ZK-STARKs — Create Verifiable Trust, even against Quantum Computers

Adam Luciano Follow

Jun 25, 2018 · 10 min read

Note: unless you have familiarity with zero-knowledge proofs or ZK-SNARKs, I would suggest reading Part 1 and Part 2 of this blog series.

Now that we covered ZK-SNARKs, let's expand into some of the issues with ZK-SNARKs, and discuss recent innovations in zero-knowledge cryptography: ZK-STARKs (Zero-Knowledge Scalable Transparent ARguments of Knowledge).

ZK-SNARKs have a few underlying issues that will lead to reduced adoption for leveraging zero-knowledge cryptography in blockchains and other potential implementations as well:

1. The trusted setup phase can be compromised (there is an underlying assumption that when using a ZK-SNARK system, the trusted setup phase is secure)
2. Scalability of ZK-SNARKs can be improved, as run-time increases, the time needed to generate and especially verify proofs needs to be improved

3. ZK-SNARK cryptography is vulnerable to attacks from quantum computers

Let's breakdown each of the above issues and compare to the up and coming ZK-STARKs.

Trusted Setup Phase

What if a government or powerful entity tried to incentivize parties involved with the setup of the ZK-SNARK system to share the setup parameters? If that powerful entity was successful, they would be able to generate false proofs in a system widely known to be trusted. If for example, a Presidential vote for a country took place on a blockchain that was using ZK-SNARKs for proving votes, there could be significant incentives for the entity to discover the setup parameters and thus change the outcome of the election. The entity could tilt the election in their intended direction by creating false proofs for votes taking place.

The biggest problem with the ZK-SNARK approach is that users need to implicitly trust in the setup phase and the parties involved to setup the system honestly. Users of the system will never actually know if the setup phase was compromised at the point of setup, or at some point in the future. So, if this is the case, the door remains open for a system where users do not implicitly need to trust the parties involved in the system's setup to be honest. When a system is used where the incentives are high to circumvent the system, there will be those who will look to find a way to circumvent it.

In ZK-STARKs, there is no external trusted setup phase and the randomness used is public information. Public randomness utilization is exceptionally important for the public to trust zero-knowledge proof systems, otherwise a powerful entity could exert their influence to obtain the setup parameters and generate false proofs. Given there is no third party trusted setup phase and publicly verifiable randomness is used instead, ZK-STARK systems create verifiable trust.

Scalability

For those who pay attention to ongoing technical challenges in the blockchain space, the scalability discussion is center stage. Although outside the scope of this blog post, there are numerous ways to scale blockchains, all with their associated tradeoffs. For ZK (zero-

knowledge) proof systems, scalability of the system is paramount to achieving widespread and sustained adoption. ZK-STARKs exhibit much higher scalability than ZK-SNARKs. Let's breakdown the associated complexity of the ZK-STARK vs. the ZK-SNARK calculations into four different categories relative to scalability (results leveraged from ZK-STARK whitepaper):

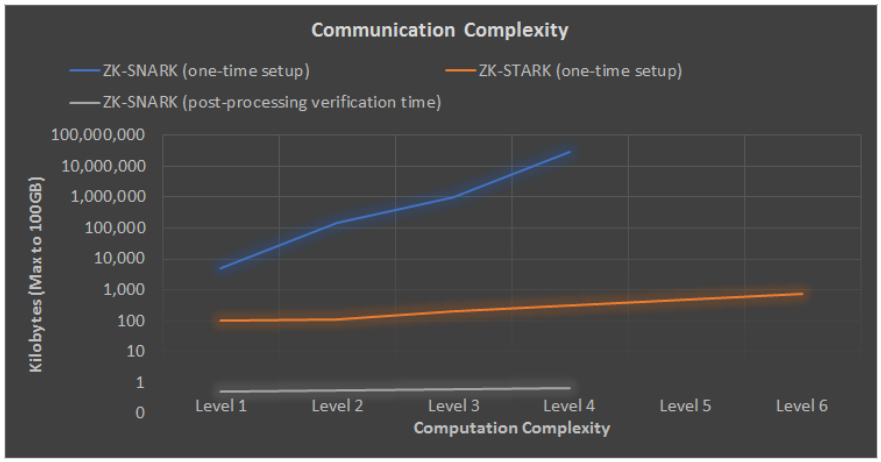
1. Arithmetic circuit complexity (an arithmetic circuit is a standard way to compute polynomials where addition and multiplication can be computed): In ZK-SNARK and ZK-STARK systems, the code to create ZK programs are written in such a way for them to be broken down into circuits, and then computed—effectively the simplicity of the circuits complexity is relative to its computational efficiency. In the below chart illustrations, the arithmetic complexity basically equates to the size and intricacy of the underlying computation used to generate the proof. (*Complexity charts below are based on overall complexity of system with primary focus on multiplication gates; for parameter setup, see page 11 of ZK-STARK whitepaper.)*

2. Communication complexity (typically defined as the amount of communication needed to solve a problem distributed among two of more parties): As the size of the computation grows, so does the communication complexity of the ZK-SNARK in a linear fashion, as opposed to the ZK-STARK which grows only slightly as computation size grows—a large advantage of ZK-STARKs vs. ZK-SNARKs when it comes to the one-time setup. After the setup phase, SNARKs currently have less communication complexity than STARKs when verifying proofs.

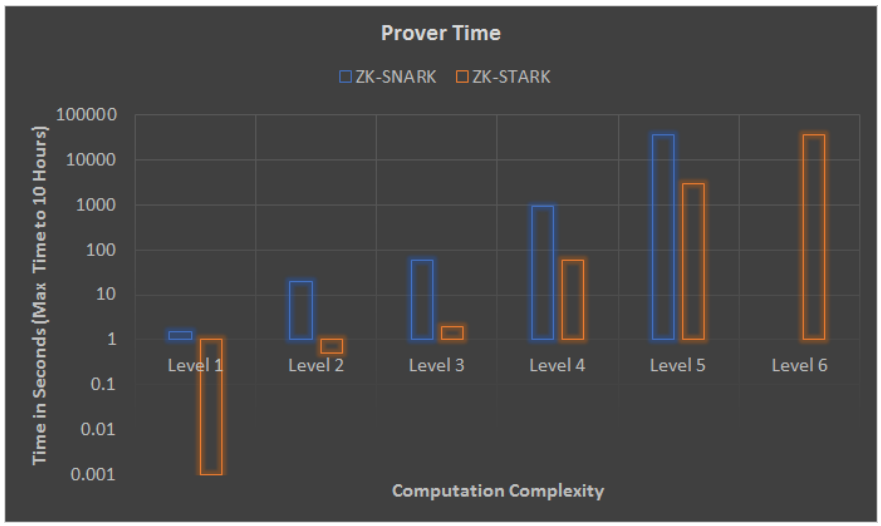
3. Prover complexity: ZK-STARKs are ~10x faster than ZK-SNARKs as computation size increases, another advantage of ZK-STARKs vs. ZK-SNARKs.

4. Verifier complexity: As computation size grows, ZK-STARKs grow only slightly vs. ZK-SNARKS, which tend to grow in a linear fashion, a significant advantage of ZK-STARKs vs. ZK-SNARKs when it comes to the one-time setup. After the setup phase, SNARKs currently need less time to verify proofs than STARKs, e.g. STARKs may take 50–100ms to verify, while SNARKs need ~10ms to verify.

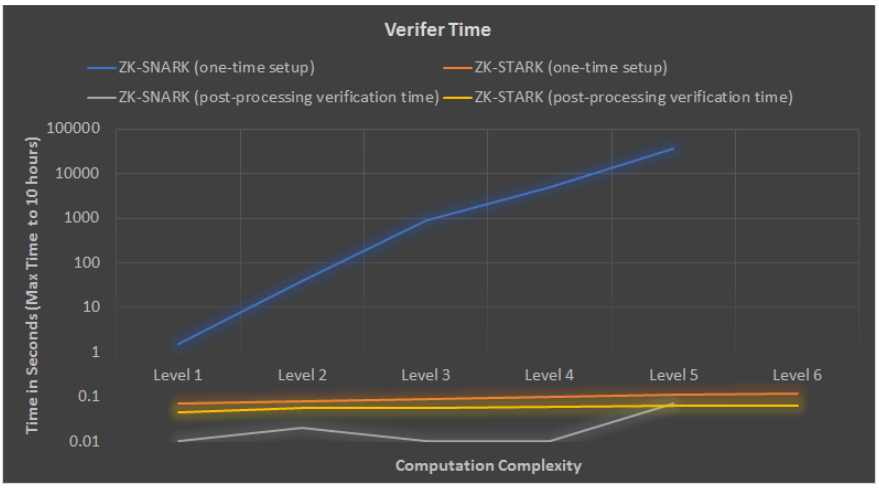
The below is a simpler view of a benchmarking analysis completed between ZK-STARKS vs. ZK-SNARKs in the ZK-STARK whitepaper.



The above simplified benchmarking illustrates that the ZK-STARK's required communication to complete the computation rises much slower than the ZK-SNARK as the underlying proof increases in complexity. Level 1—Level 6 refer to arithmetic circuit complexity (levels of multiplication gates, each level increase is ~55x of the previous level) Level 5 & 6 are not shown as the size of communication exceeds 100GB). ZK-SNARK post processing verification time refers to when a SNARK is verified.



The above simplified benchmarking illustrates that the ZK-STARK's time to generate a proof rises much slower than the ZK-SNARK as the underlying proof increases in complexity. Level 1—Level 6 refer to arithmetic circuit complexity (levels of multiplication gates, each level increase is ~55x of the previous level) Level 6 is not shown as the time to completed exceeds 10 hours).



The above simplified benchmarking illustrates that the ZK-STARK's time to verify a proof rises very slowly compared to the ZK-SNARK as the underlying proof increases in complexity. Level 1—Level 6 refer to arithmetic circuit complexity (levels of multiplication gates, each level increase is ~55x of the previous level) Level 6 is not shown as the time to completed exceeds 10 hours). ZK-STARK and SK-SNARK post processing verification time refers to when the STARK/SNARK is verified.

Charts Data Source: ZK-STARK white paper; Note charts are reproduced from whitepaper—they are simplified for comparison of a ZK-SNARK to a ZK-STARK.

In proof systems, there is a statement that the prover wants to assert as true to anyone who would like to verify the statement. For example:

Prover statement: Alice wants to prove she is the owner of a bank account with Acme Bank.

Effectively, when you slice up the above statement (statement broken up in code logic in post #2) above into pieces for the zero-knowledge circuit to calculate and then generate a proof, the verifier mathematically checks the proof for correctness along with the associated verifier key. This process (especially the verification process) has been substantially sped up by using a newer algorithm called the Fast Reed-Solomon Interactive Oracle Proof of Proximity. For more detail on the new algorithm utilized to increase the scalability of the ZK-STARK, I suggest reading Vitalik Buterin blog posts for an in-depth breakdown.

1. STARKs, Part I: Proofs with Polynomials
2. STARKs, Part II: Thank Goodness It's FRI-day
3. STARKs, Part 3: Into the Weeds

Quantum Computing

More recently, quantum computing has become a topic of interest, and to some degree in the blockchain world too (Qubit Protocol is an interesting blockchain start-up). IBM and Intel are both working on developing quantum computers however, current estimates point to quantum computing being years away from wide-spread adoption. Quantum computing does pose a risk to blockchain systems. Let's dig into why quantum computers pose a risk to some aspects of blockchain cryptography.

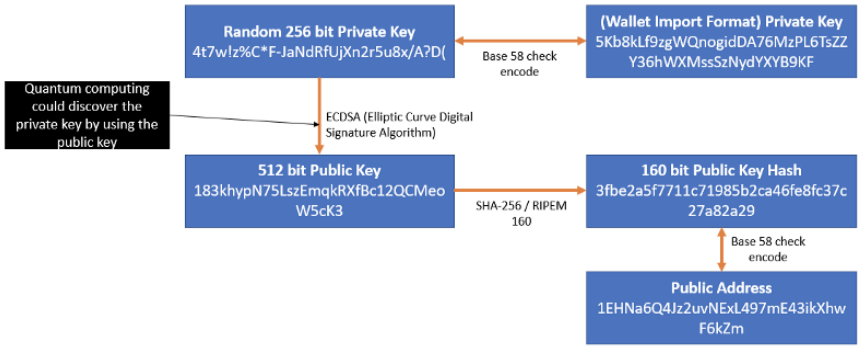
Classical computers (the computers we utilize today) operate with bits, which store one of two states, 0 or 1. Quantum computers operate with qubits, (similar to bits) which store state in either 0 or 1 however, through a principle called superposition also exist simultaneously between 0 and 1 when not being measured (an example of a qubit can be a photon, nucleus, or an electron). Due to the unique properties of qubits, they can be leveraged to do some interesting things that cannot be achieved with classical computers and bits.

Quantum computers will be designed to describe all correlations between qubits, i.e. effectively exponentially increasing computational throughput by 2^n (n = correlation per qubit in each system) for specific operations. For example, 2 qubits = 4 classical bits, 3 qubits = 8 bits, and 20 qubits = 1,048,576 bits. Considering quantum computers can process data in parallel vs. in serial (like classical computers) for specific operations, they can significantly speed up certain computations, such as database searches or finding the private key from a public key.

Implications of Quantum Computing for Blockchains

Quantum computers are not good at everything, only specific types of calculations that specific algorithms could exploit. For example, there is an algorithm called Shor's algorithm that can be run on a quantum computer and will have the capability to run very fast integer factorization calculations in parallel (integer factorization is a process to decompose a composite number into the product of small integers, which can be used to find a private key from a public key; a private key and public key are basically very large numbers), thereby finding the prime factors of any given integer.

Many of today's current encryption schemes will not be resistant to quantum computing attacks, such as RSA and ECDSA (Elliptic Curve Cryptography). Bitcoin and Ethereum use ECDSA for generation of private keys and public keys. For example, below is a basic depiction of the process that Bitcoin uses to generate the private key, public key, and public address.



Bitcoin key generation and bitcoin public address generation

The private key is generated from a number with an added level of entropy (randomness), the public key is generated from the private key, and the public address is generated from the hash of the public key. Quantum computers could utilize Shor's algorithm to derive the private key from the public key, and then leverage the private key to forge transactions or steal a user's balance (Bitcoin/Ethereum).

Shor's algorithm is primarily an issue for re-used addresses. For addresses that have not been used, quantum computers could leverage Grover's algorithms to solve for SHA-256 or SHA3–256 to find the public key from the public key hash, however quantum computers would only be able to find the public key in half the time vs. classical computers of today. Even at half the time, the key would not be found within anyone's lifetime currently on Earth. Additionally, merkle-tree hashing is not currently susceptible to quantum computing attacks.

There are currently algorithms being developed that will be difficult to break with quantum computers, such as Lattice-based cryptography or multivariate cryptography, which may be candidates for replacement of ECDSA in the future for blockchains such as Bitcoin and Ethereum.

ZK-STARKS do not rely on private-public key pairings (such as ECDSA), but rely on collision resistant hashing for interactive solutions (which

Grover's algorithm does not meaningfully break), and a random oracle model (a model that is typically used instead of general cryptographic hash functions where strong randomness assumptions are required for the oracle output) for non-interactive proofs (zk-nSTARK, n = non-interactive), therefore ZK-STARKs are currently resistant to quantum computer attacks.

Quantum computers are still years away (estimates point to 2026–2035), so there is little need to worry about their capabilities today. Also, I utilize the term quantum resistance in this post as quantum resistance and quantum proof are two terms to reflect on. Like a watch that is resistant to water to a certain depth, and a watch that is impervious to water (water proof), until the true capabilities of quantum computers are known, it is hard to say if an algorithm may be found that leverages quantum computers to circumvent current potential quantum resistant encryption algorithms today. Hence, we may not see changes to encryption algorithms used in Bitcoin or Ethereum until mainstream quantum computing is closer to becoming a reality.

An additional video that can be helpful l in understanding the difference between classical computers and quantum computers: https://www.youtube.com/watch?v=JhHMJCUmq28

Current State and Final Thoughts

Currently, ZK-SNARKs are available in the cryptocurrency Zcash, as well as the library libSNARK to build ZK-SNARK programs that can be leveraged in blockchains. ZK-STARKs are a newer technology, and not deployed in production capacity as of 6/2018. There is a new company called StarkWare Industries that is looking to solve some of the challenges with leveraging ZK-STARKs (one being the size of the proof) and also commercialize the technology, which can be leveraged across multiple industries, including blockchain implementations.

ZK-STARKs are scalable, transparent, have universal application, and currently quantum resistant. This allows for the creation of trust in the technology, as it is verifiable. There are many areas that can be enhanced by using a technology such as ZK-STARKs where trust is required and there are large incentives to cheat, such as:

1. Voting systems
2. Running a computation and verifying its results, such as a blockchain's past transactions
3. Secure verification of information, such as for proving identity or credentials

My current thinking is we will see Zcash adopt ZK-nSTARKs (n standing for non-interactive) technology into its blockchain in the future. Additionally, Ethereum may leverage ZK-STARKS in verifiable computation and potentially secure/anonymous transactions, as well as Dapps where privacy is important such as Brave's web browser that leverages the Basic Attention Token.

ZK-STARKs is an exciting technology that will enable trust to be achieved in computing systems that has not been achieved previously, as the trust created is verifiable and public. This idea is especially important, as entities that have significant incentives to falsify information will not be able to do so when using a ZK-STARK proof system. It will take time for this type of technology to be adopted however, the benefits will be significant and create something truly unique: verifiable trust in a system where there could be very high incentives to try and falsify information processed by the system.

Please let me know if you have any questions or have thoughts on the above below, look forward to discussing!

In addition, if organizations have questions around ZK-tech or tokenomics, please feel free to review more how I help organizations at https://cynapseblockchain.com/

This post can also be found at Cynapse Blockchain Consulting's website: https://cynapseblockchain.com/2018/09/07/zk-starks%e2%80%8a-%e2%80%8acreate-verifiable-trust-even-against-quantum-computers/