ZkHack. On the importance of denting the SRS

Nethermind Research

Notation. We use bracket notation to denote group elements. Namely, for an element $g_1^a \in \mathbb{G}_1$ we write $[a]_1$. Similarly for elements of $\mathbb{G}_2, \mathbb{G}_T$. We use additive notation, that is we write $[a]_1 + [b]_1$ to denote element $g_1^a \cdot g_1^b$. We use $\bullet$ to denote a bilinear pairing between elements $[a]_1$ and $[b]_2$, i.e. we write $[a]_1 \bullet [b]_2 = [ab]_T$.

High level overview of the algorithm. The commitment key of the (broken) argument equals $\mathsf{ck} = \{\left[1, \tau, \ldots, \tau^{2dim-1}\right]_1, \left[1, \ldots, \tau^{dim}\right]_2\}$.

$\mathsf{Commit}(\mathsf{ck}, \boldsymbol{v})$**:** To commit to a vector $\boldsymbol{v} = (v_0, \ldots, v_{dim-1})$ of length dim. We define polynomial $a(X) = v_0 X + v_1 X^2 + \ldots + v_{dim-1} X^{dim}$. The committer computes and outputs the KZG commitment to $a(X)$, i.e. $[a(\tau)]_1$.

$\mathsf{Open}(\mathsf{ck}, [a(\tau)]_1, \boldsymbol{w})$**:** We will denote by $\tilde{\boldsymbol{w}} = (\tilde{w}_0, \ldots, \tilde{w}_{dim-1})$ a vector of length dim such that $\tilde{w}_i = w_{dim-1-i}$, i.e. $\tilde{\boldsymbol{w}}$ is $\boldsymbol{w}$ read from right to left. Given the public vector $\boldsymbol{w}$, the committer computes:

- polynomial $b(X) = (0||\tilde{\boldsymbol{w}})^\top \cdot (1, X, X^2, \ldots, X^{dim})$,
- polynomial $c(X) = a(X) \cdot b(X)$. We note that polynomial's $c(X)$ coefficient to X^{dim+1} equals $v_0 w_0 + \ldots + v_{dim-1} w_{dim-1}$, i.e. the inner product of $\boldsymbol{v}$ and $\boldsymbol{w}$. We denote the inner product by ip. We define a polynomial $d(X) = c(X) - ip \cdot X^{dim+1}$.

Eventually, the committer outputs the KZG commitment to $d(X)$, i.e. $[d(\tau)]_1$.

$\mathsf{Verify}(\mathsf{ck}, [a(\tau)]_1, [d(\tau)]_1, ip, \boldsymbol{w})$**:** ip is the claimed inner product of vectors $\boldsymbol{v}$ and $\boldsymbol{w}$. The verifier computes $b(X) = (0||\tilde{\boldsymbol{w}})^\top \cdot (1, X, X^2, \ldots, X^{dim})$ and accepts if $[a(\tau)]_1 \bullet [b(\tau)]_2 = ip \cdot \left[\tau^{dim}\right]_1 \bullet [\tau]_2 + [d(\tau)]_1 \bullet [1]_2$.

Why it doesn't work? To build an intuition about the attack, we recall the following security assumption (which holds in the algebraic group model): If KZG polynomial commitment's key does not contain element $[\tau^q]_1$, then no adversary $\mathcal{A}$ can commit to a polynomial that has non-zero coefficient to X^q. More precisely, in the AGM, if $\mathcal{A}$ outputs a commitment to a polynomial $f(X)$ that has a non-zero coefficient to X^q, then $\mathcal{A}$ can be used to break a (variant of) discrete logarithm assumption and reveal the trapdoor τ.

We show that a malicious prover can convince the verifier on a false inner product value ip'. We note that polynomial $d(X)$, when computed correctly, has 0 as a coefficient to X^{dim+1}. Moreover, given the assumption above, it is impossible for an adversary to commit to a polynomial that has non-zero coefficient to X^{dim+1} if $\left[\tau^{dim+1}\right]_1$ is not in the SRS. However, that's not the case. Since the argument's SRS contains $\left[\tau^{dim+1}\right]_1$ the adversary can commit to polynomial $d(X)$ with non-zero coefficient to X^{dim+1}. This allows a malicious prover to coin an attack. A malicious prover follows the protocol, except it uses ip' whenever an honest prover would use ip. More precisely, it computes $d(\tau) \leftarrow c(\tau) - ip' \cdot \tau^{dim+1}$. Since the verification equation checks $a(\tau) \cdot b(\tau) = ip' \cdot \tau^{dim+1} + d(\tau)$, what is exactly the case, the verifier accepts the proof.