Seminar Innovative Internet Technologies: Zero Knowledge Proofs

Sebastian Hohl, Filip Rezabek*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: sebastian.hohl@in.tum.de, frezabek@net.in.tum.de

***Abstract*—In this paper the general idea and properties of zero knowledge proofs are discussed. The modern zero knowledge proofs zk-STARKs, zk-SNARKs (Ligero and Sonic) and Bulletproofs are compared regarding their need for a trusted setup, proof length, proving time and verification time. Zero knowledge proofs are (non-)interactive proofs that yield no information to the verifier. They are widely useable as they exist for every problem in NP. Zero knowledge proofs are used for many applications like signatures, anonymous decentralized payments on blockchains or verifying computations.**

***Index Terms*—zero knowledge proof**

1. Introduction

How much information have to be used to perform a certain action and how to minimize the released information? Any information given away might be used for attacks and other malicious activities, so information minimization is a fundamental security principle [1]. Especially if the aim is to convince someone of something, minimizing the released information seems to be hard. Zero knowledge proofs provide a solution for this problem. The idea of zero knowledge proofs (ZKPs) shown in [2] is that *they guarantee that a polynomial time verifier gains essentially no information from a proof.*
The verifier is only convinced that the given statement is valid. The prover does not release its secret information. So zero knowledge is a property a proof has. This is a way to show that a proof does not reveal too much information. [2, 3]
As ZKPs are well suited for the use on blockchains, they have got more attention with the recent rise of blockchain technology [4].
In this paper a part of the general theory about ZKPs is introduced in Section 2. Then modern ZKP systems are considered in Section 3 and in Section 4 applications of ZKPs are discussed. In the last Section 5 related works are recommended.

2. What are Zero Knowledge Proofs?

This section introduces to the basics of ZKPs. They have an extensive theory, so more advanced topics cannot be discussed in this brief paper. After reading this section the reader will know the meaning of every part of the term *(non-)interactive zero knowledge argument of knowledge*, therefore this section prepares the reader for the used terms of Section 3. At first in Subsection 2.1 the parts of a ZKP system are discussed. Then the essential properties of every ZKP are introduced in Subsection 2.2. Furthermore, differences between interactive and non-interactive ZKPs and how non-interactivity is achieved are considered in Subsection 2.3. After that the meaning of the suffix "of knowledge" is explained in Subsection 2.4. Finally, in Subsection 2.5 a possible use of the functionality of ZKPs is discussed from a blackbox perspective.

2.1. Framework

The statement to be proven is a decision problem. It consists of a language L[1] and for any given input x one has to decide whether $x \in L$. A ZKP system consists of such a decision problem and two programs[2]: One is the *verifier* and has typically limited resources like a polynomial runtime limit for its calculations. The other one is the *prover* that either has more computational power or usually a witness for the problem. The used language and the input x are known to both verifier and prover, so the secret information is the witness allowing to prove the membership of x efficiently. [2, 3, 6, 7]
Polynomial runtime limits in this Section 2 are taking the size of the common input $|x|$ as argument of the polynomial limiting the runtime.
For interactive ZKPs the verifier and prover communicate in alternating rounds or in the case of non-interactive ZKPs (NZKPs) the prover creates one proof that can be verified without further interaction (see Subsection 2.3). The purpose of the prover is to convince the verifier that $x \in L$ is correct without revealing more information. The task of the verifier is to check if $x \in L$ is correct with sufficiently high probability. Both prover and verifier can use randomness, so a recording of the view of the verifier of a proof is a random variable. [2, 3, 6, 7] This framework is illustrated in Figure 1.

2.2. Properties of a ZKP

A ZKP system fulfills the following three properties:
Completeness:

1. A language is a set of words over an alphabet.
2. Anything like algorithms or Turing machines or equally computationally powerful is considered according to the Church-Turing Hypothesis [5].

doi: 10.2313/NET-2022-07-1_08

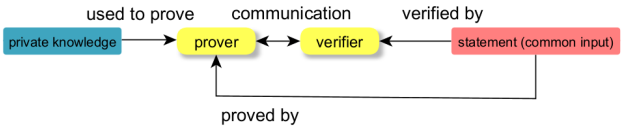

Figure 1: The prover has to convince the verifier that the statement is valid.

Completeness means that for $x \in L$ the prover having a witness or enough computational power can/will convince the verifier of this correct statements. This convincing can either be successful with sufficiently high probability or in case of *perfect completeness* with certainty. This assumes that the verifier fulfills its part of the protocol[3]. [2, 3, 6]

Soundness:

If the statement is not correct (if $x \notin L$) the verifier should only be convinced by *any* program[4] with a sufficiently low probability. This means that wrong statements are not likely accepted. Also note that this is not defined as chance of zero to convince the prover of a wrong statement. This property is often weakened to *computational soundness* like in Section 3. In this case the soundness property only has to hold against programs that have polynomial runtime. Therefore any program with more runtime could convince the verifier of at least one false statement with a higher probability. Often ZKPs with computational soundness are called zero knowledge *arguments* instead of ZKPs [3, section 2.1], like the ZKPs mentioned later in Section 3. But the term ZKPs in this work includes zero knowledge arguments aswell. [2, 3, 6]

Zero Knowledge:

For the zero knowledge property soundness and correctness are not regarded. The prover is the same as in the ZKP system, but the verifier can be any potentially adversarial program that might try to get additional information. The idea of [2] was that the *no information*[5] gain of any verifier can be defined by the notion of indistinguishability between the distribution[6] of the recorded view[7] of this verifier and another distribution that is made by a simulator running in expected polynomial time. This simulator only assumes that $x \in L$ and does not use the prover at all, so that the conclusion is that this verifier cannot gain anything new from the proof it could not compute in (expected) polynomial time on its own. [2, 3]

There are many variants and slightly changed definitions for this, but two common are:

- Perfect zero knowledge: Both distributions are identical and thus indistinguishable.
- Computational zero knowledge: Both distributions cannot be distinguished by any algorithm with polynomially bounded computation time except with a negligible small probability difference.

Note that perfect zero knowledge implies computational zero knowledge. Computational zero knowledge is the most general and therefore often used as a synonym for zero knowledge. [2, 3, 8, 9]

One disadvantage of this definition of *no information* gain is that following these definitions every interactive proof for a problem in polynomial time (P) is a ZKP, as the verifier/simulator could solve it in polynomial time on its own. So zero knowledge makes only sense for problems that do need more than (expected) polynomial time, therefore the name *zero knowledge proof* may be a bit misleading.

There are many variants [3] and slight variations of these definitions, here only the most basic can be mentioned.

Another more restrictive variant is *honest verifier zero knowledge* [3, Section 3], which means that there only has to exist such a simulator for the one verifier of the ZKP system (e.g. any dishonest/cheating verifier may get information).

A more restrictive definition is *auxiliary input zero knowledge*, where the simulator and the verifier both get access to a string of already known knowledge. This means that no previous knowledge can be used by a verifier to gain more information from the interaction than it could compute with this previous knowledge by itself. Auxiliary input zero knowledge implies that the composition of ZKPs stays a ZKP. [10]

A less restrictive variant is that of *witness indistinguishability*, where the used witness from a set of possible ones cannot be determined. This can be combined with witness hiding (e.g. no new witness can be computed from the proof). [11]

2.3. Non-Interactive ZKP

Interactivity is sometimes too difficult or too costly to achieve. This is especially important for blockchains (or other similar structures) as very relevant use case of non-interactive ZKPs (NZKPs), where the proofs must be publicly verifiable. This means that the proof must be a one-way message created by the prover, that is then saved on the blockchain. This proof must be verifiable by any participant of the blockchain without any interaction with the prover, only using the blockchain data. [6] [4].

Only very limited languages like those in BPP[8] have NZKPs [10]. Therefore the used model has to be changed. As in many ZKPs the verifier only sends randomly selected challenges to the verifier, this random selection could be done by a trustworthy public source of randomness instead. So the prover and verifier would both know the resulting random challenges, but there would be no need for the verifier to send these questions to the prover. The verifier would still receive all answers to these challenges in the one-way proof of the prover. [7, 12]

An initial approach is to have a shared random bit string per statement [13] that allows the creation of a ZKP that can be verified non-interactively. As common random strings do not simply exist, there are the following two approaches. For each approach common data is generated in an interactive setup at the start of the ZKP system. After this setup the system can be used for non-interactive

3. A "troll" verifier could always reject.
4. This means not only the prover of the system, but every program that can take its place.
5. except that the statement is true
6. It is a distribution as both programs might use randomness.
7. This includes everything it can read from and the communication.
8. In BPP are problems that can be solved with high probability in probabilistic polynomial time.

doi: 10.2313/NET-2022-07-1_08

proofs, even by non-participants of the setup.
One option is the *Common Reference String* model, where from a distribution a common string can be generated [14]. But this requires a *trusted setup* of the distribution. The trusted setup of a NZKP system consists of several participants[9] that each generate their own secret and use these in an interaction that is used to create the distribution. In the best case these individual secrets are destroyed immediately. But if anyone gets all those secrets (for example if all participants collude) the soundness and the zero knowledge property are no longer guaranteed [7, 14]–[16].
Another option to get NZKPs is to use the *Fiat-Shamir* heuristic from [12], which replaces the random decisions of the verifier by a hash function over the parameters of the proof. As this requires no trust in the confidentiality of a setup this is called *transparent* setup instead [6]. This function is used instead to make the choices that the verifier would do randomly. The hashfunction is not selected by the prover, but it is part of the common shared data. [12, 17, 18]
Although there is some controversy about it, this heuristic is secure for honest verifier ZKPs according to [18] in the *Random Oracle Model*.

2.4. Proof of Knowledge

Many ZKPs[10] are ZKPs *of knowledge*. Proof of knowledge [19] intuitively means that the prover proves its knowledge about a witness usable to prove the given problem in polynomial time. Therefore some (explicit or extractable) knowledge is guaranteed to be possessed by the prover. This can be formalized by defining another probabilistic polynomial algorithm extracting the witness using the prover as an oracle machine. [19]

2.5. Example: Blackbox Workflow of a ZKP

The properties mentioned before are about the requirements a ZKP should fullfill, not about how one would use a ZKP in a concrete way. As even simple examples of ZKPs are too long for this paper, this is now discussed from a blackbox perspective and assumes that eventual setups for NZKPs are already done. In Figure 2 this high-level view is depicted for non-interactive ZKPs and to its parts are now referred to with (1) to (8). At first (1) the statement over the common input is transformed into the input type usable by the zero knowledge proof (2), in most cases this is some low level representation like boolean or arithmetic circuits. These are circuits over finite fields. For the circuits the satisfiability problem[11] is to be proven. There are programs to transform from higher level languages like TinyRAM (a small subset of the programming language C) into this input format [4]. The resulting number of gates $|C|$ of the circuit C is one variable of the proof length and costs as shown in Section 3. This problem representation is then used by the prover

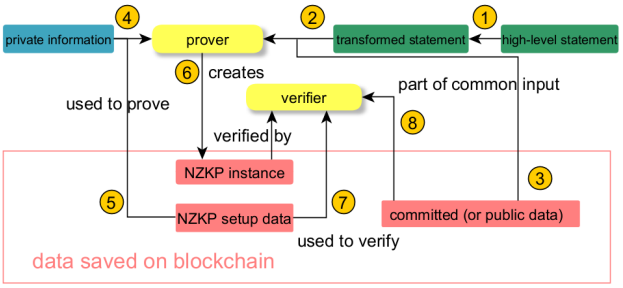

Figure 2: A high level view of the usage of non-interactive ZKPs.

(6) to either do an interactive proof or to generate the non-interactive proof over the common input (3) and its private information (4) using the proving key of the setup data (5). The verifier checks this (non-)interactively using (the common verification key (7) of the setup data and) the input data of the proof (8). [4, 6, 7].
Note that for statements like that a transaction on a blockchain [12] [20] is correct, data (either public or committed[13]) of that blockchain is part of the common input of the proof, so the proof refers to that specific blockchain. The user can formulate the statements on a higher level language and let the ZKP system do the complex mathematical part. Even without deep mathematical understanding of ZKPs usage of a provided implementations is possible. For such implementations see [4].

3. Examples of Modern ZKPs

In this section ZKPs are compared regarding their capabilities. This is a limited and simplified selection due to the complexity and extent of the topic. For this the popular term *zk-SNARKs* is introduced in Subsection 3.1. Then newer ZKPs called *zk-STARKs* are covered in Subsection 3.2. The popular ZKP *Bulletproof* and *Ligero* as zk-SNARK without and *Sonic* as a zk-Snark with trusted setup are discussed in the remaining Subsections.

3.1. zk-SNARKs

Zero Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) is a common term for NZKPs. After Section 2 only the word *succinct* has to be explained, which means that the proof size and verification time are less than linear in the input size or even constant. Some zk-SNARKs use trusted setups like Sonic [22], while more recently zk-SNARKs without trusted setup (transparent setup or also interactively useable) like Ligero [23] have been created. [4, 6, 24]

3.2. zk-STARKs

Zero Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs) use cryptographic hashfunctions

9. This group does not have to consist of everyone that ever uses the system. It can be a reasonably sized subset balancing the trustworthiness of the setup and setup costs.
10. including the ones presented in Section 3
11. Showing that there exists inputs to the circuit so that the specified output (e.g. 1 for boolean circuits) is achieved.
12. or a similar system like central signature authority [12], as long as provers and verifiers trust the integrity of the data provided by it
13. Committing data means to use a one-way hashfunction so that the given argument cannot be changed without altering the hash, but the hash also contains no useable information about the used argument [21].

doi: 10.2313/NET-2022-07-1_08

for their security assumptions. Quantum computers cannot find collisions for these hashfunctions efficiently [25]. For this reason zk-STARKs are believed to be more resistant to quantum computers than other ZKPs. For non-interactive proofs they use the Fiat-Shamir heuristic. *Scalability* means quasilinear proving time ($n \cdot polylog(n)$) and polylogarithmic verification time. In [26, 27] zk-STARKs are defined and some are constructed. Their new feature is the combination of scalability on both the provers and verifiers side combined with their transparency. Thus their prover timer outperforms zk-SNARKs (or Bulletproof) [26, Section 1.3]. But the proof sizes of STARKs are significantly larger for small input sizes (like validating blockchain transactions), which make them less suitable for this common use case on blockchains. [24, 26, 27]

3.3. Bulletproof

Bulletproof [28] is a ZKP that is used to prove that a committed secret value $v \in Z_p$ is in a given range $[0, 2^n - 1]$. Bulletproof relies on the *Discrete Logarithm Problem* as security assumption. Bulletproof has the feature to accumulate m range proofs into one bigger range proof with a smaller total size. The size of the proof[14] consists of $2(log_2(n) + log_2(m) + 4)$ elements of G and always 5 elements of Z_p. Also a multi party protocol was proposed [28, section 4.5] that can be used for merging multiple proofs of multiple parties while each party keeps their secret with linear communication in the number of proofs m and a logarithmic number of rounds in n. Therefore the total size of the accumulated proof grows logarithmic per additionally range proof over $[0, 2^n - 1]$. More generally Bulletproof can be used to prove that a given computation over an arithmetic circuit C is correct. The computation times needed for prover and verifier are each linear in n respectively $|C|$ [4, table 3]. Also the non-interactivity is achieved using the Fiat-Shamir heuristic, so Bulletproof requires no trusted setup. [28]

3.4. Ligero

Ligero [23] is a zk-SNARK made non-interactive with a transparent setup using the Fiat-Shamir transform. Like zk-STARKs its security assumption is based on cryptographic hash functions. It is used to verify for a given arithmetic circuit C, which checks the membership of a language L in NP, whether an input x is in L. The sublinear proof size is in $\Theta(\sqrt{|C|})$, therefore Ligero is called a *succinct* non-interactive argument of knowledge (zk-SNARK). Both the running times of the verifier and prover are in $O(|C| \cdot log(C))$ [4, table 3]. This protocol can also be used with a multi-party protocol merging multiple instances for a better amortized run time for the verification. [23]

14. The proof size is the length of the communication between prover and verifier in the interactive case, otherwise the length of the one-way message.

3.5. Sonic

Sonic is a zk-SNARK with trusted setup. The proof size is constant in regard to the input. The proving time is in $|C| \log |C|$ as well as the costs of the universal updatable trusted setup. Updatable means that the trusted setup can continue indefinitely, i.e. new participants can be later added to increase the trust in the setup. The universal setup has not to be repeated for different circuits, instead all circuits with circuit sizes up to a at the setup fixed size are allowed. The verification time is linear to the size N of the inputs of the gates and also logarithmic in $|C|$ [4, Table 3]. The security bases on the *Algebraic Group Model*. [22]

3.6. Comparison

In Table 1 are the asymptotic runtimes and proof sizes of the mentioned ZKPs depicted. Be aware that for small input sizes zk-STARKs create proofs of significantly larger sizes than Bulletproof or zk-SNARKs (including Sonic and Ligero). Because zk-SNARKs were introduced several years earlier than zk-STARKs, they have more available implementations to use. As shown these algorithms have different strengths and should be chosen depending on the use-case. Note that this is only a limited selection of the many ZKPs, more are listed in [4].

4. Use Cases

It was shown in [29] and [30] that every problem in NP[15] has computational (non-)interactive ZKPs. Also the ZKPs in 3 all support at least all problems in NP as input. Of this wide applicability of ZKPs some potential use cases are discussed next.
Generally, there are a lot of applications of ZKPs on blockchains like anonymous payments, voting, age verification, risk assessment, or auctions [7, sec. Zero-Knowledge Proof Applications], smart contracts, verifying computations or delegated computing [4].

4.1. Signatures

Like shown in [12] one can create non-interactive signatures having a zero knowledge property, issued by a central authority that is not required for authentication.
There are interactive zero knowledge undeniable signatures like in [31]. They cannot be verified without the signer, making it much harder (or not possible) for anyone but the private key owner to convince a third party that a message is signed correctly. Besides proving the correctness of a signed message, they can also be used to prove (also a ZKP) that a message is not correctly signed to protect the signer against false accusations. [31]

15. Decision problems that can be solved in exponential time and checked with a witness in polynomial time

Name	Trusted Setup	Proof Size	Proving Time	Verification Time
Bulletproof	No	$O(\log M)$	$O(M)$	$O(M)$
Ligero	No	$O(\sqrt{\|C\|})$	$\|C\| \log \|C\|$	$\|C\| \log \|C\|$
STARK	No	$O((\log \|C\|)^2)$	$O(\|C\| \ (\log \|C\|)^2)$	$O(\|C\|)$
Sonic	Yes	$O(1)$	$\|C\| \log \|C\|$	$N + \log \|C\|$

TABLE 1: Table part from [4, Table 3], $|C|$ is the number of gates of the computation expressed as arithmetic circuit, M the number of *And* gates in it, N the length of the inputs and outputs of the computation [4, Table 3].

4.2. Private Transactions on Blockchains

Digital currencies like Bitcoin are pseudo-anonymous. The transactions between all addresses are publicly visible to verify the correctness of the transactions. If an identity is linked to an address, the transaction history belonging to that address is retraceable. Using a new address for each new transaction or coin mixers makes this harder, but does not implicitly guarantee anonymity. [32]
Digital currencies for decentralized and trustless payments like the protocol Zerocash described in [20], solve this problem with the use of zk-SNARKS. Such currencies thereby offer a possibility for more anonymous (hidden amount, origin and destination) payments. The payment is found by scanning over the block chain using the corresponding private key searching for a commitment to the related address. [20]
Furthermore, for such currencies a trusted setup might be a disadvantage. As this requires trust in the correctness of key ceremonies with a big number of participants like that of Zcash that are only insecure if all participants are dishonest [16]. So instead NZKPs with a transparent setup might solve this issue. For example the previously mentioned Bulletproof which requires no trusted setup is used by the crypto currency Monero to prove that the sum of committed inputs is greater than the committed outputs of a transaction. [28, 33].

4.3. Verifying Computations

Even for checking computations on a von Neumann RISC architecture like vnTinyRAM in [34] verifying the computations via generating arithmetic circuits is possible for moderate code lengths. In this scenario both client and server know a function F and a given input x, the server knows or computes a secret w so that $z = F(x, w)$. The zk-SNARK used enables verifying the correct computation while the server keeps its secret and also the verification needs only very limited ressources. The cited work tests up to 32.000 machine cycles and 10.000 instructions on a desktop computer, achieving universality of the computations as only the time bound of the program execution has to be known ahead of the proof. But the work also shows that ZKPs for universal computations are still expensive for bigger and more complex programs. [34]

5. Related Works

A survey paper about ZPKs and the more recent development is [35]. An extensive document about the concepts of ZKPs is [6]. For a basic overview of some of the theory of interactive ZKPs see [3]. A basic example of a ZKP can be found in [29, Protocol 4]. [36] gives an good listing of some the theory of NIZKPs and the research history. [37] is a short overview of the use of NIZKPs in Blockchains. The later mentioned paper about Bulletproof [28] is also possibility to get a better understanding of a modern NIZKPs. For an overview of some of the many use cases and implementations of non interactive zero knowledge proofs for blockchains see [4].

6. Conclusion

As shown ZKPs have a complex theory and wide applicability. The aim of these algorithms is to minimize released information that is not to be proven, but is needed to prove something. The basic zero knowledge, soundness and completeness properties should definitely be fulfilled by all ZKPs. Especially the zero knowledge property shows a way to formalize that no usefull information except the validity of the statement to be proven is released. ZKPs are an interesting topic and wide research field with much theory and an important part of cryptography. The use of this technology on blockchains is a promising ongoing development as well as the improving NZKPs with transparent setup removing the trust issues of trusted setups. Also its many use cases like signatures, crypto currency transactions or even verifying computations are very versatile.

References

[1] C. Jackson, S. Russell, and S. Sons, *Security from First Principles*. Sebastopol, CA, USA: O'Reilly Media, Inc. [Online]. Available: https://www.oreilly.com/library/view/security-from-first/9781491996911/ch04.html

[2] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, Feb. 1989. [Online]. Available: https://doi.org/10.1137/0218012

[3] O. Goldreich, "Zero-knowledge twenty years after its invention," 01 2003.

[4] J. Partala, T. Nguyen, and S. Pirttikangas, "Non-interactive zero-knowledge for blockchain: A survey," *IEEE Access*, vol. PP, pp. 1–1, 12 2020.

[5] B. J. Copeland, "The Church-Turing Thesis," Jan 1997, [Online; accessed 5. Dec. 2021]. [Online]. Available: https://plato.stanford.edu/entries/church-turing

[6] ZKProof, "Zkproof community reference," December 2019.

[7] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[8] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway, "Everything provable is provable in zero-knowledge," in *Advances in Cryptology — CRYPTO' 88*, S. Goldwasser, Ed. New York, NY: Springer New York, 1990, pp. 37–56.

[9] J. Thaler, "Proofs, arguments, and zero-knowledge," August 2021, [Online; accessed 5. Dec. 2021]. [Online]. Available: https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf

[10] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, Dec. 1994. [Online]. Available: https://doi.org/10.1007/bf00195207

[11] U. Feige and A. Shamir, "Witness indistinguishable and witness hiding protocols," 1990.

[12] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*. Springer Berlin Heidelberg, 1987, pp. 186–194. [Online]. Available: https://doi.org/10.1007/3-540-47721-7_12

[13] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC '88*. ACM Press, 1988. [Online]. Available: https://doi.org/10.1145/62212.62222

[14] R. Canetti and M. Fischlin, "Universally composable commitments," in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 19–40.

[15] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers, "Updatable and universal common reference strings with applications to zk-snarks," in *Advances in Cryptology – CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds. Cham: Springer International Publishing, 2018, pp. 698–728.

[16] "Parameter Generation - Zcash," Aug 2019, [Online; accessed 13. Dec. 2021]. [Online]. Available: https://z.cash/technology/paramgen

[17] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, R. D. Rothblum, and D. Wichs, "Fiat-shamir: from practice to theory," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, Jun. 2019. [Online]. Available: https://doi.org/10.1145/3313276.3316380

[18] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in Cryptology — EUROCRYPT '96*, U. Maurer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 387–398.

[19] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Advances in Cryptology — CRYPTO' 92*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 390–420.

[20] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[21] A. Jain, S. Krenn, K. Piertrzak, and A. Tentes, "Commitments and effcient zero-knowledge proofs from learning parity with noise," 2021.

[22] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings," Cryptology ePrint Archive, Report 2019/099, 2019, https://ia.cr/2019/099.

[23] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2017. [Online]. Available: https://doi.org/10.1145/3133956.3134104

[24] "Zero-Knowledge Proofs: STARKs vs SNARKs | ConsenSys," Dec 2021, [Online; accessed 11. Dec. 2021]. [Online]. Available: https://consensys.net/blog/blockchain-explained/zero-knowledge-proofs-starks-vs-snarks

[25] "Defeating Quantum Algorithms with Hash Functions," Feb 2017, [Online; accessed 25. Feb. 2022]. [Online]. Available: https://research.kudelskisecurity.com/2017/02/01/defeating-quantum-algorithms-with-hash-functions

[26] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology ePrint Archive, Report 2018/046, 2018.

[27] ——, "Scalable zero knowledge with no trusted setup," in *Advances in Cryptology – CRYPTO 2019*. Springer International Publishing, 2019, pp. 701–732. [Online]. Available: https://doi.org/10.1007/978-3-030-26954-8_23

[28] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," Cryptology ePrint Archive, Report 2017/1066, 2017, https://ia.cr/2017/1066.

[29] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," *Journal of the ACM*, vol. 38, no. 3, pp. 690–728, Jul. 1991. [Online]. Available: https://doi.org/10.1145/116825.116852

[30] M. Blum, A. de Santis, S. Micali, and G. Persiano, "Noninteractive zero-knowledge," pp. 1084 – 1118, 1991.

[31] D. Chaum, "Zero-knowledge undeniable signatures (extended abstract)," in *Advances in Cryptology — EUROCRYPT '90*, I. B. Damgård, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 458–464.

[32] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 1318–1326.

[33] "Moneropedia: Bulletproofs," Dec 2021, [Online; accessed 13. Dec. 2021]. [Online]. Available: https://web.getmonero.org/resources/moneropedia/bulletproofs.html

[34] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct Non-Interactive zero knowledge for a von neumann architecture," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 781–796. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson

[35] S. Kassaras and L. A. Maglaras, "Zkps: Does this make the cut? recent advances and success of zero-knowledge security protocols," *CoRR*, vol. abs/2006.09990, 2020. [Online]. Available: https://arxiv.org/abs/2006.09990

[36] H. Wu and F. Wang, "A survey of noninteractive zero knowledge proof system and its applications," *TheScientificWorldJournal*, vol. 2014, p. 560484, 05 2014.

[37] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.