Proposal: Framework for Snarky Ceremonies

Markulf Kohlweiss[1,2], Mary Maller[3], Janno Siim[4], Mikhail Volkhov[2]

[1] IOHK
[2] The University of Edinburgh, UK
{mkohlwei, mikhail.volkhov}@ed.ac.uk
[3] Ethereum Foundation
mary.maller@ethereum.org
[4] University of Tartu, Estonia
janno.siim@ut.ee

Abstract. Succinct non-interactive arguments of knowledge (SNARKs) have found wide-scale adoption in recent years. The most efficient SNARKs require a distributed ceremony protocol to generate public parameters, also known as a structured reference string (SRS). We propose a general security framework for non-interactive zero-knowledge (NIZK) arguments with a ceremony protocol. In particular, our framework generalizes the notion of updatable reference strings, proposed by Groth, Kohlweiss, Maller, Meiklejohn, and Miers [Crypto, 2018], to multiple independent update phases. Importantly, this allows us to also capture existing setup ceremonies as performed for Groth16 SNARKs.

1 Motivation

Zero-knowledge proofs of knowledge [GMR85, BG93] allow to prove knowledge of a witness for some NP statement while not revealing any information besides the truth of the statement. The recent progress in zero-knowledge (ZK) Succinct Non-interactive Arguments of Knowledge (SNARKs) [Gro10, Lip12, PHGR13, DFGK14, Gro16] has enabled the use of zero-knowledge proofs in practical systems, especially in the context of blockchains [BCG+14, KMS+16, SBG+19, BCG+20].

Many of the most efficient SNARKs require a trusted setup. That is, both the prover and the verifier algorithm use a structured reference string (SRS) which has to be sampled by an honest party. It is of utmost importance that the SRS comes from the correct distribution and no side information is leaked. Typically the SRS is constructed from a trapdoor, which if leaked, makes it trivial to subvert SNARK's security. Correct treatment of SRS has become one of the greatest challenges in using SNARKs in practice.

A prime example is the Groth16 [Gro16] SNARK, which has the smallest proof size and fastest verifier in the literature, and is also competitive in terms of prover time. Beyond efficiency, it has several other useful properties. Groth16 is rerandomizable [LCKO19], which is a desirable property for achieving receipt-free voting [LCKO19]. Simultaneously, it also has a weak form of simulation extractability [BKSV20] which guarantees that even if the adversary has seen some simulated proofs before, it cannot prove a new statement without knowing the witness. This is important for using Groth16 in protocols modelled using Universally Composable, a target framework for many practical applications [KMS+16, KKKZ19, DGK+21]. The prover and verifier use only algebraic operations and thus proofs can be aggregated [BMMV19]. Furthermore, Groth16 is attractive to practitioners due to the vast quantity of implementation and code auditing attention it has already received. One of the main *downsides* of Groth16 is the highly structured trusted SRS.

Several approaches have been proposed over the years to alleviate the problem of trusted SRS. The works of [BCG$^+$15, BGG17, ABL$^+$19] propose specialized multi-party computation (MPC) protocols for SRS generation ceremonies. A common feature of these protocols is that they are secure if at least one of the parties is honest. However, these schemes are not robust in the sense that all parties must be fixed before the beginning of the protocol and be active throughout the whole execution. In other words if a single party goes offline between rounds then the protocol will not terminate. Bowe, Gabizon, and Miers [BGM17] showed that the latter problem could be solved if there is access to a random beacon — an oracle that periodically produces bitstrings of high entropy — which can be used to rerandomize the SRS after each protocol phase. They do not propose a formal security model for their protocol but the protocol itself has found wide-scale adoption in practice. For example, it has been used by Zcash, Aztec protocol, Filecoin, Semaphore, Loopring, and Tornado Cash.

A different approach to strengthening trust in SNARK setups is subversion security as proposed by [BFS16]. That paper studies properties of non-interactive zero-knowledge in case the SRS is created by a dishonest party. It turns out that it is impossible obtain subversion soundness (soundness when SRS is untrusted) together even with non-subversion zero-knowledge. However, they also show that subversion zero-knowledge and standard notion of soundness are achievable at the same time. Subsequent works [ABLZ17, Fuc18] showed that even efficient SNARKs like Groth16 can obtain subversion zero-knowledge. This is done by introducing an efficient SRS verification algorithm that the prover can run before outputting a proof.

The work of [GKM$^+$18] proposes a third approach. They directly construct a SNARK where the SRS is updatable, that is, anyone can update the SRS and knowledge soundness and zero-knowledge are preserved if at least one of the updaters was honest. Subsequent updatable SNARKS like Sonic [MBKM19], Marlin [CHM$^+$20], and PLONK [GWC19] have improved the efficiency of updatable SNARKs, but they are still less efficient than for example [Gro16].

1.1 Our Contribution

We recent work [KMSV21] we propose a general security framework which unifies the MPC type of SRS generation protocols of [BGM17], updatable soundness, and subversion zero-knowledge. There we also apply this framework to the ceremony of Groth16 SNARK and prove it to be secure.

We formalize the notion of non-interactive zero-knowledge (NIZK) argument with a multi-round SRS ceremony protocol, which extends the framework of updatable NIZKs in [MBKM19]. Our definitions take a game-based approach and in particular are less rigid than multi-party computation definitions. Our security notions say that an adversary cannot forge a SNARK proofs even if they can participate in the setup ceremony. We call such a SNARK ceremonial. This notion is more permissible for the setup ceremony than requiring simulatability and is therefore easier to achieve. In particular, using our definitions we do not require the use of a random beacon, whereas it is not clear that the random beacon could be easily avoided in the MPC setting. Our definitions are applicable to SNARKs with a multiple round setup ceremony as long as they are ceremonial.

From the immediate practical point of view, this proposal (and the additional material in [KMSV21]) put the widely used [BGM17] protocol on firmer foundations.

2 Ceremonial SNARKs

In this section, we put forward our definitions for NIZKs that are secure with respect to a setup ceremony. We discuss the new notions of update completeness and update soundness that apply to ceremonies that take place over many rounds. We also define subversion zero-knowledge.

Compared to standard MPC definitions, our definitions do not include a simulator that can manipulate the final SRS to look uniformly random. We believe that the attempt to realise standard MPC definitions is what led prior works to make significant practical sacrifices e.g. random beacons or players that cannot go offline. This is because a rushing adversary that plays last can manipulate the bit-decomposition, for example to enforce that the first bit of the SRS is always 0. We here choose to offer an alternative protection: we allow that the final SRS is not distributed uniformly at random provided that the adversary does not gain any meaningful advantage when attacking the soundness of the SNARK. This is in essence an extension of updatability definitions [GKM$^+$18] to ceremonies that require more than one round.

Our security framework assumes that the SRS is split into φ_{max} distinct components $\mathsf{srs} = (\mathsf{srs}_1, \ldots, \mathsf{srs}_{\varphi_{max}})$ and in each phase of the ceremony protocol one of the components gets finalized. We formalize this by enhancing the standard definition of NIZK with an Update and $\mathsf{VerifySRS}$ algorithms. Given srs and the phase number φ, the Update algorithm updates srs_φ and produces a proof ρ that the update was correct. The verification algorithm $\mathsf{VerifySRS}$ is used to check that srs and update proofs $\{\rho_i\}_i$ are valid.

Notation. PPT denotes probabilistic polynomial time, and DPT denotes deterministic polynomial time. The security parameter is denoted by λ. We write $y \overset{r}{\leftarrow} \mathcal{A}(x)$ when a PPT algorithm $\mathcal{A}$ outputs y on input x and uses random coins r. Often we neglect r for simplicity. If $\mathcal{A}$ runs with specific random coins r, we write $y \leftarrow \mathcal{A}(x; r)$. A view of an algorithm $\mathcal{A}$ is a list denoted by $\mathsf{view}_\mathcal{A}$ which contains the data that fixes $\mathcal{A}$'s execution trace: random coins, its inputs (including ones from the oracles), and outputs[5]. We sometimes refer to the "transcript" implying only the public part of the view: that is interactions of $\mathcal{A}$ with oracles and the challenger.

Argument Description. An argument system Ψ (with a ceremony protocol) for a relation $\mathcal{R}$ contains the following algorithms:

(i) A PPT parameter generator Pgen that takes the security parameter 1^λ as input and outputs a parameter p (e.g., a pairing description)[6]. We assume that $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$ and the security parameter is given as input to all algorithms without explicitly writing it.
(ii) A PPT SRS update algorithm Update that takes as input a phase number $\varphi \in \{1, \ldots, \varphi_{max}\}$, the current SRS srs, and proofs of previous updates $\{\rho_i\}_i$, and outputs a new SRS srs' and an update proof ρ'. It is expected that Update itself forces a certain phase order, e.g. the sequential one.
(iii) A DPT SRS verification algorithm $\mathsf{VerifySRS}$ that takes as an input a SRS srs and update proofs $\{\rho_i\}_i$, and outputs 0 or 1.
(iv) A PPT prover algorithm Prove that takes as an input a SRS srs, a statement ϕ, and a witness w, and outputs a proof π.

[5] The latter can be derived from the former elements of the list, and is added to $\mathsf{view}_\mathcal{A}$ for convenience, following e.g. [GM17]

[6] We do not allow to subvert p in the context of this paper but in real life systems also this part of the setup should be scrutinized. This is arguable easier since usually p is trapdoor free.

(v) A DPT verification algorithm Verify that takes as an input a SRS srs, a statement ϕ, and a proof π, and outputs 0 or 1.
(vi) A PPT simulator algorithm Sim that takes as an input a SRS srs, a trapdoor τ, and a statement ϕ, and outputs a simulated proof π.

The description of Ψ also fixes a default $\mathsf{srs}^{\mathsf{d}} = (\mathsf{srs}^{\mathsf{d}}_1, \dots, \mathsf{srs}^{\mathsf{d}}_{\varphi_{max}})$.

Security Properties. We require that a secure Ψ satisfies the following flavours of completeness, zero-knowledge, and knowledge soundness.

Completeness of Ψ requires that Update and Prove always satisfy verification.

Definition 1 (Perfect Completeness). *An argument Ψ for $\mathcal{R}$ is* perfectly complete *if for any adversary $\mathcal{A}$, it has the following properties:*

1. *Update completeness:*

$$\Pr\left[\begin{matrix}(\varphi, \mathsf{srs}, \{\rho_i\}_i) \leftarrow \mathcal{A}(1^\lambda), (\mathsf{srs}', \rho') \leftarrow \mathsf{Update}(\varphi, \mathsf{srs}, \{\rho_i\}_i) : \\ \mathsf{VerifySRS}(\mathsf{srs}, \{\rho_i\}_i) = 1 \land \mathsf{VerifySRS}(\mathsf{srs}', \{\rho_i\}_i \cup \{\rho'\}) = 0\end{matrix}\right] = 0.$$

2. *Prover completeness:*

$$\Pr\left[\begin{matrix}(\mathsf{srs}, \{\rho_i\}_i, \phi, w) \leftarrow \mathcal{A}(1^\lambda), \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \phi, w) : \\ \mathsf{VerifySRS}(\mathsf{srs}, \{\rho_i\}_i) = 1 \land (\phi, w) \in \mathcal{R} \land \mathsf{Verify}(\mathsf{srs}, \phi, \pi) \neq 1\end{matrix}\right] = 0.$$

Our definition of subversion zero-knowledge follows [ABLZ17]. Intuitively it says that an adversary that outputs a well-formed SRS knows the simulation trapdoor τ and thus could simulate a proof himself even without the witness. Therefore, proofs do not reveal any additional information. On a more technical side, we divide the adversary into an efficient SRS subverter $\mathcal{Z}$ that generates the SRS (showing knowledge of τ makes sense only for an efficient adversary) and into an unbounded distinguisher $\mathcal{A}$. We let $\mathcal{Z}$ communicate with $\mathcal{A}$ with a message st.

Definition 2 (Subversion Zero-Knowledge (sub-ZK)). *An argument Ψ for $\mathcal{R}$ is* subversion zero-knowledge *if for all PPT subverters $\mathcal{Z}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{Z}}$, such that for all (unbounded) $\mathcal{A}$, $|\varepsilon_0 - \varepsilon_1|$ is negligible in λ, where*

$$\varepsilon_b := \Pr\left[\begin{matrix}(\mathsf{srs}, \{\rho_i\}_i, st) \leftarrow \mathcal{Z}(1^\lambda), \tau \leftarrow \mathcal{E}_{\mathcal{Z}}(\mathsf{view}_{\mathcal{Z}}) : \\ \mathsf{VerifySRS}(\mathsf{srs}, \{\rho_i\}_i) = 1 \land \mathcal{A}^{\mathcal{O}_b(\mathsf{srs}, \tau, \cdot)}(st) = 1\end{matrix}\right].$$

$\mathcal{O}_b$ is a proof oracle that takes as input $(\mathsf{srs}, \tau, (\phi, w))$ and only proceeds if $(\phi, w) \in \mathcal{R}$. If $b = 0$, $\mathcal{O}_b$ returns an honest proof $\mathsf{Prove}(\mathsf{srs}, \phi, w)$ and when $b = 1$, it returns a simulated proof $\mathsf{Sim}(\mathsf{srs}, \tau, \phi)$.

Bellare et al. [BFS16] showed that it is possible to achieve soundness and subversion zero-knowledge at the same time, but also that subversion soundness is incompatible with (even non-subversion) zero-knowledge. Updatable knowledge soundness from [GKM$^+$18] can be seen as a relaxation of subversion soundness to overcome the impossibility result.

We generalize the notion of update knowledge soundness to multiple phases. SRS is initially empty (or can be thought to be set to a default value $\mathsf{srs}^{\mathsf{d}}$). In each phase φ, the adversary has to fix a part of the SRS, denoted by srs_φ, in such a way building the final srs. The adversary can ask honest updates for his own proposal of srs^*_φ, however, it has to pass the verification $\mathsf{VerifySRS}$. The adversary can query honest updates using UPDATE through a special oracle $\mathcal{O}_{\mathsf{srs}}$, described

```
$\mathcal{O}_{\mathsf{srs}}(\mathsf{intent}, \mathsf{srs}^*, Q^*)$ // Initially $Q_1 = \cdots = Q_{\varphi_{max}} = \emptyset; \varphi = 1$
if $\varphi > \varphi_{max}$ : return $\bot$; // SRS already finalized for all phases
$\mathsf{srs}_{\mathsf{new}} \leftarrow (\mathsf{srs}_1, \ldots, \mathsf{srs}_{\varphi-1}, \mathsf{srs}^*_{\varphi}, \ldots, \mathsf{srs}^*_{\varphi_{max}})$;
if $\mathsf{VerifySRS}(\mathsf{srs}_{\mathsf{new}}, Q^*) = 0$ : return $\bot$; // Invalid SRS
if $\mathsf{intent} = \text{UPDATE}$ :
  $(\mathsf{srs}', \rho') \leftarrow \mathsf{Update}(\varphi, \mathsf{srs}_{\mathsf{new}}, Q^*); Q_\varphi \leftarrow Q_\varphi \cup \{\rho'\}$;
  return $(\mathsf{srs}', \rho')$;
if $\mathsf{intent} = \text{FINALIZE} \wedge Q_\varphi \cap Q^* \neq \emptyset$ :
    Assign $\mathsf{srs}_\varphi \leftarrow \mathsf{srs}^*_\varphi; \varphi \leftarrow \varphi + 1$;
```

Fig. 1. SRS update oracle $\mathcal{O}_{\mathsf{srs}}$ given to the adversary in Definition 3. UPDATE returns $\mathcal{A}$ an honest update for φ, and FINALIZE finalizes the current phase. Current phase φ and current SRS srs (created in FINALIZE and stored up to φ) are shared with the KS challenger. $\{Q_{\varphi_i}\}_i$ is a local set of proofs for honest updates, one for each phase.

in Fig. 1. Eventually, adversary can propose some srs^*_φ with update proofs Q^* to be finalized through FINALIZE. The oracle does it if Q^* contains at least one honest update proof obtained from the oracle for the current phase. If that is the case, then srs_φ cannot be changed anymore and the phase $\varphi + 1$ starts. Once the whole SRS has been fixed, $\mathcal{A}$ outputs a statements ϕ and a proof π. The adversary wins if $(\mathsf{srs}, \phi, \pi)$ passes verification, but there is no PPT extractor $\mathcal{E}_\mathcal{A}$ that could extract a witness even when given the view of $\mathcal{A}$.

Definition 3 (Update Knowledge Soundness). *An argument Ψ for $\mathcal{R}$ is update knowledge-sound if for all PPT adversaries $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_\mathcal{A}$ such that* $\Pr[\mathsf{Game}_{\mathsf{uks}}^{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda) = 1]$ *is negligible in λ, where $\mathsf{Game}_{\mathsf{uks}}$ is defined as:*

$$\mathsf{Game}_{\mathsf{uks}}^{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda) := \left[\begin{array}{l}(\phi, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{srs}}(\cdot)}(1^\lambda); \textit{get } (\mathsf{srs}, \varphi) \textit{ from } \mathcal{O}_{\mathsf{srs}}; w \leftarrow \mathcal{E}_\mathcal{A}(\mathsf{view}_\mathcal{A}); \\ \textbf{\textit{return}} \ \mathsf{Verify}(\mathsf{srs}, \phi, \pi) = 1 \wedge (\phi, w) \notin \mathcal{R} \wedge \varphi > \varphi_{max}\end{array}\right],$$

where SRS update oracle $\mathcal{O}_{\mathsf{srs}}$, constructing srs *depending on interaction with $\mathcal{A}$, is described in Fig. 1.*

In our definition of update knowledge soundness, we require that no adversary can convince an honest verifier of a statement unless either (1) they know a valid witness; (2) the SRS does not pass the setup ceremony verification $\mathsf{VerifySRS}$; or (3) one of the phases did not include *any* honest updates. Note that completeness and zero-knowledge hold for any SRS that passes the setup ceremony verification, even if there were no honest updates at all.

If $\varphi_{max} = 1$, we obtain the standard notion of update knowledge soundness, that is the one used with updatable NIZKs (Sonic [MBKM19] and others). For Groth16, we only need to consider the case where $\varphi_{max} = 2$, as shown in our recent work [KMSV21]. In particular, in the first phase we will generate a universal SRS $\mathsf{srs}_u = \mathsf{srs}_1$ that is independent of the relation and in the second phase we generate a specialized SRS $\mathsf{srs}_s = \mathsf{srs}_2$ that depends on the concrete relation. Both srs_u and srs_s are updatable; however, the initial srs_s has to be derived from srs_u and the relation $\mathcal{R}$. Thus, parties need first to update srs_u, and only after a sufficient number of updates can they start to update srs_s. The universal srs_u can be reused for other relations. We leave it as an open question whether ceremony protocols with $\varphi_{max} > 2$ can provide any additional benefits.

It is important to explain the role of the default SRS in the definition. Our definition allows $\mathcal{A}$ to start its chain of SRS updates from any SRS, not just from the default one; the only condition that is necessary is the presence of a single honest update in the chain. The default srs $\mathsf{srs}^{\mathsf{d}}$ is only used as a reference, for honest users. This has positive real-world consequences: since the chain is not required to be connected to any "starting point", clients only need to verify the suffix of Q^*, if they are confident it contains an honest update. In particular, clients that contribute to the SRS update can start from the corresponding proof of update.

Another point is the use of adversarial view $\mathsf{view}_{\mathcal{A}}$ in sub-ZK and UKS. The extractors in these definitions assume white-box access to the adversary primarily because SNARKs are inherently white-box [GW11] in the CRS model. Although the additional presence of RO model, as in Sonic, can in theory be used to overcome this negative result, it has not been showed in practice so far for the protocols that are relevant for this proposal. With this in mind, we assume the most powerful extractors to capture a wider class of SNARKs.

3 Open Questions

In our current formalization the argument fixes the relation $\mathcal{R}$. It could be interesting to consider update knowledge soundness definition where the universal part of the SRS can be reused for other relations. This is currently not considered by our definitions. However, introducing this change likely comes with new challenges such as allowing the adversary to choose the relation adaptively, based on its current communication transcript with the challenger. Moreover, even though the relation in Groth16 can be introduced in the second phase, it is not clear how to generalize this for many phases. What the adversary specifies is technically not a relation, but an arithmetized description (QAP) of it, which contains many elements, than can potentially be added separately during different phases.

As mentioned previously, we are also unsure if $\varphi_{max} > 2$ is beneficial for any ceremonial NIZK. We are currently only aware of use cases for $\varphi_{max} \in \{1, 2\}$. It might be an interesting future direction to investigate SNARKs with $\varphi_{max} > 2$ or to simplify our framework if more than 2 phases do not add anything.

References

ABL$^+$19. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. UC-secure CRS generation for SNARKs. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 99–117. Springer, Heidelberg, July 2019.

ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017.

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

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

BCG$^+$20. Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964. IEEE Computer Society Press, May 2020.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

BG93. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

BGG17. Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602, 2017. `http://eprint.iacr.org/2017/602`.

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

BKSV20. Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-SNARK. Cryptology ePrint Archive, Report 2020/811, 2020. `https://eprint.iacr.org/2020/811`.

BMMV19. Benedikt Bünz, Mary Maller, Pratyush Mishra, and Noah Vesely. Proofs for inner pairing products and applications. Cryptology ePrint Archive, Report 2019/1177, 2019. `https://eprint.iacr.org/2019/1177`.

CHM$^+$20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

DGK$^+$21. Ivan Damgard, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain transactions. Cryptology ePrint Archive, Report 2020/1511, 2021. `https://eprint.iacr.org/2020/1511`.

Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.

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

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

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

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

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

KKKZ19. Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019.

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

KMSV21. Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. Cryptology ePrint Archive, Report 2021/219, 2021. `https://eprint.iacr.org/2021/219`.

LCKO19. Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. SAVER: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. Cryptology ePrint Archive, Report 2019/1270, 2019. `https://eprint.iacr.org/2019/1270`.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

Mal18. Mary Maller. A proof of security for the sapling generation of zk-snark parameters in the generic group model. `https://github.com/zcash/sapling-security-analysis/blob/master/MaryMallerUpdated.pdf`, 2018. Accessed 26/02/2020.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

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

SBG$^+$19. Samuel Steffen, Benjamin Bichsel, Mario Gersbach, Noa Melchior, Petar Tsankov, and Martin T. Vechev. zkay: Specifying and enforcing data privacy in smart contracts. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1759–1776. ACM Press, November 2019.