The Last Challenge Attack:

Exploiting a Vulnerable Implementation of the Fiat-Shamir Transform in a KZG-based SNARK

Oana Ciobotaru, Maxim Peter, Vesselin Velichkov

OpenZeppelin

ZKProof 6
Berlin, May 22-24, 2024

The Last Challenge Attack: Historical Context

Background of the Finding

- Finding discovered as part of a Linea PLONK verifier audit.
- Initial theoretical concern regarding the underlying Fiat-Shamir (FS) transform implementation.
- The finding was proven exploitable in practice, making it a critical vulnerability.
- Promptly communicated and fixed.
  `https://github.com/Consensys/gnark/security/advisories/GHSA-7p92-x423-vwj6`

OpenZeppelin

The Last Challenge Attack: Historical Context

Background of the Finding

- Finding discovered as part of a Linea PLONK verifier audit.
- Initial theoretical concern regarding the underlying Fiat-Shamir (FS) transform implementation.
- The finding was proven exploitable in practice, making it a critical vulnerability.
- Promptly communicated and fixed.
  `https://github.com/Consensys/gnark/security/advisories/GHSA-7p92-x423-vwj6`

Extension

The attack may affect any SNARK implementation which uses KZG as the polynomial commitment scheme.

OpenZeppelin

Can you solve this system?

Linear system of 2 equations with 2 unknowns*

$$\begin{cases} \boldsymbol{F + z_1 \cdot W_1 + u \cdot z_2 \cdot W_2 + \ldots + u^{n-1} \cdot z_n \cdot W_n = A} \\ \boldsymbol{W_1 + u \cdot W_2 + \ldots + u^{n-1} \cdot W_n = B} \end{cases}$$

with $\boldsymbol{W_1, W_2}$ the unknowns, the rest are known values.

* An attacker would need to solve the above system in the context of elliptic curve points in the first source group w.r.t a pairing. The scalars are elements of the corresponding scalar field.

Solution

A solution $\boldsymbol{(W_1, W_2)}$ exists if and only if $\boldsymbol{u \neq 0}$ and $\boldsymbol{z_1 \neq z_2}$.

OpenZeppelin

Can you solve this system?

Linear system of $\mathbf{2}$ equations with $\mathbf{2}$ unknowns*

$$\begin{cases} \boldsymbol{F + z_1 \cdot W_1 + u \cdot z_2 \cdot W_2 + \ldots + u^{n-1} \cdot z_n \cdot W_n = A} \\ \boldsymbol{W_1 + u \cdot W_2 + \ldots + u^{n-1} \cdot W_n = B} \end{cases}$$

with $\boldsymbol{W_1, W_2}$ the unknowns, the rest are known values.

* An attacker would need to solve the above system in the context of elliptic curve points in the first source group w.r.t a pairing. The scalars are elements of the corresponding scalar field.

Solution

A solution $\boldsymbol{(W_1, W_2)}$ exists if and only if $\boldsymbol{u \neq 0}$ and $\boldsymbol{z_1 \neq z_2}$.

You have just learned about the core of the Last Challenge Attack!

The Last Challenge Attack (LCA) in a Nutshell

Main Idea

- **Overview:** Targets incorrect implementations of the Fiat-Shamir (FS) transform for KZG-based SNARK verifiers*.
- **Concrete setting:** The last FS challenge $\boldsymbol{u}$ is computed incorrectly as independent of certain components of the argument** $\boldsymbol{\pi}$.
- **Outcome:** Enables a malicious SNARK prover to compute an argument $\boldsymbol{\pi'}$ for a false statement, while $\boldsymbol{\pi'}$ is accepted with high probability as valid by the affected SNARK verifier.

* In fact, LCA may apply to any batched KZG-based protocol in which the FS transform has not been implemented correctly with respect to the KZG proof batching challenge.
** For the purposes of this talk, "argument" and "proof" are used interchangeably.

OpenZeppelin

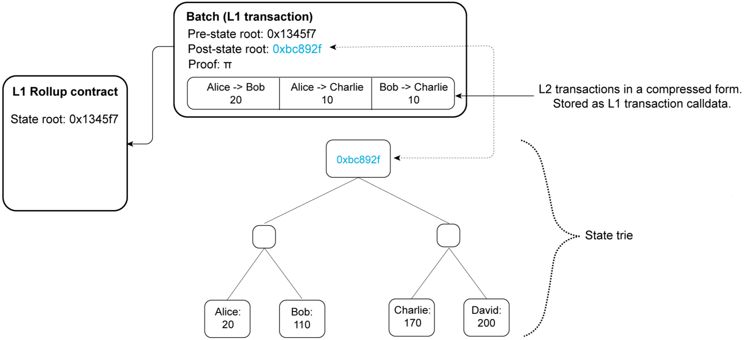
Talk Outline

1. Setting: Scaling Ethereum
2. The Fiat-Shamir Transform
3. The KZG Multipoint Evaluation Scheme
4. The Last Challenge Attack
5. Implications
6. Conclusions

OpenZeppelin

Setting: Scaling Ethereum

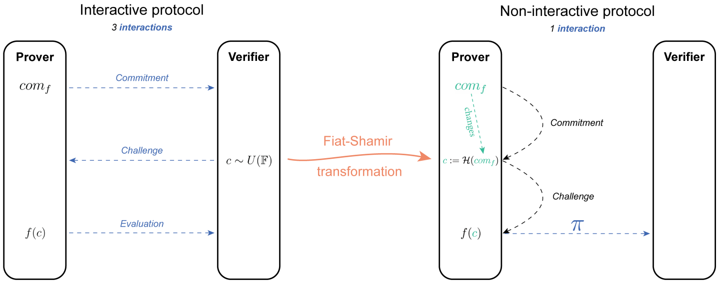
- L2 ZK-Rollups execute transactions off-chain.
- (SNARK) prover $\mathcal{P}$ provides a succinct ZK argument π on L1.
- π testifies that transactions were executed correctly.
- (SNARK) verifier $\mathcal{V}$ verifies on L1 the correctness of π.
- The state of L2 on L1 (and the state of L1) are updated accordingly.



Based on: https://vitalik.eth.limo/general/2021/01/05/rollup.html

OpenZeppelin

Interactive vs. Non-interactive Arguments



The Fiat-Shamir (FS) Transform

- By default, computing $\boldsymbol{\pi}$ is an interactive process between the prover $\boldsymbol{\mathcal{P}}$ and the verifier $\boldsymbol{\mathcal{V}}$.
- The FS transform turns that into a non-interactive process via an idealised random oracle model (ROM).
- In practice, the non-interactive prover and non-interactive verifier independently compute the same unpredictable challenges as the hash of the computation transcript up to that point.

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_{\mathbf{1}}$, $\mathbb{G}_{\mathbf{2}}$ (among others).

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_{\mathbf{1}}$, $\mathbb{G}_{\mathbf{2}}$ (among others).

$gen(d)$

Choose $\boldsymbol{x} \in \mathbb{F}$. Output $\boldsymbol{srs} = (\boldsymbol{g_1}, \boldsymbol{x} \cdot \boldsymbol{g_1}, \ldots, \boldsymbol{x^{d-1}} \cdot \boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{x} \cdot \boldsymbol{g_2}) \in \mathbb{G}_{\mathbf{1}}^{\boldsymbol{d}} \times \mathbb{G}_{\mathbf{2}}^{\mathbf{2}}$.

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_{\mathbf{1}}$, $\mathbb{G}_{\mathbf{2}}$ (among others).

$gen(d)$

Choose $\boldsymbol{x} \in \mathbb{F}$. Output $\boldsymbol{srs} = (\boldsymbol{g_1}, \boldsymbol{x} \cdot \boldsymbol{g_1}, \ldots, \boldsymbol{x^{d-1}} \cdot \boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{x} \cdot \boldsymbol{g_2}) \in \mathbb{G}_{\mathbf{1}}^{d} \times \mathbb{G}_{\mathbf{2}}^{\mathbf{2}}$.

$com(f, srs)$

Output $\mathbf{cm} = \boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{g_1}$.

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_1$, $\mathbb{G}_2$ (among others).

***gen*(d)**

Choose $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2) \in \mathbb{G}_1^d \times \mathbb{G}_2^2$.

***com*(f, srs)**

Output $\mathsf{cm} = f(x) \cdot g_1$.

***open*($\{\mathsf{cm}_j\}_{j=1}^t, z, \{s_j\}_{j=1}^t$)**

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_1$, $\mathbb{G}_2$ (among others).

gen(d)

Choose $\boldsymbol{x} \in \mathbb{F}$. Output $\boldsymbol{srs} = (\boldsymbol{g_1}, \boldsymbol{x} \cdot \boldsymbol{g_1}, \ldots, \boldsymbol{x}^{\boldsymbol{d}-1} \cdot \boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{x} \cdot \boldsymbol{g_2}) \in \mathbb{G}_1^{\boldsymbol{d}} \times \mathbb{G}_2^{2}$.

com(f, srs)

Output $\mathbf{cm} = \boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{g_1}$.

open $(\{\mathbf{cm}_j\}_{j=1}^{t}, z, \{s_j\}_{j=1}^{t})$

1. *Round 1:* $\boldsymbol{V_{KZG}}$ sends uniformly random $\boldsymbol{\gamma} \in \mathbb{F}$ to $\boldsymbol{P_{KZG}}$.

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_1$, $\mathbb{G}_2$ (among others).

$gen(d)$

Choose $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2) \in \mathbb{G}_1^d \times \mathbb{G}_2^2$.

$com(f, srs)$

Output $\mathsf{cm} = f(x) \cdot g_1$.

$open(\{\mathsf{cm}_j\}_{j=1}^t, z, \{s_j\}_{j=1}^t)$

1. *Round 1:* $\boldsymbol{V_{KZG}}$ sends uniformly random $\gamma \in \mathbb{F}$ to $\boldsymbol{P_{KZG}}$.
2. *Round 2:* $\boldsymbol{P_{KZG}}$ computes poly $h(X) = \sum_{j=1}^t \gamma^{j-1} \cdot \frac{f_j(x) - f_j(z)}{X - z}$.

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties $\boldsymbol{P_{KZG}}$ (sender/prover) and $\boldsymbol{V_{KZG}}$ (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing $\boldsymbol{e}$, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_1$, $\mathbb{G}_2$ (among others).

$gen(d)$

Choose $\boldsymbol{x} \in \mathbb{F}$. Output $\boldsymbol{srs} = (\boldsymbol{g_1}, \boldsymbol{x} \cdot \boldsymbol{g_1}, \ldots, \boldsymbol{x}^{d-1} \cdot \boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{x} \cdot \boldsymbol{g_2}) \in \mathbb{G}_1^d \times \mathbb{G}_2^2$.

$com(f, srs)$

Output $\mathbf{cm} = \boldsymbol{f}(\boldsymbol{x}) \cdot \boldsymbol{g_1}$.

$open(\{\mathbf{cm}_j\}_{j=1}^t, z, \{s_j\}_{j=1}^t)$

1. *Round 1:* $\boldsymbol{V_{KZG}}$ sends uniformly random $\boldsymbol{\gamma} \in \mathbb{F}$ to $\boldsymbol{P_{KZG}}$.
2. *Round 2:* $\boldsymbol{P_{KZG}}$ computes poly $\boldsymbol{h}(\boldsymbol{X}) = \sum_{j=1}^t \gamma^{j-1} \cdot \frac{f_j(x) - f_j(z)}{X - z}$.

   Using the $\boldsymbol{srs}$, $\boldsymbol{P_{KZG}}$ computes and sends to $\boldsymbol{V_{KZG}}$ the KZG proof $\boldsymbol{\pi_{KZG}} = \boldsymbol{W}$, where $\boldsymbol{W} = \boldsymbol{h}(\boldsymbol{x}) \cdot \boldsymbol{g_1}$.

Reminder: The KZG Polynomial Commitment Scheme

It assumes parties P_{KZG} (sender/prover) and V_{KZG} (recipient/verifier).
It requires a pairing friendly elliptic curve and, hence, an associated secure pairing e, a scalar field $\mathbb{F}$, two pairing source groups $\mathbb{G}_1$, $\mathbb{G}_2$ (among others).

gen(d)

Choose $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2) \in \mathbb{G}_1^d \times \mathbb{G}_2^2$.

com(f, srs)

Output $\mathsf{cm} = f(x) \cdot g_1$.

open($\{\mathsf{cm}_j\}_{j=1}^t, z, \{s_j\}_{j=1}^t$)

1. *Round 1:* V_{KZG} sends uniformly random $\gamma \in \mathbb{F}$ to P_{KZG}.
2. *Round 2:* P_{KZG} computes poly $h(X) = \sum_{j=1}^t \gamma^{j-1} \cdot \frac{f_j(x) - f_j(z)}{X - z}$.

   Using the srs, P_{KZG} computes and sends to V_{KZG} the KZG proof $\pi_{KZG} = W$, where $W = h(x) \cdot g_1$.
3. *Round 3:* V_{KZG} computes $F = \sum_{j=1}^t \gamma^{j-1} \cdot \mathsf{cm}_j - (\sum_{j=1}^t \gamma^{j-1} \cdot s_j) \cdot g_1$.

   V_{KZG} outputs acc if and only if $e(F + z \cdot W, g_2) = e(W, x \cdot g_2)$.

The KZG Multipoint Evaluation Scheme (KZG MES)

Let $n \geq 2$; assume parties P_{KZG} and V_{KZG} and proceed as follows:

$gen(d)$

Choose (secret) random $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2)$.

$com(f, srs)$

Output $\mathbf{cm} = f(x) \cdot g_1$.

OpenZeppelin

The KZG Multipoint Evaluation Scheme (KZG MES)

Let $n \geq 2$; assume parties P_{KZG} and V_{KZG} and proceed as follows:

$gen(d)$

Choose (secret) random $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2)$.

$com(f, srs)$

Output $\mathbf{cm} = f(x) \cdot g_1$.

$open((\{\mathbf{cm}_{i,j}\}_{j \in [t_i]})_{i \in [n]}, \{z_i\}_{i \in [n]}, (\{s_{i,j}\}_{j \in [t_i]})_{i \in [n]})$

The KZG Multipoint Evaluation Scheme (KZG MES)

Let $n \geq 2$; assume parties P_{KZG} and V_{KZG} and proceed as follows:

$gen(d)$

Choose (secret) random $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2)$.

$com(f, srs)$

Output $\mathsf{cm} = f(x) \cdot g_1$.

$open((\{\mathsf{cm}_{i,j}\}_{j \in [t_i]})_{i \in [n]}, \{z_i\}_{i \in [n]}, (\{s_{i,j}\}_{j \in [t_i]})_{i \in [n]})$

1. *Round 1:* V_{KZG} sends uniformly random $\{\gamma_i\}_{i \in [n]} \in \mathbb{F}^n$ to P_{KZG}.

OpenZeppelin

The KZG Multipoint Evaluation Scheme (KZG MES)

Let $n \geq 2$; assume parties P_{KZG} and V_{KZG} and proceed as follows:

***gen*(d)**

Choose (secret) random $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2)$.

***com*(f, srs)**

Output $\mathsf{cm} = f(x) \cdot g_1$.

***open*$((\{\mathsf{cm}_{i,j}\}_{j \in [t_i]})_{i \in [n]}, \{z_i\}_{i \in [n]}, (\{s_{i,j}\}_{j \in [t_i]})_{i \in [n]})$**

1. *Round 1:* V_{KZG} sends uniformly random $\{\gamma_i\}_{i \in [n]} \in \mathbb{F}^n$ to P_{KZG}.
2. *Round 2:* P_{KZG} computes polys $\{h_i(X) = \sum_{j=1}^{t_i} \gamma_i \cdot \frac{f_{i,j}(x) - f_{i,j}(z_i)}{X - z_i}\}_{i \in [n]}$.

OpenZeppelin

The KZG Multipoint Evaluation Scheme (KZG MES)

Let $n \geq 2$; assume parties P_{KZG} and V_{KZG} and proceed as follows:

$gen(d)$

Choose (secret) random $x \in \mathbb{F}$. Output $srs = (g_1, x \cdot g_1, \ldots, x^{d-1} \cdot g_1, g_2, x \cdot g_2)$.

$com(f, srs)$

Output $\mathsf{cm} = f(x) \cdot g_1$.

$open((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}, \{z_i\}_{i\in[n]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]})$

1. *Round 1:* V_{KZG} sends uniformly random $\{\gamma_i\}_{i\in[n]} \in \mathbb{F}^n$ to P_{KZG}.
2. *Round 2:* P_{KZG} computes polys $\{h_i(X) = \sum_{j=1}^{t_i} \gamma_i \cdot \frac{f_{i,j}(x) - f_{i,j}(z_i)}{X - z_i}\}_{i\in[n]}$.

   Using the srs, P_{KZG} computes and sends to V_{KZG} the

   KZG MES proof $\pi_{KZG} = (W_i)_{i=1}^n$, where $W_i = h_i(x) \cdot g_1$.

The KZG Multipoint Evaluation Scheme (cont.)

$\mathbf{open}((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[u]}, \{z_i\}_{i\in[u]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[u]})$

3. *Round 3:*

The KZG Multipoint Evaluation Scheme (cont.)

$\mathbf{open}((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[u]}, \{z_i\}_{i\in[u]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[u]})$

3. *Round 3:*

   $\boldsymbol{V_{KZG}}$ chooses random $\boldsymbol{u} \in \mathbb{F}$.

The KZG Multipoint Evaluation Scheme (cont.)

$\mathbf{open}((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}, \{z_i\}_{i\in[n]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]})$

3. *Round 3:*

   V_{KZG} chooses random $u \in \mathbb{F}$.

   V_{KZG} computes $\{F_i\}_{i\in[n]}$, $F_i = \sum_{j=1}^{t_i} \gamma_i \cdot \mathsf{cm}_{i,j} - (\sum_{j=1}^{t_i} \gamma_i \cdot s_{i,j}) \cdot g_1$

The KZG Multipoint Evaluation Scheme (cont.)

$\mathrm{open}((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}, \{z_i\}_{i\in[n]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]})$

3. *Round 3:*

   $\boldsymbol{V_{KZG}}$ chooses random $\boldsymbol{u} \in \mathbb{F}$.

   $\boldsymbol{V_{KZG}}$ computes $\{F_i\}_{i\in[n]}$, $F_i = \sum_{j=1}^{t_i} \gamma_i \cdot \mathsf{cm}_{i,j} - (\sum_{j=1}^{t_i} \gamma_i \cdot s_{i,j}) \cdot g_1$

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

$$e(F_1 + \ldots + u^{n-1} \cdot F_n + z_1 \cdot W_1 + \ldots + u^{n-1} \cdot z_n \cdot W_n, g_2) =$$
$$= e(W_1 + \ldots + u^{n-1} \cdot W_n, x \cdot g_2).$$

The KZG Multipoint Evaluation Scheme (cont.)

$\mathsf{open}((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}, \{z_i\}_{i\in[n]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]})$

3. *Round 3:*

   $\boldsymbol{V_{KZG}}$ chooses random $\boldsymbol{u} \in \mathbb{F}$.

   $\boldsymbol{V_{KZG}}$ computes $\{\boldsymbol{F_i}\}_{i\in[n]}$, $\boldsymbol{F_i = \sum_{j=1}^{t_i} \gamma_i \cdot \mathsf{cm}_{i,j} - (\sum_{j=1}^{t_i} \gamma_i \cdot s_{i,j}) \cdot g_1}$

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

$$\boldsymbol{e(F_1 + \ldots + u^{n-1} \cdot F_n + z_1 \cdot W_1 + \ldots + u^{n-1} \cdot z_n \cdot W_n, g_2) =}$$
$$\boldsymbol{= e(W_1 + \ldots + u^{n-1} \cdot W_n, x \cdot g_2).}$$

Quick Note (We Come Back To It Later!)

Last challenge $\boldsymbol{u}$ defined in *Round 3* is only computed and used by $\boldsymbol{V_{KZG}}$.

The KZG Multipoint Evaluation Scheme (cont.)

open$((\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}, \{z_i\}_{i\in[n]}, (\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]})$

3. *Round 3:*

   V_{KZG} chooses random $u \in \mathbb{F}$.

   V_{KZG} computes $\{F_i\}_{i\in[n]}$, $F_i = \sum_{j=1}^{t_i} \gamma_i \cdot \mathsf{cm}_{i,j} - (\sum_{j=1}^{t_i} \gamma_i \cdot s_{i,j}) \cdot g_1$

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

$$e(F_1 + \ldots + u^{n-1} \cdot F_n + z_1 \cdot W_1 + \ldots + u^{n-1} \cdot z_n \cdot W_n, g_2) =$$
$$= e(W_1 + \ldots + u^{n-1} \cdot W_n, x \cdot g_2).$$

Quick Note (We Come Back To It Later!)

Last challenge u defined in *Round 3* is only computed and used by V_{KZG}.

Lemma: Security of KZG MES in the AGM

KZG MES has *completeness and knowledge-soundness in the algebraic group model under the Q-DLOG assumption.*

(See proof of Lemma 6 from **ePrint 2024/398** for full details.)

Non-interactive Version of KZG MES and Variants

Properties of the Non-interactive Version of KZG MES

Let $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ be the non-interactive version of KZG MES prover, verifier.

The non-interactive version of KZG MES:

- is obtained by applying the FS transform;
- can be proven secure in the ROM.

OpenZeppelin

Non-interactive Version of KZG MES and Variants

Properties of the Non-interactive Version of KZG MES

Let $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ be the non-interactive version of KZG MES prover, verifier.

The non-interactive version of KZG MES:

- is obtained by applying the FS transform;
- can be proven secure in the ROM.

Reminder 1: All challenges are computed by both $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ as the hash of the entire communication transcript up to that point in the protocol.

OpenZeppelin

Non-interactive Version of KZG MES and Variants

Properties of the Non-interactive Version of KZG MES

Let $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ be the non-interactive version of KZG MES prover, verifier.

The non-interactive version of KZG MES:

- is obtained by applying the FS transform;
- can be proven secure in the ROM.

Reminder 1: All challenges are computed by both $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ as the hash of the entire communication transcript up to that point in the protocol.

Reminder 2: Last challenge $\boldsymbol{u}$ is actually only computed and used by $\boldsymbol{V_{KZGN}}$!

Non-interactive Version of KZG MES and Variants

Properties of the Non-interactive Version of KZG MES

Let $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ be the non-interactive version of KZG MES prover, verifier.

The non-interactive version of KZG MES:

- is obtained by applying the FS transform;
- can be proven secure in the ROM.

Reminder 1: All challenges are computed by both $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ as the hash of the entire communication transcript up to that point in the protocol.

Reminder 2: Last challenge $\boldsymbol{u}$ is actually only computed and used by $\boldsymbol{V_{KZGN}}$!

Assume reducing computational costs is important, including unnecessary hashing!

Non-interactive Version of KZG MES and Variants

Properties of the Non-interactive Version of KZG MES

Let $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ be the non-interactive version of KZG MES prover, verifier.

The non-interactive version of KZG MES:

- is obtained by applying the FS transform;
- can be proven secure in the ROM.

Reminder 1: All challenges are computed by both $\boldsymbol{P_{KZGN}}$, $\boldsymbol{V_{KZGN}}$ as the hash of the entire communication transcript up to that point in the protocol.

Reminder 2: Last challenge $\boldsymbol{u}$ is actually only computed and used by $\boldsymbol{V_{KZGN}}$!

Assume reducing computational costs is important, including unnecessary hashing!

Dilemma

Does non-interactive KZG MES still remain secure (i.e., *knowledge-sound*) if the non-interactive verifier (i.e., a variation on $\boldsymbol{V_{KZGN}}$) computes $\boldsymbol{u}$ as the hash of only a part of the full transcript (e.g., excluding some $\boldsymbol{\pi_{KZG}}$ components)?

OpenZeppelin

The Last Challenge Attack

Let $\boldsymbol{P'_{KZGN}}$ be a malicious non-interactive prover as per below.

Let $\boldsymbol{V'_{KZGN}}$ be the variation on $\boldsymbol{V_{KZGN}}$ verifier computing $\boldsymbol{u}$ as the hash of the full transcript excluding the first two components of $\boldsymbol{\pi_{KZG}}$.

The Last Challenge Attack

Let $\boldsymbol{P'_{KZGN}}$ be a malicious non-interactive prover as per below.

Let $\boldsymbol{V'_{KZGN}}$ be the variation on $\boldsymbol{V_{KZGN}}$ verifier computing $\boldsymbol{u}$ as the hash of the full transcript excluding the first two components of $\boldsymbol{\pi_{KZG}}$.

Steps 1–3

The Last Challenge Attack

Let $\boldsymbol{P'_{KZGN}}$ be a malicious non-interactive prover as per below.

Let $\boldsymbol{V'_{KZGN}}$ be the variation on $\boldsymbol{V_{KZGN}}$ verifier computing $\boldsymbol{u}$ as the hash of the full transcript excluding the first two components of $\boldsymbol{\pi_{KZG}}$.

Steps 1–3

1. **Bootstrapping:**

The Last Challenge Attack

Let $\boldsymbol{P'_{KZGN}}$ be a malicious non-interactive prover as per below.

Let $\boldsymbol{V'_{KZGN}}$ be the variation on $\boldsymbol{V_{KZGN}}$ verifier computing $\boldsymbol{u}$ as the hash of the full transcript excluding the first two components of $\boldsymbol{\pi_{KZG}}$.

Steps 1–3

1. **Bootstrapping:**
   - $\boldsymbol{P'_{KZGN}}$ simulates single instance single evaluation point KZG for arbitrary $\boldsymbol{f(X)} \in \mathbb{F}[\boldsymbol{X}]$ evaluated at arbitrary $\boldsymbol{z} \in \mathbb{F}$.

The Last Challenge Attack

Let P'_{KZGN} be a malicious non-interactive prover as per below.

Let V'_{KZGN} be the variation on V_{KZGN} verifier computing u as the hash of the full transcript excluding the first two components of π_{KZG}.

Steps 1–3

1. **Bootstrapping:**
   - P'_{KZGN} simulates single instance single evaluation point KZG for arbitrary $f(X) \in \mathbb{F}[X]$ evaluated at arbitrary $z \in \mathbb{F}$.
   - Using the inputs to the KZG pairing check, P'_{KZGN} produces $A, B \in \mathbb{G}_1$ such that $e(A, g_2) = e(B, x \cdot g_2)$.

The Last Challenge Attack

Let P'_{KZGN} be a malicious non-interactive prover as per below.

Let V'_{KZGN} be the variation on V_{KZGN} verifier computing u as the hash of the full transcript excluding the first two components of π_{KZG}.

Steps 1–3

1. **Bootstrapping:**
   - P'_{KZGN} simulates single instance single evaluation point KZG for arbitrary $f(X) \in \mathbb{F}[X]$ evaluated at arbitrary $z \in \mathbb{F}$.
   - Using the inputs to the KZG pairing check, P'_{KZGN} produces $A, B \in \mathbb{G}_1$ such that $e(A, g_2) = e(B, x \cdot g_2)$.
2. P'_{KZGN} sets green-font variables to domain-respecting arbitrary values:

The Last Challenge Attack

Let P'_{KZGN} be a malicious non-interactive prover as per below.

Let V'_{KZGN} be the variation on V_{KZGN} verifier computing u as the hash of the full transcript excluding the first two components of π_{KZG}.

Steps 1–3

1. **Bootstrapping:**
   - P'_{KZGN} simulates single instance single evaluation point KZG for arbitrary $f(X) \in \mathbb{F}[X]$ evaluated at arbitrary $z \in \mathbb{F}$.
   - Using the inputs to the KZG pairing check, P'_{KZGN} produces $A, B \in \mathbb{G}_1$ such that $e(A, g_2) = e(B, x \cdot g_2)$.
2. P'_{KZGN} sets green-font variables to domain-respecting arbitrary values:
   - $(\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}$, $\{z_i\}_{i\in[n]}$, $(\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]}$ —> inputs to **open**!

The Last Challenge Attack

Let P'_{KZGN} be a malicious non-interactive prover as per below.

Let V'_{KZGN} be the variation on V_{KZGN} verifier computing u as the hash of the full transcript excluding the first two components of π_{KZG}.

Steps 1–3

1. **Bootstrapping:**
   - P'_{KZGN} simulates single instance single evaluation point KZG for arbitrary $f(X) \in \mathbb{F}[X]$ evaluated at arbitrary $z \in \mathbb{F}$.
   - Using the inputs to the KZG pairing check, P'_{KZGN} produces $A, B \in \mathbb{G}_1$ such that $e(A, g_2) = e(B, x \cdot g_2)$.
2. P'_{KZGN} sets green-font variables to domain-respecting arbitrary values:
   - $(\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}$, $\{z_i\}_{i\in[n]}$, $(\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]}$ —> inputs to **open**!
   - $\pi'_{KZG} = (W_1, W_2, W_3, \ldots, W_n)$ —> $n \geq 2$!

The Last Challenge Attack

Let P'_{KZGN} be a malicious non-interactive prover as per below.

Let V'_{KZGN} be the variation on V_{KZGN} verifier computing u as the hash of the full transcript excluding the first two components of π_{KZG}.

Steps 1–3

1. **Bootstrapping:**
   - P'_{KZGN} simulates single instance single evaluation point KZG for arbitrary $f(X) \in \mathbb{F}[X]$ evaluated at arbitrary $z \in \mathbb{F}$.
   - Using the inputs to the KZG pairing check, P'_{KZGN} produces $A, B \in \mathbb{G}_1$ such that $e(A, g_2) = e(B, x \cdot g_2)$.
2. P'_{KZGN} sets green-font variables to domain-respecting arbitrary values:
   - $(\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}$, $\{z_i\}_{i\in[n]}$, $(\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]}$ —> inputs to **open**!
   - $\pi'_{KZG} = (W_1, W_2, W_3, \ldots, W_n)$ —> $n \geq 2$!

P'_{KZGN} aims to create π'_{KZG} for a false statement!

The Last Challenge Attack

Let P'_{KZGN} be a malicious non-interactive prover as per below.

Let V'_{KZGN} be the variation on V_{KZGN} verifier computing u as the hash of the full transcript excluding the first two components of π_{KZG}.

Steps 1–3

1. **Bootstrapping:**
   - P'_{KZGN} simulates single instance single evaluation point KZG for arbitrary $f(X) \in \mathbb{F}[X]$ evaluated at arbitrary $z \in \mathbb{F}$.
   - Using the inputs to the KZG pairing check, P'_{KZGN} produces $A, B \in \mathbb{G}_1$ such that $e(A, g_2) = e(B, x \cdot g_2)$.
2. P'_{KZGN} sets green-font variables to domain-respecting arbitrary values:
   - $(\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}$, $\{z_i\}_{i\in[n]}$, $(\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]}$ —> inputs to **open**!
   - $\pi'_{KZG} = (W_1, W_2, W_3, \ldots, W_n)$ —> $n \geq 2$!

   P'_{KZGN} aims to create π'_{KZG} for a false statement!
3. Using $(\{\mathsf{cm}_{i,j}\}_{j\in[t_i]})_{i\in[n]}$ and $(\{s_{i,j}\}_{j\in[t_i]})_{i\in[n]}$, P'_{KZGN} deterministically computes $F_1, \ldots, F_n \in \mathbb{G}_1$ following *Round 3* of KZG MES.

The Last Challenge Attack (cont.)

Steps 4–6

The Last Challenge Attack (cont.)

Steps 4–6

4. $\boldsymbol{V'_{KZGN}}$ computes $\boldsymbol{u}$ as the hash of the full transcript excluding $\boldsymbol{W_1}$, $\boldsymbol{W_2}$. **This is deviation from the FS transform!**

The Last Challenge Attack (cont.)

Steps 4–6

4. $\boldsymbol{V'_{KZGN}}$ computes $\boldsymbol{u}$ as the hash of the full transcript excluding $\boldsymbol{W_1}$, $\boldsymbol{W_2}$.
This is deviation from the FS transform!
$\boldsymbol{P'_{KZGN}}$ exploits that by solving the following system where $\boldsymbol{W_1, W_2}$ are the only unknowns:

$$\begin{cases} F + z_1 \cdot \boxed{W_1} + u \cdot z_2 \cdot \boxed{W_2} + \ldots + u^{n-1} \cdot z_n \cdot W_n = A \\ \boxed{W_1} + u \cdot \boxed{W_2} + \ldots + u^{n-1} \cdot W_n = B \end{cases}$$

and the rest are constants as follows (see also Steps 1–3):

$$e(\underbrace{\underbrace{F_1 + \ldots + u^{n-1} \cdot F_n}_{F} + z_1 \cdot W_1 + u \cdot z_2 \cdot W_2 + \ldots + u^{n-1} \cdot z_n \cdot W_n}_{A}, g_2) \stackrel{?}{=}$$

$$\stackrel{?}{=} e(\underbrace{W_1 + u \cdot W_2 + \ldots + u^{n-1} \cdot W_n}_{B}, x \cdot g_2).$$

The Last Challenge Attack (cont.)

Steps 4–6

4. V'_{KZGN} computes u as the hash of the full transcript excluding W_1, W_2.
This is deviation from the FS transform!
P'_{KZGN} exploits that by solving the following system where W_1, W_2 are the only unknowns:

$$\begin{cases} F + z_1 \cdot \boxed{W_1} + u \cdot z_2 \cdot \boxed{W_2} + \ldots + u^{n-1} \cdot z_n \cdot W_n = A \\ \boxed{W_1} + u \cdot \boxed{W_2} + \ldots + u^{n-1} \cdot W_n = B \end{cases}$$

and the rest are constants as follows (see also Steps 1–3):

$$e(\underbrace{\underbrace{F_1 + \ldots + u^{n-1} \cdot F_n}_{F} + z_1 \cdot W_1 + u \cdot z_2 \cdot W_2 + \ldots + u^{n-1} \cdot z_n \cdot W_n}_{A}, g_2) \stackrel{?}{=}$$

$$\stackrel{?}{=} e(\underbrace{W_1 + u \cdot W_2 + \ldots + u^{n-1} \cdot W_n}_{B}, x \cdot g_2).$$

5. P'_{KZGN} fills in the corresponding slots of π'_{KZG} with the values W_1, W_2.

The Last Challenge Attack (cont.)

Steps 4–6

4. $\boldsymbol{V'_{KZGN}}$ computes $\boldsymbol{u}$ as the hash of the full transcript excluding $\boldsymbol{W_1}$, $\boldsymbol{W_2}$.
   This is deviation from the FS transform!
   $\boldsymbol{P'_{KZGN}}$ exploits that by solving the following system where $\boldsymbol{W_1, W_2}$ are the only unknowns:

$$\begin{cases} F + z_1 \cdot \boxed{W_1} + u \cdot z_2 \cdot \boxed{W_2} + \ldots + u^{n-1} \cdot z_n \cdot W_n = A \\ \boxed{W_1} + u \cdot \boxed{W_2} + \ldots + u^{n-1} \cdot W_n = B \end{cases}$$

   and the rest are constants as follows (see also Steps 1–3):

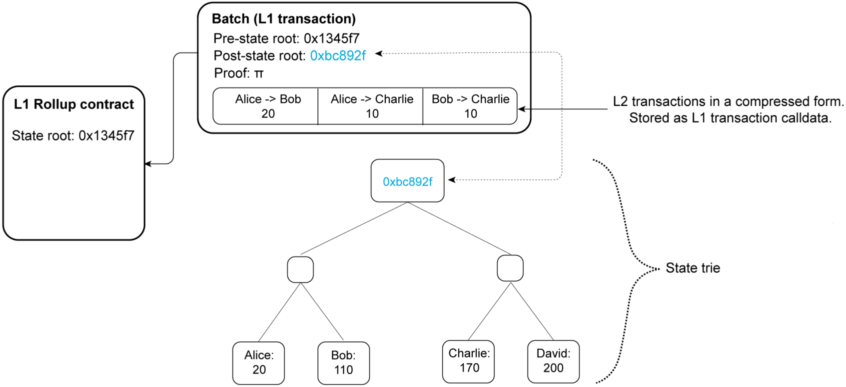
$$e(\underbrace{\underbrace{F_1 + \ldots + u^{n-1} \cdot F_n}_{F} + z_1 \cdot W_1 + u \cdot z_2 \cdot W_2 + \ldots + u^{n-1} \cdot z_n \cdot W_n}_{A}, g_2) \stackrel{?}{=}$$

$$\stackrel{?}{=} e(\underbrace{W_1 + u \cdot W_2 + \ldots + u^{n-1} \cdot W_n}_{B}, x \cdot g_2).$$

5. $\boldsymbol{P'_{KZGN}}$ fills in the corresponding slots of $\boldsymbol{\pi'_{KZG}}$ with the values $\boldsymbol{W_1, W_2}$.
6. $\boldsymbol{V'_{KZGN}}$ accepts proof $\boldsymbol{\pi'_{KZG}}$ as valid with probability 1.

Implications

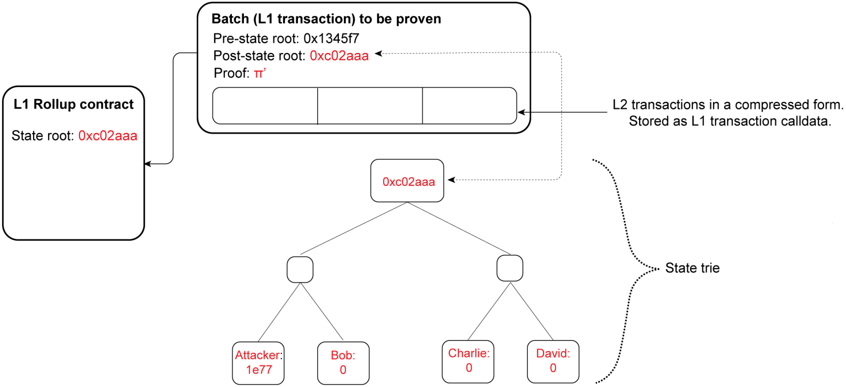
Let $\mathcal{P}'$ be a malicious SNARK prover with a $\boldsymbol{P'_{KZGN}}$ subcomponent. $\mathcal{P}'$ can set itself as the owner of all the assets by changing the Merkle root (part of the **PI**) and steal all user funds.



Based on: `https://vitalik.eth.limo/general/2021/01/05/rollup.html`

Implications

Let $\mathcal{P}'$ be a malicious SNARK prover with a $\boldsymbol{P'_{KZGN}}$ subcomponent. $\mathcal{P}'$ can set itself as the owner of all the assets by changing the Merkle root (part of the **PI**) and steal all user funds.



Based on: `https://vitalik.eth.limo/general/2021/01/05/rollup.html`

Conclusions

- Introduced LCA, a new type of attack on specific incorrect implementations of the FS transform for KZG-based SNARKs.
- LCA exploits the fact that the last challenge defined by the FS transform is incorrectly computed as independent from some of the SNARK proof components.
- LCA is related but different from the weak FS transform attacks occurring when public input or public are parameters not fully incorporated into the transcript.

Takeaways

- FS challenges must depend on the entire transcript up to that point of the computation.
- Follow the protocol!

Challenges can be challenging, so mind your Fiat-Shamir-s!

Thank you!

ePrint 2024/398