zkSNARKs Cryptography Protocol Survey

Leo Guo

July 13, 2022

Contents

1 zkSNARKs Cryptography Protocol Survey

1.1 Basic information

Item	zkSNARKs Cryptography Protocol Survey
Name	Leo Guo
Discord Account	toeinriver7694
Study Group	team3
Assignment	zkSNARKs Cryptography Protocol Survey
GitHub	`https://github.com/tlkahn/zkSNARKs-Cryptography-Protocol-Survey`

1.2 Introduction

1.2.1 Overview

zkSNARK is one of the most important Zero-Knowledge Proof (ZKP) methods used in privacy computing and blockchain technology. zkSNARK stands for: zero-knowledge Succinct Non-Interactive ARgument of Knowledge, which is a protocol through which a prover can quickly convince a verifier on knowledge of a secret without revealing anything about it. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer introduced the term zkSNARKs in 2012 [Bit+12] The core theoritecal components of zkSNARK are summarized below:

- zero-knowledge (zk) According to Goldwasser, Micali, and Rackoff [GMR85], who first introduced the "zero-knowledge" concept as "*Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question*". There are three essential attributes of zk featuring protocols:
  - Completeness: If the statement is true, then the verifier always accepts the proof π.
  - Soundness: If the statement is false, then the prover can only convince the verifier with a negligible probability.
  - Zero-knowledge: a correct proof π does not leak any information about the secret.
- Succictness (S) The first constructions of SNARK protocols were inspired by the PCP theorem [AS98] which shows that NP statements have "short" Probabilistic Checkable Proofs (PCP). New instantiations were found which allow faster and shorter proofs, when a preprocessing state is permitted. With proper pre-processing and arithematization, the proof becomes realistically feasible w.r.t proof size, running time, verification time etc. The formal definition is below [Bit+12]:

  > The length of the proof Π that $\mathcal{P}(y, w, vgrs)$ outputs, as well as the running time of nu$(priv, y, \Pi)$, is bounded by:
  >
  > $$p(k + |y|) = p(k + |M| + |x| + logt)$$
  >
  > where p is a universal polynomial that does not depend on $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$. $\mathcal{R}_{\mathcal{U}}$ is the universal relation [BG02], a.k.a the set $\mathcal{R}_{\mathcal{U}}$ of instance-witness pairs (y, w), where $y = (M, x, t)$, $|w| \leq$

> t and M is a Turing machine such that M accepts (x, w) after at most t steps.

- "Non-interactive" (N) means the prover can publish a single one-way message to the verifiers, with no need of further back-and-forth interactions.
- "ARguments" (AR) means the verifier is only protected against computationally limited provers. Provers with computationally unlimitted power can fake proofs/arguments on wrong statements (with enough computational power, any public-key encryption can be broken). This is also called “computational soundness”, as opposed to “perfect soundness”. [1]
- "of Knowledge":

  SNARKs are SNARGs where computational soundness is replaced by knowledge soundness, which roughly means the prover is not able to construct a proof without knowing the witness w. [1]

1.2.2 Big idea

As the name hints, zkSNARKs have four main components [1]:

- Encoding the problem as a polynomial

  The problems are compiled into polynomials, e.g. $t(x)h(x) = w(x)v(x)$, the Quadratic Span Programs (QSP) as in [Gen+13], where the equality holds *iff* the program is computed correctly. The prover convinces the verifier that the above equality holds. The formal definition in [Bit+12] is:

  > ... membership of an instance y in an NP language L can be verified in time that is bounded by $p(k, |y|, \log t)$, where t is the time to evaluate the NP verification relation for L on input y, p **is a fixed polynomial independent of** L, and k is a security parameter that determines the soundness error.

  In general, zkSNARKs can be contructed from four categories [Nit20]:

[1]Christian Reitwiessner’s blog: zkSNARKs in a nutshell

– PCP

The oldest SNARK construction methodology is explained:

> . . . is based on PCP characterization of NP, and it is first achieved in the random oracle model (ROM), which gave only heuristical security. The idea is to apply the random oracle-based Fiat-Shamir transform to Kilian's succinct PCP-based proof system [Kil92], achieving logarithmic proof size and verification time. Later, the construction is improved by removing the use of the random oracles and replacing them with extractable collision-resistant hash functions (ECRH).
> The work of [CL08] proposed the PCP + Merkle Tree (MT)+ Private Information Retrieval (PIR) approach to "squash" Kilian's four-message protocol into a two-message protocol.
> In both cases, we do not obtain knowledge soundness, but only plain adaptive soundness.

– QAP

The most popular and widely implemented SNARK construction shares a central starting point on quadratic programs introduced by [Gen+13]. The proof/verification framework build SNARKs for programs encoded as boolean or arithmetic circuits.

> *Arithmetic Circuits*: Informally, an arithmetic circuit consists of wires that carry values from a field F and connect to addition and multiplication gates.
> *Boolean Circuits*: A boolean circuit consists of logical gates and of a set of wires between the gates. The wires carry values over $\{0, 1\}$.

This approach has led to fast progress towards practical verifiable computations. The first implementation is Pinocchio [Par+13].

> A SNARK scheme for a circuit has to enable verification of proofs for (Arithmetic or Boolean) Circ-SAT problem, i.e., a prover, given a circuit ($\mathcal{C}$) has to convince the verifier that it knows an assignment of its inputs that makes the output true.

Circ-SAT problem is the NP-complete decision problem of determining whether a given circuit has an assignment of its inputs

that makes the output true. A very important line of works focuses on building SNARKs for circuit satisfiability and have as a central starting point the framework based on quadratic span programs (QSP) [Gen+13]. QSP consists of multiple polynomials $\nu_0, \ldots, \nu_i; \omega_0, \ldots, \omega_i$ over a Galois field F and a target polynomial t. The QSP is accepted *iff* t divides $\nu_a \times \omega_b$ where nu_a and ω_b is constructed from the witness w and the original polynomials $\nu_0, \ldots, \nu_i; \omega_0, \ldots, \omega_i$.
Further more, QSP can be improved by QAP. [Nit20]

> Parno eta. [Par+13] defined QAP,a similar notion for arithmetic circuits,namely Quadratic Arithmetic Programs (QAP). More recently, an improved version for boolean circuits, the Square Span Programs (SSP) was presented which consequentially has led to a simplified version for arithmetic circuits, Square Arithmetic Programs (SAP), proposed in [GM17].

– Linear Interactive Proof (LIP)
According to [Nit20],

> The QAP approach was generalized under the concept of Linear Interactive Proof (LIP), a form of interactive ZK proofs where security holds under the assumption that the prover is restricted to compute only linear combinations of its inputs.
> These proofs can then be turned into (designated-verifier) SNARKs by using an extractable linear-only encryption scheme.

– Polynomial Interactive Oracle Proof (PIOP)
According to [Nit20]:

> As we saw previously, SNARKs are commonly build in a modular way. Recent works propose schemes that follow a new framework, enabling more possibilities such as: **transparent constructions (without a trusted setup), recursion, aggregation properties, post-quantum security**, etc.
> Modular instantiations of recent SNARKs employs polynomial *interactive oracle proofs (IOP)* for the information theoretic step, and *polynomial commitments* for the

cryptographic compilation. One advantage of using polynomial commitments is that **the setup is only needed for the commitment scheme and is thus independent of the statement being proven.** This allows these SNARKs to be **universal** as opposed to the circuit-based SNARKs presented in the previous section. Another advantage is that the choice of polynomial commitment scheme facilitates finding the right tradeoff between performance and security.
To build such SNARKs, one commonly follows 3 steps:

- NP Characterization or Arithmetisation for Circuits:
  Start with a way to describe the computation to be proven as a system of constraints involving native operations over a finite field. Then, this system of constraints can be changed into a (low-degree) **univariate polynomial expression**.
- Polynomial IOP or Algebraic Holographic Proofs:
  The proof consists in finding a way to show that such a polynomial equation holds. In a Polynomial Interactive Oracle Proof (PIOP), the prover sends low degree polynomials to the verifier, and rather than reading the entire list of coefficients, the verifier queries evaluations of these polynomials in a random point to an oracle interface.
- Cryptographic Compiler.
  Using polynomial commitment schemes and Fiat-Shamir heuristic, the previous checks can be compiled into an efficient and non-interactive proof system. The resulting zk-SNARK has either universal updatable structured reference string (SRS), or transparent setup, depending only on the nature of the polynomial commitment scheme used in this step.
  Remark that the only step where we employ cryptographic techniques is the final one. In this cryptographic compilation step the oracles from before are replaced by a suitable cryptographic realization: Polynomial Commitments (PC) in Sonic, Marlin and Plonk ([Mal+19], [Mal+19], [GWC19]). [2] Polyno-

[2]The computation of polynomial commitment can be optimized by table lookup.

mial commitments allow to check efficiently a series of identities on low-degree polynomials resulting from the arithmetization step. This comes at the expense of introducing computational hardness assumptions for security and sometimes a trusted setup.

A typical modern zkSNARK process can be illustrated as follows:

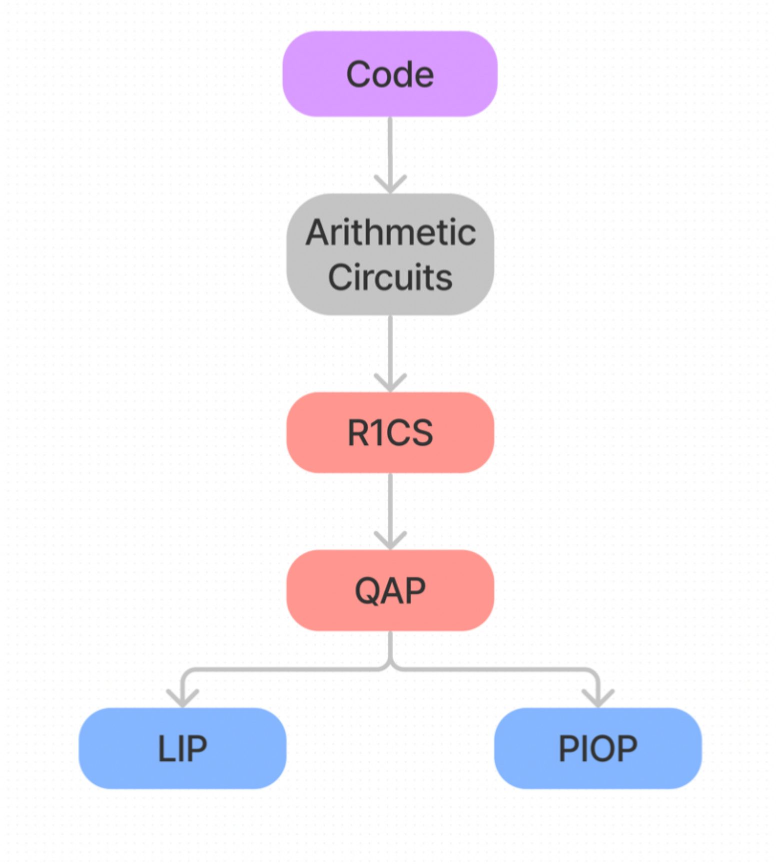

Figure 1: Source: unknown, but appears to be frequently credited to cryptographer Eran Tromer.

Polynomial commitments e.g. KZG, are a commitment scheme introduced by [KZG10] which are extremely powerful constructs that allow a prover to commit a polynomial and show evaluations of that polynomial on any point with a single group element-sized

[GW20]

proof in constant time. As such, Kate commitments and its variants lie at the heart of many PIOP-style zkSNARK constructions (such as Marlin and PLONK).

- Succictness by random sampling

  The verifier randomly samples a secret input s for evaluation with the polynomials, thus reducing the problem from polynomials operations to simple evaluation on numbers, e.g. $t(s)h(s) = w(s)v(s)$

- Homomorphic encoding/encryption

  An encoding/encryption function e is used that has some homomorphic properties. Combining the research on pairing-based double homomorphic encryption scheme [BGN05] and NIZK leads to the solution on uncondictional zero-knowledge:

ZK proof for NP-complete languages	Computational zero knowledge	Unconditional zero knowledge
Interactive	Goldreich-Wigderson-Micali 1986	Brassard-Crépeau 1986
Non-interactive	Blum-Feldman-Micali 1988	Groth-Ostrovsky-Sahai 2006

  - Core idea ([GOS06]) to prove circuit satisfiability with zk [3]:
    * Commit to wire values in circuit as $g^a h^r, g^b h^s, g^c h^t, \dots$
    * Verifier can easily add committed elements: $g^a h^r \cdot g^b h^s = g^{a+b} h^{r+s}$
    * Prover can demonstrate multiplicative relation: $e(g^a h^r, g^b h^s) = e(g^c h^t, g) e(\pi, h)$

- Obfuscation

  The prover obfuscates the values $e(g^a h^r, g^b h^s), e(g^c h^t, g), e(\pi, h)$ by multiplying with a non-zero secret point k from the contextual elliptic curve (EC) on both sides, thus the verifier can still verify the correctness on hidden values.

1.3 Overview of research scope

[3]Jens Groth's lecture Bilinear Pairings-based Zero-Knowledge Proofs

1.4 Theortical development

1.4.1 Protocol overview

- Pinocchio [4]:

- Groth16:

  Currently the fastest and smallest known zk-SNARK. It's used in Zcash, amongst others. Groth16 is non-universal; the setup is always tied to one specific circuit. Because of the speed and small proof size, performance of new zk-SNARKS is often compared to Groth16.

  - Major features:
  - Design ideas
  - Main contributions

- Sonic:

  Sonic is an early general purpose zk-SNARK protocol. The paper was published in January 2019, 10 months before this blog post was written, which is an eternity in zk-SNARK time. Sonic supports a universal and updatable common reference string. Sonic proofs are constant size, but verification is expensive. In theory, multiple proofs can be verified in batches to achieve better performance. Many of the new zk-SNARKs listed below are based on Sonic.

- Supersonic

  We present a generic compilation of any PIOP using our DARK polynomial commitment scheme. In particular, compiling the PIOP from PLONK (GWC, ePrint'19), an improvement on Sonic (MBKM, CCS'19), yields a public-coin interactive argument with quasi-linear preprocessing, quasi-linear (online) prover time, logarithmic communication, and logarithmic (online) verification time in the circuit size. Applying the Fiat-Shamir transform results in a SNARK, which we call Supersonic.

- Marlin:

  Marlin is an improvement on Sonic with 10x better prover time and 4x better verification times.

[4]See Source

- Plonk:

  Plonk is an improvement on Sonic with a 5x better prover time.

- Halo & Halo2

  Halo is a zk-SNARK that supports recursive proof composition without a trusted setup. Recursion works using "nested amortization": repeatedly collapsing multiple proofs together over cycles of elliptic curves.

  Unlike the other new constructs, Halo's verification time is linear, making it the only new construct that isn't succinct.

- Spartan
- Nova
- Quarks
- Woverine
- Poseiden

1.5 Application development

1.6 Summary

1.6.1 Comparative analysis

Protocol	Pros	Cons
Pinocchio		
Groth16		
Bulletproof		
Plonk		
Halo & Halo2		

1.6.2 Outlook

1. Recursive ZK Proofs

   Recursive zero-knowledge SNARKs are one of the most interesting use cases for zero-knowledge technology that are almost ready for practical use. As the name suggests, recursive ZK SNARKs unlock the ability to take a proof and verify it inside another proof, allowing for a lot more composability without blowing up proving/verifying times. Besides the real-world composability advantages, recursive zero-knowledge SNARKs are also of great interest because of their use case as a blockchain scaling solution, relying on the succinct- ness feature of SNARKs to compress secure verification of long blockchains. Mina Protocol ([Bonneau et al. 2020]) is currently implementing such an approach.

It should, however, come as no surprise that efficient recursive ZK SNARK constructions are quite hard to pull off. With the typical method of SNARK constructions that uses elliptic curve fields, the initial, hairy problem to solve is to find a pairing-friendly curve (see [Bitansky et al. 2013a]). Attempting to solve for this yields many interesting tradeoffs. Many of the options and possibilities have been explored in recent works such as Halo [Bowe et al. 2019], which uses elliptic curve cycles that do not require pairings and Fractal [Chiesa et al. 2020b], that eliminates the use of elliptic curves altogether. Overall, the study of efficient recursive ZK proofs is of great importance, and ripe territory for further exploration.

2. Post-Quantum Zero-Knowledge

   Many of the techniques described in previous sections depend on the computational hardness of certain problems, such as computing the discrete logarithm, which have been shown solvable by quantum computers in polynomial time by [Shor 1997]. Therefore, quantum computers may pose a significant threat to the security model and practicality of zk-SNARKs that use these techniques. There has been some work on making quantum attack resistant zk-SNARKs. For instance, [Gennaro et al. 2018] recently proposed a lattice-based zk-SNARK starting with SSP — square span programs (an alternative to the QAP intermediate for boolean circuits). Lattice problems are known to hold against quantum attacks [Ajtai 1996], so this is a very promising line of work.

3. Optimizations

   - Plonk:
     - SHPLONK: multiple commits
     - TurboPLONK: Custom gates (e.g. EC point addition gates, greatly reducing pairing's computation cost)
     - PLOOKUP: SNARK-unfriendly functions
     - Zexe PLONK: PLONK single-layer rollup with BLS12-377 + BW6-761

- Blockchain transaction fee:

  FFLONK [GW21], a modification of the popular PLONK zk-SNARK scheme, was motivated by ethereum gas fee reduction. Modification to KZG commitments that allows for reducing the scalar multiplications necessary to verify the proof by nearly three times, albeit at the cost of almost tripling proof construction time.

4. Fiat-Shamir-Compatible Hash Functions

   Currently Fiat-Shamir heuristic's security is only proven in the Random Oracle Model. It remains an open question whether there exist concrete hash functions that are compatible with the Fiat Shamir heuristic, i.e. if there exist hash functions that guarantee soundness for a transformed proof (and are also compatible with zero-knowledge proofs). There has been a lot of work on this topic, but there is still no known universal hash function that can be proven to be compatible without making strong, somewhat impractical, assumptions yet.

References

[GMR85] S Goldwasser, S Micali, and C Rackoff. "The Knowledge Complexity of Interactive Proof-Systems". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, pp. 291–304. ISBN: 0897911512. DOI: 10.1145/22145.22178. URL: https://doi.org/10.1145/22145.22178.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic Checking of Proofs: A New Characterization of NP". In: *J. ACM* 45.1 (Jan. 1998), pp. 70–122. ISSN: 0004-5411. DOI: 10.1145/273865.273901. URL: https://doi.org/10.1145/273865.273901.

[BG02] Boaz Barak and Oded Goldreich. "Universal Arguments and Their Applications". In: *Proceedings of the 17th IEEE Annual Conference on Computational Complexity*. CCC '02. USA: IEEE Computer Society, 2002, p. 194.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF Formulas on Ciphertexts". In: *Proceedings of the Second International Conference on Theory of Cryptography.* TCC'05. Cambridge, MA: Springer-Verlag, 2005, pp. 325–341. ISBN: 3540245731. DOI: `10.1007/978-3-540-30576-7_18`. URL: `https://doi.org/10.1007/978-3-540-30576-7_18`.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *Advances in Cryptology - EUROCRYPT 2006.* Ed. by Serge Vaudenay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 339–358. ISBN: 978-3-540-34547-3.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *Advances in Cryptology - ASIACRYPT 2010.* Ed. by Masayuki Abe. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 177–194. ISBN: 978-3-642-17373-8.

[Bit+12] Nir Bitansky et al. "From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference.* ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 326–349. ISBN: 9781450311151. DOI: `10.1145/2090236.2090263`. URL: `https://doi.org/10.1145/2090236.2090263`.

[Gen+13] Rosario Gennaro et al. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: *Advances in Cryptology – EUROCRYPT 2013.* Ed. by Thomas Johansson and Phong Q. Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 626–645. ISBN: 978-3-642-38348-9.

[Par+13] Bryan Parno et al. "Pinocchio: Nearly Practical Verifiable Computation". In: *2013 IEEE Symposium on Security and Privacy.* 2013, pp. 238–252. DOI: `10.1109/SP.2013.47`.

[GM17] Jens Groth and Mary Maller. "Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs". In: *Advances in Cryptology – CRYPTO 2017.* Ed. by Jonathan Katz and Hovav Shacham. Cham: Springer

International Publishing, 2017, pp. 581–612. ISBN: 978-3-319-63715-0.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana-Madalina Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 953.

[Mal+19] Mary Maller et al. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings". In: Nov. 2019, pp. 2111–2128. ISBN: 978-1-4503-6747-9. DOI: `10.1145/3319535.3339817`.

[GW20] Ariel Gabizon and Zachary J. Williamson. "plookup: A simplified polynomial protocol for lookup tables". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 315.

[Nit20] Anca Nitulescu. "zk-SNARKs: A Gentle Introduction". In: 2020.

[GW21] Ariel Gabizon and Zachary J. Williamson. *fflonk: a Fast-Fourier inspired verifier efficient version of PlonK*. Cryptology ePrint Archive, Paper 2021/1167. `https://eprint.iacr.org/2021/1167`. 2021. URL: `https://eprint.iacr.org/2021/1167`.