the security proof for the KZG scheme [3] for multiple evaluation points (KZG MES). This is the generalisation of the scheme described on page 13 from PLONK [2] to any number of distinct evaluation points. We first give the instantiation of the KZG MES, and then provide a security proof using the well-known Schwartz-Zippel Lemma and the fact that the KZG scheme is secure in the AGM for one evaluation point as, for example, shown in Section 3.1 of PLONK [2].

3.1 Security Proof for KZG MES

We begin with some specific notation. We denote the distinct evaluation points by $z_1, \ldots, z_n$ and the number of polynomials to be evaluated at $z_1, \ldots, z_n$ by $t_1, \ldots, t_n$. We denote by $\{f_{1,i}(X)\}_{i\in[t_1]}, \ldots,$ $\{f_{n,i}(X)\}_{i\in[t_n]}$ the sets of polynomials to be evaluated at $z_1, \ldots, z_n$, respectively. In the rest of the paper, we denote by $\mathbb{F}$ the prime field of characteristic q introduced in Section 2.

Instantiation 2 (KZG Multipoint Evaluation Scheme)**.** *The KZG MES assumes parties* P_{KZG} *and* V_{KZG} *and proceeds as follows*

- $\mathsf{gen}(d)$:
  Choose uniform $x \in \mathbb{F}$. *Output* $\mathsf{srs} = ([1]_1, [x]_1, \ldots, [x^{d-1}]_1, [1]_2, [x]_2)$.
- $\mathsf{com}(f, \mathsf{srs})$:
  Using the srs, *compute and output* $\mathsf{cm} = [f(x)]_1$.
- $\mathsf{open}(\{\mathsf{cm_{1,j}}\}_{j\in[t_1]}, \ldots, \{\mathsf{cm_{n,j}}\}_{j\in[t_n]}, \{z_1, \ldots, z_n\}, \{s_{1,j}\}_{j\in[t_1]}, \ldots, \{s_{n,j}\}_{j\in[t_n]})$:
  Round 1: V_{KZG} *sends random* $\gamma_1, \ldots, \gamma_n \in \mathbb{F}$ *to* P_{KZG}.

  Round 2: P_{KZG} *computes the polynomials:*

$$h_1(X) := \sum_{j=1}^{t_1} \gamma_1^{j-1} \cdot \frac{f_{1,j}(X) - f_{1,j}(z_1)}{X - z_1}$$

$$\ldots$$

$$h_n(X) := \sum_{j=1}^{t_n} \gamma_n^{j-1} \cdot \frac{f_{n,j}(X) - f_{n,j}(z_n)}{X - z_n}$$

  and using srs *computes and sends* $W_1 := [h_1(x)]_1, \ldots, W_n := [h_n(x)]_1$ *to* V_{KZG}.
  We denote $\pi_{KZG} = (W_i)_{i=1}^{n}$ *and we call it a proof for KZG MES.*
  Round 3: V_{KZG} *chooses uniformly random* $u \in \mathbb{F}$.

  V_{KZG} *computes curve points* $F_1, \ldots, F_n$ *where*

$$F_1 = \sum_{j\in[t_1]} {\gamma_1}^{j-1} \cdot \mathsf{cm_{1,j}} - \left[\sum_{j\in[t_1]} {\gamma_1}^{j-1} \cdot s_{1,j}\right]_1, \ldots, F_n = \sum_{j\in[t_n]} {\gamma_n}^{j-1} \cdot \mathsf{cm_{n,j}} - \left[\sum_{j\in[t_n]} {\gamma_n}^{j-1} \cdot s_{n,j}\right]_1$$

  V_{KZG} *outputs* acc *if and only if the following holds:*

$$e(F_1 + \ldots + u^{n-1} \cdot F_n + z_1 \cdot W_1 + \cdots + u^{n-1} \cdot z_n \cdot W_n, [1]_2) \cdot e(-W_1 - \ldots - u^{n-1} \cdot W_n, [x]_2) = 1 \quad (1)$$

[3] The initial list given to A includes the structured reference string srs.

Observation 3. *In Round 3 of Instantiation 2, the honest V_{KZG} verifier implicitly checks that*

$$W_i \in \mathbb{G}_1, \forall i \in [n] \tag{2}$$

$$\mathsf{cm_{i,j}} \in \mathbb{G}_1, \forall i \in [n], j \in [t_i] \tag{3}$$

If check (3) passes, then by the definition of F_i, that in turn, implies

$$F_i \in \mathbb{G}_1, \forall i \in [n] \tag{4}$$

In order to prove the security of Instantiation 2, we require the following assumption:

Definition 4 (Q-DLOG Assumption)**.** *For a fixed integer Q, the Q-DLOG assumption for $(\mathbb{G}_1, \mathbb{G}_2)$ states that given*

$$[1]_1, [x]_1, \ldots, [x^Q]_1, [1]_2, [x]_2, \ldots, [x^Q]_2,$$

for uniformly chosen $x \in \mathbb{F}$, the probability of an efficient A outputting x is negligible in λ.

We will also make use of the following well-known result:

Lemma 5 (Schwartz-Zippel Lemma)**.** *Let $P(X_1, \ldots, X_k) \in \mathbb{F}[X_1, \ldots, X_k]$ be a non-zero polynomial of total degree d over the finite field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$ and let $r_1, \ldots, r_k \in S$ be selected uniformly at random. Then the probability that $P(r_1, r_2, \ldots, r_k) = 0$ is at most $\frac{d}{|S|}$.*

Finally, we state and prove the main result of this section:

Lemma 6 (Security of KZG Multipoint Evaluation Scheme)**.** *Assume the Q-DLOG for $(\mathbb{G}_1, \mathbb{G}_2)$. Let n be an integer such that $n \in \mathsf{poly}(\lambda)$. Then the KZG polynomial commitment scheme for multipoint evaluation as described in Instantiation 2 has completeness and knowledge-soundness in the algebraic group model as per Definition 1.*

Proof. The completeness property is easy to verify. For the knowledge-soundness property, we proceed as follows. Let:

$$a_1 = e(F_1, [1]_2), a_2 = e(F_2, [1]_2), \ldots, a_n = e(F_n, [1]_2).$$

Analogously, let:

$$b_1 = e(W_1, [x - z_1]_2), b_2 = e(W_2, [x - z_2]_2), \ldots, b_n = e(W_n, [x - z_n]_2).$$

Let us now assume V_{KZG} has output acc. That means property (1) holds. Given the definition of $\{a_i, b_i\}_{i \in [n]}$, property (1) is equivalent to

$$a_1 \cdot (a_2)^u \cdot \ldots \cdot (a_n)^{u^{n-1}} = b_1 \cdot (b_2)^u \cdot \ldots \cdot (b_n)^{u^{n-1}} \tag{5}$$

Due to the fact that properties (2) and (4) hold (see Observation 3) and by the definition of the pairing function e, it implies that

$$a_i, b_i \in \mathbb{G}_T, \forall i \in [n] \tag{6}$$

Since $\mathbb{G}_T$ is cyclic with generator g_T, property (6) implies that there exist

$$c_i, d_i \in \mathbb{Z}_{|\mathbb{G}_T|} = \mathbb{Z}_q = \mathbb{F}, a_i = g_T^{c_i}, b_i = g_T^{d_i}, \forall i \in [n] \tag{7}$$

Due to property (7) and again, since $\mathbb{G}_T$ is cyclic with generator g_T, equation (5) becomes

$$\sum_{i=1}^{n} c_i \cdot u^{i-1} = \sum_{i=1}^{n} d_i \cdot u^{i-1} \mod q \tag{8}$$

Since u has been chosen uniformly at random after a_i, b_i have been fixed (and, hence, after c_i, d_i have been fixed), applying the Schwartz-Zippel Lemma to equality (8) which holds over $\mathbb{F}$, we obtain that $c_i = d_i, \forall i \in [n]$, except with negligible probability in λ, and hence

$$a_i = b_i, \forall i \in [n] \tag{9}$$

Substituting the definition of $\{a_i\}_{i \in [n]}$ and $\{b_i\}_{i \in [n]}$, (9) implies $e(F_i, [1]_2) = e(W_i, [x - z_i]_2), \forall i \in [n]$. Each such equality, in fact, represents the final V_{KZG} check in a KZG single evaluation point scheme as described in Section 3.1 in PLONK [2]. However, due to the security analysis performed in the same section, assuming the AGM model, this in turn implies, except with negligible probability in λ, the

existence of a universal extractor E that can extract polynomials $\{f_{1,j}(X)\}_{j\in[t_1]}, \ldots, \{f_{n,j}(X)\}_{j\in[t_n]}$ such that the following holds:

$$f_{1,j}(z_1) = s_{1,j}, \forall j \in [t_1]$$
$$\ldots$$
$$f_{n,j}(z_n) = s_{n,j}, \forall j \in [t_n]$$

This is exactly what we have set ourselves to prove.

□

3.2 Discussion

In order to conclude the security proof for Lemma 6, we made explicit use of the Schwartz-Zippel Lemma. If not all of its preconditions are met i.e.,

- if either the degree of the polynomial considered is not negligible over the size of the field $\mathbb{F}$ or
- if u is not *chosen uniformly at random* from $\mathbb{F}$,

then the security proof may not hold anymore. Note that in the proof of Lemma 6, the degree of the polynomials for which we apply the Schwartz-Zippel Lemma is at most $n-1$ while, by assumption, $n \in \mathsf{poly}(\lambda)$ and $|\mathbb{F}| \in O(2^\lambda)$.

Regarding the second precondition and, in order to protect himself from a malicious KZG prover, the verifier V_{KZG} has two options:

1. In an ideal world, V_{KZG} should follow all the steps in Instantiation 2 as prescribed. In particular, V_{KZG} should choose u uniformly at random from $\mathbb{F}$. However, that is not always efficient to implement in practice. For example, in certain real-world scenarios (e.g., in the context of blockchains), interactivity is expensive and secure non-interactive versions of protocols are preferred. In such cases, the option described next should be implemented in practice.
2. In practice and in support of an efficient implementation, the V_{KZG} verifier only simply needs to compute the value u as the hash of the entire transcript. The transcript includes all the public parameters, all the public inputs and all the communication up to that point between P_{KZG} and V_{KZG}. This is, in fact, the FS transform of the three-round interactive protocol open (see Instantiation 2), where a part of the third round is the computation by V_{KZG} of a public-coin random challenge u.

4 The Last Challenge Attack

In the previous section we have shown that the interactive KZG multipoint evaluation scheme (KZG MES) is secure if the challenge used for batching KZG evaluations proofs is chosen uniformly at random after all the other transcript values are fixed. In this section we show that the above condition is not only sufficient, but a variation of it is also necessary in order for the knowledge-soundness property of non-interactive KZG MES to hold in the context of the FS transform.

Concretely, in Section 4.1 we describe a hypothetical attack strategy that exploits an incorrect application of the FS transform which, in turn, results in a vulnerable verifier. More specifically, our hypothetical attack demonstrates that if the challenge used for batching KZG evaluation proofs is computed before at least two individual KZG evaluation proofs, then a malicious prover can break, with high probability, the knowledge-soundness of the respective variation of non-interactive KZG MES. In Section 4.2, we describe a concrete instantiation of the above attack strategy encountered while auditing a vulnerable KZG-based PLONK verifier implementation.

We choose to first describe the hypothetical attack as opposed to directly describing its special case variant encountered in practice. We believe by following this order, we enhance the clarity of the presentation and, also, we highlight that the practical vulnerability described in Section 4.2 is inherent to any KZG implementations that deviate in a specific way from correct specifications.

4.1 A Hypothetical Attack on a Variation of KZG MES

The hypothetical attack described below applies to a variation of the non-interactive version of KZG MES with at least two evaluation points. We recall that the interactive version of KZG MES described in Instantiation 2 is converted to non-interactive by the means of the FS transform. According to the FS transform, the challenges $(\gamma_i)_{i\in[n]}$ and u sent by V_{KZG} in *Round 1* and *Round 3*, respectively, are computed by an FS-compliant non-interactive verifier as the hash of the communication transcript up to the corresponding round in the protocol. Intuitively, our hypothetical attack assumes that the challenge u in *Round 3* of the open procedure is incorrectly computed by a deviating verifier as the hash of only part of the transcript. As a result, a malicious prover is able to forge a non-interactive KZG MES proof that is accepted as valid by the deviating verifier.

4.1.1 The Setting

In order to be able to describe the attack, we introduce four new entities. Let P_{KZGN} and V_{KZGN} be the non-interactive versions of P_{KZG} and V_{KZG}, respectively, obtained by applying the FS transform. Also, let P'_{KZGN} be a malicious prover trying to construct a proof π of a false statement, as described in the rest of this section. P'_{KZGN} is interacting with V'_{KZGN} which works the same as V_{KZGN} with the only difference that V'_{KZGN} is computing u as the hash of the full transcript but excluding values W_1 and W_2. In the following, we assume that $n \geq 2$, where n was introduced in Section 3.

4.1.2 The Attack

The attack proceeds as follows:

Step 1. The malicious prover P'_{KZGN} computes some group elements $A, B \in \mathbb{G}_1$ satisfying

$$e(A, [x]_2) \cdot e(-B, [1]_2) = 1 \tag{10}$$

In order to achieve that, P'_{KZGN} honestly simulates one instance of a single-point evaluation KZG scheme for some arbitrary polynomial $f(X)$ evaluated at some arbitrary point z. P'_{KZGN} does that by taking the roles of both prover and verifier in the KZG instance. Hence, at the end of the simulation, P'_{KZGN} is in possession of group elements $A, B \in \mathbb{G}_1$ such that relation (10) holds.

Step 2. P'_{KZGN} sets the inputs of the non-interactive version of the KZG open procedure to any values of its choice[4] Additionally, P'_{KZGN} sets all KZG MES proof components except for W_1 and W_2 (i.e., group elements $W_3, \ldots, W_n$, as per *Round 2* of the open procedure of Instantiation 2) to any values of its choice in $\mathbb{G}_1^{n-2}$. This is done by P'_{KZGN} in preparation to create a proof π'_{KZG} for a false statement.

Step 3. From the inputs of the KZG open procedure, P'_{KZGN} computes group elements $F_1, \ldots, F_n$ following the same logic as V_{KZG} described in Round 3 of Instantiation 2.

Step 4. As a reminder, V'_{KZGN} does not follow the FS transform and deviates from it by computing the final challenge u as the hash of the full transcript excluding values W_1 and W_2. P'_{KZGN} exploits the fact that the challenge u can be computed and, hence, fixed before W_1 and W_2 are computed. Hence, P'_{KZGN} solves the following linear system of equations with W_1, W_2 as the only unknowns. The system arises from the KZG MES pairing check (1) and also using the quantities A, B computed by P'_{KZGN} in Step 1, the group elements $W_3, \ldots, W_n$ chosen by P'_{KZGN} in Step 2, the group elements $F_1, \ldots, F_n$ computed in Step 3 and the value u:

$$\begin{cases} W_1 + u \cdot W_2 + u^2 \cdot W_3 + \ldots + u^{n-1} \cdot W_n = A \\ F_1 + \ldots + u^{n-1} \cdot F_n + z_1 \cdot W_1 + u \cdot z_2 \cdot W_2 + u^2 \cdot z_3 \cdot W_3 + \cdots + u^{n-1} \cdot z_n \cdot W_n = B \end{cases} \tag{11}$$

The linear system (11) has a solution for W_1, W_2 if and only if $u \neq 0$ and $z_1 \neq z_2$.

[4]Note that this includes the special case when both the prover P'_{KZGN} and the verifier V'_{KZGN} need to agree on the inputs of the open procedure.

Step 5. P'_{KZGN} appropriately fills in the corresponding slots of the proof π'_{KZG} with the above-computed values W_1 and W_2.

Step 6. If this step of the attack has been reached, then the vulnerable verifier V'_{KZGN} accepts proof π'_{KZG} as valid with probability 1.

We call the attack described above the *Last Challenge Attack* (LCA), as it exploits an incorrect computation of the *last challenge* u in the FS transform of the KZG MES for the verifier. More concretely, the attack exploits the fact that u is computed as the hash of only a part of the communication transcript between the interactive honest prover and honest verifier. This, in turn, invalidates option 2. from the security discussion in Section 3.2 by leaving certain elements of the resulting proof to be independent from the last challenge u. Ultimately, this allows a malicious prover to forge proofs of false statements that are, however, accepted with high probability by a vulnerable verifier.

4.2 The Last Challenge Attack on a PLONK Implementation

In this section we describe an instance of the attack presented in Section 4.1 observed in an actual implementation of a KZG-based PLONK verifier.

4.2.1 The PLONK Prover and Verifier Simplified

First, we summarise below the PLONK prover P_{PLONK} and PLONK verifier V_{PLONK}, as per [2, Sect. 8.1], highlighting only the details necessary to understand the steps of our attack.

The PLONK argument system uses the non-interactive version (i.e., the FS transform) of KZG MES for two evaluation points $z_1 = \mathfrak{z} \in \mathbb{F}$ and $z_2 = \mathfrak{z}\omega \in \mathbb{F}$ respectively, and two sets of polynomials of size $t_1 = 6$ and $t_2 = 1$, respectively:

$$\{f_{1,i}(X)\}_{i\in[t_1=6]} = \{\mathsf{r}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{S}_{\sigma_1}(X), \mathsf{S}_{\sigma_2}(X)\},\ \{f_{2,i}(X)\}_{i\in[t_2=1]} = \{\mathsf{z}(X)\}.$$

The evaluations of the polynomials $\{f_{1,i}\}_i$ and $\{f_{2,1}\}$ at $z_1 = \mathfrak{z}$ and $z_2 = \mathfrak{z}\omega$ are, respectively:

$$\{s_{1,i}\}_{i\in[t_1=6]} = \{\bar{r}, \bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma_1}, \bar{s}_{\sigma_2}\},\ \{s_{2,i}\}_{i\in[t_2=1]} = \{\bar{z}_\omega\}.$$

Prover. The PLONK prover P_{PLONK} produces a proof π_{PLONK} of the following form:

$$\pi_{PLONK} = \big([a]_1, [b]_1, [c]_1, [z]_1, [t]_1, W_1, W_2, \bar{a}, \bar{b}, \bar{c}, \bar{\mathbf{s}}_{\sigma 1}, \bar{\mathbf{s}}_{\sigma 2}, \bar{z}_\omega\big) \tag{12}$$

where $[t]_1$ is the commitment to the quotient polynomial $\mathsf{t}(X)$ obtained by dividing all batched constraints by the vanishing polynomial $Z_H(X)$ with its roots represented by the elements of the multiplicative subgroup H. Note that for the components of π_{PLONK} corresponding to slots 6 and 7, we use the notation established in Section 3, i.e., W_1 and W_2.

Verifier. The PLONK verifier V_{PLONK} takes as input the public parameters, the public inputs and a proof π_{PLONK} and outputs `accept` or `reject`. It works in 12 steps, the last of which consists of verifying a pairing check which is the special case of the check (1) for $n = 2$.

4.2.2 Implementation-based Attack Description

In the vulnerable implementation of the KZG-based PLONK verifier V'_{PLONK} that we have examined, the last challenge u is not computed as the hash of the full transcript up to that point as the group elements W_1, W_2 are omitted. This makes the hypothetical attack described in Section 4.1 directly applicable to the KZG-based PLONK verifier implementation we have examined. Below we describe the detailed attack on the vulnerable PLONK implementation following the steps defined in Section 4.1 and pointing out differences where necessary.

Step 1. From a well-formed PLONK proof computed correctly using any circuit of its choice, a malicious prover P'_{PLONK} produces A, B such that (10) holds. Alternatively, P'_{PLONK} can produce appropriate values A and B following **Step 1.** described in Section 4.1.

Step 2. P'_{PLONK} sets the public input PI and all PLONK proof components except for W_1 and W_2 to any arbitrary values of its choice in preparation to produce a proof π'_{PLONK} of a false statement. π'_{PLONK} will have the same number and type of components as π_{PLONK} described by relation (12).

Step 3. From the components of π'_{PLONK}, except for W_1 and W_2, and using steps 9, 10 and 11 of the PLONK verifier V_{PLONK} [2, Sect. 8.1]), P'_{PLONK} computes F_1, F_2. Note that W_1 and W_2 are indeed not needed in order for P'_{PLONK} to complete this computation.

Step 4. P'_{PLONK} exploits the fact that u can be computed, and hence, fixed before W_1 and W_2 are computed. This is analogous to **Step 4.** in Section 4.1 and is a direct implication of the fact that the non-interactive deviating verifier V'_{PLONK} does not include W_1 and W_2 as part of the full transcript to be hashed for deriving u. Thus, whenever a solution exists, P'_{PLONK} solves the following system of two linear equations with unknowns W_1, W_2.

$$\begin{cases} W_1 + u \cdot W_2 = A \\ F_1 + u \cdot F_2 + \mathfrak{z} \cdot W_1 + u \cdot \mathfrak{z} \cdot \omega \cdot W_2 = B \end{cases} \tag{13}$$

The linear system (13) has a solution if and only if $\mathfrak{z} \cdot u \neq 0$.

Step 5. P'_{PLONK} appropriately fills in the corresponding slots of the proof π'_{PLONK} with the above-computed values W_1 and W_2.

Step 6. If this step of the attack has been reached, the vulnerable verifier V'_{PLONK} accepts the false proof π'_{PLONK} as valid with probability 1.

4.2.3 Proof of concept

A proof of concept (POC) of the vulnerability found and explained above is made publicly available [5]. This Solidity POC is based on a PLONK verifier implementation, and follows the steps described in Section 4.2. It forges a proof for an arbitrary public input selected by the prover.

4.2.4 Implications

In the case of a ZK rollup on Ethereum, the honest PLONK prover's circuit simulates the Ethereum Virtual Machine (EVM). An honest prover can thus prove that blocks of transactions were executed correctly, and a certain state transition is valid. However, a verifier implementation vulnerable to the *Last Challenge Attack* makes it possible for a malicious prover to forge with probability 1 a proof for an invalid state transition. That would allow the malicious prover, for example, to set itself as the owner of all the assets that exist in the rollup. Thus, deviations in KZG-based SNARK verifier code that lead to a *Last Challenge Attack* can have disastrous consequences for the security of the blockchains on which they run.

5 Conclusions

In this work we have introduced the *Last Challenge Attack*, a new type of attack applicable to certain incorrect implementations of the FS transform. Our main contribution is the description of a *Last Challenge Attack* against a concrete vulnerable KZG-based PLONK verifier implementation. This attack was possible because the last challenge defined by the FS transform of the PLONK specification has been replaced with a challenge that does not depend on the entire transcript. In turn, if such a PLONK verifier implementation were to be deployed in production code, it would have allowed a malicious prover to construct proofs of false statements which could have lead to disastrous consequences. A proof of concept of the described attack was implemented and made publicly available.

We have also shown that the applicability of the *Last Challenge Attack* goes beyond the PLONK protocol. More specifically, the *Last Challenge Attack* can be a threat to any batched KZG-based

[5] `https://github.com/OpenZeppelin/publications/tree/main/papers/the-last-challenge-attack/proof-of-concept`

protocol in which the FS transform has not been implemented correctly with respect to the KZG proof batching challenge. Therefore, one of the requirements for avoiding the *Last Challenge Attack* is to ensure that all verifier challenges derived using the FS transform are always computed correctly as prescribed by standard specifications. Mind your Fiat-Shamir-s!

References

[1] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO' 86*, pp. 186–194, 1987.

[2] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge." ePrint 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

[3] A. Kate and I. G. Gregory M Zaverucha, "Constant-size commitments to polynomials and their applications," in *ASIACRYPT10*, pp. 177–194, 2010.

[4] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *CCS'93*, pp. 62–73, 1993.

[5] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *EUROCRYPT '96*, pp. 387–398, 1996.

[6] T. Okamoto, "Provably secure and practical identification schemes and corresponding signature schemes," in *Annual International Cryptology Conference*, 1992.

[7] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited." Cryptology ePrint Archive, Paper 1998/011, 1998. `https://eprint.iacr.org/1998/011`.

[8] Y. T. Kalai, G. N. Rothblum, and R. D. Rothblum, "From obfuscation to the security of fiat-shamir for proofs," in *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37*, pp. 224–251, Springer, 2017.

[9] R. Canetti, Y. Chen, L. Reyzin, and R. D. Rothblum, "Fiat-shamir and correlation intractability from strong kdm-secure encryption," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I 37*, pp. 91–122, Springer, 2018.

[10] M. Ciampi, R. Parisella, and D. Venturi, "On adaptive security of delayed-input sigma protocols and fiat-shamir nizks," in *Security and Cryptography for Networks: 12th International Conference, SCN 2020, Amalfi, Italy, September 14–16, 2020, Proceedings 12*, pp. 670–690, Springer, 2020.

[11] Z. Brakerski, V. Koppula, and T. Mour, "Nizk from lpn and trapdoor hash via correlation intractability for approximable relations," in *Annual International Cryptology Conference*, pp. 738–767, Springer, 2020.

[12] A. Lombardi and V. Vaikuntanathan, "Fiat-shamir for repeated squaring with applications to ppad-hardness and vdfs," in *Annual International Cryptology Conference*, pp. 632–651, Springer, 2020.

[13] J. Miller, "Coordinated disclosure of vulnerabilities affecting Girault, Bulletproofs, and Plonk." Trail of Bits Blog, 2022. `https://blog.trailofbits.com/2022/04/13/part-1-coordinated-disclosure-of-vulnerabilities-affecting-girault-bulletproofs-and-plonk/`.

[14] D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios." Cryptology ePrint Archive, Paper 2016/771, 2016. `https://eprint.iacr.org/2016/771`.

[15] E. McMurtry, O. Pereira, and V. Teague, "When is a test not a proof?." Cryptology ePrint Archive, Paper 2020/909, 2020. `https://eprint.iacr.org/2020/909`.

[16] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *CRYPTO 2018*, pp. 33–62, 2018.