Trust, But Verify When Using the Powers of Tau

Karim Baghery
COSIC, KU Leuven, Belgium
firstname.lastname@kuleuven.be
Version date: 09 February 2024

Abstract

To mitigate trust concerns in the setup phase of pairing-based zk-SNARKs, the primary solution has been the sampling of the Structured Reference String (SRS) using an MPC protocol. In 2017, Bowe, Gabizon, and Miers introduced the *Powers of Tau* MPC protocol for sampling a universal SRS, which has since become the main SRS generation protocol for numerous practical projects. The protocol's designers showed that for a circuit with 2^{21} multiplication gates, verifying the universal SRS for Groth16 zk-SNARK could take 55 minutes for a single update. However, they clarified that "*the verification is not run by individual users; it is done by the coordinator and anyone who wishes to verify the transcript of the protocol after completion*". This note demonstrates the importance of verifying the final SRS by either *each* individual end-user or *all* ceremony participants to mitigate potential attacks. We discuss simple attack scenarios that highlight vulnerabilities if *each* end-user or *all* participants fail to verify the final SRS. Additionally, by leveraging batching and aggregating techniques, we introduce an efficient verification algorithm for the (original) Powers of Tau protocol, substantially reducing SRS verification time and making it practical even for large-scale ceremonies. In the case of rejection, a more efficient recursive verification approach aids in identifying malicious parties more effectively. This note aims to enhance procedural understanding of SRS generation ceremonies through the Powers of Tau protocol and improve the reliability of current ceremonies against potential threats.

1 Introduction

A Non-Interactive Zero-Knowledge (NIZK) argument [GMR89, BFM88] for an NP language $\mathcal{L}$ allows a prover, knowing a witness w, to convince a verifier about the truth of a statement x, without leaking extra information about the witness w. A NIZK argument of *knowledge* enables a prover to prove that, for a public statement x she also knows a witness w which hold in an NP relation $R_{\mathcal{L}}$, $(x,w) \in R_{\mathcal{L}}$, without leaking any information about her witness w. Pairing-based Zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs) [PHGR13, Gro16] are a particular family of NIZK arguments of knowledge that due to their *succinct* proofs and *very efficient* verification, they are deployed in a wide range of practical protocols, such as verifiable computations [PHGR13], privacy preserving cryptocurrencies [BCG+14] and smart contracts [KMS+16], verifiable document redacting [NT16, CHK17], ledger verification [BMRS20], zero-knowledge rollups (as layer 2 scaling solutions aiming to increase throughput on blockchains) and so on. In terms of security, a zk-SNARK is expected to satisfy Zero-Knowledge (ZK) and knowledge soundness. ZK ensures the proof does not leak any information about the witness w; formally speaking, the proofs can be simulated without knowing the witness. Knowledge soundness guarantees that no computationally bounded malicious prover can convince honest verifier of a false statement, unless it knows the witness. To achieve (knowledge) soundness and ZK at the same time, zk-SNARKs rely on a Structured Reference String (SRS) which is supposed to be generated by a trusted third party [BFM88].

Currently, to mitigate the level of trust on the setup phase of pairing-based zk-SNARKs, the primary solution is to sample the SRS using a Multi-Party Computation (MPC) protocol. In 2015, Ben-Sasson, Chiesa, Green, Tromer, and Virza [BCG+15] proposed an efficient MPC protocol which allows one to sample the SRS of zk-SNARKs even if all but one of the participants has been corrupted. However, a limitation of their construction is that the number of parties must be known in advance. In 2017, Bowe, Gabizon, and Miers [BGM17] presented an MPC protocol for sampling the SRS of Groth's zk-SNARK [Gro16], which currently is the most efficient pairing-based zk-SNARK. The SRS of Groth's zk-SNARK consists of two groups of elements, where some (referred to as the universal SRS) are independent of the topology of the circuit that encodes the underlying NP language, while the remaining ones depend on the circuit's topology (thus named non-universal or circuit-

specific SRS). Recent constructions of zk-SNARKs, such as [GKM+18, GWC19, CHM+19, CFF+21, RZ21], only require a universal SRS.

Zk-SNARKs with a universal SRS offer an additional advantage in that a single SRS can be reused for proving the satisfiability of different circuits with up to a specific number of gates. This feature allows parties to avoid initiating new trusted setup, which are typically costly in practice.
Bowe, Gabizon, and Miers's MPC protocol [BGM17] consists of two phases, the first of which is referred to as "Powers of Tau". The *Powers of Tau* protocol can be used to sample a universal SRS for all circuits up to a given size. In the second phase, leveraging the universal SRS, the parties generate a circuit-specific SRS for Groth's zk-SNARK.

In the Powers of Tau protocol, a coordinator is used to manage messages between the participants, such that the protocol can be scaled to support hundreds or even thousands of parties. Over the past few years, Powers of Tau ceremonies have gained significant popularity, being employed to generate SRS in various practical projects such as Zcash (Sapling) [BDT17], Ethereum (Semaphore) [Fou21], Filecoin [Gab19], Tornado.cash [Cas20b], among others. Some ceremonies have involved approximately 200 participants, while others have stated to have 1114 individual participants. In Fig. 1, we provide a graphical representation of the Powers of Tau protocol's procedure in different SRS generation ceremonies. As it can be seen, in the ceremonies, there is central coordinator that gets the SRS from one participant and after verification, passes it to the next participants and finally returns the final SRS srs_n. They showed that for a circuit of size 2^{21}, the verification of the universal SRS for Groth's zk-SNARK can take 55 minutes for each update (i.e., each participant). Despite the presence of a coordinator, the output of the protocol is verifiable, and in [BGM17] authors clarified that "the verification is not run by individual users; it is done by the coordinator and anyone who wishes to verify the transcript of the protocol after completion". It is clear that when the coordinator acts honestly, the communication among the parties follows a round-robin pattern, resembling a blockchain of SRSs and proofs. Each subsequent SRS srs_i is an updated version of the previous SRS srs_{i-1} for $i \geq 1$. The participants of the protocol (i.e., ceremonies) contribute to extending the chain with their updated SRS, ultimately resulting in the final SRS as the last block.

Our Contribution. In this short paper, we show the importance of verifying the final SRS in the Powers of Tau protocol by either each individual end-user or all ceremony participants to mitigate potential attacks. We present simple attack scenarios that highlight vulnerabilities if *each* individual end-user or *all* participants fail to verify the final SRS. For instance, we illustrate that a malicious coordinator can surreptitiously cancel or substitute numerous SRS updates without detection if all ceremony participants or each individual end-user does not verify the final SRS.

We demonstrate that in a scenario where each individual end-user does not verify the final SRS themselves, and instead $t < n$ of n participants successfully verify the final SRS srs_n, then, in the best case, end-users would need to trust 1-out-of-t ceremony participants instead of 1-out-of-n. We also show that, considering the vulnerability mentioned above, all current security statements made on the web pages of different ceremonies are not clear enough and can confuse end-users.

Furthermore, we demonstrate that verifying the final SRS of current ceremonies using the original verification algorithm presented in the Powers of Tau protocol is impractical. In some cases, it can take more than a few months for each individual end-user or a ceremony participant. In practice, certain ceremonies have sampled SRSs of size 2^{27} or 2^{28}, with over 75 participants, significantly slowing down the verification process (e.g., in the range of 2-5 months). To address this concern, in the rest of paper, by leveraging batching and aggregating techniques, we introduce a novel verification algorithm for the Powers of Tau protocol, substantially reducing the SRS verification time and making it practical even for large-scale ceremonies. Our evaluations show that for a particular universal SRS [RZ21, BMS23] of size 2^{21}, updated by 128 participants, our proposed algorithm would allow one to verify the final SRS of the ceremony in less than a few minutes.

Additionally, we demonstrate that if the verification of the final SRS is rejected, verifiers can use a more efficient recursive verification approach, recently proposed by Baghery, Mertens, and Sedaghat [BMS23] in the context of the updatable SRS model, to identify the first malicious SRS updater very efficiently. The recursive verification significantly, i.e., logarithmically, reduces the time required to identify a malicious party compared to the naive approach, and further optimizations through pre-computations can improve its practical efficiency.

Finally, we discuss practical strategies that, depending on the parameters of an SRS generation ceremony, can be employed to conduct ceremonies with more efficient SRS verification.

2 Preliminaries

Notations. Throughout, we suppose the security parameter of the scheme and its unary representation to be denoted by λ and 1^λ, respectively. For all positive functions $\varepsilon(\lambda)$, a mapping function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is called negligible function if there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$ we have, $\mathsf{negl}(\lambda) < 1/\varepsilon(\lambda)$. We use $x \leftarrow\!\$\, X$ to denote x sampled uniformly according to the distribution X, and $y \leftarrow\!\$\, A(x)$ to denote the output of a randomized algorithm given a uniformly random string x.
We work with pairing-based groups and use additive and the bracket notation, i.e., in group $\mathbb{G}_\zeta$, $[a]_\zeta = a[1]_\zeta$, where $[1]_\zeta$ is

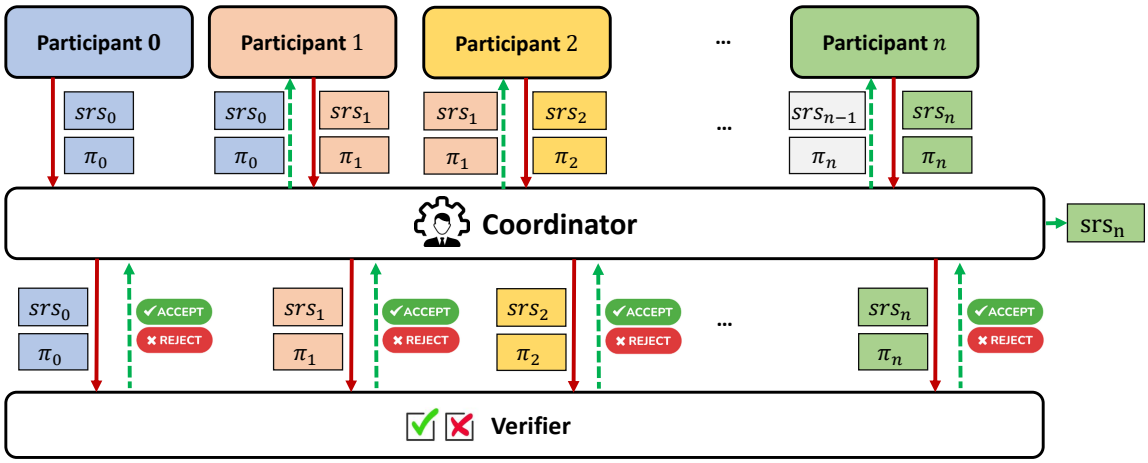


Figure 1: A graphical representation of the Powers of Tau protocol.

the generator of $\mathbb{G}_\zeta$ for $\zeta \in \{1,2,T\}$. A *bilinear group generator* $\mathsf{BGgen}(1^\lambda)$ returns $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2)$, where p (a large prime) is the order of cyclic abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Finally, $\hat{e}:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1,[b]_2)=[ab]_T$. Denote $[a]_1\bullet[b]_2=\hat{e}([a]_1,[b]_2)$.

Zk-SNARKs with Trusted or Updatable and Universal Setups. We adopt the definition of zk-SNARKs with trusted or updatable and universal setups from [GKM+18]. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $R=\{(x,w)\}$, where x is the statement and w is the witness. We assume one can deduce λ from the description of R. Let $\mathcal{L}=\{x:\exists w\mid(x,w)\in R\}$ be an *NP*-language including all the statements which there exist corresponding witnesses in relation R. A NIZK argument Ψ_{NIZK} with universal and updatable SRS for $\mathcal{R}$ consists of the following PPT algorithms:

- $(\mathsf{srs}_0,\pi_0)\leftarrow\mathsf{SG}(R)$: Given R, the SRS generator SG first deduces the upper bound N on the relation size. Next, samples the trapdoor ts and then use it to generate srs_0 along with π_0 as a proof of its well-formedness. Finally, returns (srs_0,π_0) as the output.
- $(\mathsf{srs}_i,\pi_i)\leftarrow\mathsf{SU}(\mathsf{srs}_{i-1},\{\pi_j\}_{j=0}^{i-1})$: Given $(\mathsf{srs}_{i-1},\{\pi_j\}_{j=0}^{i-1})$, an SRS updater SU returns the pair of (srs_i,π_i), where srs_i is the updated SRS and π_i is a proof for correct updating.
- $(\perp/1)\leftarrow\mathsf{SV}(\mathsf{srs}_i,\{\pi_j\}_{j=0}^{i})$: Given a potentially updated srs_i, and $\{\pi_j\}_{j=0}^{i}$, an SRS verifier SV returns either $\perp$ (if srs_i is incorrectly formed or updated) or 1 (if srs_i is correctly formed or updated).
- $(\pi'/\perp)\leftarrow\mathsf{P}(R,\mathsf{srs}_i,x,w)$: Given the tuple of (R,srs_i,x,w), such that $(x,w)\in R$, a prover P outputs an argument π'. Otherwise, it returns $\perp$.
- $(0/1)\leftarrow\mathsf{V}(R,\mathsf{srs}_i,x,\pi')$: Given $(R,\mathsf{srs}_i,x,\pi')$, a verifier V verifies the proof π' and returns either 0 (reject) or 1 (accept).

In the standard SRS model, which the setup phase is supposed to be trusted, a zk-SNARK for $\mathcal{R}$ has a tuple of algorithms $(\mathsf{SG},\mathsf{P},\mathsf{V})$ (and SG does not return the π_0), while in the zk-SNARKs with universal and updatable SRS [GKM+18], we additionally have an SU and SV algorithm which allows various parties to update the SRS with their private shares and verify an updated SRS. Note that in the later case, the algorithm SG does not necessarily need R, and it only deduces security parameter 1^λ and the upper bound N from it.

The Universal SRS of Groth's zk-SNARK [Gro16]. As mentioned earlier, the SRS of Groth's zk-SNARK [Gro16] consists of two groups of elements, with some referred to as the universal SRS. The initial phase of Bowe, Gabizon, and Miers's MPC protocol [BGM17], known as the "Powers of Tau," enables a set of parties to sample the universal SRS of Groth's zk-SNARK. It's noteworthy that the Powers of Tau protocol is general enough to be applied to different forms of universal SRSs.

Next, we recall the list of universal SRS elements in Groth's zk-SNARK, based on the categorization outlined in [BGM17], which will be used later in this paper.

$$\mathsf{srs}\leftarrow\begin{pmatrix}[1]_1,[1]_2,[\alpha]_1,[\beta]_1,[\beta]_2,\{[x^k]_1\}_{k=1}^{2m-2}\\ \{[\alpha x^k]_1,[\beta x^k]_1,[x^k]_2\}_{k=1}^{m},\end{pmatrix}. \quad (1)$$

3 What If the Coordinator is Not Honest?

As previously observed (in Fig. 1), in the Powers of Tau protocol there exists a coordinator which is supposed to honestly manage messages between the participants and also verify all

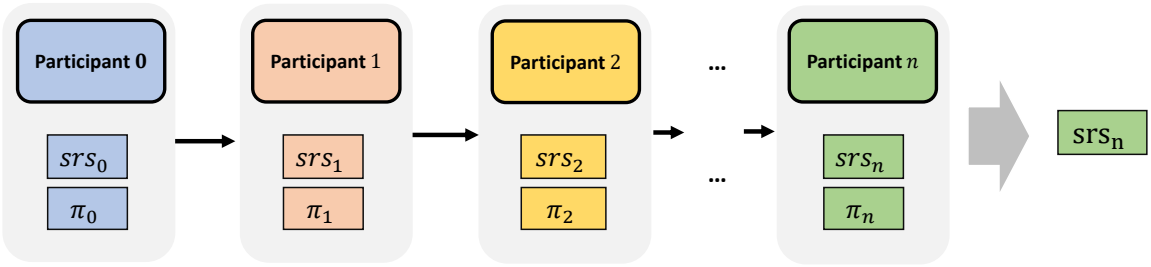


Figure 2: SRS generation ceremony with an honest coordinator acts as a round-robin MPC protocol.

the proofs. For further details related to the protocol we refer to the original paper [BGM17].

Verification of Original Powers of Tau is Impractical. In [BGM17], the authors demonstrated that for a circuit of size $m = 2^{21}$, the verification of the universal SRS for Groth's zk-SNARK, given in eq. (1), can take up to 55 minutes for a party (i.e., an update). Based on that, to verify the final SRS which is generated by n parties (i.e., is updated n times) one has to spent approximately $50 \times n$ minutes in the Powers of Tau protocol. In practical scenarios, some ceremonies have sampled SRSs of size 2^{27} or 2^{28}, and that can significantly slow down the verification process, especially when there are numerous participants in the protocols. As an example, let's consider the "Perpetual Powers of Tau" ceremony conducted by the Semaphore team. In this ceremony, 71 participants utilized the Powers of Tau protocol to generate parameters for circuits containing up to 2^{28} constraints, and in the ceremony each contribution required a 97 Gigabyte download, a 1-day computation, and a 49 Gigabyte upload [Fou21]. According to an email sent on December 5, 2019, from the Semaphore team to a public Google group (sent by one of the coordinators), the verification process for a 15-time updated SRS "took 1.5 weeks on an Azure Standard F4s_v2 (4 vcpus, 8 GiB memory) VM" [Koh19]. Considering this information, verifying the entire transcript of their ceremony involving 71 participants could require over 7 weeks on the same machine. If we extrapolate this to the Tornado.cash ceremony [Cas20b,Cas20a], which have had 1114 participants, assuming the transcript is still available [1], the verification process for their ceremony could take even longer.

In the original paper [BGM17], the designers of the Powers of Tau clarify that,

> "the verification is not run by individual users, it is done by the coordinator and anyone who wishes to verify the transcript of the protocol after completion".

The above statement has likely been the primary reason behind organizing ceremonies involving a significant number of participants without *clearly asking all* the ceremony participants or *each* individual end-users to verify the *final* SRS. This approach can also be attributed to the fact that the original SRS verification algorithm of the Powers of Tau protocol was found to be highly inefficient and impractical. As a result, the emphasis has been placed on relying on the coordinator and offering interested parties the opportunity to verify the transcript retrospectively.

Undetectable Forks by a Malicious Coordinator. Fig. 2 serves as an illustration of a scenario where the coordinator is assumed to be honest. In that case, the SRS update among the parties follows a round-robin pattern, resembling a blockchain of SRSs and proofs. The participants contribute to extending the chain with their updated SRS, eventually obtaining the final SRS after the last update. However, in practical implementations, this may not always be the case. In fact, a malicious coordinator has the capability to create a fork in the original chain and manipulate the generation of the final SRS. For instance, instead of providing participant i with the original SRS srs_{i-1}, the coordinator can intentionally provide a different SRS such as srs_2 (or srs_2 after $(i-3)$-time fake updates). In such cases, if participant i do not verify the entire transcript, namely all the previous SRSs $\{\mathsf{srs}_j\}_{j=1}^{i-1}$ and the proofs $\{\pi_j\}_{j=1}^{i-1}$ on the main chain, the participant cannot be certain that their share is being injected into the original SRS.

Moreover, even if participant i successfully verifies all the previous SRSs and proofs, and proceeds to honestly update the original SRS srs_{i-1}, broadcast srs_i, and then exits the protocol [2], there is still a possibility for a malicious coordinator to exclude their honest share and update from future updates and the final SRS (srs_n in Fig. 2).

Due to the vulnerability mentioned above, statements like,

> "Only one participant needs to do this successfully to ensure the final parameters are secure"

made by Zcash Research Team [BDT17], or the ones like,

[1] Unfortunately, we were unable to locate the transcript of the Tornado.cash ceremony online. Considering the significant number of participants involved, it is reasonable to anticipate that the transcript's size would be in the hundred gigabytes (or even terabytes) range.

[2] Some projects state that their protocol "allowed participants to join the protocol, do their part and leave immediately." For example, see https://z.cash/technology/paramgen/.

> "As long as one party in the ceremony behaves honestly and is not comprimised, the entire setup is trustworthy"

made on Ethereum Foundation Blog [Fou21], made by File Coin Research Team [Gab19], or made on Loopring's Git Page [DW19], or statements like,

> "If at least 1 participant is honest and destroys their secret, then there should be no way for a malicious prover to create fake proofs"

made on Aleo's Git Page [Pru19], and many similar ones, made on different web sites are not complete and clear enough, and they can confuse the participants and the end-users.

The fact that the coordinator(s) should be trusted unless *all* the participants or *each* end-user verify the final SRS, is hidden in all the above statements, and seems their wordings need to be clarified.

In fact, it is crucial to consider the potential risks and attacks associated with relying on a single verifier or a few verifiers, such as the coordinator and a handful of the ceremony participants. As discussed earlier, simple attacks can be executed against the ceremony, nullifying the shares of a significant number of participants, if the participants or each individual end-user fails to verify the final SRS.

One might argue that if at least one participant verifies the final SRS, it is adequate to ensure the security (knowledge soundness) of the underlying zk-SNARK. While this statement holds true, it implies that all other participants and end-users must place their trust solely in that particular participant who verifies the final SRS and broadcasts a public attestation. However, it is clear that this contradicts the primary goal of ceremonies, which is to distribute trust among a large number of participants. If the mentioned participant colludes with the coordinator, they can together nullify the shares of other participants and learn the trapdoors of final SRS. This, in turn, would enable the generation of fake proofs in the underlying zk-SNARK.

4 Trust the Coordinator(s) of Ceremony or Verify the Final SRS

In this section, we explore potential solutions to address the described concern.

Trusting the Coordinator. The most straightforward solution to address the mentioned concern would be to place trust solely in the Coordinator. However, this approach contradicts the fundamental goal of distributed SRS generation, which aims to minimize dependence on a single entity and distribute trust among multiple parties. In reality, if no one verifies the final SRS and the ceremony's transcript, relying solely on the coordinator is essentially equivalent to placing trust in a single party for SRS generation. This undermines the core principle of decentralization and the purpose of involving multiple participants in the SRS generation.

Verifying the Final SRS. Based on our evaluations, if we desire end-users to trust only 1-out-of-the-n participants in the ceremony, then there are generally two solutions:

1. either all ceremony participants individually should verify the final SRS srs_n and publicly share their attestations for successful verification.

2. or each end-user has to independently (successfully) verify the final SRS srs_n, using the original transcript of the protocol (signed by the original participants).

Regarding the first solution, the statements like "we need only 1-out-of-n participants to be honest", holds true only if all the n participants have successfully verified the final SRS srs_n and have publicly attested to it. Otherwise, even if a participant i, where $i < n$, honestly updates the SRS srs_{i-1} after verifying previous updates, if they fail to successfully verify the final CRS srs_n, there is still a possibility of cheating and gaining knowledge of the trapdoor in the final SRS. We assume, the participants in the Powers of Tau protocol do not want themselves to trust another third party. In practice, if only t out of the total n participants in a ceremony (where $t \leq n$) verify the final SRS and publicly provide their attestations, it can be stated that end-users need to trust one out of the t participants who have verified the final SRS. In current ceremonies, some participants have attested that they have participated in the ceremony [3], but to the best of our knowledge, in none of the ceremonies, all the participants (or a large number of them, i.e., $t < n$) have publicly attested that they have successfully verified the final SRS and the entire transcript of the protocol. Therefore still there is a level of trust on the coordinator(s).

Regarding the second solution, it is crucial to ensure that end-users are explicitly informed of their options. They should be aware that, *additional* to trusting 1-out-of-n SRS updaters, they need either trust the coordinator of the ceremony (or someone who has verified the final SRS and the entire transcript), or they can choose to verify the final SRS themselves. However, it appears that this crucial information is not clearly stated on the webpages of several ceremonies or protocols that use the output of these ceremonies. While some ceremonies have implemented and provided the original SRS verification algorithm of the Powers of Tau protocol, along with the transcript of the ceremony, the slow nature of their SRS verification algorithm renders it impractical. Consequently, it is highly likely that many participants have not verified the final SRS despite the availability of these resources. In some cases, such as the Tornado.cash ceremony [Cas20a], we could not find the transcript of the ceremony online. This absence of

[3] See an example on https://twitter.com/IPFSMain/status/1192855448098992129.

accessibility hinders the ability of end-users to verify the ceremony's transcript and forces them to trust the coordinator(s). Considering the significant number of participants involved in their ceremony, namely 1114 participants [Cas20a], the transcript's size of their ceremony can be huge (e.g., in range of terabyte).

In the rest of section, our focus will be on developing more efficient algorithms for SRS verification.

4.1 Verifying the Final SRS: The Naive Approach

As mentioned before, the general solution to deal with the discussed concern is that the final SRS and the whole transcript of the ceremony need to be (successfully) verified by either all the n participants or by each single end-user. A naive solution for this is to use the SRS verification algorithm proposed in the original paper [BGM17]. If the ceremony is ended, the verifier (either a participant or an end-user) needs to run the SRS verification algorithm on whole transcripts of the ceremony, which one-by-one will verify $\{\mathsf{srs}_j, \pi_j\}_{j=1}^{n}$ and will return 1 if all the verifications were successful. In cases the ceremony is not ended yet, a participant can verify the previous SRSs and proofs before his update, and then update the SRS. At the end of ceremony, to get convinced that his contribution or share still is used in the final SRS srs_n, the participant needs to verify all the SRS updates that are done after his/her contribution.

As mentioned before, the participants can also delegate the verification of protocol to the end-users, such that each end-user would need to verify the whole transcript of the ceremony. This way, the participant will never get sure that his/her share is used in the final SRS, and they would need to trust to someone else who has verified the final SRS (using their outputs). If the number of end-users would be larger than the number of SRS updaters, which usually is the case, then the total computational cost of verification would be more than the total computational cost of the case that the participants themselves verify the transcript.

Considering the fact that verifying an SRS of size 2^{21} took 50 minutes in the original Powers of Tau protocol [BGM17], then verifying an n-time updated SRS of the same size can take about $50n$ minutes. If the ceremony is not ended yet, a participant $1 \leq i \leq n$ would need to spend $(i-1)50$ minutes before his update, and $(n-i)50$ minutes to verify the final SRS srs_n. Considering the current real-world numbers for the number of participants, n, and the length of the SRS, e.g., $(n = 87, |\mathsf{srs}| = 2^{22})$ for Zcash [4], $(n = 71, |\mathsf{srs}| = 2^{29})$ for Semaphore [5], $(n = 19, |\mathsf{srs}| = 2^{28})$ for FileCoin [6], and $(n = 1114, |\mathsf{srs}| = 2^{21})$ for Tornado Cash [7], verification can take more than a few weeks or a month. Probably, this is also one reason that currently many of the participants in the ceremonies or the end-users have not verified the final SRS, which highlights that there should be a trust on the coordinator.

4.2 Verifying the Final SRS: A More Efficient Approach

Considering the fact that the original SRS verification algorithm of Powers of Tau protocol is impractical and currently many of the SRS generation ceremonies are ended and their final SRSs are out, a more desire solution can be constructing an efficient SRS verification algorithm that would allow one to verify the final SRS more efficiently. That would allow the participants or end-users to verify the final SRS srs_n once in a reasonable time and will bypass the trust on the coordinator, and only will trust to 1-out-of-n SRS updaters.

In this section, we present an efficient verification algorithm for the original Powers of Tau protocol, which would allow one to verify the final SRS, significantly faster than the verification algorithm discussed in [BGM17]. The presented verification algorithm can be adapted to either of the ceremonies, which might have a different form of monomial SRS. Verifying all the SRSs and proofs, $\{\mathsf{srs}_j, \pi_j\}_{j=1}^{n}$, using the original verification algorithm [BGM17] can be very slow, but it has the advantage that not only a potential malicious behaviour can be detected, but also the adversary (a malicious party, i.e., a participant or the coordinator) can be identified by the verification. However, in practice, we first might only be interested to know if the final SRS is valid, meaning that it is computed using the shares of all participants. Particularity, in the situations that the ceremonies are ended and the parties only want to get sure that their shares are used in the generation of final SRS. After that, if the final SRS srs_n was invalid, then they will be interested to identify the malicious party.

The SRS Verification (SV) algorithm proposed in [BGM17], given srs_i uses pairing product equations and check two sets of equations. The first set is for checking the proof of knowledge given by participant i, and the second set is for verifying the consistency of SRS elements using the already verified proof elements. More formally, the srs_i can be represented as $\mathsf{srs}_i := (\pi_i, \mathsf{srs}'_i)$, where within the SV algorithm, the first set of verification equations verify π_i and if it was successful, then the second set of verification equations verify srs'_i using pairings and the group elements given in π_i. The efficiency of their SV algorithm is dominated by $O(m)$ pairing equations. Hence, to verify an n-time updated SRS, the verifier needs to compute $O(mn)$ pairing equations.

To improve the efficiency of their SRS Verification (SV) algorithm [BGM17], one approach can be using the standard

[4] Stated in https://z.cash/technology/paramgen/.

[5] Stated in https://github.com/weijiekoh/perpetualpowersoftau.

[6] Stated in https://filecoin.io/blog/posts/update-trusted-setup/.

[7] Stated in https://tornado-cash.medium.com/the-biggest-trusted-setup-ceremony-in-the-world-3c6ab9c8fffa.

batching techniques [BGR98] and constructing a Batched SV (BSV) algorithm. This would allow one to verify an n-time updated SRS of length m by $O(nm)$ exponentiations (mostly, short-exponent) and $O(n)$ number of pairing checks. Using the batched variant already can give a huge speed up on the running time of SV algorithm. However, for a large number of participants, i.e., $n \geq 32$ and $m \geq 2^{22}$, still verifying the final SRS srs_n can take more than a few hours, which can be a bottleneck in large-scale ceremonies.

Recently, Kohlweiss, Maller, Siim, and Volkhov [KMSV21] used the idea of updatable SRS model [GKM+18] and modified Bowe et al.'s MPC protocol [BGM17], and presented a simplified and more efficient version of the Powers of Tau protocol. In the updatable SRS model [GKM+18], similar to the Powers of Tau protocol, the participants sequentially update the SRS and return a proof and an updated SRS, and after n times updating the SRS (by n participants), they obtain the final SRS srs_n. But there is a subtle difference in the updatable SRS model, that the SRS verification algorithm does not need all the proofs and intermediate SRSs, i.e., $\{(\pi_j, \mathsf{srs}_j)\}_{j=1}^n$, but only the whole proofs and the final SRS, i.e., $(\{\pi_j\}_{j=1}^n, \mathsf{srs}_n)$. In this way, the SRS verification can be considerably faster, as the participants can wait till the end, and then verify the final SRS srs_n by one time executing the SRS verification algorithm. Using batching techniques [BGR98], again the SRS verification algorithm can get even more efficient. Kohlweiss et al. [KMSV21] removed the need for a random beacon in the Powers of Tau protocol, but they added a random oracle based proof system to the original protocol.

In Fig. 3, similarly, we use the idea of updatable SRS model [GKM+18, KMSV21] and present an efficient SRS Verification (SV) algorithm for the original Powers of Tau protocol. This algorithm can be considerably faster than the original SRS verification algorithm of the Powers of Tau [BGM17]. The reason is that in this case the parties only need to verify the chain of proofs and the final SRS, rather than all the proofs and SRSs. The new SV algorithm allows one to verify the final SRS with $O(m+n)$ pairings, instead of $O(nm)$ pairings. In Fig. 4, we also use the standard batching techniques and present a batched version of our proposed SV (BSV) algorithm. The new BSV algorithm would allow the participants (or an end-user) of the Powers of Tau protocol to verify the final SRS very efficiently. The proposed algorithms can be considered as a variant of the ones proposed in [KMSV21, BMS23], which are adapted for the original Powers of Tau protocol (i.e., for the first round of their MPC protocol [BGM17]). In practice, following the Powers of Tau protocol, since the random beacon is publicly applied after the final SRS update, therefore it does not need to be verified (by the participants), and we assume it is applied after successfully verifying the final SRS srs_n. For further details on this matter, we refer to the original paper [BGM17].

Efficiency and Security of New BSV Algorithm. The proposed BSV algorithm, allows one to verify an n-time updated SRS of length m, in $O(m+n)$ exponentiations (mostly, short-exponent) and $O(n)$ pairings. As we saw earlier, in current practical projects, n is considerably smaller than m, e.g., $m = 2^{28}$, and $n = 19$, therefore the efficiency of the proposed BSV algorithm is dominated with $O(m)$ exponentiations (mostly, short-exponent).

To build the BSV algorithms, we use a corollary of the Schwartz-Zippel lemma stating that if $\sum_{i=1}^{s-1} t_i X_i + X_s = 0$ is a polynomial in $\mathbb{Z}_q[t_i]$ with coefficients $X_1, \ldots, X_s$, $t_i \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $i < s$, then $X_i = 0$ for each i, with probability $1 - 1/2^\kappa$. Namely, if $\sum_{i=1}^{s-1} t_i([a_i]_1 \bullet [b_i]_2) = \sum_{i=1}^{s-1} t_i [c]_T$ for uniformly random t_i, then w.h.p., $[a_i]_1 \bullet [b_i]_2 = [c]_T$ for each individual $i = 1, 2, \cdots, s-1$. In practice, the BSV algorithm can be considerably faster (at the information-theoretical security level 2^{-80}) and even faster at the security level 2^{-40}. It is worth to mention that, using the batching technique comes at the cost of a small loss of soundness: even if the batched equation verifies, there is a probability of at most $2^{-\kappa}$ that one of the non-batched (original) equations was false.

5 Identifying a Malicious SRS Updater

In the previous section, we introduced the BSV algorithm for the Powers of Tau protocol, a highly efficient method for verifying an n-time updated SRS, which enables participants or end-users to perform efficient validation of the final SRS. However, the BSV algorithm solely focuses on verifying the correctness of the final SRS and does not address the identification of malicious parties responsible for incorrect updates. In this section, our main objective is to introduce a more efficient approach specifically tailored to detect malicious participants in similar scenarios.

Recently, Baghery, Mertens, and Sedaghat [BMS23] conducted a study and benchmarked the setup phase of various pairing-based updatable and universal zk-SNARKs. They observed that the constructions in the updatable SRS model can only achieve security with abort if the parties refrain from verifying the updated SRS after each update. In other words, they demonstrated that if the parties wait until the end of the SRS generation protocol to verify only the final SRS srs_n, the SRS verification process will be fast, but the parties will be unable to identify the malicious party (either an updater or the coordinator) in the event of a failed final verification. A naive approach to address this concern would be to run the SRS verification algorithm SV (or BSV) after each update, similar to the original Powers of Tau protocol. However, this would lead to impractical verification and the parties would still need to verify the updates that occurred after their own.

To achieve the best of both worlds - faster SRS verification and identifiable security - Baghery, Mertens, and Sedaghat [BMS23] introduced a recursive SRS verification approach designed to efficiently identify the malicious party (i.e., an SRS updater) in the updatable SRS model. In the event of an unsuccessful verification of the final SRS srs_n,

SRS Verification, $(\perp/1) \leftarrow \mathsf{SV}(\mathsf{srs}_i, (\pi_j)_{j=0}^{i})$**:** To verify (an i-time updated) $\mathsf{srs}_i := (([x_i^k]_1, [x_i^k]_2, [\alpha_i x_i^k]_1)_{k=0}^{m-1}, ([x_i^k]_1)_{k=m}^{2m-2}, ([\beta_i x_i^k]_1)_{k=1}^{m-1}, [\beta_i]_1, [\beta_i]_2$, and the proofs $\pi_j := (\pi_j^{\mathsf{Agg}}, \pi_j^{\mathsf{Ind}}) := (([x_j]_1, [\alpha_j]_1, [\beta_j]_1), ([\bar{x}_j]_1, [\bar{x}_j r_{\bar{x}_j}]_2, [\bar{\alpha}_j]_1, [\bar{\alpha}_j r_{\bar{\alpha}_j}]_2, [\bar{\beta}_j]_1, [r_{\bar{\beta}_j} \bar{\beta}_j]_2, \zeta_j)$ for $j = 0, 1, \cdots, i$, act as follows. Recall that ζ_j denotes the transcript of the protocol after j-th update (by party P_j), RO stands for Random Oracle, and from an honset update we know that $\eta_j = \bar{\eta}_j \eta_{j-1}$, where $\bar{\eta}_j$ denotes the share of party P_j for $\eta \in \{x, \alpha, \beta\}$.

1. **Verifying the individual proofs**. For $j = 1, 2, \cdots, i$:
   - Set $[r_{\bar{x}_j}]_2 \leftarrow RO([\bar{x}_j]_1, \zeta_{j-1}) \in \mathbb{G}_2$, & check if $[\bar{x}_j]_1 \bullet [r_{\bar{x}_j}]_2 = [1]_1 \bullet [\bar{x}_j r_{\bar{x}_j}]_2$;
   - Set $[r_{\bar{\alpha}_j}]_2 \leftarrow RO([\bar{\alpha}_j]_1, \zeta_{j-1}) \in \mathbb{G}_2$, & check if $[\bar{\alpha}_j]_1 \bullet [r_{\bar{\alpha}_j}]_2 = [1]_1 \bullet [\bar{\alpha}_j r_{\bar{\alpha}_j}]_2$;
   - Set $[r_{\bar{\beta}_j}]_2 \leftarrow RO([\bar{\beta}_j]_1, \zeta_{j-1}) \in \mathbb{G}_2$, & check if $[\bar{\beta}_j]_1 \bullet [r_{\bar{\beta}_j}]_2 = [1]_1 \bullet [\bar{\beta}_j r_{\bar{\beta}_j}]_2$;
2. **Verifying the aggregated proofs**. For $j = 1, 2, \cdots, i$:
   - Use $[r_{\bar{x}_j}]_2$ and check if $[x_{j-1}]_1 \bullet [\bar{x}_j r_{\bar{x}_j}]_2 = [x_j]_1 \bullet [r_{\bar{x}_j}]_2$;
   - Use $[r_{\bar{\alpha}_j}]_2$ and check if $[\alpha_{j-1}]_1 \bullet [\bar{\alpha}_j r_{\bar{\alpha}_j}]_2 = [\alpha_j]_1 \bullet [r_{\bar{\alpha}_j}]_2$;
   - Use $[r_{\bar{\beta}_j}]_2$ and check if $[\beta_{j-1}]_1 \bullet [\bar{\beta}_j r_{\bar{\beta}_j}]_2 = [\beta_j]_1 \bullet [r_{\bar{\beta}_j}]_2$;
3. **Verifying (the well-formedness of) the final SRS,** srs_i.
   - Check if $[\beta_i]_1 \bullet [1]_2 = [1]_1 \bullet [\beta_i]_2$;
   - For $k = 1, \ldots, 2m-2$: Check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;
   - For $k = 2, \ldots, m-1$: Check if $[x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [x_i^k]_2$;
   - For $k = 1, \ldots, m-1$: Check if $[\alpha_i]_1 \bullet [x_i^k]_2 = [\alpha_i x_i^k]_1 \bullet [1]_2$;
   - For $k = 1, \ldots, m-1$: Check if $[\beta_i]_1 \bullet [x_i^k]_2 = [\beta_i x_i^k]_1 \bullet [1]_2$;

Return 1 if all the checks passed, otherwise return $\perp$.

Figure 3: SV: An efficient SRS verification algorithm for the Powers of Tau ceremony protocol.

the recursive SRS verification approach allows the parties to identify the initial malicious updater by running the SV (or BSV) algorithm $\lceil \log n \rceil$ times, instead of executing the same algorithm in linear time, i.e., maximum n times. This approach significantly reduces the time required to identify the first malicious party involved. However, it is important to note that the resulting protocol is less robust compared to the SRS verification of original Powers of Tau protocol. In the case of an attack, for example, during update i, the parties would need to repeat the last $n-i$ updates after identifying the malicious updater at step i. To address this concern and minimize the latency in the setup phase of updatable zk-SNARKs, Baghery et al. propose to verify the SRS after a suitable number of updates instead of waiting until the final update (i.e., until the end of ceremony).

The recursive SRS verification approach proposed by Baghery, Mertens, and Sedaghat [BMS23] can also be employed in the verification of Powers of Tau protocol, for verifying the transcripts of completed ceremonies and identifying malicious participants in the case of a failed verification. Specifically, if the verification of the final SRS using the BSV algorithm (shown in Fig. 4) fails, we can leverage their recursive verification approach to efficiently identify the malicious party. Fig. 5 demonstrates an example of the recursive execution of the BSV algorithm for identifying the first malicious updater or the coordinator [BMS23]. In this particular example, after $\lceil \log(15) \rceil + 1 = 5$ iterations of the BSV algorithm, we can detect the first malicious updater, denoted as $(\mathsf{srs}_{11}, \pi_{11})$. Furthermore, their implementation results indicate that both the BSV algorithm and the recursive identification approach are highly practical in real-world scenarios. For instance, when verifying an SRS of the form $\mathsf{srs}_i := ([x^k]_1, [x]_2)_{k=1}^{2^{20}}$, which is utilized in Basilisk zk-SNARK [RZ21], a 1000-time updated SRS can be verified in under 30 seconds. Importantly, in cases where the verification of the final SRS fails, the identification of a malicious updater can be accomplished in less than 5 minutes. Moreover, by employing pre-computation techniques, the identification time can be further improved.

It is important to note that in case of necessity, the proposed SRS verification algorithm can be readily adapted for a specific ceremony with different universal SRS.

6 Ethics Considerations

Data Collection. We collected data, including universal SRS sizes, ceremony transcript sizes, SRS structures, number of participants in the ceremony, and security statements, from public websites and Git pages of the ceremonies.

Batched SRS Verification, $(\perp/1) \leftarrow \mathsf{BSV}(\mathsf{srs}_i, (\pi_j)_{j=0}^{i})$**:** To verify (an i-time updated) $\mathsf{srs}_i := ((\left[x_i^k\right]_1, \left[x_i^k\right]_2, \left[\alpha_i x_i^k\right]_1)_{k=0}^{m-1}, (\left[x_i^k\right]_1)_{k=m}^{2m-2}, (\left[\beta_i x_i^k\right]_1)_{k=1}^{m-1}, [\beta_i]_1, [\beta_i]_2$, and the proofs $\pi_j := (\pi_j^{\mathsf{Agg}}, \pi_j^{\mathsf{Ind}}) := ((\left[x_j\right]_1, \left[\alpha_j\right]_1, \left[\beta_j\right]_1), (\left[\bar{x}_j\right]_1, \left[\bar{x}_j r_{\bar{x}_j}\right]_2, \left[\bar{\alpha}_j\right]_1, \left[\bar{\alpha}_j r_{\bar{\alpha}_j}\right]_2, \left[\bar{\beta}_j\right]_1, \left[r_{\bar{\beta}_j}\bar{\beta}_j\right]_2, \zeta_j)$ for $j = 0, 1, \cdots, i$, act as follows. Note that ζ_j denotes the transcript of the protocol after j-th update (by party P_j), RO stands for Random Oracle, and from an honset update we know that $\eta_j = \bar{\eta}_j \eta_{j-1}$, where $\bar{\eta}_j$ denotes the share of party P_j for $\eta \in \{x, \alpha, \beta\}$.

1. **Verifying the individual proofs**. For $j = 1, 2, \cdots, i$:
   - Sample $\{r_{1,j}, r_{2,j}, r_{3,j}\} \leftarrow_{\$} \{1, \ldots, 2^\kappa\}$;
   - Set $\left[r_{\bar{x}_j}\right]_2 \leftarrow RO(\left[\bar{x}_j\right]_1, \zeta_{j-1}) \in \mathbb{G}_2$, and check if $\sum_j \left(\left[r_{1,j}\bar{x}_j\right]_1 \bullet \left[r_{\bar{x}_j}\right]_2\right) = [1]_1 \bullet \left(\sum_j \left[r_{1,j}\bar{x}_j r_{\bar{x}_j}\right]_2\right)$;
   - Set $\left[r_{\bar{\alpha}_j}\right]_2 \leftarrow RO(\left[\bar{\alpha}_j\right]_1, \zeta_{j-1}) \in \mathbb{G}_2$, and check if $\sum_j \left(\left[r_{2,j}\bar{\alpha}_j\right]_1 \bullet \left[r_{\bar{\alpha}_j}\right]_2\right) = [1]_1 \bullet \left(\sum_j \left[r_{2,j}\bar{\alpha}_j r_{\bar{\alpha}_j}\right]_2\right)$;
   - Set $\left[r_{\bar{\beta}_j}\right]_2 \leftarrow RO(\left[\bar{\beta}_j\right]_1, \zeta_{j-1}) \in \mathbb{G}_2$, and check if $\sum_j \left(\left[r_{3,j}\bar{\beta}_j\right]_1 \bullet \left[r_{\bar{\beta}_j}\right]_2\right) = [1]_1 \bullet \left(\sum_j \left[r_{3,j}\bar{\beta}_j r_{\bar{\beta}_j}\right]_2\right)$;
2. **Verifying the aggregated proofs**. For $j = 1, 2, \cdots, i$:
   - Use $\left[r_{\bar{x}_j}\right]_2$ and check if $\left[x_{j-1}\right]_1 \bullet \left[\bar{x}_j r_{\bar{x}_j}\right]_2 = \left[x_j\right]_1 \bullet \left[r_{\bar{x}_j}\right]_2$;
   - Use $\left[r_{\bar{\alpha}_j}\right]_2$ and check if $\left[\alpha_{j-1}\right]_1 \bullet \left[\bar{\alpha}_j r_{\bar{\alpha}_j}\right]_2 = \left[\alpha_j\right]_1 \bullet \left[r_{\bar{\alpha}_j}\right]_2$;
   - Use $\left[r_{\bar{\beta}_j}\right]_2$ and check if $\left[\beta_{j-1}\right]_1 \bullet \left[\bar{\beta}_j r_{\bar{\beta}_j}\right]_2 = \left[\beta_j\right]_1 \bullet \left[r_{\bar{\beta}_j}\right]_2$;
3. **Verifying (the well-formedness of) the final SRS,** srs_i.
   - For $k = 1, \ldots, m-1$: Sample $\{r_{4,k}, r_{4,m-1+k}, r_{5,k}, r_{6,k}\} \leftarrow_{\$} \{1, \ldots, 2^\kappa\}$;
   - Check if $[\beta_i]_1 \bullet [1]_2 = [1]_1 \bullet [\beta_i]_2$;
   - Check if $\left(\sum_{k=1}^{2m-2} \left[r_{4,k}x_i^k\right]_1\right) \bullet [1]_2 = \left(\sum_{k=1}^{2m-2} \left[r_{4,k}x_i^{k-1}\right]_1\right) \bullet [x_i]_2$;
   - Check if $\left(\sum_{k=2}^{m-1} \left[r_{4,k}x_i^k\right]_1\right) \bullet [1]_2 = [1]_1 \bullet \left(\sum_{k=2}^{m-1} \left[r_{4,k}x_i^k\right]_2\right)$;
   - Check if $[\alpha_i]_1 \bullet \left(\sum_{k=1}^{m-1} \left[r_{5,k}x_i^k\right]_2\right) = \left(\sum_{k=1}^{m-1} \left[r_{5,k}\alpha_i x_i^k\right]_1\right) \bullet [1]_2$;
   - Check if $[\beta_i]_1 \bullet \left(\sum_{k=1}^{m-1} \left[r_{6,k}x_i^k\right]_2\right) = \left(\sum_{k=1}^{m-1} \left[r_{6,k}\beta_i x_i^k\right]_1\right) \bullet [1]_2$;

Return 1 if all the checks passed, otherwise return $\perp$.

Figure 4: BSV: An efficient *batched* SRS verification algorithm for the Powers of Tau ceremony protocol.



Figure 5: Recursive execution of BSV to identify a malicious SRS updater or coordinator in the Powers of Tau based SRS generation ceremonies [BMS23]. Light green: verification was successful. Light red: verification was failed. Dashed red square: Malicious update.

Disclosures. We reported the concern and the hidden trust in the ceremonies more than a year ago (On August 2023) to the individuals who designed the Powers of Tau protocol and the coordinators who utilized the Powers of Tau protocol to conduct the ceremonies. In our (exchanged and follow-up) emails, we provided technical details and verification instructions to facilitate addressing the reported issue (On February 2024). Over a year ago (On February 2024), we shared an updated version of the draft with them, which included more efficient SRS verification algorithms for the original Powers

of Tau protocol. These algorithms can be used and in case of need easily adapted for the transcript of each ceremony, considering potential variations in their universal SRS structures. Our primary communication was with the ZCash, Ethereum, and FileCoin research teams, as others mostly utilized outputs from one of these three ceremonies, and in some instances, coordinators of new ceremonies continued from a previous one.

To avoid any misunderstanding, we explicitly state that, *we have not made any claims regarding the insecurity of any specific ceremony employing the Powers of Tau protocol*. Our presented findings are based on theoretical analysis, and we have not checked or verified the transcripts of any existing ceremony. It is crucial to recognize the substantial efforts undertaken by various projects in conducting ceremonies. This note does not intend to cast doubt on the credibility and security of any specific project; rather, it advocates for clear and robust verification practices within the broader cryptographic community.

7 Conclusion and Remarks

The Powers of Tau MPC protocol [BGM17] has been widely adopted as a means to generate the SRS of zk-SNARKs for various prominent practical projects. In the original paper [BGM17], the authors show that verifying the output of the protocol (i.e., the universal SRS for Groth's zk-SNARK [Gro16] with 2^{21} multiplication gates) can take about $50n$ minutes, for n participants. Then, they stated that the verification is not run by individual users, it is done by the coordinator and anyone who wishes to verify the transcript of the protocol after completion.

In this short paper, we highlighted the importance of verifying the outputs of SRS generation ceremonies based on the Powers of Tau protocol. We showed that if we desire end-users to trust only 1-out-of-the-n participants in the ceremony, then either *all* the participants or *each* individual end-user need to verify the final SRS, using the original transcript of the protocol. To verify the final SRS efficiency, we introduced a practical SRS verification algorithm for the original Powers of Tau protocol. The presented algorithm can be adapted and used to verify the transcript of ceremonies with different form of monomial SRS. We showed that, if the verification of final SRS was failed, given the full transcript of the ceremony, parties can use a recursive verification approach [BMS23] and identify the malicious party (either a participant, or the coordinator).

It is important to note that the new proposed SRS verification algorithm significantly reduces the size of the transcript compared to the original version of the Powers of Tau protocol. With our proposed algorithm, to verify the final SRS, it is sufficient to store the intermediate proofs and the final SRS. This results in a compact transcript size in the ceremonies. However, in cases where the verification of the final SRS fails, the verifiers cannot identify the specific malicious party with only this limited information. To identify a malicious party, access to the complete transcript of the ceremony is necessary. To address this issue, one approach is to designate the coordinator as responsible for storing the entire transcript of the protocol and verifying the SRS after each update. In case of detecting any malicious behaviour like invalid update, the coordinator will raise an alarm and will reject that particular update. At the end of the ceremony, the coordinator would provide the final SRS and the (only) intermediate proofs to the final verifiers, whether they are participants or end-users. This approach ensures both efficiency and accountability in the ceremony protocols. The reduced transcript size and new verification algorithm improves efficiency, while the coordinator's role in verifying and storing the full transcript enhances accountability and facilitates the identification of any malicious participants.

We demonstrated that the final SRS verification needs to be carried out by either *all* participants or *each* individual end-user. The preference between these approaches depends on the specific context and application at hand. In situations where the number of participants in the ceremony is considerably smaller than the number of end-users, and the participants are available, it is likely more convenient to request the participants to verify the final SRS. It should be noted that if only t out of n participants (where $t < n$) verify the final SRS and publicly attest to their successful verification, it implies that end-users would need to trust 1 out of t participants who were involved in the ceremony and the verification of the final SRS. To maintain the motivation and presence of participants for verifying the final SRS, mechanisms like rewarding can be employed. However, in scenarios where the size of the final SRS (including intermediate proofs) is not substantial and the participants are unavailable, an alternative solution would be to involve the end-users in the verification process. Nevertheless, even after the verification by end-users, they would still need to trust at least 1 out of n participants. Therefore, we believe that in practical terms, the former solution would be more efficient. Especially considering that with the first solution, participants can continue the ceremony until they are certain that at least t participants will be present to verify the final SRS and provide a public attestation of their verification.

Acknowledgments

We would like to thank Ian Miers, Janno Siim, Anca Nitulescu, and Mary Maller for their valuable discussions about this note. This work was supported by the Flemish Government through the Cybersecurity Research Program with grant number VOEWICS02. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Cyber Security Research Flanders.

References

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015.

[BDT17] Sean Bowe, Gareth Davies, and Peter Todd. Powers of tau. https://github.com/ebfull/powersoftau., 2017. Published: 2017-11-16, Accessed: 2022-10-22, 04:45.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch Verification with Applications to Cryptography and Checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191, Campinas, Brazil, April 20–24, 1998. Springer, Heidelberg.

[BMRS20] Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352, 2020. https://eprint.iacr.org/2020/352.

[BMS23] Karim Baghery, Axel Mertens, and Mahdi Sedaghat. Benchmarking the setup of updatable zk-snarks. In Abdelrahaman Aly and Mehdi Tibouchi, editors, *Progress in Cryptology - LATINCRYPT 2023 - 8th International Conference on Cryptology and Information Security in Latin America, LATINCRYPT 2023, Quito, Ecuador, October 3-6, 2023, Proceedings*, volume 14168 of *Lecture Notes in Computer Science*, pages 375–396. Springer, 2023.

[Cas20a] Tornado Cash. The biggest trusted setup ceremony in the world. https://tornado-cash.medium.com/the-biggest-trusted-setup-ceremony-in-the-world-3c6ab9c8fffa., 2020. Published: 2020-05-13, Accessed: 2022-10-22, 05:28.

[Cas20b] Tornado Cash. Tornado.cash trusted setup ceremony. https://tornado-cash.medium.com/tornado-cash-trusted-setup-ceremony-b846e1e00be1., 2020. Published: 2020-05-01, Accessed: 2022-10-22, 05:08.

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2021.

[CHK17] Hervé Chabanne, Rodolphe Hugel, and Julien Keuffer. Verifiable document redacting. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part I*, volume 10492 of *LNCS*, pages 334–351. Springer, Heidelberg, September 2017.

[CHM+19] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. Cryptology ePrint Archive, Report 2019/1047, 2019. https://eprint.iacr.org/2019/1047.

[DW19] Brecht Devos and Daniel Wang. Loopring ZKP trusted setup multi-party computation ceremony. https://github.com/Loopring/trusted_setup/blob/loopring-3.6.2/README.md., 2019. Published: 2019-11-10, Accessed: 2022-10-22, 05:11.

[Fou21] Ethereum Foundation. EF-supported teams: Research & development roundup. https://blog.ethereum.org/2021/08/12/ef-supported-teams-research-and-development-update-2021-pt-2., 2021. Published: 2021-08-12, Accessed: 2022-10-22, 04:59.

[Gab19] Ariel Gabizon. Perpetual powers of tau (for BLS381). https://github.com/arielgabizon/perpetualpowersoftau/#ceremony-progress., 2019. Published: 2019-11-05, Accessed: 2022-10-22, 05:05.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[KMS+16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.

[KMSV21] Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 98–127. Springer, Heidelberg, December 2021.

[Koh19] W. J. Koh. Transcript verification up to challenge #15. To Perpetual Powers of Tau Ceremony Google Group, https://groups.google.com/a/ethereum.org/g/perpetual-powers-of-tau-ceremony-group/c/7WiPMo7GbIQ., 2019. Published: 2019-12-5, 01:27, Accessed: 2022-10-22, 04:30.

[NT16] Assa Naveh and Eran Tromer. PhotoProof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy*, pages 255–271. IEEE Computer Society Press, May 2016.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

[Pru19] Alex Pruden. Aleo setup I. https://github.com/AleoHQ/aleo-setup., 2019. Published: 2021-01-15, Accessed: 2022-10-22, 05:25.

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. Cryptology ePrint Archive, Report 2021/590, 2021. https://eprint.iacr.org/2021/590.