SoK: Systematically Evaluating and Constructing zkSNARKs

YUNCONG ZHANG

ACM Reference Format:
Yuncong Zhang. 2020. SoK: Systematically Evaluating and Constructing zkSNARKs. 1, 1 (July 2020), 8 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. This family of proof systems is particularly useful in blockchains [2, 3], where it conceals part or all of the transaction details from the public. Recent years have witnessed an explosion of zkSNARK constructions and implementations enjoying different properties, including constant-size proofs [4–8], universal or trustless setups [9–14], and post-quantum security [12, 13].

The existing large number of zkSNARK constructions have a wide variety of property combinations related to their efficiency, security, and functionality. As balancing these properties is complicated, each construction can claim itself to be the best for a particular choice of focuses. This complexity makes it hard for engineers to select a suitable zkSNARK for specific tasks and also difficult for researchers to identify the state-of-the-art. Moreover, this research field has diversified into many branches [10, 14–17] that rely on different toolsets [11, 18–20] or models [5, 21, 22]. The barriers between these branches increase with the rapid development of zkSNARKs, resulting in a higher possibility of reinventing tools or repeating mistakes. A survey of knowledge for zkSNARKs is needed to provide a unifying vision of this field in terms of its theoretical results, techniques, targets, and challenges, to assist researchers to move forward in this field.

In this paper, we present a survey that summarizes the knowledge of zkSNARKs. This work aims at the following types of readers:

- readers interested in the academic history of zkSNARKs
- researchers to understand the theoretical contributions of zkSNARKs and the challenges this field is facing
- designers to understand the existing technologies in this field
- developers to select a zkSNARK construction or implementation suitable for their tasks

Recall of history. We introduce the history of zkSNARKs (Sect. 3) from the landmarking work of Goldwasser, Micali, and Rackoff [23] that proposed *zero-knowledge proofs (ZKP)*, a precursor of zkSNARK, to the recent explosion of new constructions and implementations.

Author's address: Yuncong Zhang.

XXXX-XXXX/2020/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

As a variant of ZKP, zkSNARK is the product of decades of works devoted to improving zero-knowledge proofs. We briefly introduce the motivations and techniques behind these works. Fig. 1 summarizes the history of zkSNARKs.

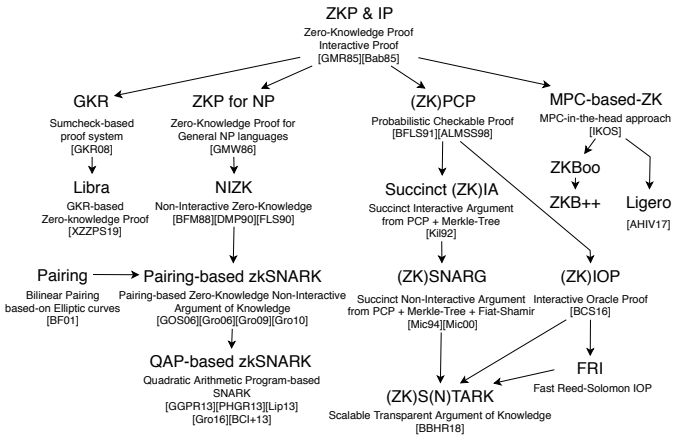

Fig. 1. A Summary of zkSNARK History

Theoretical results and challenges. We provide definitions for the concepts related to zkSNARKs (Sect. 4.2 to 4.3), starting from general concepts e.g., proof systems, NP languages, to variations of zkSNARKs, e.g., preprocessing zkSNARK, transparent zkSNARK. Based on these definitions, we illustrate the theoretical results (Sect. 4.4), including widely believed assumptions and proved theorems. We intuitively explain why these statements are true, or widely believed to be correct, and refer to the literature for readers interested in the proof details. We also point out the challenges and open problems in zkSNARKs.

Building tools. We describe the toolset used in the existing zkSNARK constructions (Sect. 5). We focus on describing the syntax and properties of the tools, i.e., how to use them and what to expect from them. We also mention the ideas to realize some tools, for example, the Merkle-Tree as a realization of vector commitment. We refer to the literature for readers interested in the details of how to construct these tools.

Existing constructions. We start describing the existing zkSNARK constructions (Sect. 6) by explaining existing frameworks in the literature (Sect. 6.1) that try to unify these constructions. However, none of these frameworks covers the entire state-of-the-art.

We build a new framework by noticing that all zkSNARKs are essentially determined by how they address the following issues:

- Which NP-Complete problem to choose;
- What is the underlying interactive model;
- How to simulate the verifier randomness unpredictably;
- How to compress the prover message;

Table 1. Measurement metrics for zkSNARKs

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

- How to save the verifier computation.

We then describe the current constructions grouped according to the research branches they reside in:

- PCP-based zkSNARKs (Sect. 6.2)
- Pairing-based zkSNARKs (Sect. 6.3)
- IP-based zkSNARKs (Sect. 6.4)
- Recursive zkSNARKs (Sect. 6.5)

We position each branch and the inside constructions into our framework by illustrating how they address the issues we mentioned above. For each construction, we provide a simplified version of the algorithms, and refer to the original literature for details.

Analysis and comparison. We address the issue that most zkSNARK users care about: how to select a suitable zkSNARK construction for a particular task? For this question, we summarize the properties in three groups: efficiency, security, and functionality, as shown in Table 1.

For efficiency properties, e.g., verifier complexity, the property values are expressed as asymptotic functions in some parameters, e.g., the security parameter, the length of the NP witness instance. For other properties, we enumerate and explain the possible values. For example, the possible values for "universality" include: universal, bounded-size universal, updatable, circuit-specific, parallel-circuit-only.

We list all the property values for each zkSNARK construction. We also select some application scenarios. For each scenario, we discuss which properties are most important, and suggest some constructions that are suitable for this scenario.

2 RELATED WORKS

3 A BIOGRAPHY OF ZKSNARK

The zkSNARKs are variations of *zero-knowledge proofs (ZKP)* that originated from the ground-breaking work by Goldwasser, Micali, and Rackoff [23]. Zero-knowledge proofs are protocols allowing a party, called the prover, to prove a statement to another party, called the verifier, interactively without leaking secret information. This work also introduced the broader concept of *interactive proofs*, which was simultaneously and independently proposed in the work of Babai [24] in the name of Arthur-Merlin (AM) proofs. The major difference between AM proofs and interactive proofs is that AM proofs are public-coin proofs, which means all randomnesses the verifier uses must be revealed later to the prover.

All the precursors of zkSNARKs, including zkSNARKs themselves, can be viewed as special cases of *interactive proofs* with different property combinations. Particularly, non-interactive proofs are special interactive proofs where the interaction consists of a single message. For clarity, we prefer to use the term *proof systems* for this broader concept, and refer to interactive proofs only when the interaction contains at least two messages from different parties.

Zero-knowledge proofs demonstrated wide applications [25, 26] soon after proposition. However, the zero-knowledge proof scheme provided by Goldwasser et al. is only of theoretical interest and not practical for real-world applications. Followup works focused on two goals: improving the efficiency to achieve *succinct* interactive proofs, and constructing non-interactive zero-knowledge (NIZK) schemes. Succinctness means the communication cost and optionally the verifier computation cost are lower than the case when the prover directly reveals the witness to the verifier.

The first breakthrough in improving the efficiency is the proposition of the *Probabilistically Checkable Proof (PCP)* model by Babai et al [21]. The PCP model allows the prover to send to the verifier a proof oracle that the verifier can query probabilistically, instead of a complete proof string that the verifier must read in the whole. This additional power makes constructing succinct PCP easier than constructing succinct interactive proofs. The PCP theorem [27] later proved by Arora et al. indicates the existence of succinct PCPs for any NP languages.

However, the PCP is only an ideal model as the proof oracle is unrealistic to implement. Regarding this issue, Kilian proposed a method to transform any PCP into a four-message interactive proof [28] using cryptographic tools, including collision-resistant hash functions and Merkle-trees. Kilian's construction is zero-knowledge if the underlying PCP is zero-knowledge. Meanwhile, the introduction of cryptography restricts the security to hold only against computationally bounded provers, since all-powerful provers can successfully cheat the verifier with false statements by breaking the hash function. Proof systems with this relaxed security requirement are called *argument systems* [29].

For the other goal regarding non-interactivity, Blum et al. [30] showed that NIZK is possible if the prover and the verifier are assumed to share a common random string before the prover generates any proofs. They also proposed the first NIZK, under the computational assumption that products of two large primes are indifferentiable from products of three large primes. Furthermore...

4 DEFINITIONS AND THEORETICAL RESULTS

The zkSNARKs are special proof systems satisfying a specific set of properties. We first give the definitions of proof systems and several properties that a proof system may have. Then we define zkSNARKs by referring to these definitions. Finally, we discuss the theoretical results about different properties of proof systems

and demonstrate the minimum assumptions underlying desired combinations of properties.

4.1 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, 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).

4.2 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. 2.

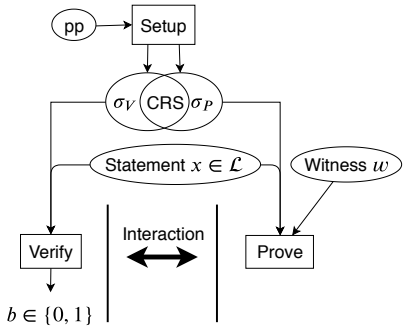

Fig. 2. Proof System
An example of a proof system execution. The rectangles represent algorithms. The ovals represent inputs to these algorithms.

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

Definition 4.1 (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 4.2 (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 4.2 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 4.3 (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 4.4 (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 $\mathcal{D}$:

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

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 4.5 (Succinctness). A proof system $\Pi_{\mathcal{L}}$ is succinct if the communication cost of $\Pi_{\mathcal{L}}$ is at most polylogarithmic to the witness size.

Definition 4.6 (Scalability). A proof system $\Pi_{\mathcal{L}}$ is scalable if...

4.3 zkSNARK

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

Definition 4.7 (zkSNARK). A *zkSNARK* is a proof system that is zero-knowledge (Definition 4.4), succinct (Definition 4.5), non-interactive, and argument-of-knowledge (Definition 4.3).

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 ...

4.4 Security Assumptions

After the introduction of zero-knowledge proofs by Goldwasser et al. [23], Goldreich, Micali, and Wigderson showed that zero-knowledge proofs exist for any NP languages, under the assumption that one-way function exists [25]. The existence of one-way functions is almost the weakest assumption in cryptography. The only weaker one is P≠NP. It is also proved that statistical ZK proofs (i.e., both ZK and soundness are statistical) are impossible for NP-complete languages unless P=NP [31–33]. Therefore, considering general NP languages, at least one side—ZK or soundness—must be computational. In the interactive case, it is demonstrated that both flavors exist: statistical ZK with computational soundness, or statistical soundness with computational ZK [25, 29, 34].

Table 2. Security Assumptions for Different Proof Systems

Name	Inter.	ZK	Succ.	Sound	Assum.	Model
NP	×	×	×	Uncond.	None	Std.
IP	✓	×	×	Stat.	None	Std.
ZKP	✓	✓	×	Stat.	OWF	Std.
NIZK	×	✓	×	Stat.	TDP	CRS
sARG	✓	×	✓	Comp.	CRH	Std.
sZKA	✓	✓	✓	Comp.	CRH	Std.
sNIZK	×	✓	✓	Comp.	CRH	CRS / RO
SNARG	×	×	✓	Comp.	nonf.	CRS / RO
SNARK	×	×	✓	AoK	nonf.	CRS / RO
zkSNARK	×	✓	✓	AoK	nonf.	CRS / RO

In the non-interactive case, NIZK constructions only had computational ZK until ...

Succinctness is another property that poses stronger security assumptions on the proof system. Succinctness is only possible for argument systems, i.e. proof systems with only computational soundness [35–38]. We provide an intuitive proof for the nonexistence of succinct statistically-sound proof systems for NP-complete language $\mathcal{L}$. Recall that succinct proof systems have only $O(\log n)$ computation cost where n is the size of NP witness w. If a succinct proof system is also statistically-sound, i.e. it is secure against unbounded prover, the prover should not be able to find valid responses with probability over 1/2. Since the prover can search through the entire response space, this means valid responses do not exist at all. However, that would facilitate a probabilistic $O(n)$ algorithm $\mathcal{A}$ deciding the language $\mathcal{L}$ with error 1/2: for any $x \in \mathcal{L}$, $\mathcal{A}$ randomly sample the verifier challenges, loop through the response space in $O(n)$ time, and if $\mathcal{A}$ does not find any valid response, $\mathcal{A}$ outputs 0. Therefore, $\mathcal{L}$ is a degenerated language in BPP, i.e. languages having p.p.t. deciding algorithms, which would indicate the collapse of the polynomial hierarchy. Hence we proved succinct statistically-sound proof systems should not exist for any interesting NP problems.

Furthermore, ...

Table 2 summarizes the minimum security assumptions underlying different proof systems.

5 BUILDING TOOLS

Here we illustrate some tools in cryptography and computation complextiy theory that zkSNARKs are built on.

5.1 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 5.1 (Binding). A commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ is binding if ...

A commitment scheme can optionally be *hiding*.

Definition 5.2 (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 committed polynomial [18]. 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 for the binding and hiding properties, a polynomial commitment scheme Γ is also expected to be *correct* and *evaluation binding*.

Definition 5.3 (Correct). A polynomial commitment sheme $\Gamma = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open}, \mathsf{Eval})$ is correct if ...

Definition 5.4 (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 5.5 (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 committing a single element, an accumulator allows committing to a list of elements, and ...

Merkle-tree is an example of an accumulator ...

5.2 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 ...

5.3 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 [19] is a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies...

6 EXISTING ZKSNARK CONSTRUCTIONS

Having introduced the theoretical definitions, we are ready to investigate the details of current constructions.

6.1 Construction Framework

Before explaining the details, it would be convenient to put the various constructions into a common framework. We will briefly review the existing frameworks and demonstrate our unified framework.

Bünz et al. [11] proposed a framework by noting that many SNARKs can be obtained by compiling a polynomial IOP with a polynomial commitment scheme and the Fiat-Shamir protocol. A polynomial IOP is an IOP where the proof oracles are restricted to a polynomial, i.e. an oracle that responses with $f(x)$ at query x for some polynomial f of bounded degree. The polynomial commitment scheme is responsible for ensuring the prover to behave like a polynomial oracle.

The ZKProof Community in a reference document [39] presents a more general framework saying that all the ZKPs are obtained by applying a cryptographic transformation to an information-theoretic proof system.

Ben-Sasson recently showed another perspective to view all cryptographic proofs, not just zkSNARKs, as a combination of an *arithmetization* method and a polynomial *low degree checking (LDC)* scheme [40]. The arithmetization step transforms a computation satisfaction problem into an algebraic problem that is more friendly with proof systems. The algebraic problem is usually described as a polynomial equation. The low degree checking step verifies that the polynomials provided by the prover are within appropriate degree limits.

These different frameworks focus on different parts of the same zkSNARK construction pipeline, illustrated in Fig. 3. Bünz et al. [11] focuses on abstracting all the proof systems with polynomial IOP and the first compilation into argument systems. Ben-Sasson [40] focuses on arithmetization and views the other three steps in the whole as an LDC scheme.

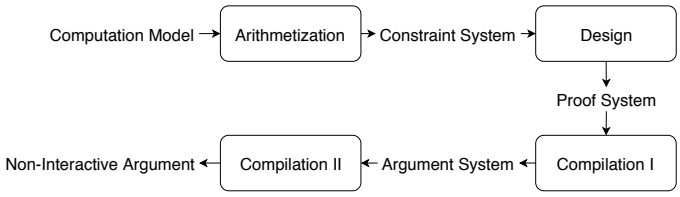

Fig. 3. zkSNARK Construction Pipeline

We propose a new framework based on this pipeline, ...

We will introduce the current zkSNARK constructions grouped by the research lines. In each group, we will extract the common techniques they share and dive into the details of their differentiations ...

6.2 PCP-based zkSNARKs

The PCP-based approach is the result of the series of works of Blum et al. [21], Kilian [28], and Micali [41]. The idea is to design a ZK-PCP for the given language, then compile it into a succinct

interactive argument using a commitment scheme [28], and finally apply the Fiat-Shamir transformation to obtain a zkSNARK [41]. For those zkSNARKs based on generalizations of ZK-PCP, e.g. ZK-IPCP, ZK-IOP, we also categorize them into this group.

Most of the researches in this category focus on the design of ZK-PCP. Therefore, they mainly differ in the arithmetization and proof system design. Another two lines of work fall into this category: the Multi-Party Computation (MPC)-based, and the Polynomial IOP (PIOP)-based zkSNARKs.

STARK..

Aurora.

Fractal.

6.2.1 MPC-based zkSNARKs. The MPC-based approach presents a way to transform a secure multi-party computation for any NP language into a ZK-IPCP. This line of work originated from Ishai, Kushilevitz, Ostrovsky, and Sahai (IKOS) [42], ...

A multi-party computation scheme is ...

ZKBoo.

ZKB++.

Ligero.

6.2.2 PIOP-based zkSNARKs.

Sonic.

PLONK.

Supersonic.

6.3 Pairing-based zkSNARKs

The pairing-based zkSNARKs have one of the most efficient verifier and the smallest proof size. As a sacrifice, they rely on an expensive per-circuit trusted setup. This line of research originated from the series of works by Groth, Ostrovsky, and Sahai [43–45], after the introduction of bilinear-pairing by Boneh et al. [19].

GGRP13.

Pinocchio.

BCGTV13.

Groth16.

6.4 IP-based zkSNARKs

The interactive proof (IP)-based line of research started from the Muggles paper [46] by Goldwasser et al. that brings the GKR protocol. However, these studies did not produce a zkSNARK until T. Xie et al. proposed Libra [15] which introduces the zero-knowledge property to GKR.

To learn about the Libra zkSNARK, it is inevitable to first introduce the famous sum-check protocol, which is the foundation of GKR protocol.

The Sumcheck protocol.

The GKR protocol.

Libra.

6.5 Recursive zkSNARKs

The recursive technique is based on the insight that we can further verify the zkSNARK verification computation recursively by another zkSNARK. Recursive zkSNARKs work with two primitives: proof-carrying data (PCD) and bootstrapping...

7 ANALYSIS FRAMEWORK

Evaluating and comparing the zkSNARK constructions is challenging because of the high dimensionality of metrics. To mitigate this problem, we categorize these metrics into three groups: efficiency, security, and functionality. Instead of presenting all the dimensions simultaneously as efficiency analysis usually does, we design several real-world application scenarios. Each scenario specifies two sets of criteria: the hard criteria filter for zkSNARKs that satisfy given conditions, and the soft criteria define an order over these filtered zkSNARKs.

7.1 Computation Delegation

A common application of zkSNARKs is computation delegation. In this scenario, the verifier is usually a device with weak computation power, e.g. a smartphone, while the prover has much more computational resources, e.g. a cloud server. The verifier may delegate computation tasks to the prover, but the prover is not completely trustable. In this case...

7.2 Organization with Authorized Leader

7.3 Value-transfer-only Permissionless Blockchain

In a permissionless blockchain, if most activities are transfers of currency ownerships, e.g. in Bitcoin or Zerocash, the verifier efficiency, the proof size, and the public verifiability are crucial. Meanwhile, other properties e.g. high prover efficiency, post-quantum security, transparency are nice but not fate-determining. Expressiveness and universality are less interesting as the blockchain functionality is simple and stable. In this case...

7.4 Permissionless Blockchain with Smart Contract

In permissionless blockchains supporting Turing complete smart contracts, e.g. Ethereum, Nervos, the verifier efficiency, the proof size, and the public verifiability are as important as in value-transfer-only blockchains. Meanwhile, expressiveness and universality are as significant as efficiency...

8 CONCLUSION

We recalled the history of zkSNARKs and summarized the knowledge of zkSNARKs in three levels: the theoretical level, the technical level, and the application level. At the theoretical level, by recalling the history, we introduced the proof systems and presented a definition for zkSNARKs in this broader context. To better understand the zkSNARK constructions, we demonstrated the theoretical results about proof systems and zkSNARKs, e.g. the security assumptions required for different combinations of efficiency and functionality. At the technical level, we introduced existing zkSNARK construction frameworks and presented a unified framework. We provided

the details of different zkSNARK research lines in the perspective of this framework. Finally, at the application level, we analyzed and compared the zkSNARKs in terms of efficiency, security, and functionality, in different application scenarios.

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] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 733–764. Springer, 2019.

[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] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from rams to delegatable succinct constraint satisfaction problems: extended abstract. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 401–414. ACM, 2013.

[23] 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.

[24] László Babai. Trading group theory for randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187. IEEE Computer Society, 1986.

[26] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[27] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[28] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992.

[29] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[30] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112. ACM, 1988.

[31] William Aiello and Johan Håstad. Perfect zero-knowledge languages can be recognized in two rounds. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 439–448. IEEE Computer Society, 1987.

[32] Lance Fortnow. The complexity of perfect zero-knowledge (extended abstract). In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 204–209. ACM, 1987.

[33] Rafael Pass and Abhi Shelat. Unconditional characterizations of non-interactive zero-knowledge. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 118–134. Springer, 2005.

[34] Gilles Brassard and Claude Crépeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 188–195. IEEE Computer Society, 1986.

[35] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[36] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[37] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1-2):1–53, 2002.

[38] Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2005.

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

[40] Eli Ben-Sasson. The cambrian explosion of crypto proofs, 2020. https://medium.com/starkware/the-cambrian-explosion-of-crypto-proofs-7ac080ac9aed.

[41] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[42] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30. ACM, 2007.

[43] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2006.

[44] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.

[45] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. 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 321–340. Springer, 2010.

[46] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122. ACM, 2008.