An Exploration of Zero-Knowledge Proofs and zk-SNARKs

Terrence Jo

EAS499

Fall 2019

Advisor: Professor David Crosbie

Table of Contents

Abstract

Over the past few years, with the rise of cryptocurrency prices and new blockchain startups, blockchain technology has become popular and a variety of use cases have become apparent – in banking, identification, cryptocurrency, supply chains, payments and more. With new developments in cryptography and privacy, related cryptographic technology is becoming more advanced and furthering its applications. One such recent development is zero knowledge proofs, and specifically zk-SNARKs (Zero Knowledge Succinct Non-interactive Argument of Knowledge), a new form of zero-knowledge cryptography. In this paper, first I will give an overview of zero-knowledge proofs, then investigate the emerging new technology of zk-SNARKs, why and how it works, and explore recent implementations and their efficiencies. Then, I will also explore the practicality and usability of zk-SNARKs in a business context, by investigating companies and startups that claim to use zero-knowledge proofs in their products. Finally, I will discuss current challenges in the development of zero-knowledge proofs and the future outlook.

1. Introduction and Overview of Zero-Knowledge Proofs

Zero-knowledge proofs are a very interesting and fascinating cryptographic concept and prove to be useful in various applications, mainly in privacy and blockchain technology. Zero-knowledge proofs involve two parties: a prover and a verifier. The prover makes an assertion that his or her proof is valid, which the verifier must approve, without the prover leaking any "knowledge" other than the assertion itself. How can the verifier validate an assertion without any knowledge of the content and minimal interaction with the prover? This is where zero-knowledge proofs become useful and can be used in many cryptographic protocols to "provide zero-knowledge proofs of correctness of... secret-based actions, without revealing these secrets". [1]

We can imagine use cases almost immediately – imagine frequent scenarios when one reveals private information to verify something about one's identity. When one applies for a credit card, he or she must give up his social security number, or when one checks in for a flight at the airport, he or she must reveal all the private information present on the passport, such as birthdate or passport number. Even on a day-to-day basis, when a user logs on to a web or mobile application, the user must enter a password to the account and send this sensitive information to the server, trust that the application securely handles this information, and that the password is not intercepted over the network. Dr. Manuel Blum from the University of California, Berkeley suggested a zero-knowledge solution to this problem – the "password" could be an interactive "procedure", such as changing letters in a string, rather than a string itself, through which the user proves knowledge of the "password" without leaking the password itself. [2] An outside observer would no longer have a string password to intercept and fraudulently log on to another user's account. [2]

So how do zero-knowledge proofs work and how are they structured? As mentioned previously, there is a "prover" and "verifier". The "prover" wants to prove the knowledge of a solution to a specific problem (or truth of a statement), without leaking any knowledge of the solution itself to the "verifier", who wants to be reassured that the prover actually does know the solution (or that the statement is true). According to Goldwasser, Micali, and Rackoff, who first introduced the "zero-knowledge" concept[3], there are three main essential properties of zero-knowledge protocols:

1. *Completeness*: If statement is true, the verifier would be convinced of the truth of statement by the prover.

---

[1] Goldreich, Oded. "A Short Tutorial of Zero-Knowledge." *Weizmann Institute of Science*, 2010.

[2] Gleick, James. "A NEW APPROACH TO PROTECTING SECRETS IS DISCOVERED." *The New York Times*, The New York Times, 17 Feb. 1987, https://www.nytimes.com/1987/02/17/science/a-new-approach-to-protecting-secrets-is-discovered.html.

[3] Green, Matthew. "Zero Knowledge Proofs: An Illustrated Primer." *A Few Thoughts on Cryptographic Engineering*, 27 Nov. 2014, https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/.

2. *Soundness*: The prover can only convince the verifier of the truth of the statement if that statement itself is actually true, except with some very small probability.
3. *Zero-Knowledge*: The verifier does not know any knowledge (zero knowledge) of the prover's statement or solution that, other than the truth of the statement. [4]

One of the first such valid general zero-knowledge proof systems was proposed by Goldreich, Micali, and Widgerson, specifically to verify that a prover knew the 3-coloring of a graph. The 3-coloring graph problem is an NP-Complete problem stated as follows:

"Given a graph G, can you color the nodes with ≤3 colors such that for every edge {u, v} we have f(u) =/= f(v)?" [5]

In this zero-knowledge proof system, the prover wants to prove to the verifier that he or she knows the 3-coloring of a given graph to a verifier, without revealing to the verifier the actual 3-coloring solution. Thus, the zero-knowledge proof system works in the following way:

1. The prover covers each vertex of a 3-coloring solution of the graph such that it is not observable to the verifier.
2. The verifier randomly chooses an edge of the graph and the prover reveals the two vertices of the chosen edge. The prover shows that the two vertices are of a different color. If the two vertices are of the same color, we know that the prover is dishonest and does not have the solution. If the two vertices are of different color, the verifier has some (but not full) confidence that the prover is telling the truth. We note that the prover has *(E-1)/E* probability of cheating, where *E* is the number of edges in the graph. Then we continue to the next step.
3. The prover now covers all vertices of the graph again, and randomly switches the ordering of the three colors in the prover's solution. Again, the verifier then chooses a random edge to check that the edge is valid (the two vertices are of different colors). Although the prover could be cheating again, we see that now the probability that the prover successfully cheated both rounds is $((E-1)/E) * ((E-1)/E) = ((E-1)/E)\text{^}2$, which is lower than the previous round.
4. After repeating this process for multiple *n* rounds, we can lower the probability that the prover can cheat the verifier, to a negligible value. [6]
    a. Probability of Prover Cheating: $\left(\frac{E-1}{E}\right)^n$.

One of the most important aspects of this protocol proposed by Goldreich, Micali, and Widgerson is the zero-knowledge aspect – we have to show that the verifier cannot identify the actual solution to the 3-coloring solution. This is where the randomness of the coloring at each round comes in. If at each round, the order of the coloring of the vertices of the graph is different, the verifier cannot link the edges revealed between subsequent rounds to construct a

[4] Goldwasser et al. "Interactive Proof Systems." Computational Complexity Theory Proceedings of Symposia in Applied Mathematics, 1989, pp. 108–128.
[5] https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/sat.pdf
[6] Green, Matthew. "Zero Knowledge Proofs: An Illustrated Primer."

valid solution to the 3-coloring graph problem. So, we can see that this zero-knowledge protocol to proving knowledge of a solution to the 3-coloring graph problem is complete, sound, and has zero-knowledge. Additionally, since the 3-coloring graph problem is NP-Complete, we know that any problem in "the class NP can be reduced into an instance of that problem". [7] Thus, in essence, Goldreich, Micali, and Widgerson have shown that all statements in NP can be verified through this zero-knowledge protocol.[8]

We might wonder in practice how to convert every zero-knowledge proof into a 3-coloring graph problem, and also run the interactive aspect of the zero-knowledge protocol for enough rounds efficiently to be useful in practice and in every-day applications. This leads us to the development of zk-SNARKs (Zero Knowledge Succinct Non-interactive Argument of Knowledge), which becomes more efficient and more applicable in practice.

2. zk-SNARKs Introduction

The first zero-knowledge proofs described were introduced in the late 1980's, by Goldwasser, Micali, and Rackoff[4], but the modern development of zk-SNARKs happened only in the past decade (introduced by Alessandro Chiesa, professor at UC Berkeley) and is constantly being improved. Zk-SNARKs (Zero Knowledge Succinct Non-interactive Argument of Knowledge) are specifically zero-knowledge proofs that are "succinct", meaning they can be verified in the matter of milliseconds with a proof length of a few hundred bytes. The "non-interactive" aspect refers to fact that the prover can send a single message to the verifier, without having many back-and-forth interactions.[9]

As mentioned previously, Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer introduced the term zk-SNARKs in their paper "From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again" in 2011. They introduced the idea of an extractable collision hash function (ECRH)[10] and proved that the existence of ECRH implies that a modified version of "Di Crecsenzo and Lipmaa's protocol is a succinct non-interactive argument for NP."[11] This modified version of the protocol was introduced as SNARKs, and the zero-knowledge version of such protocol is zk-SNARKs, which we will deep further into the details of in the next section.

---

[7] Green, Matthew. "Zero Knowledge Proofs: An Illustrated Primer."

[8] Goldreich, Oded, et al. "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, Jan. 1991, pp. 690–728.

[9] "What Are Zk-SNARKs?" Zcash, Electric Coin Company, https://z.cash/technology/zksnarks/.

[10] Bitansky, Nir, et al. "From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again." *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS 12, 2012, doi:10.1145/2090236.2090263.*

[11] Crescenzo, Giovanni Di, and Helger Lipmaa. "Succinct NP Proofs from an Extractability Assumption." *Logic and Theory of Algorithms Lecture Notes in Computer Science*, 2008, pp. 175–185., doi:10.1007/978-3-540-69407-6_21.

3. zk-SNARKs Technical Overview

The initial condition begins with a prover that wants to prove to a verifier that he or she knows an input w such that $z = f(x, w)$, where x is a publicly known input to the function f. The verifier wants to be assured that the z provided by the prover is correct, and the verifier wants to be assured that the prover gains no knowledge about the private input w. This situation, with *completeness*, *soundness*, and *zero-knowledge*[4] is similar to the more general zero-knowledge proofs that we have discussed earlier in this paper and refers to the security aspects of zk-SNARKs. However, to show the true innovative benefit of zk-SNARKs, a few more requirements regarding the efficiency of this proving system must be met as follows:

1. *Non-interactive*: Unlike the back-and-forth interactions between the prover and verifier of the zero knowledge protocol we saw earlier with the 3-coloring graph example, in zk-SNARKs, the prover provides only z along with a string π, which proves to the verifier that z is the correct output of f (in other words, prover knows the secret input w). The efficiency comes from this one-way interaction, as z and π is sufficient to the verifier and does not need to ask any further questions to the prover.[12]
2. *Succinct*: Given a λ, which is a security parameter, the π provided by the prover must have size $O_\lambda(1)$, and the verification should be able to be finished in the following runtime:

$$O_\lambda(|f| + |x| + |z|) \qquad [12]$$

To successfully construct and implement such zk-SNARKs, there are four main ingredients as follows:[13]

1. Encoding into Polynomial Problem
2. Succinctness through Random Samples
3. Homomorphic Encryption
4. Zero-Knowledge

In the paper "Quadratic Span Programs and Succinct NIZKs without PCP's" by Gennaro, Gentry, Parno, and Raykova[14], the authors discovered that reducing original problems to QSP's (Quadratic Span Programs), were helpful in constructing zk-SNARKs. Simply put, a QSP consists of multiple polynomials $v_0 \dots v_i, w_0 \dots w_j$ over a field F and a target polynomial t. The QSP is accepted for an input and a witness if and only if t divides $v_a * w_b$, where v_a and w_b is constructed from the witness and the original polynomials $v_0 \dots v_i, w_0 \dots w_j$.[14] Thus the prover shows that $t * k = v_a * w_b$ for some other polynomial k. However, due to

[12] Ben-Sasson, et al. "Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture." Feb, 2019. https://eprint.iacr.org/2013/879.pdf

[13] Reitweissner, Christian. "zk-SNARKs in a Nutshell." 5 December, 2016. http://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf

[14] Gennaro, et al. "Quadratic Span Programs and Succinct NIZKs without PCPs." May, 2013. https://eprint.iacr.org/2012/215.pdf

the complexity of large polynomials and large runtime of multiplying and dividing polynomials, this QSP is hard to completely verify in practice, and thus we will go into the details of verifying a secret point s in the polynomials. In other words, the verifier chooses a secret point s such that

$$t(s) * k(s) = v_a(s) * w_b(s). \quad [13]$$

Although one may see that verifying a QSP at a single point, rather than for all points, reduces security, since there are relatively few zeroes to make the above equation satisfy, we can see that it is relatively safe in real application.

3a. CRS Setup Phase and Prover Computation

Currently, the most common constructions of zk-SNARKs involve a CRS (Common Reference String) and a set-up of initial parameters. Firstly, we choose a group and a generator g, and an encryption scheme E where $E(x) = g^x$. Then, the verifier secretly chooses s as well as another value z and publicly posts as part of the CRS the following:

$$E(s^0), E(s^1), \dots, E(s^d)$$
$$E(zs^0), E(zs^1), \dots, E(zs^d) \quad [13]$$

where d is the maximum degree of all polynomials in the program.[13] Once these values are calculated and posted, the verifier must discard the secret point s for security reasons, so that the prover cannot obtain it to falsely create proofs. The prover must then use these published values above to prove that he can compute a polynomial function f. We can see that any prover can compute $m = E(f(s))$ for any function f without knowing the verifier's secret value s. As an example, consider a function $f(x) = x^2 + x$. The prover computes the following:

$$E\big(f(s)\big) = E(s^2 + s)$$
$$= g^{s^2+s} = g^{s^2} * g^s$$
$$= E(s^2) * E(s). \quad [13]$$

Each of $E(s^2)$ and $E(s)$ can be taken from the publicly published CRS. By the same token, the prover can also compute $n = E(z * f(s))$, and sends both m and n to the verifier.

3b. Pairing Function and Verifier Computation

The reason that the verifier needs n in addition to m is that earlier the verifier had discarded s, so there is no way to check that the prover correctly solved the polynomial f at s. Thus a way around this is once the verifier receives the values m and n, the verifier must check that m and n match through an pairing function p, which is chosen with the group chosen in the CRS setup phase, such that the following holds for all inputs values x and y:

$$p(g^x, g^y) = p(g, g)^{xy} \quad [13]$$

where a pairing function p is a computable bijection such that p: $N\ X\ N \rightarrow N$.[15] We can see immediately that this pairing function p becomes useful, as we can plug in the pairs n, g and m, g^z into the pairing function, and if the results match, then we know that the prover solved the polynomial correctly at s, as shown by the following equations:

$$p(m, g^z) = p\left(g^{f(s)}, g^z\right) = p(g, g)^{z*f(s)}$$
$$p(n, g) = p\left(g^{z*f(s)}, g\right) = p(g, g)^{z*f(s)} \quad [13]$$

So, if the prover correctly solved the polynomial f at s, we can see that the following should hold:

$$p(m, g^z) = p(n, g) \quad [13]$$

We can see that the verifier can verify the prover's solution without actually having to replicate the prover's computation. Even with much more complex polynomials and programs than our given example of f, the verifier does not need to put in as much work as the prover, and only needs to compute the pairing function p for various inputs to finish the job as a verifier.

The example above also shows that the protocol is *non-interactive* – that is, the prover only needs to send a sequence of values to the verifier once and in one-direction. The verifier does not need to ask additional questions to the prover to verify the correctness of the prover's statement. However, we must also show that the protocol is *succinct*, as well as *zero-knowledge.*

Currently, the example as it is above is not completely zero-knowledge. It is important that the verifier does not know anything about the value of $f(s)$, which is sensitive information to the prover, or even better, the encrypted version of the value, $E(f(s))$.[13] The verifier can obtain some information about both with our current system. $E(f(s))$ is known for obvious reasons, as the prover sends this exact value to the verifier for verification. For $f(s)$, since the verifier knows the public value of $E(z) = g^z$,[13] the value of $f(s)$ can be backed out from the result of the pairing function p, through a process that we will call "$f(s)$ Leakage Procedure from Malicious Verifier":

$f(s)$ Leakage Procedure from Malicious Verifier:

1. In the verification process, the verifier computes:
    a. $p(n, g) = p(g, g)^{z*f(s)}$
2. The verifier can also compute the following value using the pairing function:

[15] Alvarez, Carmé. "Algorithmics and Theory of Computation." https://www.cs.upc.edu/~alvarez/calculabilitat/enumerabilitat.pdf

a. $p(g^z, g^z) = p(g, g)^z$

3. Then, the verifier can take the log of $p(n, g)$ to obtain the value of $f(s)$:
   a. $log_{p(g,g)^z}\big(p(g, g)^{z*f(s)}\big) = f(s)$.

Since we have shown that the verifier can retrieve the value of $f(s)$ which is sensitive to the prover, we must add zero-knowledge to the protocol.

3c. Zero-knowledge

To add zero-knowledge, we modify the example above with the prover choosing a random value φ to "shift" the value of $f(s)$ before encryption.[13] So instead of computing $E(f(s))$ and $E(z * f(s))$, the prover computes $m' = E(\varphi + f(s))$ and $n' = E(z * (\varphi + f(s)))$ and sends it to the verifier:

$$E\big(\varphi + f(s)\big) = g^{\varphi + f(s)} = g^{\varphi} * g^{f(s)} = E\big(f(s)\big) * E(\varphi)$$ [13]

We can see from above that the prover can still compute m' from the public parameters in the CRS, and by the same token, the prover can also compute n'. Once the verifier receives m' and n', the values are inputted into the pairing function p in a similar fashion to the example above:

$$p(m', g^z) = p\big(g^{\varphi + f(s)}, g^z\big) = p(g, g)^{z*(\varphi + f(s))}$$
$$p(n', g) = p\big(g^{z(\varphi + f(s))}, g\big) = p(g, g)^{z*(\varphi + f(s))}$$

From the equations above, we see that verification process still functions properly, and the verifier's computation is still limited to the pairing function. An added benefit of this modified protocol is the zero-knowledge. As mentioned previously, we want to protect knowledge of $E(f(s))$ and $E(s)$ from leaking to the verifier. It is clear that $E(f(s))$ is not leaked, as the prover no longer sends this value to the verifier for validation. For $f(s)$, even if we apply the same "$f(s)$ Leakage Procedure from Malicious Verifier" stated above, the only useful information that a malicious verifier can extract from the values m' and n' is $\varphi + f(s)$. Since φ is a random value only known to the prover[13], it is now apparent that the malicious verifier can no longer deduce the value of $f(s)$, and thus we have shown the new modified protocol has zero-knowledge.

3d. QSP Problem and Succinctness of zk-SNARKs

The above example described in detail was for a zk-SNARK protocol for a single polynomial f, but most systems require reduction to QSP's, which were described earlier as a problem involving multiple polynomials that is derived from the original program to be proved.[14] Our above example is just applied to the various polynomials $(v_0 \dots v_i, w_0 \dots w_j, t)$ that are presented in the reduced QSP problem.

Instead of simply publishing $E(s^0), E(s^1), \dots, E(s^d), E(zs^0), E(zs^1), \dots, E(zs^d)$ to the CRS, in the CRS set up phase, we publish the following:

$$E\big(t(s)\big), E\big(z * t(s)\big)$$
$$E\big(v_0(s)\big) \dots E\big(v_i(s)\big), E\big(z * v_0(s)\big) \dots E(z * v_i(s))$$
$$E\big(w_0(s)\big) \dots E\big(w_i(s)\big), E\big(z * w_0(s)\big) \dots E(z * w_i(s))$$ [13]

Yet there are some additional parameters that must be published in the CRS that we did not mention previously. A hole in our logic in the simple example in parts 3a.-3c. is that the prover can maliciously come up with values m and n such that the verifier's computation still checks out. In other words, the prover can theoretically find values that makes the statement $p(m, g^z) = p(n, g)$ still true, without actually computing $E\big(f(s)\big)$ or $E\big(z * f(s)\big)$[13], violating the *soundness* aspect of our zero-knowledge system. Thus, in the setup phase of the CRS, we must add additional parameters to show that the values that the prover sends to the verifier actually involves a computation of the satisfying assignment to the QSP and the initial polynomials.[13] Then, the following procedure of the zk-SNARK is similar to the example in 3a.-3c., with the prover sending more polynomials to the verifier, who uses a pairing function to match the inputs. Again, the prover uses a random value φ to "shift" the values that he sends to the verifier, maintaining the zero-knowledge property.[13]

Notice that *succinctness* of zk-SNARKs is with respect to the verification process only and does not refer to the prover side.[12] With more complex problems to prove in the zk-SNARK protocol, we end up with more complex polynomials in the Quadratic Span Program. However, we maintain succinctness in the verification process, as we do not need to do polynomial multiplication for verification – rather, we only need to "check polynomial identity for a single point"[13]. Although this leads to a slight reduction in security, this reduction is negligible as described previously, and thus we maintain *soundness*. Then, the only things affecting the efficiency of our verification process are the inputs to the original program, as well as the group size chosen in our CRS setup phase[13]. This maintenance of succinctness no matter the problem complexity, and the non-interactive nature is the largest added benefit of the recent development of zk-SNARKs, as opposed to the original inefficient, interactive zero-knowledge proofs introduced in the 1980's,[4] and is the reason zk-SNARKs can be used in real-life application.

Note, however, there is one downside of zk-SNARKs implemented currently, which is the CRS phase discussed in our above example. This CRS setup phase requires trust that the generated parameters that are published publicly are not compromised[16]. We can imagine a case in which a parameter such as z (mentioned in our example) is not properly discarded and can allow a malicious prover to generate fake proofs. This problem of trust in these common

[16] Binance Academy. "Zk-SNARKs and Zk-STARKs Explained." *Binance Academy*, Binance Academy, 18 Nov. 2019, https://www.binance.vision/blockchain/zk-snarks-and-zk-starks-explained.

parameters is solved in a newer development in zero-knowledge proofs called zk-STARKs[17], which will be examined in more detail later in this paper.

4. zk-SNARKs in Application

One of the most interesting engineering challenges of zk-SNARKs in application is the translation of the original computational problem to the QSP[14], or Quadratic Arithmetic Programs (QAP)[18] mentioned earlier in the paper. For zk-SNARKs to be applied to a problem that the prover wants to prove to a verifier, there are 3 main preparatory procedures:

1. "Flattening Procedure" or Translation into Gates
2. Conversion from Gates to R1CS
3. Conversion from R1CS to QAP form. [18]

Starting from the original program, in step number 1, we translate or "flatten" all complex statements into a series of very simple equations, or logic gates, of the form "x = y" or "x = y (operation) z", where an operation is "+, -, *, / and y and z can be variables, numbers or themselves sub-expressions."[18]

Then these very simple "logic gates" are then converted into an important form called the Rank-1 Constraint System (R1CS). An R1CS involves a sequence of three vectors a, b, and c, and the solution s to the R1CS, or the witness that the prover provides, is another vector such that the following equation is satisfied:

$$a \cdot s + b \cdot s - c \cdot s = 0$$ [18]

where '·' stands for the dot product between the vectors on either side. This transformation into the Rank-1 Constraint System is done for each of the logic gates in the previous step – for instance, if we had 5 logic gates in step 1, at the end of step 2, we would have a series of 5 vectors for each of a, b, and c.[18]

Then, we convert these series of vectors in R1CS form to polynomial form, using Lagrange Interpolation[18] to create polynomials such that evaluating each polynomial at coordinates $1 \ldots i$ where i is the number logic gates in step 1, creates the values of vectors that were created in the R1CS system in step 2.[18] Finally, this series of polynomials form the QAP problem, to which zk-SNARKs can be directly applied.

4a. Implementations

There are various implementations of the above transformations as well as the

[17] Ben-Sasson, et al. "Scalable, transparent, and post-quantum secure computational integrity." 6 Mar. 2018, https://eprint.iacr.org/2018/046.pdf

[18] Buterin, Vitalik. "Quadratic Arithmetic Programs: Zero to Hero." 11 Dec. 2016, https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649

proof-generating and verification process of zk-SNARKs. A few open-source projects, such as Snårkl and Pequin[19], attempt to take programs written in higher-level languages such as C and Haskell and compile them into zk-SNARKs, from pre-processing to verification.

Currently, one of the most popular implementations of zk-SNARKs is libsnark, a C++ library used in the backend of the most major widespread zk-SNARK applications, such as Z-Cash.[20] Libsnark is currently being developed by a collaboration amongst a group of academic institutions called SCIPR (Succinct Computational Integrity and Privacy Research) Lab[21], and led by some of the top researchers in the zero-knowledge space, such as Eli Ben-Sasson and Alessandro Chiesa.

One of the major challenges, however, is the usability and documentation of these libraries. Although there are some instructional guidelines on installation and usage, documentation is very limited, and many bugs and issues are not addressed. A large milestone for further exploration and more access for interested developers in the usage of zk-SNARKs in their applications would be more detailed documentation, ease of use, and easy integration. Most of these implementations are not ready for widespread use, not only because of the usability, but also because of technical challenges facing zk-SNARKs that will be discussed later in this paper. Thus, libraries such as Pequin of the Pepper Project, a joint effort by UT Austin and NYU, highlight the "potential applicability of the theory" rather than being optimized for use in "real systems".[22]

5. Applications with zk-SNARKs

There are some blockchain-related applications that use zk-SNARKs to provide privacy to transactions completed over the blockchain network. One of the major benefits when blockchain technology was initially introduced was the transparency and traceability of transactions. Typically, in widely used blockchain networks such as Bitcoin or Ethereum, a publicly distributed ledger allows one to trace the transactions and determine the balance of any given address. As stated on Bitcoin's website:

> *"All Bitcoin transactions are public, traceable, and permanently stored in the Bitcoin network. Bitcoin addresses are the only information used to define where bitcoins are allocated and where they are sent. These addresses are created privately by each user's wallets. However, once addresses are used, they become tainted by the history of all transactions they are involved with. Anyone can see the balance and all transactions of any address. Since users usually have to reveal their identity in order to receive services or goods, Bitcoin addresses cannot remain fully anonymous."* [23]

---

[19] "Zero-Knowledge Proofs: What are they, how do they work, and are they fast yet?" https://zkp.science/
[20] Samman, George. "The Trend Towards Blockchain Privacy: Zero Knowledge Proofs" 12 September, 2016. https://www.coindesk.com/trend-towards-blockchain-privacy-zero-knowledge-proofs
[21] http://www.scipr-lab.org/
[22] https://www.pepper-project.org/
[23] https://bitcoin.org/en/protect-your-privacy

Thus, when a user on a blockchain network such as Bitcoin wants to protect his or her privacy regarding transaction history and balance, many precautions must be taken, such as using multiple addresses or wallets to make and receive payments, and using tools to hide the user's computer IP address.[23] This lack of privacy and inconvenience in protecting one's identity on a blockchain network has led to the development of Zcash, a blockchain that uses zk-SNARKs to allow users to transact while protecting their privacy. We will now explore Zcash, focusing on the privacy aspects, assuming knowledge of the workings of a typical blockchain network such as Bitcoin.

5a. Zcash

Zcash is a blockchain-based digital cryptocurrency that has two types of addresses, "private" or "transparent" addresses.[24] Transactions done through transparent addresses are essentially the same as ones done on Bitcoin in protocol, but the more interesting zero-knowledge technology is implemented in "private" address transactions.[25] The zk-SNARKs in Zcash allow users to show that all conditions (such as no double-spending, or having enough balance to complete a transaction) are satisfied to form a valid transaction, without "revealing any crucial information about the addresses or values involved."[26] In the Bitcoin protocol, a user that wants to spend money must have the necessary Unspent Transaction Outputs (UTXO). In Zcash, however, transaction outputs are called *commitments*, and to spend a *commitment,* the spender must publish a *nullifier* using his spending key. Thus, every *commitment* has a corresponding *nullifier*, each of which are hashed and stored on Zcash nodes, with no way determining which *nullifier* corresponds to which *commitment*. [26]

More specifically, the input values, or note provided by the spender in the shielded, private payment to a recipient, is hashed with the recipient's address, a "rho", or unique identifier value corresponding to the specific note, as well as a random value, called a nonce in the following way:

$$Commitment = HASH(recipient\ address,\ amount,\ rho,\ r)$$ [26]

Then, when such recipient wants to spend a note, the recipient hashes his or her spending key along with the rho of a previous commitment to publish a nullifier, where such resulting hash cannot match an existing nullifier, to prevent double spending, or spending a note that was already spent[26]:

$$Nullifier = HASH(spending\ key,\ rho)$$ [26]

Even if the spender shows through the nullifier (with rho) that the note was not spent

[24] https://z.cash/technology/
[25] Bowe, et al. "Zcash Protocol Specification." 24, September 2019. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf
[26] https://z.cash/technology/zksnarks/

previously, he or she still has not shown that the note itself exists and is not fraudulently created by the spender. This is where zk-SNARKs come into the Zcash application.[27] To accomplish this without revealing the detailed addresses and spent amounts, the sender (or prover in the zk-SNARK) of the private, shielded transaction also provides a zero-knowledge proof, or a string π, as described above in the technical overview, using a proving key[26], that the following statements are true:

1. The commitments and nullifiers were correctly computed.
2. The sum of input values and the sum of output values of the transaction equal each other, and the corresponding notes exist.
3. The sender has authority to spend the input notes of the transaction (by verifying the spending keys). [26]

The miners of the Zcash network then use verifying keys to complete the verification process of the zk-SNARK proofs provided by the spenders, subsequently adding the verified transaction to the Zcash blockchain. Currently, the Zcash network uses Bellman, a Rust-language library to generate arithmetic circuits in zk-SNARKs, which is intended to provide higher security and more efficiency than its predecessor, libsnark.

5b. zk-SNARKs in Enterprise

In addition to blockchain networks that involve individual users that desire privacy for their identities as well as their transactions, zk-SNARKs are very applicable for enterprise blockchain networks. There are many industries, such as pharmaceutical, food, or healthcare, where traceability and transaction verification through distributed ledgers are desired either from the consumer perspective, or by regulation. For instance, in 2013, the Drug Supply Chain Security Act (DSCSA), Title II of the Drug Quality and Security Act, required the pharmaceutical industry to adopt a digital system to "track and trace prescription drugs in the United States"[28]:

> *"... (DSCSA), outlines steps to build an electronic, interoperable system to identify and trace certain prescription drugs as they are distributed in the United States. This will enhance FDA's ability to help protect consumers from exposure to drugs that may be counterfeit, stolen, contaminated, or otherwise harmful. The system will also improve detection and removal of potentially dangerous drugs from the drug supply chain to protect U.S. consumers."* [29]

It seems likely that a blockchain network would be useful for such an "electronic, interoperable" system, but a typical decentralized application built with smart contracts on a

[27] Gabizon, Ariel. "How Transactions Between Shielded Addresses Work." 1 October, 2018. https://electriccoin.co/blog/zcash-private-transactions/

[28] Chronicled. "Chronicled and The LinkLab Announce The MediLedger Project, a Revolutionary Blockchain-backed System to Safeguard the Pharmaceutical Industry." *PR Newswire.* 21 September, 2017.

[29] https://www.fda.gov/drugs/drug-supply-chain-integrity/drug-supply-chain-security-act-dscsa

blockchain network such as Ethereum faces severe challenges in an enterprise setting. Because such an industry-specific blockchain network would involve nodes of collaborators along a supply chain as well as competitors, there are privacy concerns. A typical blockchain allows for a decentralized system to trace transfers of pharmaceutical goods, but also reveals the amount of goods, pricing, and other company-specific sensitive information that a player might not want to reveal to competitors. Additionally, simple "obfuscation" of data through hashing and other methods[30] is not enough, as most use cases of enterprise blockchains, such as authenticity of a drug, require more than just knowing the pure existence of a transaction. Therefore, naturally zk-SNARKs becomes useful in this type of enterprise setting, specifically verifying complex proofs without giving away sensitive company information.[31]

5b(i) Chronicled/MediLedger

Chronicled is a venture backed startup founded in 2014 with $28 million in funding[32], and focuses on enterprise software solutions for providing smarter and more secure supply chains. One of the core ideas in Chronicled's solutions is privacy, specifically for players in an enterprise blockchain network.

One of their main solutions is called The MediLedger Project[33], which essentially solves the problem mentioned above in the pharmaceutical industry. In the pharmaceutical industry, a very secure system of tracing drugs throughout a supply chain is necessary because of the large number of transfers that products can undergo before being used by the end consumer. According to Maurizio Greco, the CTO of Chronicled, a "…large number of drugs are returned by pharmacies to distributors, which must verify the serial number of the drug before reselling it to another pharmacy or hospital."[34]

A previously suggested solution to this problem of making drugs is called the Proof of Existence (PoE), which involves hashing data along with a timestamp before committing it to a block on the blockchain, such that the hash cannot be "tampered" with in the future.[34] So in this case, we can imagine hashing a serial number associated with the manufacturing of a drug and committing to the relevant blockchain. Then an end consumer of the drug can again hash the serial number and check the blockchain for previous existence. However, this solution only proves the existence of a serial number, rather than revealing the actual origin of a drug later down the line – nothing prevents a malicious actor to hash a counterfeit serial number and commit it to the blockchain, and thus we need to find a way to "authorize" a serial number, or proving that a serial number is of authentic origin, without revealing disclosing the actual "creator" or the serial number.[34]

---

[30] Petkus, Maksym. "Game-changing Year for Private Blockchains." 3 January, 2018. https://blog.chronicled.com/game-changing-year-for-private-blockchains-5b91eec0a0e4

[31] Gavigan, Jack. "ZSL: zk-SNARKs for the Enterprise." 23 March, 2017. https://electriccoin.co/blog/zsl/

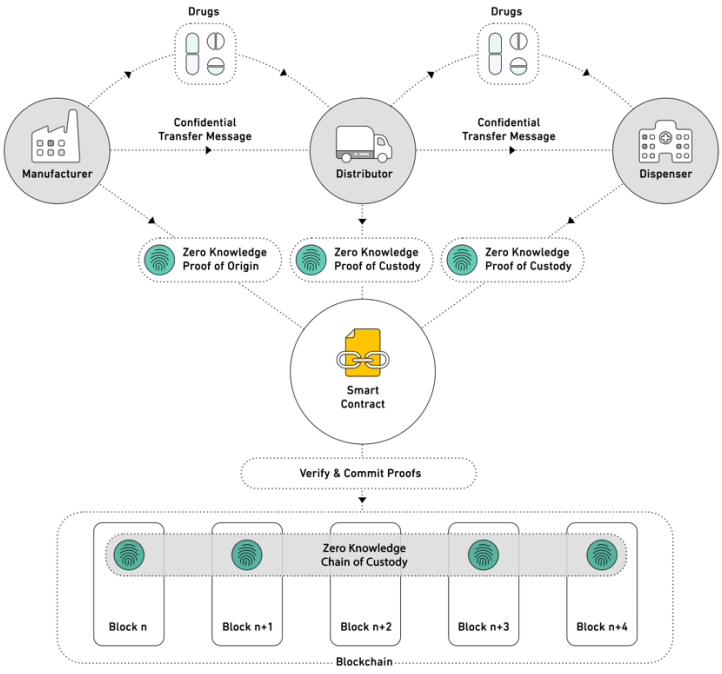
[32] https://www.crunchbase.com/organization/chronicled#section-overview

[33] https://www.mediledger.com/solution-protocols

[34] Greco, Maurizio. "Does Proof of Existence establish Provenance?" 10 April, 2018. https://blog.chronicled.com/does-proof-of-existence-establish-provenance-5028fbd8c6da

Thus, the MediLedger blockchain solution is called a "Confidential Chain of Custody" (3C), which is a combination of two systems: an immutable manufacturer registry, and another system called the "Chain of Custody."[34] The immutable registry on the blockchain lists a group of legitimate "creators", who expose verification endpoints for users to verify the origin of their drug products, and this verification process can be done in a decentralized matter. Additionally, the "Chain of Custody" involves transforming a serial number into a "token" and transferring such token to different users in the blockchain network along with the transfer of the physical good. At one time, there is only one "custodian", or controller, of the drug that can demonstrate ownership or transfer to another player in the network. Through the "Chain of Custody", a new counterfeit drug with the same serial number cannot be introduced into the network, unless the existing legitimate physical asset is replaced as well.[34]

At each transfer or transaction, unlike in the Proof of Existence solution, the details of data are not committed to the blockchain – instead, zero-knowledge proofs are pushed through an Ethereum smart contract that proves that the sender is the valid custodian of the drug being transferred, and that the transfer was valid without revealing the details of the transfer. Once the proof is verified through the zk-SNARK protocol, the recipient of the drug "completes the transfer by proving that he or she is the intended recipient"[34]



Source: Chronicled

As seen by the description of the MediLedger solution and diagram above, 3C provides an immutable and efficient supply chain solution for the pharmaceutical industry, while maintaining privacy of company sensitive information. In terms of efficiency, the current

MediLedger solution has response times of "400ms for Coast-to-Coast verifications"[33], so they have reached "production-level scalability."[30] We can see that this combination of blockchain and zk-SNARKs can be applied to supply chains of other industries as well, including automotive, agriculture, and fashion industries, to improve traceability and authenticity guarantees of products for the end consumer.

5c. Self-Sovereign Identity

Another application of zk-SNARKs is in the concept of self-sovereign identity – the idea of using zero knowledge proofs to prove claims about aspects of an individual's identity is a major use case for zk-SNARKs. Questions like "Are you over 21", "Do you live in the US", or "Are you employed" could be answered and verified through zk-SNARKs without giving the details of one's identity or sensitive information away to the party that is asking the question.[35]

5c(i). The Sovrin Network/Hyperledger Indy

The Sovrin Foundation is a nonprofit organization dedicated to supporting one of the most prominent efforts to achieve digital self-sovereign identity, the Sovrin Network.[36] In essence, the Sovrin Network is trying to create a secure, easy-to-use "digital equivalent" of a passport or birth certificate[37], to prove identity without having to manage many usernames and passwords that users trust companies to keep secure. There are two main problems that the Sovrin protocol is trying to tackle:

1. Standardizing the format for a digital credential.
2. Creating a "standard way to verify the source and integrity of these digital credentials"[37]

Until now, the Public Key Infrastructure (PKI) has been used to verify digital signatures and help determine the identity of services and users on the Internet. This public key infrastructure relies on a central authority called Certificate Authorities (CAs)[38], which issues digital credentials to users and is trusted to ensure that the digital certificates are associated with the legitimate and right identity. We can already see that there is a huge security vulnerability in trusting these CA's – if anything happens to CA's or if they make a mistake with providing digital certificates, the entire PKI system is in jeopardy and it may be difficult to trust that an online connection is secure and that data is sent to the correct counterparty.[37] Additionally, the

[35] "The Sovrin Network and Zero Konwledge Proofs." 3 October, 2018. https://sovrin.org/the-sovrin-network-and-zero-knowledge-proofs/

[36] "Self-Sovereign Identity Advocates Support the Sovrin Network." 25 February, 2019. https://www.globenewswire.com/news-release/2019/02/25/1741723/0/en/Self-Sovereign-Identity-Advocates-Support-the-Sovrin-Network.html

[37] The Sovrin Foundation. "Sovrin: A Protocol and Token for Self-Sovereign Identity and Decentralized Trust." Jan 2018. https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf

[38] https://www.thalesesecurity.com/faq/public-key-infrastructure-pki/what-public-key-infrastructure-pki

current infrastructure and the reliance of CA's makes it difficult and costly to obtain certificates.[37]

Therefore, the Sovrin protocol proposes a new solution, called a decentralized PKI (DPKI)[37], which involves the of storing these certificates, or "proof of ownership" of public keys, on a blockchain so that there is an immutable decentralized ledger and no reliance on a central authority. With partners such as the Linux Foundation's HyperLedger Indy Project as well as the World Wide Web Consortium (W3C)[39], the Sovrin Network is attempting use Digital Identifiers (DIDs) to create an identifier that can be controlled solely by the owner of that identity. Additionally, any connections between two parties would involve looking up the counterparty's DID and the associated public key, similar to how the Doman Name System (DNS) works currently.[39]

One of the highest priorities of the Sovrin Network is privacy, which is where the zero-knowledge proof comes in. There are three main privacy requirements that the Sovrin networks sets out to achieve:

> *"1. Pseudonymity by default. Sovrin supports pairwise-unique DIDs and public keys.*
> *2. Private agents by default. To prevent correlation, no private data is stored on the ledger, even in encrypted form.*
> *3. **Selective disclosure by default**. Sovrin verifiable claims use cryptographic **zero-knowledge proofs** so they can automatically support data minimization"*[37]

As seen above, the Sovrin Network and its proposed protocol strives to limit the amount of data leaked in any digital connection between two parties through the idea of "selective disclosure".[37] Most of the zero knowledge technology in the Sovrin network is built into HyperLedger Indy, which is the codebase on which Sovrin is built on top of.[40] In Indy's protocol, for a "prover" to prove that he or she owns a digital credential, or that the data in a set of claims is true, Indy's implementation uses zk-SNARKs to keep the prover's identity hidden in the verification process.[41]

6. Challenges of zk-SNARKs and Future Outlook

There are still many challenges to zk-SNARKs and zero-knowledge proofs that are preventing widespread application, and the two major ones are scalability, as well as security.

In our technical overview, we mentioned that one of the greatest improvements in zero-knowledge proofs provided by zk-SNARKs is the succinctness and efficiency with which verifiers can verify provers' claims. However, one of the issues with many implementations of zk-SNARKs is the costs to the prover in generating the actual proof that is

[39] "Decentralized Identifiers (DIDs) v1.0: Core Data Model and Syntaxes." w3.org/TR/did-core/
[40] https://github.com/hyperledger/indy-node
[41] https://hyperledger-indy.readthedocs.io/en/master/

to be sent to the verifier. As mentioned by developers of the Pepper Project, the "*CPU costs to the prover are currently immense: order of magnitude (factors between a thousand and a million) more than simply executing the computation*"[42]. Not only are the CPU costs to generate a proof very expensive, but the pure memory to store the transcript of a proof is also a bottleneck mentioned by the Pepper Project.[42] Due to these costs, zk-SNARKs can currently only be applied to small-scale computations, and overly complex programs are still impractical and difficult to prove through current implementations of zk-SNARKs like the Pequin of the Pepper Project. Even Maksym Petkus from Chronicled, mentioned earlier in this paper, describes the "notorious computations" of zk-SNARKs[30], and although they were able to achieve production-level scalability for the <10 enterprise clients they have on their network, it is unclear how viable it is to scale such a complex application with zero-knowledge proofs for millions of users.

The second major issue is the set-up process of current zk-SNARKs: naturally, the existence of parameters in the common string, or CRS, that are created from private randomness[16] decreases security of zero knowledge systems significantly. For instance, in Zcash, anyone who has access to the private parameters can generate fake proofs and hence fraudulently create cryptocurrency value for him or herself.[43] This vulnerability in the trusted setup has become a topic of discussion amongst researchers in zero knowledge proofs, and there is an emerging solution called zk-STARKs.

zk-STARKs stands for zero-knowledge, scalable and transparent argument of knowledge. As seen in the name, zk-STARKs provide scalability and transparency to zero-knowledge proofs. The "transparency" refers to the lack of the trusted set-up process that involves private parameters which could be compromised. Instead, the proofs in zk-STARKs only uses public parameters, and thus a malicious party would not have an unfair advantage or a way to generate fake proofs. Additionally, zk-STARKs provide scalability that current zero knowledge solutions do not have, because proofs provided in zk-STARK systems can be verified much faster than zk-SNARKs.[44] zk-STARKs provide "exponentially decreasing verification time", and a node in a blockchain network can produce proofs that can convince other nodes without requiring these nodes to "store the entire blockchain's state" or re-execute the computation.[45]

However, a downside of zk-STARKs given its quick verification time and a lack of need for a trusted "set-up phase" is its long proofs. With zk-SNARKs as long as it is currently, zk-STARK proofs are 1000x longer than zk-SNARK proofs[45], and a lot of research would have to be done to shorten this proof length to have viability on the blockchain. In

---

[42] https://www.pepper-project.org/summary-perf.htm

[43] Drygin, Alexander. "The Dark Side of Zero Knowledge: Undetectable Backdoor in zk-SNARK." 11 January, 2019. https://blog.smartdec.net/the-dark-side-of-zero-knowledge-undetectable-backdoor-in-zk-snark-a9093ffe49bf

[44] Whittle, Ben. "From zk-SNARKs to zk-STARKs: The Application of Zero-Knowledge Proofs." *CoinCentral.* 22 January, 2019. https://coincentral.com/zk-starks/

[45] Ben-Sasson, et al. "Scalable, transparent, and post-quantum secure computational integrity." 6 March, 2018. https://eprint.iacr.org/2018/046.pdf

2018, Ben-Sasson, one of the co-authors of the zk-STARK whitepaper, founded a venture-backed company called StarkWare Industries to continue improving and developing blockchain solutions involving zk-STARKs.[46]

7. Conclusion

Zero-knowledge proofs and zk-SNARKs are a fascinating cryptography concept that has been developing rapidly over the past couple of years. With various new applications on the blockchain and in enterprise as seen in ZCash, Sovrin, and Chronicled, we can see that this revolutionary technology has the potential to provide true privacy for users and companies that interact and transact digitally. Although currently there are various challenges including security, scalability, and efficiency, we will see many developments and research, such as zk-STARKs, in this space in the upcoming years, and we will eventually experience a new type of secure digital privacy through an exploding number of applications utilizing this zero-knowledge technology.

[46] https://starkware.co/