Survey of ZK-SNARK

YUNCONG ZHANG

ACM Reference Format:
Yuncong Zhang. 2020. Survey of ZK-SNARK. 1, 1 (June 2020), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Zero-Knowledge Succinct Non-interactive ARgument of Knowledge (zkSNARK) [1] enables constant-or-logarithmic-time verification of computation outputs without knowing the inputs. The zkSNARKs are particularly useful in blockchains [2, 3], where zkSNARKs facilitate creating confidential transactions that conceal part or all of the transaction details. Currently, zkSNARKs are under active research, and recent years have seen an explosion of zkSNARK constructions enjoying different properties, including constant-size proofs [4–8], universal or trustless setups [9–14], and post-quantum security [12, 13].

However, the rapid development of zkSNARK poses considerable challenges for researchers to keep up with the state-of-the-art. Existing zkSNARK constructions rely on a large and growing number of underlying tools, examples shown in Table 1. Existing reviews of literature [15–17] only introduced these tools separately instead of in a unified perspective. It is also tricky to assess and compare existing schemes due to the high-dimensionality of measurement metrics, summarized in Table 2. Most of the studies that propose new constructions include an efficiency analysis and compare their products with previous ones. But these analysis are often incomplete and, to make things worse, diverse in notations, metrics, and parameters. A comprehensive survey of literature in zkSNARKs can serve as an anchor of knowledge in this field, to enhance understanding of why and how zkSNARKs appeared and thrived, to help selecting appropriate zkSNARKs for different application scenarios, to reveal the insights behind current constructions, and to uncover the full potential of the existing techniques to construct more efficient and secure zkSNARKs.

In this paper, we present a survey that summarizes the knowledge of zkSNARKs in a systematic way. To begin the story, we recall the history of zkSNARKs to undertsand the role zkSNARK plays in a broader field called "proof systems". For the most part of this survey, we present the knowledge of zkSNARKs in three levels:

(1) In the theoretical level, we provide a definition for zkSNARK as a special case of proof systems. Based on this definition, we illustrate current theoretic results on zkSNARKs, e.g. the best efficiency we can achieve under certain security assumptions.
(2) In the technical level, we review and unify the existing frameworks of zkSNARK constructions. In this unified framework, we classify and examine the existing zkSNARKs to reveal the insights behind the construction techniques.
(3) In the application level, we propose an analysis system for measuring and comparing zkSNARKs. Instead of the traditional way that emphasizes the asymptotic complexities and practical performance in "toy examples", our system focus on evaluating the practicality of zkSNARKs in different application scenarios. Using this system, we conduct a comprehensive comparison of existing constructions.

Table 1. Examples of zkSNARK building tools

Proof Model	Computation Model	Cryptography
IP [18]	Boolean circuit	Polynomial commitments [19]
PCP/PCPP [20]	Arithmetic circuit	CRH/ECRH
IPCP	Structured circuit	Bilinear pairing [21]
IOP/IOPP [22]	Layered circuit	Fiat-Shamir [23]
LPCP/LIP [24]	Random access machine	Accumulator
	QSP/QAP/SSP [5]	Multi-party computation
	AIR/ACSP [12]	Lattic-based
	R1CS	Group of unknown order [11]

Table 2. Measurement metrics for zkSNARKs

Efficiency	Security	Functionality
Prover complexity	Soundness	Expressiveness
Verifier complexity	Zero-knowledgeness	Public verifiability
Setup complexity	Trusted setup	Universality
Proof length	Post-quantum security	(Non)Preprocessing
CRS length	Cryptographic assumption	

2 RELATED WORKS

3 PRELIMINARIES

Designing zkSNARKs requires systematic use of cryptography and computational complexity theory. We will illustrate the concepts in these fields that are most relevant to zkSNARKs.

3.1 Notations

We use $\mathcal{R}$ for an NP relationship, i.e. a set of pairs $\{(x, w) : f(x, w) = 0\}$ where x, w are bit strings of size n and f is a deterministic polynomial-time algorithm. The NP language $\mathcal{L}$ induced from $\mathcal{R}$ is the set $\{x : \exists(x, w) \in \mathcal{R}\}$. For $(x, w) \in \mathcal{R}$, we say x is an instance of $\mathcal{L}$ and string w is a witness of the fact $x \in \mathcal{L}$.

We say a system has λ-bit security if breaking this system costs at least 2^λ units of computation power. We use n for the bit-lengths of algorithms inputs. We use $\eta(n)$ for a negligible function in n,

Author's address: Yuncong Zhang.

XXXX-XXXX/2020/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

polylog(n) for a poly-logarithmic function, and poly(n) for a polynomial function. We say an algorithm is p.p.t. if the algorithm is probabilistic and can only execute for a period of length poly(n).

3.2 Proof systems

A proof system is a protocol that allows a party, namely the prover, to prove a statement to another party, namely the verifier.

Statements. Proof systems usually deal with two types of statements related to an NP language $\mathcal{L}$:

(1) given string x, the statement claims that $x \in \mathcal{L}$;
(2) given string x, the statement claims knowledge of w such that $(x, w) \in \mathcal{R}$.

Syntax. Given an NP language $\mathcal{L}$, a proof system $\Pi_{\mathcal{L}}$ consists of three algorithms (Setup, Prove, Verify).

(1) $\mathsf{Setup}(\mathsf{pp}) \to (\sigma_P, \sigma_V)$. The Setup algorithm takes public parameters pp as inputs and generates reference strings σ_P and σ_V for the prover and the verifier respectively. The reference strings σ_P and σ_V may intersect with each other, in which case, the intersection is called a *common reference string (CRS)*. The Setup algorithm must be executed before any instance of Π can start.
(2) $\langle \mathsf{Prove}(\sigma_P, x, w) \rightleftharpoons \mathsf{Verify}(\sigma_V, x) \rangle \to 0/1$. Given a pair $(x, w) \in \mathcal{R}$, the Prove algorithm takes a pair (x, w) and the reference string σ_P as inputs. The Verify algorithm takes x and the reference string σ_V as inputs. When being executed, the algorithms may exchange messages between each other. Finally, the Verify algorithm outputs 0 or 1, which is regarded as the output of this protocol.

The execution of a proof system is illustrated in Fig. 1.

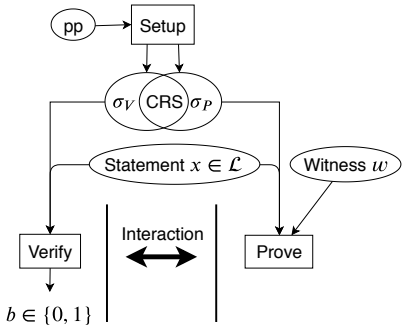

Fig. 1. Proof System

An example of a proof system execution. The rectangles represent algorithms. The ovals represent inputs to these algorithms.

Properties. A proof system $\Pi_{\mathcal{L}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is expected to satisfy the following properties:

Definition 3.1 (Completeness). A proof system $\Pi_{\mathcal{L}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ has *completeness* if for any $(x, w) \in \mathcal{R}$, correctly executing the protocol always outputs 1, i.e.

$$\Pr\left[b = 1 \,\middle|\, \begin{array}{c} \mathsf{Setup}(\mathsf{pp}) \to (\sigma_P, \sigma_V); \\ \langle \mathsf{Prove}(\sigma_P, x, w) \rightleftharpoons \mathsf{Verify}(\sigma_V, x) \rangle \to b \end{array}\right] = 1 \tag{1}$$

Definition 3.2 (Soundness). A proof system $\Pi_{\mathcal{L}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ has *soundness* if for any $x \notin \mathcal{L}$, for any adversary $\mathcal{A}$ acting as the prover, the protocol outputs 1 with probability $\epsilon < 1/2$, where ϵ is called the soundness error of $\Pi_{\mathcal{L}}$.

$$\Pr\left[b = 1 \,\middle|\, \begin{array}{c} \mathsf{Setup}(\mathsf{pp}) \to (\sigma_P, \sigma_V); \\ \langle \mathcal{A}(\sigma_P, x) \rightleftharpoons \mathsf{Verify}(\sigma_V, x) \rangle \to b \end{array}\right] = \epsilon < 1/2 \tag{2}$$

If equation (2) holds only for p.p.t. adversaries $\mathcal{A}$, we say $\Pi_{\mathcal{L}}$ has *computational soundness*. In this case, we say $\Pi_{\mathcal{L}}$ is an *argument system*.

Note that given $\Pi_{\mathcal{L}}$ with non-negligible soundness error $\epsilon < 1/2$, we can always construct $\Pi'_{\mathcal{L}}$ with exponentially small soundness error ϵ', by repeating the protocol a polynomial number of times. Therefore, requiring the soundness error to be smaller than $1/2$ is sufficient for a proof system to be useful in practice.

Variations. Based on the standard definitions described above, proof systems may take variations in many aspects.

(1) *Proof-of-knowledge.* Informally, a proof system is said to have proof-of-knowledge, if Definition 3.2 holds not only for $x \notin \mathcal{L}$, but also for $x \in \mathcal{L}$ if the adversary does not "know" any witness of x. The notion "knowledge" is formally defined by an extractor algorithm.

Definition 3.3 (Proof-of-Knowledge). A proof system $\Pi_{\mathcal{L}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ has *proof-of-knowledge* if for any $x \in \{0, 1\}^n$, for any adversary $\mathcal{A}$ acting as the prover, there exists a negligible function $\eta(n)$ and an extractor $\mathsf{Ext}^{\mathcal{A}}$ which has non-blackbox access to the adversary, such that whenever the adversary successfully cheats the verifier into outputing 1, $\mathsf{Ext}^{\mathcal{A}}$ outputs a witness w s.t. $(x, w) \in \mathcal{R}$ with overwhelming probability, i.e.

$$\Pr\left[\begin{array}{c} b = 1 \wedge \\ (x, w) \notin \mathcal{R} \end{array} \,\middle|\, \begin{array}{c} \mathsf{Setup}(\mathsf{pp}) \to (\sigma_P, \sigma_V); \\ \langle \mathcal{A}(\sigma_P, x) \rightleftharpoons \mathsf{Verify}(\sigma_V, x) \rangle \to b; \\ \mathsf{Ext}^{\mathcal{A}}(\sigma_P, x) \to w \end{array}\right] = \eta(n) \tag{3}$$

If equation (3) only holds for computationally bounded adversaries, we say the proof system has *argument-of-knowledge*.

(2) *Zero-knowledge.* A proof system is said to be zero-knowledge if the verifier cannot extract any "knowledge" from the interaction transcripts, i.e. the collection of all the messages sent during the protocol execution. The zero-knowledge property is formally defined by a simulator algorithm.

Definition 3.4 (Zero-knowledgeness). Let $\mathsf{tr} = [\mathsf{Prove}(\sigma_P, x) \rightleftharpoons \mathsf{Verify}(\sigma_V, x)]$ be the interaction transcript between the prover and the verifier, and call (σ_V, tr) the view of the verifier. A proof system $\Pi_{\mathcal{L}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is zero-knowledge if for any adversary $\mathcal{A}$ acting as the verifier, there exists a simulator $\mathsf{Sim}(\cdot)$ such that for any $(x, w) \in \mathcal{R}$, $\mathsf{Sim}(x)$ is

indifferentiable from the view of $\mathcal{A}$ during the protocol execution, i.e. there exists negligible function $\eta(n)$, s.t. for any differentiator $\mathcal{D}$:

$$\left|\begin{array}{l}\Pr\left[\mathcal{D}(\sigma_V, tr) = 1 \left| \begin{array}{c}\mathsf{Setup}(\mathsf{pp}) \to (\sigma_P, \sigma_V); \\ [\mathsf{Prove}(\sigma_P, x) \rightleftharpoons \mathcal{A}(\sigma_V, x)] \to \mathsf{tr}\end{array}\right.\right] \\ \\ \quad - \Pr\left[\mathcal{D}(\sigma_V, tr) = 1 \left| \mathsf{Sim}(x) \to (\sigma_V, tr)\right.\right]\end{array}\right| = \eta(n) \quad (4)$$

The definition of zero-knowledge has some variations. *Honest-verifier zero-knowledge ... Unconditional zero-knowledge* or *perfect zero-knowledge ... Statistical zero-knowledge ... Computational zero-knowledge ...*

(3) *Interaction models.* A standard proof system only admits normal interactions, i.e. the prover and the verifier send messages to each other, and each message is read in the whole by the receiver...
Non-interactive ... Probabilistically Checkable Proof (PCP) ... Interactive PCP (IPCP) ... Linear PCP (LPCP) ... Interactive Oracle Proof (IOP) ...

Efficiency. The efficiency of a proof system is measured by the efficiency of each of the algorithms (Setup, Prove, Verify), sizes of the reference strings (σ_P, σ_V), and the communication cost...

Definition 3.5 (Succinctness). A proof system $\Pi_{\mathcal{L}}$ is succinct if...

Definition 3.6 (Scalability). A proof system $\Pi_{\mathcal{L}}$ is scalable if...

3.3 zkSNARK

With the above definitions on proof systems and their properties, we can now give the definition of zkSNARKs.

Definition 3.7 (zkSNARK). A *zkSNARK* is a proof system that is zero-knowledge (Definition 3.4), succinct (Definition 3.5), non-interactive, and argument-of-knowledge (Definition 3.3).

Depending on how Setup works, a zkSNARK may be in one or more of the following models.

Preprocessing. A zkSNARK is called *preprocessing* if ...

Universal setup. A zkSNARK has *universal setup* if ...

Transparent setup. A zkSNARK has *transparent setup* if ...

3.4 Cryptography

All the existing zkSNARKs are obtained by applying a cryptographic transformation to an information-theoretic proof system [15]. We will introduce the necessary cryptographic tools behind the major contributions to zkSNARKs.

Commitment scheme. A cryptographic commitment scheme enables ... A commitment scheme is a tuple $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ of p.p.t. algorithms.

(1) $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ generates public parameters given the security bit number
(2) $\mathsf{Comm}(\mathsf{pp}, m) \to (\mathsf{cm}, r)$ takes a message m and generates a commitment cm, together with an opening hint r
(3) $\mathsf{Open}(\mathsf{pp}, \mathsf{cm}, m, r) \to b \in \{0, 1\}$ takes a message m, a commitment cm and an opening hint r, and verifies if cm is a valid commitment of m. If $b = 1$, we say (m, r) is a correct opening of cm.

A commitment scheme is expected to be *binding*.

Definition 3.8 (Binding). A commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ is binding if ...

A commitment scheme can optionally be *hiding*.

Definition 3.9 (Hiding). A commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ is hiding if ...

Polynomial commitment. Polynomial commitment schemes are variants of commitment schemes that have message space $R[X]$, i.e. polynomials over ring R, and allow opening a single evaluation of the commited polynomial [19]. A polynomial commitment scheme is a tuple $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open}, \mathsf{Eval})$ where $(\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ is a commitment scheme and $\mathsf{Eval}(\mathsf{pp}, \mathsf{cm}, x, y, d) \to b \in \{0, 1\}$ is a protocol ... Except the binding and hiding properties, a polynomial commitment scheme Γ is also expected to be *correct* and *evaluation binding*.

Definition 3.10 (Correct). A polynomial commitment sheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open}, \mathsf{Eval})$ is correct if ...

Definition 3.11 (Evaluation binding). A polynomial commitment sheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open}, \mathsf{Eval})$ is evaluation binding if ...

Sometimes we need a stronger property than the evaluation binding called *knowledge soundness* that requires the prover to "know" the committed polynomial [11].

Definition 3.12 (Knowledge soundness). A polynomial commitment sheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open}, \mathsf{Eval})$ has knowledge soundness if ...

Accumulator. Accumulators are another kind of variations of commitment schemes. Instead of commiting a single element, an accumulator allows commiting to a list of elements, and ...
Merkle-tree is an example of accumulator ...

Fiat-Shamir transformation and random oracle. Fiat-Shamir transformation is the standard way to transform a public-coin interactive protocol to a non-interactive scheme. The Fiat-Shamir transformation works by simulating the verifier challenges by the hash value of prover messages. However, the security of the resulting non-interactive scheme is established only in the random oracle model. Current security proving techniques require modeling the hash function by a random oracle, which is an ideal functionality that does not exist in reality.
A random oracle is ...

Bilinear pairing. Given groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, with $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$, a bilinear pairing [21] is a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies...

4 ZKSNARK CONSTRUCTION PARADIGMS

Having learned about the definitions and the building tools, we are ready to investigate the details of current constructions. As mentioned above,

9.1 Efficiency

In evaluating the efficiency of zkSNAKRs, the properties we care most about are proving time, verification time, proof sizes, and common reference string sizes. Fig. 2 summarizes the asymptotic complexities of current zkSNARK implementations.

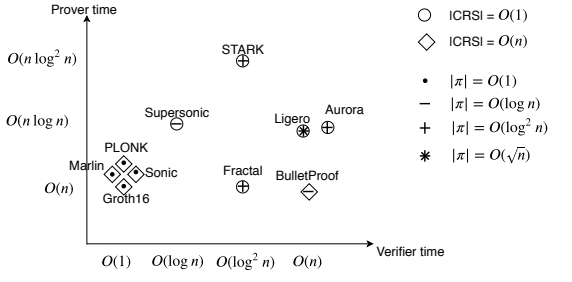

Fig. 2. Asymptotic complexities of zkSNARK implementations

The big-O notation here hides the security parameter λ. The n denotes the statement size. For circuit-based zkSNARKs, n is the circuit size. For statements with succinct representation, e.g. STARK, n is the length of execution trace, i.e. the size of the circuit representing the unrolled computation.

9.2 Security

9.3 Functionality

10 CONCLUSION

REFERENCES

[1] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 326–349. ACM, 2012.

[2] 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, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014.

[3] Shifeng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, volume 10493 of *Lecture Notes in Computer Science*, pages 456–474. Springer, 2017.

[4] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

[5] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, 2013.

[6] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[7] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252. IEEE Computer Society, 2013.

[8] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive arguments for a von neumann architecture. *IACR Cryptol. ePrint Arch.*, 2013:879, 2013.

[9] 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, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 698–728. Springer, 2018.

[10] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptol. ePrint Arch.*, 2019:99, 2019.

[11] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 677–706. Springer, 2020.

[12] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, 2018:46, 2018.

[13] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019.

[14] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104. ACM, 2017.

[15] E. Tromer. D. Benarroch, L. T. A. N. BrandaÌČo. Zkproof community reference, 2019. https://zkproof.org.

[16] Anca Nitulescu. A gentle introduction to snarks. 2019.

[17] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, 2015.

[18] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985.

[19] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.

[20] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31. ACM, 1991.

[21] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[22] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

[23] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in*

Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. https://doi.org/10.1007/3-540-47721-7_12.

[24] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 315–333. Springer, 2013.