# On-chain KZG Setup Ceremony - Technical Report

Valeria Nikolaenko, Sam Ragsdale, Joseph Bonneau
a16z crypto research

September 6, 2022

## 1 Notation and setup

We assume three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, each of prime order $p$, with generators $B_1, B_2, B_T$ respectively, addition as a group operation, and a bilinear pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Our goal is to construct a "powers of $\tau$" structured reference string (SRS) of the form:

$$\begin{aligned} \mathsf{pp} &= [\quad P_1, \quad P_2, \quad P_3, \quad \ldots, \quad P_n; \quad P_+] && (1)\\ &= [\tau B_1, \quad \tau^2 B_1, \quad \tau^3 B_1, \quad \ldots, \quad \tau^n B_1; \quad \tau B_2, \tau^2 B_2, \ldots, \tau^k B_2] && (2)\\ &= (\,[\tau]_1, \quad [\tau^2]_1, \quad [\tau^3]_1, \quad \ldots, \quad [\tau^n]_1; \quad [\tau]_2, [\tau^2]_2, \ldots, [\tau^k]_2) && (3) \end{aligned}$$

It is essential that $\tau$ be kept secret in the final string, $\mathsf{pp}$.

The protocol for constructing $\mathsf{pp}$ will be a sequential multi-party computation between $m$ contributors in $m$ rounds, such that each contributor, $C_j$, contributes only in the $j^{\text{th}}$ round. Each contributor can efficiently prove that their participation was correct. The protocol should be secure as long as any individual contributor used good randomness in their round and was honest, i.e. only used locally generated secrets as intended by the protocol and destroyed them successfully after the protocol's completion. In this way it is possible to conduct a permissionless setup in which any contributor is free to contribute, mediated by a smart contract which verifies each participant's contribution.

### 1.1 Initialization

The initial state (after round 0) consists of the string[1]:

$$\begin{aligned} \mathsf{pp} &= [P_{1,0}, \quad P_{2,0}, \quad P_{3,0}, \quad \ldots, \quad P_{n,0}; \quad P_{+,0}] && (4)\\ &= [\;B_1, \quad B_1, \quad B_1, \quad \ldots, \quad B_1; \quad B_2] && (5) \end{aligned}$$

That is, $n$ copies of the generator $B_1$ plus $k$ copies of the generator $B_2$. This is equivalent to an SRS with $\tau = 1$. This is trivially insecure as everybody knows $\tau$, but is trivially easy to check for well-formedness.

### 1.2 Update procedure

At the beginning of round $j$, we assume the current string is of the form:

[1] We focus here on the case when the setup contains only one power in $\mathbb{G}_2$, i.e. when $k = 1$, however it is straightforward to generalize to $k > 1$. It is not strictly necessary to have $k > 1$, but it benefits the efficiency of certain applications, e.g. KZG polynoimal commitments with multi-point evaluation proofs.

$$\begin{aligned}
\mathsf{pp} &= [\ P_{1,j-1}, \quad P_{2,j-1}, \quad P_{3,j-1}, \quad \dots, \quad P_{n,j-1}; \quad P_{+,j-1}] && (6)\\
&= [\tau_{j-1}B_1, \quad \tau_{j-1}^2 B_1, \quad \tau_{j-1}^3 B_1, \quad \dots, \quad \tau_{j-1}^n B_1; \quad \tau_{j-1}B_2] && (7)
\end{aligned}$$

The value $\tau_{j-1}$ is of course hidden. Contributor $C_j$ chooses a random value $r_j \xleftarrow{\$} \mathbb{Z}_p^*$ and publishes a new string:

$$\begin{aligned}
\mathsf{pp} =&[P_{1,j}, & P_{2,j}, & & P_{3,j}, & & \dots, & & P_{n,j}; & & P_{+,j}] && (8)\\
=&[(P_{1,j-1}) \cdot r_j, & (P_{1,j-1}) \cdot r_j^2, & & (P_{1,j-1}) \cdot r_j^3, & & \dots, & & (P_{1,j-1}) \cdot r_j^n; & & (P_{+,j-1}) \cdot r_j] && (9)\\
=&[r_j \tau_{j-1} B_1, & r_j^2 \tau_{j-1}^2 \cdot B_1, & & r_j^3 \tau_{j-1}^3 B_1, & & \dots, & & r_j^n \tau_{j-1}^n B_1; & & r_j \tau_{j-1} B_2] && (10)\\
=&[\tau_j B_1, & \tau_j^2 B_1, & & \tau_j^3 B_1, & & \dots, & & \tau_j^n B_1; & & \tau_j B_2] && (11)
\end{aligned}$$

The new setup has $\tau_j = r_j \cdot \tau_{j-1}$ as its secret. If an attacker knows $\tau_{j-1}$ but not $r_j$, and $r_j$ was chosen uniformly at random from $\mathbb{Z}_p^*$ (meaning in particular that $r_j \neq 0$), then the attacker will have no information about $\tau_j$ (since the operations are done modular a large prime $p$ of roughly 256-bits length). In other words, each new honest contributor randomizes the setup completely. If at least one of the contributors supplies their update, $r_j$, randomly and properly destroys it (and forgets), then the resulting secret ($\tau_m = r_1 \cdot r_2 \cdot \ldots \cdot r_m$) is randomly distributed and unknown to anybody.

## 1.3 Update proofs

Contributor $C_j$ must convince the verifier (the smart contract) that the following three statements are true about its contribution:

1. **The prover knows** $r_j$: a proof that the latest contribution to the ceremony builds on the work of the preceding participants.
2. **The new parameters, $\mathsf{pp}_j$, are well-formed**: the contract should verify that $\mathsf{pp}_j$ consists of consecutive powers of some $\tau_j$.
3. **The update is non-degenerative,** $r_j \neq 0$: a defense against attackers trying to erase the setup thus undermining the contributions of previous participants.

Only if the verifier (smart contract) is convinced that all of the above is true, it updates the setup $\mathsf{pp}$ with the contribution from $C_j$.

We now give the details of how each of these statements is verified on-chain and what proofs (if any) the contributor needs to send to facilitate the verification:

1. **The prover knows** $r_j$. The contributor computes a zero-knowledge proof $\pi$ demonstrating that it knows $r_j$ s.t. $P_{1,j} = P_{1,j-1} \cdot r_j$ The could be a simple Fiat-Shamir version of Schnorr's $\Sigma$-protocol, and it works as follows.

   The prover, the contributor $C_j$, samples a random $z \xleftarrow{\$} \mathbb{Z}_p^*$ and computes

   $$\begin{aligned}
   h &= \mathsf{HASH}(P_{1,j} \mid\mid P_{1,j-1} \mid\mid z \cdot P_{1,j-1})\\
   \pi &= (z \cdot P_{1,j-1}, \;\; z + h \cdot r_j)
   \end{aligned}$$

   where $\mathsf{HASH}$ is a collision-resistant hash function (typically for a 256-bits prime $p$ the hash function needs to output a 512-bits number to argue the uniformity of the distribution of $h$ for zero-knowledge).

   The verifier, the smart contract verifies the proof $\pi = (\pi_1, \pi_2)$ as follows:

   $$P_{1,j-1}^{\pi_2} = P_{1,j}^{\mathsf{HASH}(P_{1,j} \mid\mid P_{1,j-1} \mid\mid \pi_1)} \cdot \pi_1. \tag{12}$$

2. **The new parameters, $\mathsf{pp}_j$, are well-formed**. To verify that $\mathsf{pp}_j$ is correctly formed as stated in Eq. (11), the verifier will sample $n$ random scalars $\rho_0, \rho_1, \ldots, \rho_{n-1} \xleftarrow{\$} \left(\mathbb{Z}_p^*\right)^n$ and verify that:

$$e(\rho \cdot B_1 + \sum_{i=1}^{n-1}(\rho_i \cdot P_{i,j}),\ P_{+,j}) \quad = \quad e(\rho_0 \cdot P_{1,j} + \sum_{i=1}^{n-1}(\rho_i \cdot P_{i+1,j}),\ B_2) \tag{13}$$

For an honest prover this will always hold since:

$$e\left(\rho_0 \cdot B_1 + \sum_{i=1}^{n-1}\left(\tau_j^i \cdot B_1\right) \cdot \rho_i, \quad \tau_j \cdot B_2\right) = e\left(\rho_0 \tau_j B_1 + \sum_{i=1}^{n-1}\left(\tau_j^{i+1} B_1\right) \cdot \rho_i, \quad B_2\right)$$

3. Finally to verify that $r_j \neq 0$ the verifier simply checks that the first element in the new setup is non-zero:

$$P_{1,j} \neq 0. \tag{14}$$

**Correctness:** it is easy to see that an honest prover that updated the setup correctly and produced correct proof $\pi$ will convince the verfier about the correctness of its setup.

**Zero-knowledge:** it is also easy to simulate a satisfying proof with the same distribution for Eq. (12) without knowing $r_j$ by programming the random oracle (using the random-oracle (RO) assumption) as follows. Choose random $w, h \xleftarrow{\$} \mathbb{Z}_p^*$, and set $\pi_1 = P_{1,j-1}^w / P_{1,j}^h$, $\pi_2 = w$ and program the random oracle: $\mathsf{HASH}(P_{1,j} \mid\mid P_{1,j-1} \mid\mid P_{1,j-1}^w) = h$.

**Knowledge soundness:** this is a harder part to argue. We need to show that for any convincing prover the witness $r_j$ can be extracted.

We first prove that if Eq. 13 holds for random $\rho$-strings, then the public setup parameters should be of the form $\mathsf{pp} = [P_1, P_2, \ldots, P_n; P_+] = [\tau B_1, \tau^2 B_1, \ldots, \tau^n B_1; \tau B_2]$ for some $\tau$. We first denote by $\tau$ the "discrete log" of $P_+$ base $B_2$ (i.e, $P_+ = \tau B_2$), and assuming Eq. 13 holds for n distinct random $\rho$-strings, we get that the following holds of a random matrix $\Gamma$ of size $n \times n$ (here $\Gamma$ is constructed by putting $\rho$-elements row by row into a matrix):

$$\tau\Gamma \times \begin{bmatrix} B_1 \\ P_1 \\ P_2 \\ \vdots \\ P_{n-1} \end{bmatrix} = \Gamma \times \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_n \end{bmatrix}$$

We know that with an overwhelming probability a random matrix is invertible, thus we can multiply both parts by $\Gamma^{-1}$ and get:

$$\begin{gathered} \tau B_1 = P_1 \\ \tau P_1 = P_2 \\ \tau P_2 = P_3 \\ \vdots \\ \tau P_{n-1} = P_n \end{gathered}$$

Which immediately implies that the public parameters are of the form:

$$\mathsf{pp} = [P_1, P_2, \ldots, P_n; P_+] = [\tau B_1, \tau^2 B_2, \ldots, \tau^n B_1; \tau B_2].$$

And finally, we exploit knowledge soundness of the $\Sigma$-protocol of the proof $\pi_j$ to extract the discrete log $r_j = \tau_j / \tau_{j-1}$ of $\tau_j B_1$ to the base $\tau_{j-1} B_1$.

## 2 Related Work

Ben-Sasson et al. [BSCG+15] constructed the first protocol to solve the problem of sampling the public parameters for zk-proofs. The protocol requires a pre-commitment phase, and thus relies on the parties to remain available making it challenging to scale this protocol in practice. Bowe et al. [BGG18] instantiated the protocol with Pinnochio for Zcash Sprout. Abdolmaleki et al. [ABL+19] proved the UC-security of this protocol for Groth16.

Another family of protocols grows out of the work of Bowe et al. [BGM17] who designed a protocol for Groth16, the parties do not have to stay on-line, so the protocol scales well in practice. However it requires a random beacon - an auxiliary process that produces publicly verifiable unpredictable and unbiasable randomness. It has two phases, the first phase is commonly referred as powers-of-tau and is the style of setup described above. And the second phase of the ceremony depends on the SNARK circuit. In the work of Kohlweiss, Maller, Siim, and Volkhov [KMSV21] the need of a random beacon in the setup was eliminated.

Vitalik Buterin [But22] suggested a simple way to verify the update to the setup that opens the possibility of a gas-efficient on-chain deployment.

## Acknowledgment

We would like to thank Dan Boneh for useful feedback at the early stages of this work.

## References

[ABL+19] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michał Zajac. Uc-secure crs generation for snarks. In *AfricaCrypt*, 2019.

[BGG18] Sean Bowe, Ariel Gabizon, and Matthew D Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. In *Financial Cryptography*, 2018.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. *Cryptology ePrint Archive*, 2017.

[BSCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *IEEE Symposium on Security and Privacy*, 2015.

[But22] Vitalik Buterin. "How do trusted setups work?". https://vitalik.ca/general/2022/03/14/trustedsetup.html, 2022.

[KMSV21] Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In *AsiaCrypt*, 2021.