Halo 0.9: A Halo Protocol with Fully-Succinctness

Lira Wang[1]

[1]lirawang@yahoo.com

Abstract. Zero-Knowledge Proof is a crucial tool for privacy preserving and stake proving. It allows the Prover to convince the Verifier about the validity of some statement without leaking any knowledge of his own. Quantities of zero knowledge protocols have been proposed by now and one of the state-of-the-art works is Halo [1], which is brought about by Bowe, Grigg and Hopwood. Even though nested amortization technique used in Halo, the Verifier still has to compute an O(n) operation ultimately. As a result, Halo is not a fully succinct zero-knowledge scheme and infeasible to be utilized in some scenarios such as Ethereum Smart Contract applications.

We propose Halo 0.9, which is an enhanced version of Halo aiming at the issue above. Specifically, we introduce the SRS in [2] as the substitute for the random vector in the inner product and thus transform the Pedersen vector commitment to Kate polynomial commitment [2]. On the premise of original Halo protocol remained, the computation of Verifier is in logarithmic time.

Keywords: Succinct Proof, Halo, Zero-knowledge

1 Introduction

Zero knowledge proof is a crucial tool for privacy preserving and stake proving, which was proposed by GoldWasser, Micali and Rackoff [3]. There are two parties engaged in the protocol, where the Prover is able to convince the Verifier about the validity of some statement without leaking his own knowledge. At the end of the protocol, the Verifier acquires nothing more than the statement itself. Blum, Feldman and Micali [4] further extended the notion to Non-interactive zero-knowledge proof, as known as NIZK. In 1992, Kilian [5] demonstrated the first sublinear zero-knowledge scheme in communication that has less proof size than the statement to be proved. Ever since then, quantities of researches aimed at this area with impressive progress. At present, zero-knowledge proof is wildly applied to outsourced computation, credential-identity verification and so on. The emerging technology of blockchain hugely promotes the development of zero knowledge proof. In fact, some mature applications such as Zcash [6] and Ethereum have already deployed zero knowledge proof in their system.

Generally speaking, zero knowledge proof schemes make use of random variants in order to combine multiple constraints into single one for quick proving. One of the most efficient method is zero-knowledge succinct non-interactive argument of knowledge (zkSNARK). By utilizing trusted setup and bilinear pairing, zkSNARK-based protocols [7][8][9][10] realize succinct proof with fast verification. However, the procedure of trusted setup highly depends on the specific circuits, meaning that minor modification requires a brand-new common reference string. On the other hand, in order to ensure security, multi-party computation is usually applied during trusted setup phase. This process called ceremony [11] is relatively complicated and unsuitable for updating, becoming a barrier for those methods to develop. In order to solve this issue, Groth et.al [12] proposes universal and updatable structured reference string (SRS) that one set of parameters supports various statements, and is capable of updating when needed. Though creative, the scheme in [12] still can be promoted in both proof size and verifying time. Sonic [13] as the very first potential practical zero-knowledge protocol with universal SRS, realizes fast verification with untrusted "helper". Yet the Verifier still has to compute an O(n) operation in the end of the protocol, making it not fully-succinct. Marlin [14] and Plonk [15] were successively proposed in 2019, shortly after Sonic [13], that both are fully-succinct zero knowledge proof based on SRS. It is relatively back-handed to compare them. [14] adopts R1CS constraint system and is more adaptable for "fully-dense" circuits. While [15] utilized original linear constraints focus on constant fan-in circuits, resulting in more gates and bigger size of circuits.

Structured reference string is indeed a solution for trusted setup, it is still a kind of setup phase to an extent. There are alternatives without this procedure such as zkSTARKs [16] which uses polynomial interpolation and FRI algorithm [17] that achieves fast proving and verifying. However, the proof size of [16] is relatively large even for small circuits. [18] and the enhanced version [19] base on inner product and weaker assumption that acquires logarithmic proof size yet higher verifying time, leading it suitable for proving of simple relation.

1.1 Halo Protocol

Based on the outcome of [19] and [13], Bowe, Grigg and Hopwood proposed Halo protocol in 2019 [1]. Aiming at incrementally verifiable computation, Halo realizes batched-verification via modified inner product argument and vector commitment. By compositing multiple proofs together, the Verifier' work load is greatly alleviated. Specifically, Halo fixes the second vector $\boldsymbol{b}$ with bisected representation in inner product argument and proves satisfaction of arithmetic circuits by demonstrating the constant term of a bivariate polynomial equals to zero. Moreover, it also uses the helper mode in [13] therefore delays the verification of a stream of arguments to the last one. There are several amortizing strategies used in Halo which efficiently reduce verifier's work to almost logarithmic.

Unfortunately, even though the nested amortization is utilized, the Verifier of Halo still has to compute an inner product $\langle \boldsymbol{s}, \boldsymbol{G} \rangle$ ultimately in the last argument, where $\boldsymbol{s}$ is a scalar vector and $\boldsymbol{G}$ is a random vector in cyclic group. This operation is of O(n) sized that makes Halo not fully succinct. Due to this circumstance, scheme in [1] is not compatible for scenarios where the Verifier's work is strictly restricted. As a concrete motivation, the Ethereum Smart Contract demands 6,000 gas fees for every multiplication on elliptic curve [21]. If we employ Halo as the zero-knowledge proof protocol, considering the upper bound of gas limit of Ethereum smart contract is 12 million [20], then the circuits would only allow 2,000 gates because the expensive gas fee when executing linear time operation. Therefore, the original Halo protocol in [1] is inefficient and impractical. In all, those scenarios present a rigorous challenge to the workload of the Verifier so that it is significant if we could reduce the $O(n)$ operation in [1].

1.2 Our Halo 0.9 Scheme

Based on the analysis above, we propose an enhanced version of [1] as Halo 0.9. We compare the merits and demerits of Pedersen vector commitment and Kate polynomial commitment [2]. By introducing CRS used in [2] to take over from the random group vector used in [1], therefore tactfully transform the Pedersen commitment to Kate commitment. The Verifier merely need to take $log(n)$ multiplications in finite field instead of linear time operation. This modification makes Halo fully succinct. Specifically, in the original scheme, the Verifier has to compute an inner product $\langle \boldsymbol{s}, \boldsymbol{G} \rangle$ at last, where $\boldsymbol{s} \in F^n$ is a scalar vector and $\boldsymbol{G} \in \mathbb{G}^n$ is a random group vector. This inner product can be viewed as the Pedersen vector commitment of $\boldsymbol{s}$ without referring to blind factor. Therefore, we replace $\boldsymbol{G}$ with the CRS used in [2] and the Pedersen commitment can be transformed to the Kate commitment of $\tilde{\boldsymbol{s}}$, where $\tilde{\boldsymbol{s}}$ is the dual vector of $\boldsymbol{s}$. The verifier could compute this modified formula via Open operation, which is far more efficient than naïve calculation. For more details and full protocol, see Section 3.

1.3 Comparison with Marlin and Plonk

We compare Halo 0.9 with recently proposed zero knowledge proof schemes [14] [15] that are also fully succinct. The result is shown in Table 1. For an arbitrary statement, [14] utilizes R1CS as their constraint system. Halo 0.9 adopts constraint system that is similar to that used in [13]. This constraint system composed of multiplication constraints and linear constraints which we call it as Bootle-Constraint System. On the other hand, [15] puts forward an original constraint system aiming at constant fan-in circuits that can be reduced to a permutation check. We call it as Plonk-Constraint System. Roughly speaking, Plonk-Constraint System has more gates in circuits than Bootle and R1CS, leading it more expensive when computing multiplication on both elliptic curves and finite field.

Table 1. Comparison with Marlin and Plonk

Method	Constraint System	Verifier Work
Marlin	R1CS	18 EC MUL + 2 pairing
Plonk	PLONK-Constraint System	18 EC MUL + 2 pairing
Halo 0.9 (Ours)	**Bootle-Constraint System**	**11 EC MUL + 2 pairing + log(n) FP MUL**

As described above, original Halo protocol is not suitable for scenarios such as Ethereum smart contract because its high workload for the Verifier. Aiming at this shortcoming, we propose the enhanced version of it and compare our Halo 0.9 with

Marlin and Plonk about the amount of work of the Verifier. The result is shown in Table 1, where EC MUL and FP MUL represent the multiplicate operation on the elliptic curve and in finite field respectively. By applying our improvement, Halo 0.9 only need 11 EC MUL, 2 pairing and $\log(n)$ FP MUL. It can be concluded that among the state-of-art zero-knowledge protocols with universal SRS, Halo 0.9 is the most gas-cost saving one for Ethereum smart contract applications considering FP MUL costs less gas fee than EC MUL.

2 Preliminaries

Before we present our Halo 0.9 scheme, we will first review some underlying tools and cryptographic primitives. We denote by λ the security parameter, and assume all algorithms and adversaries are probabilistic interactive Turing machine with respect to this security parameter. Sometimes we will leave the security parameter implicitly without ambiguity. We denote by $\varepsilon(\lambda)$ an unspecified function that is negligible in λ, which means that this function vanishes faster than the inverse of any polynomial in λ. We write PPT/DPT for algorithms running in probabilistic polynomial time and deterministic polynomial time in the size of their inputs respectively.

2.1 Terminology

Let $\mathbb{G}$ denotes a cyclic group of prime order p, and we assume our field $\mathbb{F}$ is of prime order p and denote by $\mathbb{F}_p$ for brevity. Let $\mathbb{G}^n$ and $\mathbb{F}_p^n$ represent vector spaces of dimension n over $\mathbb{G}$ and $\mathbb{F}_p$ respectively. Normally the generators of cyclic group are denoted by $g, h \in \mathbb{G}$. We use the notation $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ to denote a type I bilinear pairing in $\mathbb{G}$. In our paper, we use bold fonts denote vectors, e.g., $\boldsymbol{v}$ is a vector of scalars and $\boldsymbol{G}$ is a vector of group elements with their ith entry is v_i and G_i respectively. For scalars in finite field, lower letters are used and scalar multiplication is denoted by $[a]G = G^a$ for $a \in \mathbb{F}_p$ and $G \in \mathbb{G}$. We write $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = a_0 b_0 + a_1 b_1 + \cdots + a_{n-1} b_{n-1}$ for inner product of two scalars $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_p^n$ and denote the inner product of multi-scalar with group vector by $\langle \boldsymbol{G}, \boldsymbol{a} \rangle = \prod_{i=0}^{n-1} {g_i}^{a_i}$, where $\boldsymbol{a} \in \mathbb{F}_p^n$ and $\boldsymbol{G} \in \mathbb{G}^n$.
Finally, we write $y \leftarrow A(x; r)$ when algorithm A outputs y on inputs x and randomness r. For the sake of simplicity, we sometimes omit randomness r thus denote as $y \leftarrow A(x)$. We write $r \overset{\$}{\leftarrow} \mathbb{F}_p$ represents uniformly sampling r from $\mathbb{F}_p$. $x, y, z \in \mathbb{F}_p$ represents random challenge if not explicitly noted.

2.2 Commitment

Definition 1 (Commitment). *A non-interactive commitment scheme consists of a pair of probabilistic polynomial time algorithms (Setup, Com). The Setup algorithm* $(pk, sk) \leftarrow Setup(1^\lambda)$ *generates public and private key corresponding to the security parameter. The Com algorithm* $C \leftarrow Commit(pk, b;\ r)$ *takes public key* pk*, message*

$b \in M$, and randomness $r \in R$ as inputs and outputs a commitment to that specific message, where M is the message space and R is the randomness space. For ease of notation, we write $Com(b;r) = Commit(pk, b; r)$.

Definition 2 (Hiding Commitment). *A commitment scheme is said to be hiding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\varepsilon(\lambda)$ such that*

$$\left| P\left[b = b' \,\middle|\, \begin{matrix} (pk, sk) \leftarrow Setup(1^\lambda); \\ (x_0, x_1) \in M^2 \leftarrow \mathcal{A}(pk), b \overset{\$}{\leftarrow} \{0,1\}, r \overset{\$}{\leftarrow} R, \\ C = Com(\, x_b;\, r), b' \leftarrow (pk, C) \end{matrix} \right] - \frac{1}{2} \right| \leq \varepsilon(\lambda)$$

Which means that the adversaries $\mathcal{A}$ has negligible advantage given the result of commitment. If $\varepsilon(\lambda) = 0$, we say the commitment scheme is perfectly hiding.

Definition 3 (Binding Commitment). *A commitment scheme is said to be binding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\varepsilon(\lambda)$ such that*

$$P\left[|Com(x_0; r_0) = Com(x_1; r_1) \wedge x_0 \neq x_1 \,\middle|\, \begin{matrix} (pk, sk) \leftarrow Setup(1^\lambda), \\ x_0, x_1, r_0, r_1 \leftarrow \mathcal{A}(pk) \end{matrix} \right] \leq \varepsilon(\lambda)$$

If $\varepsilon(\lambda) = 0$, we say the commitment scheme is perfectly binding.

Definition 4 (Pedersen Commitment). *$x, r \in \mathbb{F}_p$, such that*

Setup: $g, h \overset{\$}{\leftarrow} \mathbb{G}$

$Com(x; r) = (g^x h^r)$

Definition 5 (Pedersen Polynomial Commitment). *For polynomial $p(x)$ with degree lower than d, suppose its coefficients is $\boldsymbol{a}$ such that $p(x) = \sum_{i=0}^{d-1} a_i x^i$. The Pedersen polynomial commitment is defined as*

Setup: $\boldsymbol{G} \overset{\$}{\leftarrow} \mathbb{G}^n, H \overset{\$}{\leftarrow} \mathbb{G}$

$Com(\boldsymbol{a}; r) = \langle \boldsymbol{a}, \boldsymbol{G} \rangle \cdot [r]H$

Pedersen polynomial commitment is perfectly hiding.

Definition 6 (Kate Polynomial Commitment). For polynomial $p(x)$ with degree lower than d, suppose its coefficients is $\boldsymbol{a}$ such that $p(x) = \sum_{i=0}^{d-1} a_i x^i$. *The Setup and Commitment phase of Kate Commitment is defined as*

Setup: $e, \alpha \overset{\$}{\leftarrow} \mathbb{F}_p, g \overset{\$}{\leftarrow} \mathbb{G}, \left(g, g^\alpha, \ldots, g^{\alpha^{d-1}}\right)$

$Com(\boldsymbol{a}; r) = \prod_{i=0}^{d-1} \left(g^{\alpha^i}\right)^{a_i}$

For the sake of brevity, we call Pedersen Commitment as PedCom and Kate Commitment as KateCom in our paper.

3 Our Scheme

Now in this section, we first give some observations about PedCom and KateCom. Then we bring about our Halo 0.9 Scheme.

3.1 Commitment Observation

As defined in Section 2, PedCom and KateCom can both commit to a certain polynomial. The main difference between them is the opening size and workload of the Verifier. Concretely speaking, comparison when committing to n-degree polynomial between PedCom and KateCom is shown in Table 2 below.

Table 2. Table captions should be placed above the tables.

Commitment scheme	Zero-knowledge	Verify Size	Prover Work	Verifier Work
PedCom	**Yes**	$O(n)$	$O(n)$	$O(n)$
KateCom	**No**	$O(1)$	$O(n)$	$O(1)$

It is clear that KateCom is more efficient than PedCom with half of the workload and less verifying size. However, KateCom doesn't support zero-knowledge open, even by making use of random polynomial mask, it can only perform limited zero-knowledge opening. This characteristic restricts KateCom from being exerted on various applications. However, we observe that the inner product computed in [1] is of the form $\langle \boldsymbol{s}, \boldsymbol{G} \rangle = \prod_{i=0}^{n-1} {g_i}^{s_i}$, which can be viewed as the PedCom of vector $\boldsymbol{s}$ without blinding factor. By taking advantage of this trait, we can tactfully circumvent the zero-knowledge issue and therefore transform the PedCom to KateCom for efficiency.

3.2 Main Idea

In Halo [1], the verifier still must ultimately perform a linear-time operation $G = \langle \boldsymbol{s}, \boldsymbol{G} \rangle$ with a scalar vector $\boldsymbol{s} \in F^n$ and a vector of group elements $\boldsymbol{G} \in \mathbb{G}^n$. The vector $\boldsymbol{s}$ has a special structure by defining a polynomial

$$f(x) = \prod_{i=1}^{k} (u_i + u_i^{-1} X^{2^{i-1}})$$

such that the vector $\boldsymbol{s}$ is the coefficients of $f(x)$. Since u_i is public knowledge, verifier could compute s_i in logarithmic time, e.g., assuming $n = 2^k$ for $k > 0$, we have $s_0 = \prod_{i=1}^{k} u_i$, $s_{n-1} = \prod_{i=1}^{k} u_i^{-1}$ etc.

One of our novel observation is to apply the SRS in [2] as a substitute for the original vector $\boldsymbol{G}$, therefore transform $\langle \boldsymbol{s}, \boldsymbol{G} \rangle$ to Kate polynomial commitment [2] which can be proved by the prover and verified by the verifier in logarithm time. The detail of our method is as following.

Let $P_v(X)$ denotes the Lagrange interpolating polynomial of vector v. For any vector $\boldsymbol{s}$, there exists a vector $\tilde{\boldsymbol{s}} := (\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n)$ satisfying that the ith degree term of $P_{\tilde{s}}(X)$ equals to the ith entry of $\boldsymbol{s}$, e.g.

$$P_{\tilde{\boldsymbol{s}}}(X) = \sum_{i=0}^{n-1} \tilde{s}_i \cdot L_i(X) = \sum_{i=0}^{n-1} s_i X^i$$

where $L_i(X)$ is the ith Lagrange basis, $\tilde{s}_i$ (resp. s_i) is the ith entry of $\tilde{\boldsymbol{s}}$ (resp. $\boldsymbol{s}$). There is $KateCom(\tilde{\boldsymbol{s}}) = g_1^{s_0} g_1^{s_1 \alpha} g_1^{s_2 \alpha^2} \ldots g_1^{s_{n-1} \alpha^{n-1}}$, where α is the SRS secret defined in [2].

To avoid the computation of $\langle \boldsymbol{s}, \boldsymbol{G} \rangle$, the verifier asks the prover to compute $\langle \boldsymbol{s}, \boldsymbol{G} \rangle$ and prove it. To prove $G = \langle \boldsymbol{s}, \boldsymbol{G} \rangle$, the Prover first computes the vector $\tilde{\boldsymbol{s}}$**.** Note that G=$\langle \boldsymbol{s}, \boldsymbol{G} \rangle$=$KateCom(\tilde{\boldsymbol{s}})$. The Verifier chooses a random challenge r and sends it to the Prover. The Prover now opens commitment $KateCom(\tilde{\boldsymbol{s}})$ at r to z, such that

$$z = \langle \boldsymbol{s}, \boldsymbol{r} \rangle$$

where $\mathbf{r} := (1, r, r^2, \ldots, r^{n-1})$. Verifier can compute $\langle \boldsymbol{s}, \boldsymbol{r} \rangle$ by

$$\langle \boldsymbol{s}, \boldsymbol{r} \rangle = \prod_{i=1}^{k} \left(u_i + u_i^{-1} r^{2^{i-1}} \right)$$

and thus verifies the validity of z with log(n) multiplications in finite field.

3.3 Full Protocol

As described in previous section, the full protocol of Halo 0.9 is showed in Figure.1.

```
Common Input:
        $g \in \mathbb{G}, (g, g^{\alpha}, \ldots, g^{\alpha^{n-1}})$        //$\alpha \overset{\$}{\leftarrow} \mathbb{F}_p$ is generated by a (possibly distributed)
                                                                            //trusted authority
        $u_i \in \mathbb{F}_p, i \in [1, n]$
        $f(x) = \prod_{i=0}^{n-1} u_i + u_i^{-1} X^{2^{i-1}}$
$\mathcal{P}$:
        $\tilde{\boldsymbol{s}} \leftarrow \mathrm{dual}(\boldsymbol{s})$,                  //compute dual vector of $\boldsymbol{s}$
        satisfying $P_{\tilde{s}}(X) = \sum_{i=0}^{n-1} \tilde{s}_i \cdot L_i(X) = \sum_{i=0}^{n-1} s_i X^i$
        $G = \prod_{i=0}^{n-1} \left( g^{\alpha^i} \right)^{s_i}$                //KateCom to $P_{\tilde{s}}(x)$
$\mathcal{P} \rightarrow \mathcal{V}$:  $G$
$\mathcal{V} \rightarrow \mathcal{P}$:  $r \overset{\$}{\leftarrow} \mathbb{F}_p$
$\mathcal{P}$:
        $z = P_{\tilde{s}}(r)$
        $\psi_r(x) = \frac{P_{\tilde{s}}(x) - P_{\tilde{s}}(r)}{x - r} := \sum_{i=0}^{n-1} \phi_i X^i$      //Open KateCom($P_{\tilde{s}}(x)$) at $r$
        $\varphi_{\mathrm{r}} = g^{\psi_r(\alpha)} = \prod_{i=0}^{n-1} \left( g^{\alpha^i} \right)^{\phi_i}$
$\mathcal{P} \rightarrow \mathcal{V}$:  $(z, \varphi_{\mathrm{r}})$
$\mathcal{V}$:
        $z' = f(r) = \prod_{i=0}^{n-1} (u_i + u_i^{-1} r^{2^{i-1}})$
        $z' \overset{?}{\Leftrightarrow} z$
        $e(G, g) \overset{?}{\Leftrightarrow} e(\varphi_{\mathrm{r}}, g^{\alpha}/g^{r}) e(g, g)^{z}$
        return 1 if all checks pass, else return 0
```

Fig. 1. Computation of $\langle \boldsymbol{s}, \boldsymbol{G} \rangle$ in Halo 0.9 Protocol

4 Conclusion and Future Work

We propose Halo 0.9, which is an enhanced version of [1] aiming at the issue that the Verifier still must compute an O(n) operation ultimately. By utilizing SRS in [2], we cleverly transform the PedCom in the original inner product to KateCom, acquiring more efficient verification and fully-succinct zero knowledge proof scheme. The comparison between Halo 0.9 and other state-of-the-art protocols illustrates that our method is the most gas-cost saving SRS-based zero-knowledge proof scheme on Ethereum smart contract with the lowest workload for the Verifier. In the future, we will further integrate Halo 0.9 with more scenarios that has high restrictions on the workload of the Verifier.

References

[1] S. Bowe, J. Grigg, D. Hopwood, Recursive proof composition without a trusted setup, Cryptology ePrint Archive, Report 2019/1021, https://eprint.iacr. org/2019/1021 (2019).

[2] Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings, pages 177–194. Springer, 2010.

[3] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. SIAM Journal on Computing, 18(1):186–208, 1989.

[4] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In STOC, pages 103–112, 1988.

[5] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In Symposium on Theory of Computing Conference – TCC 1992, pages 723–732, 1992.

[6] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, Zerocash: Decentralized anonymous payments from bitcoin, Cryptology ePrint Archive, Report 2014/349, https://eprint.iacr.org/2014/349 (2014).

[7] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Advances in Cryptology - ASIACRYPT 2010, pages 321–340, 2010.

[8] Jens Groth. On the size of pairing-based non-interactive arguments. In Advances in Cryptology - EUROCRYPT 2016, pages 305–326, 2016.

[9] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In Advances in Cryptology - EUROCRYPT 2013, pages 626–645, 2013.

[10] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: nearly practical verifiable computation. Commun. ACM, 59(2):103–112, 2016.

[11] Zooko Wilcox. The Design of the Ceremony. 2016. url: https://z.cash/blog/ the-design-of-the-ceremony/ (visited on 2018-05-01).

[12] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-snarks. In Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III, pages 698–728, 2018.

[13] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero- knowledge snarks from linear-size universal and updateable structured ref- erence strings. IACR Cryptology ePrint Archive, 2019:99, 2019.

[14] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. IACR Cryptology ePrint Archive, 2019:1047, 2019.

[15] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptology ePrint Archive, 2019:953, 2019.

[16] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046. https://eprint.iacr.org/2018/046. 2018.

[17] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, Fast reed-solomon interactive oracle proofs of proximity, Vol. 107, Prague, Czech republic, 2018, pp. Avast; RSJ. URL http://dx.doi.org/10.4230/LIPIcs.ICALP.2018.14.

[18] 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.

[19] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In Proceedings of the IEEE Sympo- sium on Security & Privacy, 2018.

[20] Wood, G. (2014). Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 1–32.

[21] Antonio Salazar Cardozo, Zachary Williamson, "EIP-1108: Reduce alt_bn128 precompile gas costs," Ethereum Improvement Proposals, no. 1108, May 2018. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-1108.