SoK: Trusted setups for powers-of-tau strings

Faxing Wang[1,2], Shaanan Cohney[1], and Joseph Bonneau[2,3]

[1] University of Melbourne
[2] New York University
[3] a16z crypto research

Abstract. Many cryptographic protocols rely upon an initial *trusted setup* to generate public parameters. While the concept is decades old, trusted setups have gained prominence with the advent of blockchain applications utilizing zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs), many of which rely on a "powers-of-tau" setup. Because such setups feature a dangerous trapdoor which undermines security if leaked, multiparty protocols are used to prevent the trapdoor from being known by any one party. Practical setups utilize an elaborate public ceremony to build confidence that the setup was not subveted. In this paper, we aim to systematize existing knowledge on trusted setups, drawing the distinction between setup *protocols* and *ceremonies*, and shed light on the different features of various approaches. We establish a taxonomy of protocols and evaluate real-world ceremonies based on their design principles, strengths, and weaknesses.

1 Introduction

Consider the following well-known cryptographic setup procedure: sampling two independent generators in a finite group $g, h \in \mathbb{G}$. Doing so is necessary for a variety of schemes, such as Pedersen commitments [1] and (notoriously) the Dual_EC pseudorandom bit generator [2]. For any two such generators in a cyclic group, there exists a value τ such that $g^\tau = h$ (the discrete logarithm of h to the base g). The value τ is variously called a *trapdoor*, *backdoor* or *toxic waste* [3] in that disclosure of τ undermines the security properties of the system.

The risk of disclosure motivates a variety of approaches to sample g and h such that *no party learns* τ. Given some public constant c and a hash function H which outputs elements in the group $\mathbb{G}$ [4], it suffices to choose $g = H(c||0), h = H(c||1)$. This fits into the class of *public-coin* protocols: all parties can see the value c, which could be sampled from a randomness beacon or the output of a multi-party randomness protocol [5].

Unfortunately, some schemes inherently require secrets or *private coins*. The best-known example is *powers-of-tau* setups, in which the output is a sequence of elements $(g^\tau, g^{\tau^2}, g^{\tau^3}, \cdots, g^{\tau^k})$ in a finite group. Amongst the "Cambrian explosion" of zero-knowledge proof systems [6] used in modern blockchain applications, the most efficient protocols rely on a powers-of-tau-like string. These protocols have real stakes, some with billions of dollars on the line if the trusted setup is compromised. They power proofs for many applications, including scaling blockchains via zk-rollups [7], connecting disparate blockchains via zkBridges [8], and facilitating anonymous payments[9].

Setting up the powers-of-tau string is easy with a simple *trusted setup*: a trusted party can generate the string and promise not to use the trapdoor τ maliciously. It is cliched to note that placing trust in a single entity is risky, as that entity is a single-point-of-failure inherently exposed to targeted attacks. It is also impossible to generally cryptographically prove trapdoor deletion since digital data can be arbitrarily copied and pasted.[4] Another potential solution is to outsource setup to a secure hardware enclave [16]–[19]. However, this model is reliant on significant assumptions as to the security of the enclave, which has been found not to hold in real implementations, [20]–[22].

Instead, practical instantiations bootstrap trust from a multi-party trusted setup protocol. In a typical powers-of-tau setup protocol, each participant ends up with a share τ_i such that $\tau = \prod \tau_i$. An appealing feature is that all parties must collude to reconstruct τ, and if any party deletes their τ_i then τ can never be recovered (ensuring forward secrecy even if an honest party is later corrupted).

While in principle such a protocol could be run once and re-used for many applications, in practice, it is difficult to find a universally trusted party, leading to a number of disparate trusted setups for different applications. Additionally, some proof systems require additional application-specific parameters, requiring distinct setups. Hence, the blockchain space has already seen dozens of powers-of-tau setup protocols. While notions of a trusted setup have long been known in the literature, in practice, the bulk of practical instantiations have been powers-of-tau setups ocurring within the past several years, making them the most important case study for trusted setups.

In this work, we systematize the emerging body of knowledge on trusted setups, specifically for powers-of-tau strings.

Protocols vs. ceremonies. While there are widespread discrepancies in terminology within both the research literature and zk-SNARKs community, we draw a distinction between trusted setup *protocols* and trusted setup *ceremonies*:

- Trusted setup *protocols* which are specified purely mathematically, enabling an arbitrary group of n participants to construct public parameters. They often utilize abstractions such as a public bulletin board or broadcast channels, and come with a rigorous cryptographic security proof.
- Trusted setup *ceremonies* are real-world instantiations of these protocols. They operate with a specific user base or orgnaization and aim to provide operational security, verifiability and social trust among the intended user base. Ceremonies require, for example, implementing the adopted protocol in software, choosing participants, instantiating required tools like bulletin boards, and ensuring public access to data output during the protocol run.

[4] Surprisingly, under specific settings and using quantum computation, it is theoretically possible to produce a proof-of-deletion [10]–[15], but we will not consider these approaches in detail here.

The research literature often conflates these two notions, but we draw a distinction as it allows us to define desiderata for each independently. Both are essential to achieve security. In the current research literature, protocols are highly formalized and often come with formal proofs, while ceremonies are often loosely described and practitioners have had to design many details on the fly. The purpose of this work is not to solve the significant challenge of formalizing all aspects of ceremonies. However, though practitioners have now conducted cryptographic ceremonies over a significant time period, there has until now been insufficient scrutiny of practices at a systematic level. This work aims to partially fill that gap, by reviewing past practices, and presenting the first step towards understanding ceremonies (and their tradeoffs) in context.

Types of setups Adapting from GMR [23], we refer to a setup which requires generating and erasing secrets as a private-coin setup, whereas a secret-free setup is called a public-coin setup. Public-coin setups are often called *transparent setups* [24], [25] in the context of zk-SNARKs. They do not constitute a trusted setup as the trapdoor, though it may be guaranteed to exist mathematically, does not become available to any computationally-bounded party.

By contrast, many zk-SNARK systems (e.g. the popular Groth16 [26]) require a circuit-specific private-coin setup and a new setup for any new circuit to be proved. In contrast, a *universal setup* [27], [28] is one-size-fits-all that can be used in proving any circuit. Once complete, multiple statements can be proved by reusing the materials from the ceremony assuming bounded-size statements. *Updatable setups* [29] enable the parameters to be updated at any time, with cumulative security: if any update is done securely, the trapdoor is hidden permanently. Note that both *universal* and *updatable* setups are independent of any specific circuit.

Why Trusted Setups? From a security standpoint, transparent setups are obviously superior to trusted setups (even universal or updateable ones). Given this, why do we care about the trusted setup model at all? For better or for worse, SNARKs built with private-coin *circuit-specific* setups provide the most efficient proof size, prover time, and verification time. Even if the trusted setup is very complex and slow, it is a one-time cost (which may only be borne by large, powerful nodes participating) and the system may enjoy substantial performance benefits in the long run. For these reasons, as we will see, in practice many real-world blockchain systems have chosen to undertake the risks and costs of trusted setups. Hence, we consider it crucial to design the most secure and efficient trusted setups possible.

Scope. Reviewing and evaluating all the existing *trusted setups* and *ceremonies* is far beyond this SoK. In particular, we focus on powers-of-tau strings and derivatives, though there are many other important types of setups such as groups-of-unknown-order [30]. We also exclude trusted ceremonies like DNSSEC since the corresponding underlying protocols are not trusted setups (they are signing ceremonies which must be repeated periodically).

Paper organization. We begin with preliminaries in **Section 2**, including our algebraic primitives, security assumption, and security definitions for trusted setup protocols. **Section 3** overviews trusted setup protocols and categorizes which applications require which type of protocol. In **Section 4** we discuss practical trusted setup ceremonies, introducing a set of desirable properties for ceremonies and surveying over ceremonies which have been conducted. We conclude (**Section 5**) with insights from all the ceremonies we cover above and open research questions.

2 Preliminaries

We present key notation and notions we will use henceforth. PPT denotes probabilistic polynomial time. We set λ as our security parameter. We use $\mathsf{negl}(\lambda)$ to denote a negligible function that vanishes faster than the inverse of any polynomial of λ. We denote uniform sampling x from a set A as $x \leftarrow\$ A$. For a cyclic group $\mathbb{G}$ with a generator of order q, we denote its generators as g and h.

Definition 1. *q-strong Diffie-Hellman (q-sDH): Given a tuple* $(g, g^x, g^{x^2}, \cdots, g^{x^q}) \in \mathbb{G}^{q+1}$*, where* $x \leftarrow\$ \mathbb{Z}_q$*, q-sDH expects a pair where* $(g^{\frac{1}{x+c}}, c) \in \mathbb{G} \times \mathbb{Z}_q$*. q-sDH is believed to be intractable, that is, for any* PPT *adversary* $\mathcal{A}$*, it is hard to find such a* c*. Formally,*

$$\Pr\left[O = g^{\frac{1}{x+c}} \;\middle|\; (O, c) \leftarrow \mathcal{A}(g, g^x, g^{x^2}, \cdots, g^{x^q})\right] \leq \mathsf{negl}(\lambda) \tag{1}$$

Definition 2. *Powers-of-tau: Upon a given trapdoor secret* τ*, the powers-of-tau is a* k-size *tuple* $(g^{\tau}, g^{\tau^2}, g^{\tau^3}, \cdots, g^{\tau^k})$*. As long as q-sDH is secure, given the powers-of-tau tuple, there is no* PPT $\mathcal{A}$ *able to obtain the trapdoor* τ*.*

2.1 Cryptographic Reference Strings

The community has not employed consistent terminology among the terms *CRS* (Common Reference String), *SRS* (Structured Reference String), and *URS* (Uniform Reference String). In this definition, a CRS can be either a URS or an SRS. We prefer this use, as in practice CRS has been used to mean both.[5] The essential distinction between the two is that an SRS requires a realized trapdoor during setup which must be discarded, whereas a URS is secret-free.

Shamir [31] showed that interactive zero-knowledge proofs are of an equivalent complexity to a much larger complexity class (**IP** = **PSPACE**). However, as a practical matter interaction in the real world implicates additional challenges. Among them are dealing with network latency, synchronizing the prover and verifier, and ensuring the liveness of participants until a proof is complete. These challenges naturally led to an interest in non-interactive zero-knowledge systems

[5] Unhelpfully, CRS is sometimes taken to mean "comment *random* string" with the same initialism CRS.

(NIZKs) [32]. However, non-interactive zero-knowledge proofs have been proven to be impossible in the standard model (in which there is no random oracle) [32]. Instead, NIZK's are constructed under the assumption that participants share a "*random uniform string*": a shared random number sampled by a trusted third party. Such a construction is widely used [33]–[41]. Although originally Blum, Feldman and Micali [32] defined a *CRS* as "*sharing a common, short, random string*", *URS* would have been more appropriate to their context. Formally, with a PPT trusted third party (TTP) running a **Setup** algorithm to generate and distribute a random string, for any PPT prover P, verifier V, and adversary$\mathcal{A}$, a *URS* is a string that has:

- **Randomness**: given a *urs*, it is computationally indistinguishable from a uniform string r of the size l.

$$\Pr\left[\begin{array}{ll} urs \leftarrow\!\!\$_{\mathcal{D}} \mathbf{Setup}(1^\lambda) & \\ r \leftarrow\!\!\$_{\mathcal{D}} \{0,1\}^\lambda & b' \leftarrow \mathcal{A}(\lambda, c) \\ b \leftarrow\!\!\$ \{0,1\} & : b = b' \\ c = \begin{cases} urs, & b = 0 \\ r, & b = 1 \end{cases} & \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \tag{2}$$

where $\mathsf{negl}(\lambda)$ is a negligible function with sufficiently large λ.

- **Uniqueness**: after TTP setting a *urs* for the Prover and the Verifier, urs_P, urs_V, there is but one identical *urs* shared by the two.

$$\Pr\left[\begin{array}{ll} urs \leftarrow\!\!\$_{\mathcal{D}} \mathbf{Setup}(1^\lambda) & (urs'_P, urs'_V) \leftarrow \mathcal{A}(1^\lambda) \\ (urs_P, urs_V) \leftarrow TTP(urs) : & urs'_P = urs'_V \end{array}\right] = 1 \tag{3}$$

The notion of shared randomness naturally leads to considering NIZK constructions. However, it still seems not enough for an efficient construction of a SNARK. Practical zk-SNARKs are based on "*CRS*" like $(g^\tau, g^{\tau^2}, g^{\tau^3}, \cdots, g^{\tau^k})$. It's clear to see that widespread CRSs are essentially SRSs. The embedded algebraic structure of SRS somehow promises us a significant improvement in the performance of proof systems. Essentially, for PPT P and V, given a PPT trusted third party (TTP) running a **Setup** algorithm to generate and distribute random string with a secret s, we define a secure *SRS* such that for any PPT $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\lambda)$ with sufficiently large λ satisfying:

- **Randomness** and **Uniqueness** follow as in definition 2 and definition 3.

- **Correctness**: a *structured reference string* must be of the correct form, relative to some public deterministic polynomial time **Verify** algorithm. A *srs* along with a correctness-proof π of **Setup** must satisfy:

$$\Pr\left[\,(srs, \pi) \leftarrow\!\!\$_{\mathcal{D}} \mathbf{Setup}(1^\lambda, s) \;:\; 1 \leftarrow \mathbf{Verify}(srs, \pi)\,\right] = 1 \tag{4}$$

Remark 1. The correctness proof is essentially a proof-of-knowledge of τ to demonstrate that the SRS is well-formed, after which τ should be deleted. This rules out some attacks by the trusted setup party, but cannot rule out that they maintain knowledge of the trapdoor τ.

- **Secrecy**: the secret s of an SRS is the trapdoor reserved for its inner algebraic structure, such as the τ in $(g^{\tau}, g^{\tau^2}, g^{\tau^3}, \cdots, g^{\tau^k})$. For any PPT adversary $\mathcal{A}$, given an **Setup** output srs and the corresponding correct-proof π, we have

$$\Pr\left[(srs, \pi) \leftarrow_{\$\mathcal{D}} \mathbf{Setup}(1^{\lambda}, s) : \begin{array}{l} s' \leftarrow \mathcal{A}(srs, \pi) \\ s = s' \end{array}\right] < \mathsf{negl}(\lambda) \tag{5}$$

3 Trusted Setup Protocols

We begin by formalizing the notion of a trusted setup *protocol* as a purely mathematical representation of a trusted setup (**Section 3.1**). We then systematize the core ideas of setup protocols for powers-of-tau-like strings (**Section 3.2**).

3.1 Trusted Setup Protocol Formalization

We present a generic definition of trusted setup protocols with formally defined security properties. Our definition aims to capture the core notion of a *process*, which enforces internal computations and communicates with other *processes* through corresponding external *input* and *output* channels. We assume two key pieces of context:

Round-Robin Communication. Power-of-tau trusted setups are typically non-parallelizable, also known as round-robin protocols [42]. They proceed in rounds, with only one party active in each round. The round-robin communication model is a special case of synchronized communication, with only one party allowed to talk in a single round. A *broadcast channel* is a more general synchronized communication channel in which every party may talk in any round and every message is guaranteed to be delivered to every endpoint.

(Dis)Honest Participants. A protocol is secure as long as there are at most $n-1$ dishonest participants, where n is the number of participants. We call participants who follow the protocol description *honest participants* whereas participants with deviant behaviors are called *dishonest participants*. Specifically, we refer to an environment in an *honest majority* setting if over half of the participants are honest. In contrast, if more than half of the participants are dishonest, this reflects the *dishonest majority* setting. Due to the danger of leaking the trapdoor, most powers-of-tau setup protocols are designed to work in a dishonest majority setting.

General Framework: A *process* captures a family of probability distributions among multiple runs indexed by security parameters. A *process* could have multiple runs. A run of a *process* not only includes the *process* description but also the security parameter and all random coins. Note that *processes* could be composed to form a new *process*. We denote *process* A composed with *process* B as $A||B$. Typically, we model *process* as probabilistic polynomial-time systems of probabilistic polynomial-time interactive Turing machines (PPT ITM).

Channels are categorized into two types of visibility: *private, public*. A *public* channel can be seen as broadcasting, whereas *private* channel is modeled as point-to-point (P2P) communication. With external input channels I and output channels O, we denote the set of all *processes* by $\Pi(I,O)$. We denote $I_A = I_B$ if the same data is transmitted in channels I_A, I_B, and I_A and I_B connect the same endpoints.

We proceed with our formal definition of *trusted setup protocols*:

Definition 3. *A trusted setup protocol is a tuple,*
$TS = (S, C, OUT, \{\Pi_p\}_{p\in S}, \{\Pi'_p\}_{p\in S}, t, s)$*, where:*

- $S = (p_1, \cdots, p_n; op)$ *is a finite set of* participants $p_{i\in n}$ *and a special participant* operator op. C *contains all the channels of* TS*;*
- $I, O \in C$ *are channels connecting participants in* S *such that* $O(p \in S)$ *and* $O(p_1 \in S)$ *are disjoint for all* $p \neq p_1$ *and* $I(p)$ *and* $I(p_1)$ *are disjoint for all* $p \neq p_1$ *as well. We call* $I(p)$ *and* $O(p)$ *the set of external* input *and* output *channels of participant* p *respectively;*
- *Explicitly, we define* OUT *to be output of a trusted setup. Note that post-ceremony materials (PCM) are included in* OUT*;*
- $\Pi_{p\in S} \subseteq \Pi(I(p), O(p))$*, is the set of* all the possible runs *of a participant* p*;*
- $\Pi'_{p\in S} \subseteq \Pi_{p\in S}$*, is the set of* all the honest runs *of a participant* p*. Note that the behaviors of participants are reflected in their corresponding runs. Thus, our definition captures both* honest *and* dishonest *participants. An* instance *of* TS *is a process in the form of* $\pi = (\pi_{p_1} || \cdots, || \pi_{p_n})$ *where* $\pi_{p_i} \in \Pi_{a_i}$*. We call a participant* p_i *honest if* $\pi_{p_i} \in \Pi'_{p_i \in S}$*. As trusted setups tolerate corrupted participants, we call a process* π *honest if the number of corrupted participants is within the corruption threshold* t*;*
- s *is the trapdoor of a trusted setup, which is supposed to be known to no one during the lifetime of the trusted setup.*

Desired properties of trusted setup protocols. We propose the following security properties for a trusted setup protocol:

1. **Correctness**: a correct trusted setup should comply with the trusted setup procedure and produce the deterministic output to any two participants on accepted inputs from honest participants. Consider two instances $\pi, \bar{\pi}$ of a trusted setup with the same input $I = \bar{I}, (I, \bar{I}) \in C, \bar{C}$. For any PPT distinguisher D, a correct trusted setup satisfies

$$\Pr\left[\begin{matrix} OUT \leftarrow \pi(S, C, t) \\ \overline{OUT} \leftarrow \pi(S, \bar{C}, t) \end{matrix} \;\middle|\; 1 \leftarrow D(OUT, \overline{OUT})\right] \leq \mathsf{negl}(\lambda) \tag{6}$$

   Informally we define correctness with the following observation: if the output is deterministic, then anyone with the trapdoor t can rerun the trusted setup and get the identical output.

2. γ-ϵ **trapdoor-confidentiality**: Trusted setup contain a secret (the trapdoor). Let s be the trapdoor of the setup procedure π. A trusted setup provides γ-ϵ trapdoor confidentiality if, for any PPT adversary $\mathcal{A}$ who could compromise participants $[p_A]_{A \leq t}$ up to the threshold t.

$$\Pr\left[s' \leftarrow \mathcal{A}(S, C, [\pi^i(I,O)]_{i\in\gamma}, OUT) \;\middle|\; s' = s\right] \leq \mathsf{negl}(\lambda) \tag{7}$$

where π^i indicates that $\mathcal{A}$ is allowed to repeat the instance π with the same S, I, O up to γ times. We model a powerful $\mathcal{A}$ who is able to rewind the instance π of TS. To win the game, the $\mathcal{A}$ must derive the trapdoor with non-negligible probability.

3. **Consistency**: a consistent trusted setup is an *instance* π *of* TS that outputs OUT_i to a participant process π_{p_i} and OUT_j to party j,

$$\Pr\left[\pi_{p_i}, \pi_{p_j} \in \Pi'_{p \in S} | OUT_i = OUT_j\right] = 1 \tag{8}$$

Trusted setup results are intended to be publicly accessible to anyone. More importantly, every honest party's view of the output should be consistent—even for those who did not commit contributions.

4. **(m, ϵ)-Robustness**. we define a trusted setup to be $(m$-$\epsilon)$ robust if and only if the following holds: Let m be the number of participants, $\pi_{p_j\, j \in m} \subseteq \pi$ be the instances of trusted setup TS under the control of $\mathcal{A}$. Then, for channels C with inputs I and outputs O the distribution of the trusted setup output OUT must satisfy:

$$\Pr\left[\begin{array}{l|l} [\pi_{p_j}]_{j \in m}, \pi_{p_j} \notin \Pi'_{p \in S} & \\ m \leq t < n & \\ OUT \leftarrow \pi(S, C, t) & OUT \approx \overline{OUT} \\ \overline{OUT} \leftarrow \pi(\bar{S}, \bar{C}, t) & \end{array}\right] \leq 1 - \mathsf{negl}(\epsilon) \tag{9}$$

where $\bar{S}$ is a set full of honest participants and $\overline{OUT}$ *and* $\bar{C}$ is the corresponding output and channels of the same instance π. The intuition is that an adversary $\mathcal{A}$ who controls m participants must not be able to bias the output of the trusted setup in a non-neglible fashion.

3.2 Powers-of-Tau Setup Protocols

We summarize popular powers-of-tau-based proof systems in Table 1 and polynomial commitment schemes (PCS) in Table 2 (where we highlight the ideal polynomial commitment scheme in red). These two applications are inherently related, many SNARK proof systems are built on KZG commitments [56] which benefit from fast verification and constant-size proofs. In return for these advantages, of course, is the required trusted setup. Transparent proof systems remain significantly more expensive in proof and/or verification time. Developing a PCS with transparent setup, constant-sized proof, and constant verification is still an open question.

KZG commitments depend inherently on an SRS powers-of-tau tuple. Knowledge of the trapdoor τ completely undermines the binding property, which in turn breaks the soundness of every SNARK proof system built on it. This has motivated considerable work on generating the powers-of-tau SRS in a distributed manner, ideally in the dishonest majority model where the trapdoor τ is secure so long as one participant has behaved honestly (and deleted the toxic waste corresponding to their particular contribution to the protocol).

We introduce a toy protocol to illustrate this approach.

SNARKs	SRS size	Universal	Proof Size	Year	Constraint system
Pari[43]	$4a\ \mathbb{G}_1$	✗	$2\ \mathbb{G}_1 + 1\ 2\ \lvert F\rvert$	2024	Square R1CS[44]
Polymath[45]	$(2a+24m)\ \mathbb{G}_1,$	✗	$3\ \mathbb{G}_1 + 1\ \lvert F\rvert$	2024	SAP[26]
Tesutdo[46]	$\sqrt{N}\mathbb{G}_1, \sqrt{N}\mathbb{G}_2,$	✓	$\frac{\sqrt{N}}{2}\mathbb{G}_t, \frac{\sqrt{N}}{2}\mathbb{G}_1, \frac{\sqrt{N}}{2}\mathbb{G}_2$	2023	R1CS[47]
Gemini[48]	$(N+2)\ \mathbb{G}_1, 2\ \mathbb{G}_2$	✓	$3n\ \mathbb{G}_2$	2022	R1CS
Vampire[49]	$(12M+n_K)\ \mathbb{G}_1, (4M+n_k)\ \mathbb{G}_2$	✓	$4\ \lvert\mathbb{G}_1\rvert + 2\ \lvert F\rvert$	2022	R1CSLite, sparse matrices
Basilisk[50]	$M\ \mathbb{G}_1, 1\ \mathbb{G}_2$	✓	$10\ \lvert\mathbb{G}_1\rvert + 3\ \lvert F\rvert$	2021	Plonk constraints
Lunar[51]	$M\ \mathbb{G}_1, M\ \mathbb{G}_2$	✓	$11\ \lvert\mathbb{G}_1\rvert + 2\ \lvert F\rvert$	2020	R1CSLite, sparse matrices
Sonic[27]	$4M\ \mathbb{G}_1,\ 4M\ \mathbb{G}_2$	✓	$20\ \lvert\mathbb{G}_1\rvert + 16\ \lvert F\rvert$	2019	Hadamard Product Constraint[52]
Marlin[28]	$(3n_k+3)\ \mathbb{G}_1, 2\ \mathbb{G}_2$	✓	$13\ \lvert\mathbb{G}_1\rvert + 8\ \lvert F\rvert$	2019	R1CS, sparse matrices
Plonk[53]	$3N\ \mathbb{G}_1, 1\ \mathbb{G}_2$	✓	$7\ \lvert\mathbb{G}_1\rvert + 7\ \lvert F\rvert$	2019	Plonkish[53]
Groth16[26]	$(a+2m)\ \mathbb{G}_1\ ,\ m\ \mathbb{G}_2$	✗	$2\ \mathbb{G}_1, 1\ \mathbb{G}_2$	2016	R1CS,QAP[47]
BCTV14[54]	$(6a+m+l)\ \mathbb{G}_1, m\ \mathbb{G}_2$	✗	$7\ \mathbb{G}_1, 1\ \mathbb{G}_2$	2014	R1CS,QAP
Pinocchio[55]	$(7a+m-2l)\ \mathbb{G}$	✗	$8\ \mathbb{G}$	2013	R1CS,QAP

Table 1. Non-transparent proof systems and their SRS. l linear relations for the left and right inputs. Assume a circuit is of size $N = 2^n$, M is the upper bound of the number of multiplication gates and m is the number of multiplication gates in the circuit. n_k is the number of nonzero entries in R1CS(-lite) matrices encoding the circuit. a is the number of wires in the circuit.

PCS		SRS size	Setup	Open proof size	Prove	Verify
KZG[56]	Pairing	$O(d)\ \mathbb{G}_1$	Univ	$1\ \mathbb{G}_1$	$O(d)$	$O(1)$
Bulletproof[57]	DL	$O(d)\ \mathbb{G}$	Trans	$2logd\ \mathbb{G}$	$O(d)$	$O(d)$
FRI[58]	RO	$O(1)\ \mathbb{G}$	Trans	$\lambda log^2(d)\ \mathbb{G}_1$	$O(\lambda d)$	$O(\lambda log^2(d))$
DARK[24], [52]	GUO	$O(1)\ \mathbb{G}$	Trans	$\lambda log^2(d)\ \mathbb{G}_1$	$O(\lambda d)$	$O(\lambda log^2(d))$
Dory[25]	Pairing	$O(d)\ \mathbb{G}$	Trans	$6log(d)\ \mathbb{G}_1$	$O(d^{1/2})$	$O(log\ d)$
Dew[59]	GUO	$O(1)\ \mathbb{G}$	Trans	$66\ \mathbb{G}_2$	$O(d^3/log\ d)$	$O(log\ d)$
Behemoth[60]	Pairing	$O(d)\ \mathbb{G}$	Trans	$47\ \mathbb{G}_2 + 19\ \mathbb{F}$	$O(d^3/log\ d)$	$O(1)$
KZG[56]	Pairing	$O(n)\ \mathbb{G}_1$	Univ	$v\ \mathbb{G}_1$	$O(n)$	$O(v)$
Dory[25]	Pairing	$O(n)\ \mathbb{G}_1$	Trans	$6v\mathbb{G}_T$	$O(n)$	$O(v)$
Bulletproof[61]	GUO	$O(n)\ \mathbb{G}_1$	Univ	$2v\ \mathbb{G}$	$O(n)$	$O(n)$
Gemini[48]	Pairing	$O(n)\ \mathbb{G}_1$	Univ	$(v+4)\mathbb{G}_1 + (v+1)\mathbb{F}$	$O(n)$	$O(n)$
Brakedown[58]	Coding	$O(n)\ \mathbb{G}$	Trans	$\sqrt{\lambda n}\mathbb{F}$	$O(n)$	$\sqrt{\lambda n}$
Orion[62]	Coding	$O(n)\ \mathbb{G}_1$	Trans	$\lambda v^2\mathbb{G}_1$	$O(n)$	$O(\lambda v^2)$
Zeromorph[63]	Pairing	$O(n)\ \mathbb{G}_1$	Univ	$v + 3\mathbb{G}_1$	$O(n)$	$O(\lambda v^2)$
Orion+[64]	Pairing	$O(n)\ \mathbb{G}_1$	Univ	$4v\mathbb{G}_1$	$O(n)$	$O(v)$
BaseFold[65]	Coding	$O(n)\ \mathbb{G}_1$	Trans	$4v\mathbb{G}_1$	$O(n)$	$O(v)$
Ideal PCS	TBA	$O(1)\ \mathbb{G}$	Trans	$O(1)\mathbb{G}$	$O(d)$	$O(1)$

Table 2. Polynomial Commitment Schemes (PCS). We separate *univariate* PCS and *multilinear PCS* with the dashline. DL is discrete logarithm, RO is random oracle, GUO is groups-of-unknown-order. We let d be the degree of the univariate polynomial, $\mathbb{G}_1, \mathbb{G}_2, and\ \mathbb{G}_T$ be generators of bilinear groups, λ be the security parameter, and $\mathbb{F}$ be a finite field. For the v-variate multilinear polynomial, $n = 2^v$.

Toy powers-of-tau setup protocol. Intuitively, our goal is for parties to contribute secrets to the SRS such that no single party knows the combined and final τ of the SRS. The simplest protocol relies on sequential contributions by participants.

That is, upon receiving the powers-of-tau string $S = (g^{\tau}, g^{\tau^2}, g^{\tau^3}, \cdots, g^{\tau^k})$ from the last participant, participant j samples a random τ_j and obfuscates S by raising each element to the corresponding powers of τ_j:

$$S' = (g^{\tau\tau_j}, g^{(\tau\tau_j)^2}, g^{(\tau\tau_j)^3}, \cdots, g^{(\tau\tau_j)^k})$$

At first glance, this scheme works so long as one participant is honest. However, a malicious participant might *adaptively* generate their secret τ to bias the final SRS. For this reason, some protocols [66]–[69] aim to eliminate the adaptive bias by adding a mandatory commitment layer beforehand. In this case, each party is required to commit to their secret before contributing it and running the remainder of the protocol. Thus each τ_j is guaranteed to be independent even under an adaptive adversary. Obviously, asking parties to commit their τ_j at minimum incurs another round of interaction. In addition, this imposes a significant additional constraint as the participants must be known and remain online throughout the trusted setup, which could take multiple days to finalize in practice.

Additional re-randomization A key of Bowe, Gabizon and Miers [70] is that the last powers-of-tau SRS can be mixed with another random τ' from a public, independent, source. As a result even an adaptively chosen τ_j can not bias the outcome and parties are no longer required to remain online during the setup. Bowe, Gabizon and Miers [70] instantiated the random source with the *random beacon* [71] primitive, designed to periodically produce fresh randomness. This allows the design of a protocol without the commitment phase. Further it permits constructions wherein participants can join the round-robin contribution process in an ad-hoc manner. Participants are thus able to join the SRS trusted setup in their reserved time slot and add their contributions, after which they are free to leave. Restrictions on the identity and the time of participants are also removed. This approach leads us to the MMORPG framework [70].

MMORPG Trusted Setups MMORPG (Massively Multiparty Open Reusable Parameter Generation) is a protocol for generating the SRS used by Groth16 [26] (a pairing-based zk-SNARK). The MMORPG approach has seen broad adoption in systems such as Semaphore [72], TornadoCash [73], and Aztec [74]. Groth16 is not a universal proof system—instead, proofs apply only for the specific program for which they were generated. This leads to a two-phase design, with an initial universal phase and a circuit-specific phase. The universal phase is conducted according to the powers-of-tau private-coin trusted setup we mentioned above. The second phase is a circuit-specific phase that tailors the construction to the specific program for which proofs are to be generated.

Generally, an MMORPG private-coin trusted setup involves three types of entities: the coordinator, the participants, and a randomness beacon (which may be a multiparty protocol itself [5], [75], [76]). We describe it in more detail as follows:

1: The coordinator initializes the protocol by executing a procure $\mathsf{Init}(k, g)$ with the prescribed size of the powers-of-tau tuple k, and a group generator g. After initialization, the coordinator chooses the first party who expressed interest in joining the protocol and sends them the initial powers-of-tau tuple PoT_0^1.
2: After receiving the powers-of-tau tuple from the coordinator (or prior participant), the participant (p_i) first checks the correctness of prior contributions to the string. After checking for correctness, p_i picks a random τ_i to update the power-of-tau tuple in the way we mentioned in the toy proposal and then generates the corresponding correctness proof of their contribution. Specifically, every participant must provide a zero-knowledge proof of the following three properties to ensure the correctness of the tuple: First, the participant must prove knowledge of their corresponding τ_i. Second, the participant must show that the structure of the string has been preserved: each exponent should be the square of the prior exponent. Third, the participant showed the update was non-degenerative (where a malicious p_i erases all prior contributions by using a contribution $\tau_i = 0$). p_i then transmits the tuple and its correctness proof.
3: The coordinator receives the update from p_n, checks the correctness of p_n's contribution, and samples a random contribution τ^1 from the *randomness beacon*. Note that as long as the beacon is secure (unpredictable, live, and unbiased), the protocol remains secure—even if all the participants are malicious. The coordinator finalizes the powers-of-tau phase by updating with τ^1 and generating the corresponding correctness proof.
4: After generating the powers-of-tau tuple, the coordinator first converts the statement to an equivalent arithmetic circuit and then linearly combines elements in the PoT according to the circuit C to initialize the SRS_0^1 for the second phase protocol.
5: The coordinator begins the second phase by transmitting the SRS from phase one SRS_0^1 over to the first registered phase two participant. Similar to phase 1, participants contribute their secrets to the SRS. Each participant checks the correctness of the last contribution, samples a random secret, and updates the SRS with the secret.
6: In the final step, the coordinator checks the SRS_m^2 transmitted from the prior participant, retrieves another output τ^2, from the random beacon, and updates the SRS with τ^2 to finalize the output of the whole protocol.

Remark 2. Note that for clarity, we assume that each participant passes their contribution consecutively to the next participant, and everyone verifies the correctness of the tuple from the last round. In practice, communication is centralized through the coordinator, who also verifies the correctness proofs. In addition, transcripts of the whole protocol are stored and made available to the public for further inspection, and a mandatory correctness check is not compulsory for participants.

Remark 3. Note that both phases are necessary to finalize an SRS, though essentially, participants in both phases work in the same manner. That is because the corresponding circuit-dependent SRS is a linear combination of the final $\tau = \prod_{i=1}^{n} \tau_i$. The additional contribution from the *random beacon* provides an important property: even where participants collude, no set will know the final powers-of-tau tuple $(g^{\tau}, g^{\tau^2}, g^{\tau^3}, \cdots, g^{\tau^k})$. Therefore, to transform powers-of-tau to the Groth16 SRS the coordinator must ensure the powers-of-tau process has been finalized with the random beacon.

Remark 4. A simplified and optimized version of the MMORPG framework is Snarky Ceremonies [77]. Snarky ceremonies are a more general framework that captures both circuit-specific [26], [55], [78] and updatable SRS [27], [28], [53]. By relaxing the security definition, it removes the need for random beacons as slightly biased SRS is not sufficient to break SNARKs. We summarize existing proposals for trusted setup protocol in Table 3.

Protocols	Rounds	Individual-Rounds	Corruption	Round-robin	Decentralized	Incentive	UC-secure	Target
BCDTV15[66]	$O(n)$	$O(n)$	$n-1$	✗	✗	✗	✗	General
BGG17[67]	O(1)	4	$n-1$	✗	✗	✗	✗	Pinocchio
BGM17[70]	$O(n)*$	1	n	✓	✗	✗	✗	Groth16
KMSV21[77]	$O(n)$	1	$n-1$	✓	✗	✗	✗	Groth16
KKK21[79]	$O(n)*$	1	$n-1$	✓	✓	✓	✓	General
CDKS22[42]	$O(\sqrt{n})$	$O(\sqrt{n})$	$n-1$	✗	✓	✗	✓	PoT
NRBB22[80]	$O(n)*$	1	$n-1$	✓	✓	✗	✗	PoT

Table 3. Trusted Setup Protocols. We denote the number of participants in a protocol as n. Protocol with an additional round of applying a random beacon is denoted as $*$. We use rounds for the communication complexity, individual-rounds for communications of individuals in running the protocol, corruption to denote the threshold of corruption, round-robin to indicate if protocols are running in round-robin communication model, decentralized to distinct whether a central coordinator is necessary, incentive to indicate if participants are incentively driven to be hoenst, UC to indicate if protocols are proven to be UC-Secure, target for their design goals.

3.3 Powers-of-tau beyond SNARKs

Powers-of-tau strings are not only for constructing polynomial commitments in SNARKs. Among other further applications, they enable an adaptively secure DKG [81] in an asynchronous network. They also enable a customized weighted-threshold signature scheme [82] and eliminate the prohibitively expensive DKG [83] in threshold signatures. Though DKG protocols are possible without a trusted setup, they either assume a weaker static adversary [84] or incur more overhead [85]. A natural open question is to explore other potential applications that benefit from the powers-of-tau strings.

4 Ceremonies

Trusted setup protocols can not be executed on paper. Instead, public trust in these setup protocols is based on "celebrating" a ceremony to convince the public that the protocol was run correctly and securely. Ceremonies are often conducted with requirements on real-world conduct—(for example lack of conflicts-of-interest or collusion between participants, geographic diversity, rules mandating recording of proceedings). Such elements are designed to bootstrap social trust in the systems, beyond provabile mathematical notions.

We attempted to identify all real-world ceremonies conducted to date to construct a powers-of-tau string, proving a tabulation of more than 40 instances **Table 4**. We note that all of the ceremonies we were able to find are variations of the MMORPG framework [66], [70]. We collected details from various projects with their corresponding trusted setup ceremonies based on an exhaustive online search for the *trusted setup ceremony* keyword on Google (yielding 18 pages in total). The details of many ceremonies, like Hermez, are already missing and scattered despite being conducted only a few years prior to our study. In these cases, we attempted to find their corresponding ceremony pages and blog posts using the Internet Archive [86]. Since 2022, ceremony information can be found on one page thanks to the DefinitelySetup project [87].

4.1 Desired Properties of Ceremonies

Based on our study of real-world trusted setup ceremonies, as well as the public communication around trusted setup ceremonies we have observed in practice, we distill a set of desirable properties. We propose a set of properties by the acronym **"ADOPT"**, that an ideal ceremony should follow:

- **A**vailable says that the protocol runs for a sufficient period of time that all participants have time to broadcast contributions. Due to the round-robin nature of protocols, it is important to leave enough time for many contributors. cannot be suppressed by a denial-of-service attack.
- **D**ecentralization means the absence of reliance on a central coordinator. This coordinator might be a target for attacks, or could censor specific contributors.
- **O**pen indicates that the ceremony welcomes any interested participant to join and contribute.
- **P**ersistent means that a ceremony maintains its published information for future verification or extension even after the ceremony is finished. Note that this may require ensuring long-term access to a non-trivial amount of data (i.e. multiple gigabytes).
- **T**ransparency requires that the ceremony documents all procedures for public scrutiny, including the exact protocol specification to enable independent implementation, as well as all procedures, identities of operators and contributors, data formats for intermediate transcripts, etc.

Note that none of the ceremonies we observed actually satisfied all five properties, in particular the Decentralization property. Ensuring all the above properties without a centralized coordinator requires some form of multi-party computation (MPC). In general, conducting a secure MPC without an honest majority or a trusted coordinator is impossible. Fortunately, the general MPC impossibility results on identifiable abort [88], [89] and output delivery guarantee [90], [91] under a dishonest majority model, do not apply to ceremonies. Ceremonies can operate in a fashion where an execution is attempted, and if it fails, rounds can proceed by excluding parties that contributed to the failure—without losing their security. As all the ceremonies that operate in a round-robin manner can easily identify individual malicious behaviors, aborts are easily identifiable. Given this, a malicious party's only option is to abort the ceremony to prevent others from learning the output with their contribution. However, since all ceremonies implement a timeout mechanism and the output is independent of individual contributions, contributors who deliberately attempt to abort the process can be excluded from future executions of the ceremony.

As noted above, we also observed widespread problems with the Persistence and Transparency goals. Many projects did not provide clear, complete documentation sufficient for an independent third-party implementation for contribution or for verification, requiring participants to download and run the reference source code to participate. Many projects also have not persisted the intermediate values or in some cases even the final results of the ceremony, making it impossible retroactively to assess or extend their powers-of-tau string.

4.2 A case study: The Sprout Ceremony

As a concrete case study, we consider the Sprout ceremony [66] of the Zcash project, an early ceremony conducted in 2016 (perhaps the world's first). Lessons learned from this ceremony have motivated much follow-up research in the area.

First, the identity of participants in the Sprout Ceremony was required to be known to a coordinator in advance. Therefore, conspiracies and rumors of the static flavor of MPC protocol were widely spread across the ZCash community. This restriction undermined the Open and Decentralized goals. This led Bowe, Gabizon and Miers [70] to propose a *player-exchangeable* MPC SRS framework. MMORPG is a two-phase round-robin process where participants are free to join or leave in every phase. In this case, the security of the secret τ is escalated to the next level – you are welcome to commit your contribution to generate the SRS if you do not trust anyone else in the ceremony. Security is preserved even if others deviate from the protocol.

Second, the Sprout ceremony is a two-phase SRS generation framework originally designed for [26] but it also works similarly for Linear PCP (Probalistic Checkable Proof) zk-SNARKs [26], [55], [92]–[96] which significantly simplified the ceremony and quickly became the de-facto standardization in the industry. Note that though it's customized for pairing-based linear PCP SNARK systems, Polynomial IOPs [27], [28], [53], [97]–[101] (Interactive Oracle Proof) built from pairing-based commitment [56] benefit from it as well. Specifically, the first phase of the MMORPG ceremony is known as *powers-of-tau* ceremony, which is completely orthogonal to the NP statement. Trivial to see that the *powers-of-tau* SRS works for arbitrary statements of bounded size. The second phase runs sequentially after the first phase to generate the final SRS that summarizes the NP statement. Note that the general approach to proving an NP statement in zk-SNARKs is to first convert the NP statement to the corresponding arithmetic circuit, which we call arithmetization. The second phase essentially encodes the circuit into the SRS. Therefore, two phases are enough to finalize an SRS private-coin setup. MMORPG drastically improves ceremonies overall (2 rounds vs 4 rounds [66]). Concretely speaking, it took six days to finish the Sprout Ceremony [66] even though there were only six participants, whereas the *Sapling* ceremony only took 2.5 hours. Note that a large portion of the cost of an MMORPG ceremony can be amortized over multiple circuits as powers-of-taus are reusable.

Third, the robustness of the Sprout ceremony was not guaranteed as even one of the fixed parties could deliberately force a setup to terminate thus bias the output SRS. As a result,[70] proposed to use *random beacons* [5], [71], [102]–[119] to rerandomize the protocol output for each phase so that no single participant is able to bias it. However, the then-common practice of instantiating a *random beacon* with a hash function on a recent block in Bitcoin/Ethereum is not an *secure random beacon* as miners/sequencers in roll-ups are capable of manipulating the outcome [120]. To improve this, *distributed random beacons* [5], [75], [76] can bootstrap the randomness of the SRS in a more sound and reliable way. But secure random beacons significantly increase the complexity of security analysis because most constructions expect (public) private-coin setups as well as MMORPG ceremonies. Examples are *random beacons* built from PVSS (Publicly Verifiable Secret Share) (a dealer distributes shared-secret), threshold signature (an additional layer of DKG is needed), and delay-based protocols rely on *group of unknown order setups and customized hardware*.

4.3 Recovering from a compromised SRS

We now present, as a case study, an illustrating real-world example of how a trusted setup can fail, outlining properties that could be preserved even under a *subverted SRS*. We also note that *recovering* from a compromised setup is itself a second type of ceremony, designed to build public trust.

The most prominent example was in the ZCash Sprout ceremony [66]. In this case, the problem was not with the ceremony but the protocol itself. Specifically, Gabizon [155] found a small but critical vulnerability in the private-coin SRS protocol as described in the research paper by Ben-Sasson et al.[78], which was the protocol used in the ZCash Sproud ceremony. Ben-Sasson et al. [78] described an extra, unneeded parameter, which was generated in the ZCash Sprout ceremony, but could be exploited by an adversary to generate proof of any statement based on a single valid proof. This implies that an adversary can generate arbitrary proofs for any statement given a single proof—allowing an adversary to mint ZCash coins at will. Note that the vulnerability (which we present below

Project	PPOT	Update	System	Data	Time	# Contrib	Curve	Type	Available
Aleo universal [121]	N/A	✓	Marlin[28]	44GB	9.56 hours	146	BLS12_377	Powers-of-tau	✗
Aleo inner [121]	N/A	✗	Groth16[26]	771MB	40 mins	1059	BLS12_377	MMORPG	✗
Aleo outer [121]	N/A	✗	Groth16[26]	2.56GB	42 mins	1036	BW6_761	MMORPG	✗
Anon Aadhaar V2 [122]	54	✗	Groth16[26]	583 MB	6.81 mins	105	BN254	MMORPG-2	✓
Axiom [123]	78	✓	KZG[56]	-	-	-	BN256	MMOPRG-1	✓
Aztec [74]	N/A	✗	Sigma[124]	5.71GB	3.3 hours	176	BN254	MMORPG-1	✓
Celo Plumo [125]	N/A	✗	Groth16[26]	Unknown	10 hours	111	BLS12_377	MMORPG	✗
clr.fund [126]	54	✗	Groth16[26]	252MB	10 mins	2562	BN254	MMORPG-2	✗*
DeGate [127]	71	✗	Groth16[26]	370GB	14 hours	5	BN254	MMORPG-2	✗
Email Wallet [128]	54	✗	Groth16[26]	2.83GB	50 mins	29	BN254	MMORPG-2	✗*
FileCoin Mainnet [129]	19	✗	Groth16[26]	52G	6 hours	17	BLS12-381	MMORPG-2	✓
	19	✗		77GB	3 hours	19		MMORPG-1	✓
FileCoin SnapDeals [130]	19	✗	Groth16[26]	25 GB	30 mins	12	BLS12-381	MMORPG-2	✓
Fractal Cash [131]	30	✗	Groth16[26]	17.4MB	2 mins	7	BN256	MMORPG-2	✓
Hermez [132]	54*	✗	Groth16[26]	92GB	9 horus	6	BN254	MMORPG-2	✓
KZG ceremony [133]	N/A	✗	KZG[56]	7.21MB	3 mins	141416	BLS12-381	MMORPG-2	✓
Loopring [134]	11	✗	Groth16[26]	75GB	5.0 hours	16	BN254	MMORPG-2	✗*
MACI [135]	54	✗	Groth16[26]	1.23GB	10 mins	45	BN254	MMORPG-2	✗*
MACI v1.2 [136]	54	✗	Groth16[26]	2093 MB	33 mins	41	BN254	MMORPG-2	✓
MACI V2 [137]	54	✗	Groth16[26]	2010 MB	20.5 mins	246	BN254	MMORPG-2	✓
Manta [138]	72	✗	Groth16[26]	30MB	10 mins	4382	BN254	MMOPRG	✓
Namada [139]	ZCash POT	✗	Groth16[26]	80.8MB	2 mins	2510	BLS12-377	MMORPG-2	✓
Panther [140]	54	✗	Groth16[26]	38.5MB	60s	11	BN254	MMORPG-2	✓
PPOT [141]	N/A	✓	N/A	97GB	24 hours	80	BN254	Powers-of-tau	✗*
RLN [142]	54	✗	Groth16[26]	68.2MB	17 mins	62	BN254	MMORPG-2	✗*
RISCZERO [143]	54	✗	Groth16[26]	3452MB	25 mins	238	BN254	MMORPG-2	✓
Semaphore [72]	25	✗	Groth16[26]	208MB	10 mins	360	BN256	MMORPG-2	✗*
Semaphore V4 [144]	54	✗	Groth16[26]	102 MB	25 mins	389	BN254	MMORPG-2	✓
Tornado cash [145]	30	✗	Groth16[26]	10.64MB	3 s	1114	BN256	MMORPG-2	✓
Unirep v2 [146]	54	✗	Groth16[26]	41.8MB	2 mins	5347	BN254	MMORPG-2	✗*
ZCash Sprout [147]	N/A	✗	BCTV14[78]	7GB	2 days*	6	BN254	BGG	✓
Zcash Sapling [148]	N/A	✗	Groth16[26]	1.1GB	2.5hours	87	BLS12-381	MMORPG-1	✓
	N/A	✗		741MB	40 mins	91		MMORPG-2	✓
ZKOPRU [149]	54	✗	Groth16[26]	400MB	40 mins	369	BN254	MMORPG-2	✓
ZKP2P [150]	23	✗	Groth16[26]	8.2GB	82 mins	25	BN254	MMORPG-2	✗*
ZKP2P v2.4 [151]	54	✗	Groth16[26]	17.1 GB	71 mins	8	BN254	MMORPG-2	✓
ZKP2P Domain [152]	54	✗	Groth16[26]	1662 MB	34.5 mins	17	BN254	MMORPG-2	✓
ZKSync [153]	N/A	✓	PLONK[53]	5.71GB	3.3 hours	176	BN256	MMORPG-1	✓
ZK Ticket [154]	54	✗	Groth16[26]	3.62MB	2.5 mins	17	BN254	MMORPG-2	✗*

f

Table 4. SNARK Ceremonies up-to-date in alphabetic order. PPOT denotes the index of the powers-of-tau contributions from the PPOT ceremony. "Update" denotes if it is an updatable ceremony. The "Data" and "Time" columns measure the costs to each participant in the ceremony. For a ceremony running over on multiple circuits, we denote time and size as the amounts of efforts for one to contribute to all circuits. "Available" indicates whether all the transcripts of a ceremony are still publicly accessible. We denote a ceremony with both phases in MMORPG as "MMORPG", whereas a ceremony only runs the first phase as "MMORPG-1" and the one only runs the second phase as "MMORPG-2".

for illustrative purposes) is not a result of the ceremony itself but is a flaw in the underlying protocol.

Abstractly, for n polynomials $P_j(x_j) = \sum_{i=0}^{n} a_{i,j} x_j^i, j \in n$, the *SRS* of P_j described in [78] is $g^{P_j(\tau)}, g^{\alpha P_j(\tau)}, \alpha \leftarrow_r \mathbb{F}$. For a prover, with public input $w_{in} = w_0, \cdots, w_m$ and witness $w_{m+1}, \cdots, w_n$, the proof $\pi[0]$ is $\prod_{i=m+1}^{n}(g^{P_i(\tau)})^{w_i}$ and $\pi[1] = \prod_{i=m+1}^{n}(g^{\alpha P_i(\tau)})^{w_i}$. A valid proof will pass the check $\pi[0]^\alpha = \pi[1]$. Obviously, SRS like $g^{P_j(\tau), j \in m}$ are redundant and are never used. So a malicious prover with a valid π could generate valid proofs to other public inputs $w'_{in} = w'_0, \cdots, w'_m$, $\pi'[0] = \pi[0]\prod_{i=1}^{m}(g^{P_i(\tau)})^{w_i - w'_i}$, $\pi'[1] = \pi[1]\prod_{i=1}^{m}(g^{\alpha P_i(\tau)})^{w_i - w'_i}$. Essentially, $\pi'[0] = g^{\sum_{i=0}^{n} P_i(\tau) w_i - \sum_{i=m+1}^{n} P_i w'_i}$, $\pi'[1] = g^{\sum_{i=0}^{n} \alpha P_i(\tau) w_i - \sum_{i=m+1}^{n} \alpha P_i w'_i}$. Therefore, $\pi'[0]^\alpha$ will always be $\pi'[1]$.

The above flaw was implemented by the practical Zcash ceremony directly from the paper[78], and the results were widely published before the bug was noticed. This brings up two natural questions. First, what occurs when an adversary is able to set the SRS and how much security can one retain under a maliciously generated SRS? Second, how can a practical project recover from a compromised SRS?

Noting that circuit-specific SRS in SNARKs [26] depends on the circuit's corresponding statement, a modified SRS undermines the protocol by leading to a different statement. Further, if the secure trapdoor τ is not random, the verifier can learn one bit of the witness by checking the proof. Last, a sophisticated adversary could set all strings to 0 except the string on the chosen spot. This leaks one bit of the witness to the verifier, which inevitably breaks the zero-knowledge property. More details can be found in [156].

Bellare, Fuchsbauer and Scafuro [157] studied subversion-resistant zk-SNARKs and systematically answered the extent to which security can be retained in the event of a subverted SRS: one generated by a malicious party who knows the trapdoor. Following the principle of trust-but-verify, subversion-resistant zk-SNARKs additionally allow a prover to verify if a generated SRS is well-formed. Thus, it bypasses trust and the need for a trusted third party. Bellare, Fuchsbauer and Scafuro [157] provides a counterintuitive result: even where an SRS/URS is subverted, it is possible to nonetheless retain *soundness of the overall protocol* even while preserving the zero-knowledge property. However, the paper also provides a negative result: one can not retain both *subversion-soundness* (a property requiring that it is hard for an adversary to generate a malicious CRS) and classical zero-knowledge properties at the same time. This is a result of the existence of a simulator that can output valid proofs of false statements under a valid proof.

The general approach to ensuring that SNARKs are subversion-resistant is to make the SRS verifiable. Fuchsbauer [158] shows that given an appropriate SRS verification procedure: (a) no additional strings are needed to ensure verifiable SRS in Groth16 [26] whereas (b) four extra group elements are necessary for provers to verify the SRS in Pinocchio [55].

In the event of the Sprout ceremony, the bug was discovered by Zcash engineers in March 2018 [159]. After confirmation, the published ceremony transcripts containing the additional were quietly removed. As there was no way to fix the vulnerability if anybody had downloaded and cached the transcripts, the bug was kept under embargo until after the already-planned second setup ceremony, Sapling, was conducted in November 2018. At this point, after the network had hard-forked away from using the parameters from the vulnerable Sprout ceremony, the bug was publicly disclosed in February 2019. There is (fortunately) no evidence that the bug was ever exploited.

5 Concluding Discussion

We conclude with outlining key lessons from our systematization. First are research goals emerging from our systematization of protocols, followed by lessons from our systematization of ceremonies:

5.1 Research directions for trusted setup protocols

Reducing the size of SRS from linear to sublinear. For a comparison of the size of the SRS for different schemes, we refer the reader to [59], [60]. Recent work [46] shows that SNARKs can be built with a linear-time prover, constant-sized proofs, and square root-sized SRS. It is also possible [160] to achieve a sublinear prover with a linear-sized SRS, and a constant-sized proof. It is still an open question to construct a SNARK-proof system with a sublinear prover, sublinear SRS, and constant-sized proof. This would dramatically alleviate participants'/coordinators' loads to participate in expensive SRS setup ceremonies.

Accelerating SNARK ceremonies via reusing SRS: note that the powers-of-tau ceremony can be upgraded to a perpetual version. We defer the discussion of the perpetual powers-of-tau (PPOT) ceremony to Appendix A. Powers-of-tau strings relieve new zk-SNARK players from generating system parameters from scratch. To properly launch a publicly trustworthy zk-SNARK system, they only have to fork any PPOT contribution out to continue their specific phase 2 ceremony. Moreover, noticeably, the phase 2 ceremony is significantly lightweight compared to the powers-of-tau ceremony. As reported in [70], phase 2 runs 4x faster than phase 1 while incurring 3x less transmission overhead.

Asynchronous ceremonies: Existing ceremonies are designed in a synchronized model, which requires contributors to register to reserve a timeslot for online contribution or a contributor has to queue in a line until its turn to formally join a ceremony. To activate a fully *come-contribute-go* ceremony paradigm, one requires an asynchronous ceremony. Initial work is done by Das et al. [161], outlining an asynchronous powers-of-tau ceremony that significantly improves the sequential MMORPG protocol. However, it is still far away from an optimal asynchronous PPOT setup since it introduces $O(\lambda n^3)$ communication overhead overall. This protocol[161] only targets the power-of-tau setup, though we can trivially extend it to phase 2 since both phases essentially work in the same manner. It's still not a fully asynchronous protocol as participants in phase

2 have to wait for the central coordinator to finalize the polynomial SRS $g^{P(\tau)}$. It remains an open challenge to design a fully asynchronous protocol that is capable of handling the whole ceremony without a centralized coordinator.

Fully decentralized and asynchronous ceremonies: A further enhancement is to get rid of the centralized coordinator completely. Nikolaenko et. al. [80] designed and evaluated an on-chain powers-of-tau ceremony to eliminate the assumption of the centralized coordinator. For the phase 2 circuit-dependent ceremony, Kerber et al. [79] observed that updatable SRS [27]–[29] generation can be integrated as part of a consensus protocol so that no additional security assumptions or off-chain computation are needed for the security of the SRS. Both [80] and [79] open the door to providing incentives for honest participants. However, it is still a synchronous ceremony requiring sequential updates. Still, it opens the possibility of a fully *decentralized* and *asynchronous* phase one ceremony for zk-SNARKs integrated with [161]. Furthermore, we envision a fully asynchronous and decentralized ceremony that is capable of handling both phases of zk-SNARKs' protocols.

Curve transformation: So far, the PPOT ceremonies in practice have been performed on the fixed BN254 curve[162]. However, deployed applications may be built on other curves. For example, ZCash uses BLS12-381[163]. Practitioners face trade-offs when considering implementing their applications in a specific curve. However, outputs from the PPOT ceremony are incompatible with applications that rely on a different curve. An interesting research question is to design an efficient transformation scheme to enable using the output of power-of-tau ceremonies on the BN254 curve for other pairing-friendly popular curves [164].

The transparency requirement requires not only a commitment to open-sourcing code, but also requires explainable system parameters. This is an often overlooked point in SNARK ceremonies when deciding to run it on the BN254 curve. There are concerns that the base point [165] (generator) may be too small (the generator's x coordinate is chosen to be one) to be secure. Though the generator is fast for pairing, it is not a truly random point.

5.2 Lessons for trustworthy setup ceremonies.

We summarize the following six lessons for running a sound and trustworthy ceremony, many of which were not explicitly followed in multiple ceremonies we studied. Note that those are universal principles distilled from the MMOPRG ceremony that can be beneficial to all ceremonies.

Multiple open-sourced implementations and third-part auditing: to reduce the risk of coordinated failure due to software bugs, contributors should be encouraged to implement their own version of the ceremony and several official implementations should be sanctioned, with third party audits. The MPC protocol [166] of the Sprout Ceremony is open-sourced and is audited by a third party – NCC [167]. The ceremony code of generating *SRS* for ZCash depends on a specific circuit, and ZCash functions are open-sourced and audited. A model example is the recent Ethereum KZG ceremony [168], community members not only implement the ceremony in command-line with different programming languages to hedge potential security issues in libraries [169], but also provide web-based interfaces implemented based on arkworks [170] and gnark [171], which are more user-friendlier. This effort results in 10 different implementations.

Diverse participation: To boost the confidence that all participants in the ceremony are not colluding, the ceremony should include people from different backgrounds and interests. This may require active recruitment by the organizers. To counter a critique of anonymous participation, the Sprout Ceremony chose participants from academia, the ZCash founder team, an independent Bitcoin core developer, Peter Todd, who is skeptical and critical of the ceremony, and the whistleblower Edward Snowden. People of unrelated interests and even criticism prevent the ceremony from the interior collusion and counterfeits of the ZCash team.

Air-gapped and one-time machines: To minimize the risk of vulnerabilities, air-gapped machines should be the only places processing participants' contributions to the ceremony. Some ceremonies have gone beyond that, for example requiring that every machine was from a random store and it was exclusive to the ceremony. In this manner most cyber-attacks can be prevented since the machines will never be connected to any network during their lifetime. To jointly finish an MPC in the Sprout ceremony, an additional networked laptop was used to burn the communication transcripts onto a disc so that air-gapped machines can talk to each other. Every laptop was physically destroyed after the Sprout Ceremony. In that case, the ceremony is solely vulnerable to participants' willful collusion.

Geographically isolation: participants should be selected from diverse geogrpahic locations and isolated worldwide to lower the chance that all of them could be targeted physically.

Diversed source of randomness: participants should be encouraged to secure their contributions as much as they can by sampling randomness from diverse physical sources. Creative ideas include[172]: sampling randomness from space aboard the Crypto 2 satellite, from the random fluctuations of a lava lamp, from scatted biscuits over the floor after feeding a pet, or from the noisy city Sydney, and so forth.

Publicly verifiable operation log: all communications from the ceremony should be stored permanently and made public for later attestation. The logs cannot convince the public that the participants did not collude, but can enable them to verify that the transcript is consistent with the protocol. Additionally, old transcripts can be used to extend a ceremony in the future by adding additional contributions. For example, restored transcripts from the Sprout Ceremony guaranteed that the updated CRS of ZCash after the Sapling Ceremony (a newer and larger ceremony) is genuine from the original six participants (not substituted by any others who might keep his τ).

6 Acknowledgements

Faxing Wang is supported by Melbourne research scholarship. Joseph Bonneau was additionally supported by DARPA Agreement HR00112020022 and NSF Grant CNS-2239975. The views and conclusions contained in this material are those of the authors and do not necessarily reflect the official policies or endorsements of the United States Government, DARPA, a16z Crypto, or any other supporting organization.

References

[1] T. P. Pedersen Non-interactive and information-theoretic secure verifiable secret sharing, in *CRYPTO*, 1991 (cit. on p. 1).

[2] E. Barker and J. Kelsey, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, *NIST Special Publication*, 2006 (cit. on p. 1).

[3] Z. Wilcox *The Design of the Ceremony*, `electriccoin.co/blog/the-design-of-the-ceremony/`, 2016 (cit. on p. 1).

[4] A. Faz-Hernandez, S. Scott, N. Sullivan, R. S. Wahby and C. A. Wood *Hashing to Elliptic Curves*, RFC 9380, 2023 (cit. on p. 1).

[5] K. Choi, A. Manoj and J. Bonneau SoK: Distributed Randomness Beacons, in *IEEE Security & Privacy*, 2023 (cit. on pp. 1, 10, 15).

[6] E. Ben-Sasson *The Cambrian Explosion of Crypto Proofs*, `medium.com/starkware/the-cambrian-explosion-of-crypto-proofs-7ac080ac9aed`, 2020 (cit. on p. 1).

[7] K. Nazirkhanova, J. Neu and D. Tse Information dispersal with provable retrievability for rollups, in *ACM AFT*, 2022 (cit. on p. 2).

[8] T. Xie, J. Zhang, Z. Cheng, et al Zkbridge: Trustless cross-chain bridges made practical, in *ACM CCS*, 2022 (cit. on p. 2).

[9] I. Miers, C. Garman, M. Green and A. D. Rubin Zerocoin: Anonymous distributed e-cash from bitcoin, in *IEEE Security & Privacy*, 2013 (cit. on p. 2).

[10] J. Bartusek and D. Khurana Cryptography with certified deletion, in *CRYPTO*, 2023 (cit. on p. 2).

[11] A. Broadbent and R. Islam Quantum encryption with certified deletion, in *TCC*, 2020 (cit. on p. 2).

[12] S. Garg, S. Goldwasser and P. N. Vasudevan Formalizing data deletion in the context of the right to be forgotten, in *Eurocrypt*, 2020 (cit. on p. 2).

[13] T. Hiroka, T. Morimae, R. Nishimaki and T. Yamakawa Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication, in *Asiacrypt*, 2021 (cit. on p. 2).

[14] A. Poremba *Quantum proofs of deletion for learning with errors*, 2023. arXiv: `2203.01610 [quant-ph]` (cit. on p. 2).

[15] J. Bartusek, D. Khurana and A. Poremba *Publicly-verifiable deletion via target-collapsing functions*, 2023. arXiv: `2303.08676 [quant-ph]` (cit. on p. 2).

[16] V. Costan and S. Devadas *Intel SGX explained*, Cryptology ePrint Archive, Paper 2016/086, 2016 (cit. on p. 2).

[17] V. Costan, I. Lebedev and S. Devadas Sanctum: Minimal hardware extensions for strong software isolation, in *USENIX Security*, 2016 (cit. on p. 2).

[18] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović and D. Song Keystone: An open framework for architecting trusted execution environments, in *EuroSys*, 2020 (cit. on p. 2).
[19] S. Pinto and N. Santos, Demystifying arm trustzone: A comprehensive survey, *ACM Computing Surveys*, 2019 (cit. on p. 2).
[20] J. Van Bulck, M. Minkin, O. Weisse, et al Foreshadow: Extracting the keys to the intel SGX kingdom with transient Out-of-Order execution, in *USENIX Security*, 2018 (cit. on p. 2).
[21] S. van Schaik, A. Seto, T. Yurek, et al SoK: SGX.Fail: How Stuff Gets eXposed, in *IEEE Security & Privacy*, 2024 (cit. on p. 2).
[22] Z. Zhang, M. Tao, S. O'Connell, C. Chuengsatiansup, D. Genkin and Y. Yarom BunnyHop: Exploiting the instruction prefetcher, in *USENIX Security*, 2023 (cit. on p. 2).
[23] S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof-systems, *SIAM Journal on Computing*, 1989 (cit. on p. 3).
[24] B. Bünz, B. Fisch and A. Szepieniec Transparent SNARKs from DARK compilers, in *Eurocrypt*, 2020 (cit. on pp. 3, 9).
[25] J. Lee Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments, in *TCC*, 2021 (cit. on pp. 3, 9).
[26] J. Groth On the size of pairing-based non-interactive arguments, in *Eurocrypt*, 2016 (cit. on pp. 3, 9, 10, 12, 14, 16, 17, 29).
[27] M. Maller, S. Bowe, M. Kohlweiss and S. Meiklejohn Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings, in *ACM CCS*, 2019 (cit. on pp. 3, 9, 12, 14, 19).
[28] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely and N. Ward Marlin: Preprocessing zkSNARKs with universal and updatable SRS, in *Eurocrypt*, 2020 (cit. on pp. 3, 9, 12, 14, 16, 19).
[29] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn and I. Miers Updatable and Universal Common Reference Strings with Applications to zk-SNARKs, in *CRYPTO*, 2018 (cit. on pp. 3, 19).
[30] M. Chen, C. Hazay, Y. Ishai, et al Diogenes: Lightweight scalable RSA modulus generation with a dishonest majority, in *IEEE Security & Privacy*, 2021 (cit. on p. 3).
[31] A. Shamir, IP= PSPACE, *Journal of the ACM*, 1992 (cit. on p. 4).
[32] M. Blum, P. Feldman and S. Micali Non-interactive zero-knowledge and its applications, in *STOC*, 1988 (cit. on p. 5).
[33] U. Feige, D. Lapidot and A. Shamir Multiple non-interactive zero knowledge proofs based on a single random string, in *FOCS*, 1990 (cit. on p. 5).
[34] M. Blum, A. De Santis, S. Micali and G. Persiano, Noninteractive zero-knowledge, *SIAM Journal on Computing*, 1991 (cit. on p. 5).
[35] I. Damgård Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing, in *Eurocrypt*, 1992 (cit. on p. 5).
[36] A. De Santis and G. Persiano Zero-knowledge proofs of knowledge without interaction, in *FOCS*, 1992 (cit. on p. 5).
[37] J. Kilian and E. Petrank, An efficient noninteractive zero-knowledge proof system for np with general assumptions, *Journal of Cryptology*, 1998 (cit. on p. 5).
[38] A. D. Santis, G. D. Crescenzo and G. Persiano Necessary and Sufficient Assumptions for Non-Iterative Zero-Knowledge Proofs of Knowledge for All NP Relations, in *ICALP*, 2000 (cit. on p. 5).

[39] J. Groth Short non-interactive zero-knowledge proofs, in *Asiacrypt*, 2010 (cit. on p. 5).

[40] C. Gentry, J. Groth, Y. Ishai, C. Peikert, A. Sahai and A. D. Smith, Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs, *Journal of Cryptology*, 2015 (cit. on p. 5).

[41] J. Groth and R. M. Ostrovsky, Cryptography in the multi-string model, *Journal of Cryptology*, 2007 (cit. on p. 5).

[42] R. Cohen, J. Doerner, Y. Kondi and A. Shelat Guaranteed output in $O(\sqrt{n})$ rounds for round-robin sampling protocols, in *Eurocrypt*, 2022 (cit. on pp. 6, 12).

[43] M. Dellepere, P. Mishra and A. Shirzad, Garuda and pari: Smaller and faster snarks via equifficient polynomial commitments, *Cryptology ePrint Archive*, 2024 (cit. on p. 9).

[44] J. Groth and M. Maller Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks, in *CRYPTO*, 2017 (cit. on p. 9).

[45] H. Lipmaa Polymath: Groth16 is not the limit, in *CRYPTO*, 2024 (cit. on p. 9).

[46] M. Campanelli, N. Gailly, R. Gennaro, P. Jovanovic, M. Mihali and J. Thaler Testudo: Linear Time Prover SNARKs with Constant Size Proofs and Square Root Size Universal Setup, in *Latincrypt*, 2023 (cit. on pp. 9, 18).

[47] R. Gennaro, C. Gentry, B. Parno and M. Raykova Quadratic span programs and succinct NIZKs without PCPs, in *Eurocrypt*, 2013 (cit. on p. 9).

[48] J. Bootle, A. Chiesa, Y. Hu and M. Orru Gemini: Elastic SNARKs for diverse environments, in *Eurocrypt*, 2022 (cit. on p. 9).

[49] H. Lipmaa, J. Siim and M. Zając Counting vampires: From univariate sumcheck to updatable zk-snark, in *Asiacrypt*, 2022 (cit. on p. 9).

[50] C. Ràfols and A. Zapico An algebraic framework for universal and updatable SNARKs, in *CRYPTO*, 2021 (cit. on p. 9).

[51] M. Campanelli, A. Faonio, D. Fiore, A. Querol and H. Rodríguez Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions, in *Asiacrypt*, 2021 (cit. on p. 9).

[52] J. Bootle, A. Cerulli, P. Chaidos, J. Groth and C. Petit Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting, in *Eurocrypt*, 2016 (cit. on p. 9).

[53] A. Gabizon, Z. J. Williamson and O. Ciobotaru *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*, Cryptology ePrint Archive, Paper 2019/953, 2019 (cit. on pp. 9, 12, 14, 16, 29).

[54] E. Ben-Sasson, A. Chiesa, E. Tromer and M. Virza Scalable zero knowledge via cycles of elliptic curves, in *CRYPTO*, 2014 (cit. on p. 9).

[55] B. Parno, J. Howell, C. Gentry and M. Raykova Pinocchio: Nearly practical verifiable computation, in *IEEE Security & Privacy*, 2013 (cit. on pp. 9, 12, 14, 17).

[56] A. Kate, G. M. Zaverucha and I. Goldberg Constant-size commitments to polynomials and their applications, in *Asiacrypt*, 2010 (cit. on pp. 8, 9, 14, 16, 29).

[57] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille and G. Maxwell Bulletproofs: Short proofs for confidential transactions and more, in *IEEE Security & Privacy*, 2018 (cit. on p. 9).

[58] E. Ben-Sasson, I. Bentov, Y. Horesh and M. Riabzev Fast reed-solomon interactive oracle proofs of proximity, in *ICALP*, 2018 (cit. on p. 9).

[59] A. Arun, C. Ganesh, S. Lokam, T. Mopuri and S. Sridhar Dew: A transparent constant-sized polynomial commitment scheme, in *PKC*, 2023 (cit. on pp. 9, 18).

[60] I. A. Seres and P. Burcsi *Behemoth: Transparent polynomial commitment scheme with constant opening proof size and verifier time*, Cryptology ePrint Archive, Paper 2023/670, 2023 (cit. on pp. 9, 18).

[61] C. Papamanthou, E. Shi and R. Tamassia Signatures of correct computation, in *TCC*, 2013 (cit. on p. 9).

[62] T. Xie, Y. Zhang and D. Song Orion: Zero knowledge proof with linear prover time, in *CRYPTO*, 2022 (cit. on p. 9).

[63] T. Kohrita and P. Towa, Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments, *Journal of Cryptology*, 2024 (cit. on p. 9).

[64] B. Chen, B. Bünz, D. Boneh and Z. Zhang Hyperplonk: Plonk with linear-time prover and high-degree custom gates, in *EurocryptO*, 2023 (cit. on p. 9).

[65] H. Zeilberger, B. Chen and B. Fisch *Basefold: Efficient field-agnostic polynomial commitment schemes from foldable codes*, Cryptology ePrint Archive, Paper 2023/1705, 2023 (cit. on p. 9).

[66] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer and M. Virza Secure sampling of public parameters for succinct zero knowledge proofs, in *IEEE Security & Privacy*, 2015 (cit. on pp. 10, 12–15).

[67] S. Bowe, A. Gabizon and M. D. Green A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK, in *FC*, 2019 (cit. on pp. 10, 12).

[68] B. Abdolmaleki, K. Baghery, H. Lipmaa, J. Siim and M. Zajac Uc-secure CRS generation for SNARKs, in *Africacrypt*, 2019 (cit. on p. 10).

[69] A. Aggelakis, P. Fauzi, G. Korfiatis, et al A non-interactive shuffle argument with low trust assumptions, in *CT-RSA*, 2020 (cit. on p. 10).

[70] S. Bowe, A. Gabizon and I. Miers *Scalable multi-party computation for zk-SNARK parameters in the random beacon model*, Cryptology ePrint Archive, Paper 2017/1050, 2017 (cit. on pp. 10, 12–15, 18, 29).

[71] M. O. Rabin, Transaction protection by beacons, *Journal of Computer and System Sciences*, 1983 (cit. on pp. 10, 15).

[72] *Semaphore trusted setup*. [Online]. Available: `https://github.com/privacy-scaling-explorations/semaphore-phase2-setup` (cit. on pp. 10, 16).

[73] *Tornado.cash trusted setup ceremony app*. [Online]. Available: `https://github.com/tornadocash/trusted-setup-server` (cit. on p. 10).

[74] *Aztec trusted ceremony*. [Online]. Available: `https://medium.com/aztec-protocol/aztec-crs-the-biggest-mpc-setup-in-history-has-successfully-finished-74c6909cd0c4` (cit. on pp. 10, 16).

[75] M. Raikwar and D. Gligoroski SoK: Decentralized randomness beacon protocols, in *Australasian Conference on Information Security and Privacy*, 2022 (cit. on pp. 10, 15).

[76] A. Kavousi, Z. Wang and P. Jovanovic SoK: Public Randomness, in *Euro S&P*, 2024 (cit. on pp. 10, 15).

[77] M. Kohlweiss, M. Maller, J. Siim and M. Volkhov Snarky ceremonies, in *Asiacrypt*, 2021 (cit. on p. 12).

[78] E. Ben-Sasson, A. Chiesa, E. Tromer and M. Virza Succinct non-interactive zero knowledge for a von neumann architecture, in *USENIX Security*, 2014 (cit. on pp. 12, 15–17).

[79] T. Kerber, A. Kiayias and M. Kohlweiss Mining for Privacy: How to Bootstrap a Snarky Blockchain, in *Financial Crypto*, 2021 (cit. on pp. 12, 19).

[80] V. Nikolaenko, S. Ragsdale, J. Bonneau and D. Boneh Powers-of-tau to the people: Decentralizing setup ceremonies, in *ACNS*, 2024 (cit. on pp. 12, 19).

[81] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn and G. Stern Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation, in *CRYPTO*, 2023 (cit. on p. 12).

[82] S. Das, P. Camacho, Z. Xiang, J. Nieto, B. Bunz and L. Ren Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold, in *ACM CCS*, 2023 (cit. on p. 12).

[83] S. Garg, A. Jain, P. Mukherjee, R. Sinha, M. Wang and Y. Zhang Hints: Threshold signatures with silent setup, in *IEEE Security & Privacy*, 2024 (cit. on p. 12).

[84] T. Yurek, L. Luo, J. Fairoze, A. Kate and A. Miller Hbacss: How to robustly share many secrets, in *NDSS*, 2022 (cit. on p. 12).

[85] S. Das, T. Yurek, Z. Xiang, A. Miller, L. Kokoris-Kogias and L. Ren Practical asynchronous distributed key generation, in *IEEE Security & Privacy*, 2022 (cit. on p. 12).

[86] *The internet archive*, 2024. [Online]. Available: https://archive.org/ (cit. on p. 13).

[87] *Definitely setup project*, 2024. [Online]. Available: https://github.com/privacy-scaling-explorations/DefinitelySetup (cit. on p. 13).

[88] Y. Ishai, R. Ostrovsky and H. Seyalioglu Identifying cheaters without an honest majority, in *TCC*, 2012 (cit. on p. 14).

[89] Y. Ishai, R. Ostrovsky and V. Zikas Secure multi-party computation with identifiable abort, in *CRYPTO*, 2014 (cit. on p. 14).

[90] R. Cleve Limits on the security of coin flips when half the processors are faulty, in *STOC*, 1986 (cit. on p. 14).

[91] Y. Ishai, A. Patra, S. Patranabis, D. Ravi and A. Srinivasan Fully-secure MPC with minimal trust, in *TCC*, 2022 (cit. on p. 14).

[92] Y. Ishai, E. Kushilevitz and R. Ostrovsky Efficient arguments without short pcps, in *CCC*, 2007 (cit. on p. 14).

[93] N. Bitansky, A. Chiesa, Y. Ishai, O. Paneth and R. Ostrovsky Succinct non-interactive arguments via linear interactive proofs, in *TCC*, 2013 (cit. on p. 14).

[94] S. T. Setty, R. McPherson, A. J. Blumberg and M. Walfish Making argument systems for outsourced computation practical (sometimes), in *NDSS*, 2012 (cit. on p. 14).

[95] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg and M. Walfish Taking Proof-Based verified computation a few steps closer to practicality, in *USENIX Security*, 2012 (cit. on p. 14).

[96] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno and M. Walfish Resolving the conflict between generality and plausibility in verified computation, in *EuroSys*, 2013 (cit. on p. 14).

[97] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou and D. Song Libra: Succinct zero-knowledge proofs with optimal prover computation, in *CRYPTO*, 2019 (cit. on p. 14).

[98] C. Ràfols and A. Zapico An algebraic framework for universal and updatable snarks, in *CRYPTO*, 2021 (cit. on p. 14).

[99] S. Setty, J. Thaler and R. Wahby *Customizable constraint systems for succinct arguments*, Cryptology ePrint Archive, Paper 2023/552, 2023 (cit. on p. 14).

[100] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos and C. Papamanthou Vsql: Verifying arbitrary sql queries over dynamic outsourced databases, in *IEEE Security & Privacy*, 2017 (cit. on p. 14).

[101] A. L. Xiong, B. Chen, Z. Zhang, et al VeriZexe: Decentralized Private Computation with Universal Setup, in *USENIX Security* (cit. on p. 14).

[102] M. J. Fischer, M. Iorga and R. Peralta A public randomness service, in *SECRYPT*, 2011 (cit. on p. 15).

[103] J. Clark and U. Hengartner On the use of financial data as a random beacon, in *EVT/WOTE*, 2010 (cit. on p. 15).

[104] J. Bonneau, J. Clark and S. Goldfeder *On bitcoin as a public randomness source*, Cryptology ePrint Archive, Paper 2015/1015, 2015 (cit. on p. 15).

[105] A. K. Lenstra and B. Wesolowski *A random zoo: Sloth, unicorn, and trx*, Cryptology ePrint Archive, Paper 2015/366, 2015 (cit. on p. 15).

[106] K. Choi, A. Arun, N. Tyagi and J. Bonneau Bicorn: An optimistically efficient distributed randomness beacon, in *FC*, 2023 (cit. on p. 15).

[107] D. Yakira, A. Asayag, I. Grayevsky and I. Keidar *Economically viable randomness*, 2020. arXiv: `2007.03531 [cs.CR]` (cit. on p. 15).

[108] I. Cascudo and B. David SCRAPE: Scalable randomness attested by public entities, in *ACNS*, 2017 (cit. on p. 15).

[109] I. Cascudo and B. David ALBATROSS: Publicly attestable batched randomness based on secret sharing, in *Asiacrypt*, 2020 (cit. on p. 15).

[110] E. Syta, P. Jovanovic, E. K. Kogias, et al Scalable bias-resistant distributed randomness, in *IEEE Security & Privacy*, 2017 (cit. on p. 15).

[111] A. Cherniaeva, I. Shirobokov and O. Shlomovits *Homomorphic encryption random beacon*, Cryptology ePrint Archive, Paper 2019/1320, 2019 (cit. on p. 15).

[112] A. Bhat, N. Shrestha, Z. Luo, A. Kate and K. Nayak Randpiper – reconfiguration-friendly random beacons with quadratic communication, in *ACM CCS*, 2021 (cit. on p. 15).

[113] P. Schindler, A. Judmayer, N. Stifter and E. Weippl Hydrand: Efficient continuous distributed randomness, in *IEEE Security & Privacy*, 2020 (cit. on p. 15).

[114] A. Kiayias, A. Russell, B. David and R. Oliynykov Ouroboros: A provably secure proof-of-stake blockchain protocol, in *CRYPTO*, 2017 (cit. on p. 15).

[115] S. Das, V. Krishnan, I. M. Isaac and L. Ren Spurt: Scalable distributed randomness beacon with transparent setup, in *IEEE Security & Privacy*, 2022 (cit. on p. 15).

[116] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup and D. Williams Internet computer consensus, in *PODC*, 2022 (cit. on p. 15).

[117] D. Galindo, J. Liu, M. Ordean and J.-M. Wong Fully distributed verifiable random functions and their application to decentralised random beacons, in *IEEE EuroS&P*, 2021 (cit. on p. 15).

[118] D. Beaver, K. Chalkias, M. Kelkar, et al Strobe: Stake-based threshold random beacons, in *AFT*, 2023 (cit. on p. 15).

[119] Y. Gilad, R. Hemo, S. Micali, G. Vlachos and N. Zeldovich Algorand: Scaling byzantine agreements for cryptocurrencies, in *SOSP*, 2017 (cit. on p. 15).

[120] J. Bonneau, J. Clark and S. Goldfeder *On Bitcoin as a public randomness source*, Cryptology ePrint Archive, Paper 2015/1015, 2015 (cit. on p. 15).

[121] *Aleo trusted setup*. [Online]. Available: https://setup.aleo.org/ (cit. on p. 16).

[122] *Anon adahaar trusted ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/Anon%20Aadhaar%20V2%20Trusted%20Setup%20Ceremony (cit. on p. 16).

[123] *Axiom trusted ceremony.* [Online]. Available: https://docs.axiom.xyz/docs/transparency-and-security/kzg-trusted-setup (cit. on p. 16).

[124] I. Damgård "On Σ-protocols", Tech. Rep., 2002 (cit. on p. 16).

[125] *Celo plumo trusted ceremony.* [Online]. Available: https://blog.celo.org/the-plumo-ceremony-ac7649e9c8d8 (cit. on p. 16).

[126] *Clr-fund trusted ceremony.* [Online]. Available: https://blog.clr.fund/trusted-setup-completed (cit. on p. 16).

[127] *Degate trusted setup.* [Online]. Available: https://medium.com/degate/degate-completes-zk-trusted-setup-ceremony-4752301e379f (cit. on p. 16).

[128] *Email wallet trusted ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/Email%20Wallet%20Trusted%20Setup%20Ceremony (cit. on p. 16).

[129] *Filecoin mainnet trusted setup.* [Online]. Available: https://filecoin.io/blog/posts/trusted-setup-complete/ (cit. on p. 16).

[130] *Filecoin snapdeal trusted setup.* [Online]. Available: https://filecoin.io/blog/posts/trusted-setup-complete-for-network-v15-upgrade/ (cit. on p. 16).

[131] *Fractal cash trusted setup ceremony.* [Online]. Available: https://medium.com/@fractalcash/fractal-cash-announces-trusted-setup-ceremony-652445cfe176 (cit. on p. 16).

[132] *Hermez trusted setup ceremony.* [Online]. Available: https://github.com/hermeznetwork/phase2ceremony_4 (cit. on p. 16).

[133] *Ethereum kzg ceremony.* [Online]. Available: https://github.com/ethereum/kzg-ceremony (cit. on p. 16).

[134] *Loopring trusted setup ceremony.* [Online]. Available: https://loopring.org/#/ceremony (cit. on p. 16).

[135] *Maci v1 trusted setup ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/Maci%20v1%20Trusted%20Setup%20Ceremony (cit. on p. 16).

[136] *Maci v1 trusted setup ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/MACI%20V1.2.0%20Trusted%20Setup%20Ceremony (cit. on p. 16).

[137] *Maci v2 trusted setup ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/MACI%20v2%20Trusted%20Setup%20Ceremony (cit. on p. 16).

[138] *Manta trusted setup.* [Online]. Available: https://cointelegraph.com/news/manta-network-conducts-record-breaking-trusted-setup-ceremony-4-000-contribute (cit. on p. 16).

[139] *Namanda trusted setup.* [Online]. Available: https://namada.net/trusted-setup (cit. on p. 16).

[140] *Panther trusted setup ceremony.* [Online]. Available: https://github.com/pantherprotocol/preZKPceremony (cit. on p. 16).

[141] *Ppot trusted setup.* [Online]. Available: https://perpetualpowersoftau.com/ (cit. on p. 16).

[142] *Rln trusted setup ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/RLN%20Trusted%20Setup%20Ceremony (cit. on p. 16).

[143] *Risczero trusted setup ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/RISC%20Zero%20STARK-to-SNARK%20Prover (cit. on p. 16).

[144] *Semaphore v4 trusted setup ceremony.* [Online]. Available: https://ceremony.pse.dev/projects/Semaphore%20V4%20Ceremony (cit. on p. 16).

[145] *Tornadocash trusted setup ceremony.* [Online]. Available: https://tornado-cash.medium.com/the-biggest-trusted-setup-ceremony-in-the-world-3c6ab9c8fffa (cit. on p. 16).
[146] *Unirep trusted setup ceremony.* [Online]. Available: https://ceremony.unirep.io/ (cit. on p. 16).
[147] *Zcash sprout setup ceremony.* [Online]. Available: https://electriccoin.co/blog/the-design-of-the-ceremony/ (cit. on p. 16).
[148] *Zcash sapling mpc.* [Online]. Available: https://electriccoin.co/blog/completion-of-the-sapling-mpc/ (cit. on p. 16).
[149] *Zkopru trusted setup ceremony.* [Online]. Available: https://medium.com/privacy-scaling-explorations/zkopru-trusted-setup-ceremony-f2824bfebb0f (cit. on p. 16).
[150] *Zkp2p trusted setup.* [Online]. Available: https://ceremony.pse.dev/projects/ZKP2P%20Trusted%20Setup%20Ceremony (cit. on p. 16).
[151] *Zkp2p v2 trusted setup.* [Online]. Available: https://ceremony.pse.dev/projects/ZKP2P%20Trusted%20Setup%20Ceremony%20V2 (cit. on p. 16).
[152] *Zkp2p domain trusted setup.* [Online]. Available: https://ceremony.pse.dev/projects/ZKP2P%20Domain%20Marketplace (cit. on p. 16).
[153] *Zksync trusted setup.* [Online]. Available: https://docs.lite.zksync.io/userdocs/security/#universal-crs-setup (cit. on p. 16).
[154] *Zkticket trusted setup.* [Online]. Available: https://ceremony.pse.dev/projects/ZK%20Ticket%20Trusted%20Setup%20Ceremony (cit. on p. 16).
[155] A. Gabizon *On the security of the BCTV Pinocchio zk-SNARK variant*, Cryptology ePrint Archive, Paper 2019/119, 2019 (cit. on p. 15).
[156] M. Campanelli, R. Gennaro, S. Goldfeder and L. Nizzardo Zero-knowledge contingent payments revisited: Attacks and payments for services, in *ACM CCS*, 2017 (cit. on p. 17).
[157] M. Bellare, G. Fuchsbauer and A. Scafuro NIZKs with an untrusted CRS: security in the face of parameter subversion, in *Asiacrypt*, 2016 (cit. on p. 17).
[158] G. Fuchsbauer Subversion-zero-knowledge SNARKs, in *PKC*, 2018 (cit. on p. 17).
[159] J. Swihart *Zcash Counterfeiting Vulnerability Successfully Remediated*, ECC Blog, Feb. 2019 (cit. on p. 17).
[160] A. R. Choudhuri, S. Garg, A. Goel, S. Sekar and R. Sinha *Sublonk: Sublinear prover plonk*, Cryptology ePrint Archive, Paper 2023/902, 2023 (cit. on p. 18).
[161] S. Das, Z. Xiang and L. Ren Powers of tau in asynchrony, in *NDSS*, 2024 (cit. on pp. 18, 19).
[162] P. S. L. M. Barreto and M. Naehrig Pairing-friendly elliptic curves of prime order, in *SAC*, 2006 (cit. on pp. 19, 29).
[163] P. S. Barreto, B. Lynn and M. Scott Constructing elliptic curves with prescribed embedding degrees, in *SCN*, 2003 (cit. on p. 19).
[164] M. Bellés-Muñoz, B. Whitehat, J. Baylina, V. Daza and J. L. Muñoz-Tapia, Twisted edwards elliptic curves for zero-knowledge circuits, *Mathematics*, 2021 (cit. on p. 19).
[165] V. Buterin and C. Reitwiessner *EIP-197: Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128*, 2017 (cit. on p. 19).
[166] *Multi-party computation protocol for the key-generation step of Pinocchio zk-SNARKs*, github.com/zcash/mpc (cit. on p. 19).
[167] N. G. R. Blog *Zcash cryptography and code review* (cit. on p. 19).
[168] *KZG ceremony*, github.com/ethereum/kzg-ceremony, 2023 (cit. on p. 19).

[169] S. Kim, S. Woo, H. Lee and H. Oh Vuddy: A scalable approach for vulnerable code clone discovery, in *IEEE Security & Privacy*, 2017 (cit. on p. 20).
[170] arkworks contributors *Arkworks zksnark ecosystem*, arkworks.rs, 2022 (cit. on p. 20).
[171] G. Botrel, T. Piellard, Y. E. Housni, I. Kubjas and A. Tabaie *Consensys/gnark: V0.9.0*, doi.org/10.5281/zenodo.5819104, 2023 (cit. on p. 20).
[172] *KZG Ceremony Special Contributions*, 2023. [Online]. Available: https://github.com/ethereum/kzg-ceremony/blob/main/special_contributions.md (cit. on p. 20).
[173] *Semaphore project*, semaphore.appliedzkp.org/ (cit. on p. 29).
[174] K. W. Jie *Announcing the perpetual powers of tau ceremony to benefit all zk-snark projects*, 2019 (cit. on p. 29).
[175] D. Wang, J. Zhou, A. Wang and M. Finestone *Loopring: A decentralized token exchange protocol*, 2018 (cit. on p. 29).
[176] *Loopring trusted setup*, loopring.org/ceremony, 2023 (cit. on p. 29).
[177] Tornado.cash *Tornado.cash trusted setup ceremony* (cit. on p. 29).
[178] *Polygon hermez: Scalable payments. decentralised by design, open for everyone* (cit. on p. 29).

A Perpetual Powers-of-tau Ceremony

Ethereum project Semaphore [173], an anonymous membership proof system, launched a large private coin setup, the Perpetual Powers-of-Tau (PPOT) ceremony, for its internal Groth16 [26] proof system. They followed the impactful Sapling ceremony paradigm, developed by [70]. Two distinctions stand out in the PPOT ceremony. First, the Semaphore team visions a once-for-all Powers-of-Tau ceremony to enable SNARKs on as many circuits (aka constraints) as possible. The size of powers-of-tau targeted up to 2^{28}, which is even capable of covering the zk-rollup circuit (260 million constraints) [174]. Such a computation-intensive ceremony will take almost a whole day on a fast machine with 97GB downstream and 49GB upstream bandwidth per participant to contribute to the ceremony. Although the underlying proof system of Semphore is Groth16, other zk-SNARKs like Plonkish proof systems [53] can still benefit from the PPOT ceremony. Generally speaking, any protocol (not necessarily SNARKs) built on the KZG commitment [56] is relieved from the burden of running a powers-of-tau private coin ceremony.

Second, PPOT is a *perprtual* ceremony that supports fluid participants and continuous running mode. Though it started back in 2019, it is still open to accepting contributions on the BN254 curve [162]. Indeed, the 77th contribution was just submitted last week by the time of writing. Unlike any other past ceremonies, projects are free to choose whichever contribution to start their specific phase two ceremonies. Popular examples around the community include: Loopring [175] moves on their phase two setup [176] with the 11th rounds of PPOT ceremony, Semaphore [173] heads to its phase two ceremony after 25th contribution, and Tornado.cash [177] built their circuit-depend ceremony based on the 30th contribution to PPOT ceremony, polygon Hermez [178] selecting the 54th contribution as its starting point. The best practice, which is adopted by the above projects based on existing contributions, is to rotate the one you chose into the customized next round with your contribution. In addition, unpredictable randomness (even if all the participants colluded) reinforces the all-but-one-honest security. In practice, a random beacon derived from a designated future block in Bitcoin/Ethereum is utilized to add extra randomness and boost the trustworthiness of the whole ceremony. Upon that, the phase 1 ceremony (powers-of-tau) can be efficiently bootstrapped without organizing, coordinating, and repeating the same process repeatedly for every project. As mentioned, a successful contribution indicates a participant does no more than raise corresponding numbers to powers of an identical τ. Raised criticism of the PPOT ceremony lies in the coordinator. Specifically, the *The Sword of Damocles* of PPOT ceremonies is the centralized coordinator who orders and verifies participants' contributions. Trust in a reputation-based coordinator can be risky if he is capable of deliberately introducing biased randomness in the final round. Specifically, in practice biased randomness comes from a random beacon, which is usually instantiated with future blocks of blockchain systems, as the coordinator is usually the one who runs a ceremony.