

Protocol Foundations 005: Zero-knowledge proofs

Mario Havel and Tim Beiko

Zero-knowledge, often abbreviated as ZK, is a domain of cryptography, similar to encryption which we covered in the first Protocol Foundations issue in this series.¹

ZK employs cryptographic principles in a different way: instead of obfuscating data cryptographically, ZK protocols allow us to prove knowledge of data without fully revealing it. If we analogize traditional encryption to a light switch that can be either on or off, hiding or revealing everything, zero-knowledge proofs are more like a dimmer, controlling the amount of information that is visible.

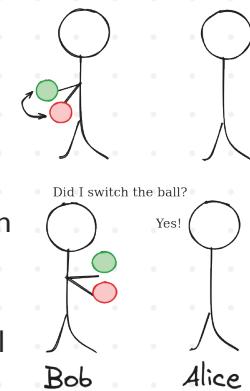
In other words, zero-knowledge proofs enable us to probabilistically prove a statement or the possession of information without revealing the information itself. Proving the validity of a statement about the data without revealing the actual data might seem counterintuitive or almost impossible—that’s why these systems are so revolutionary. This breakthrough property opens up a new paradigm of private and scalable applications over large amounts of data. Let’s examine a simple example to illustrate what this means:

Bob is a color-blind person. His friend Alice has two balls, one red and one green. To color-blind Bob both balls appear the same. He does not believe Alice can reliably distinguish between them. Alice can prove that she can tell them apart without Bob knowing what color the balls are. To do this, Bob holds one ball in each hand in front of Alice, who sees which hand holds which ball. Bob then puts the balls behind his back and either switches them or keeps them in the same hand. He then shows them to Alice again, asking whether they changed hands. Assuming Alice is not also color-blind, she’ll reliably tell Bob whether or not he switched the balls in his hands while they were behind his back.

The first time Bob does this, Alice has a 50% chance of getting lucky. Bob can then repeat

the sequence, over and over. Each time Alice guesses correctly, the probability she has simply gotten lucky on each chance is halved. If they repeat this 100 times and Alice guesses correctly each time, the odds she would have gotten the right answer each time by luck are 1 out of 2^{100} —a larger number than the number of atoms in the universe.

We call this a *zero-knowledge proof* because Bob never learns which ball is green or red and at no time can he distinguish them himself, even though he learns with high confidence that Alice can distinguish them.



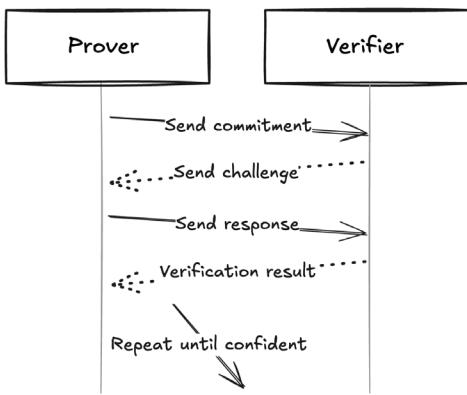
Provers and verifiers. In a proof mechanism like Alice and Bob and the two colored balls, participants have two distinct roles: *prover* and *verifier*. The verifier (Bob) challenges the prover (Alice) after changing the order of balls; the prover creates the proof by sharing the result of their observation. With each iteration of this process, the probability that the correct answer is a coincidence rapidly asymptotes to zero.

Arguments. The obvious limitation of this approach is the interactivity of the protocol, requiring many rounds of communication between the prover and verifier. The constant back and forth is impractical even with today’s fast computation because it limits how the protocol can operate. In practice, proofs involve complex logical statements broken down into smaller components, similarly to how high-level coding languages get broken down to sequences of 0s and 1s by computers. In ZK systems, the statement that is evaluated is called an *assertion*. The result obtained by this evaluation is called an *argument*, also often referred to as a *proof*.

Scalability. ZK proof systems have been an active area of research since the 1980s. Recent

1. Mario Havel and Tim Beiko, “Protocol Foundations 001: Cryptography,” *Summer of Protocols*, 2023. summerofprotocols.com/wp-content/uploads/2023/12/53-BEIKO-001-2023-12-13.pdf

INTERACTIVE ZK PROOF



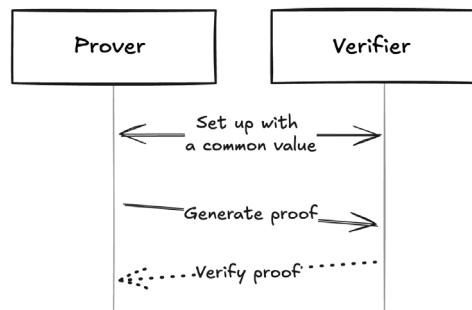
cryptographic protocols have been able to improve proving efficiency, making ZK a viable solution for some real-world applications. In parallel to ZK research, systems like the sum-check protocol² were invented to generate and verify proofs more easily by calculating a single point on a polynomial. Even before the ZK application, the sum-check protocol was a remarkable achievement that opened new possibilities. A widely quoted sentence from an important 1993 paper gets to the heart of the importance of scalability:

In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.³

The protocol reduces checking the entire sum to checking only a single, randomly-chosen point. Using a single point allows the actual ZK proof to be very small, making it possible to verify correctness without heavy computation. These newer, smaller, proofs are called *succinct* or *scalable*. Scalability in this context means that we are able to generate a relatively small proof from large amounts of data and verification of this proof scales sublinearly.

Reference strings. Another important innovation in ZK proofs was the introduction of *reference strings* shared between the prover

NON-INTERACTIVE ZK PROOF



and verifier. By sharing a secret value between the prover and verifier, we can eliminate the need for interactive verification. Of course, this comes at the cost of an additional trust assumption: that the shared secret actually is secret!

Trusted setups. This trade-off simplifies the proving but creates overhead for operating such a system. Leaking the reference string value outside of verifying parties would result in an insecure proof system where anyone could generate fake proofs. If the initial setup has been done securely, for example by using multi-party computation (MPC), the system can be considered safe. This is known as *trusted setup*. Trusted setups usually have an “n=1” trust assumption, meaning that as long as a single participant in the MPC keeps their input to the secret generation protocol hidden, the secret cannot be reconstructed. In other words, every input to the secret is necessary to calculate the computation and a single participant honestly destroying their input to the process is sufficient to guarantee the security of the entire system. With processes to create a secure trusted setup, the trade-off is manageable and these types of systems have been in wide public use for almost a decade. Notably, the Ethereum community recently used a trusted setup with over 141,000 participants.⁴

SNARKs. Succinctness and trusted steps form the basis for the most commonly used ZK

-
2. Presented originally in Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan, “Algebraic methods for interactive proof systems,” *Journal of the ACM* 39, no. 4 (October 1992): 859–868 (doi.org/10.1145/146585.146605). See also some interesting retrospection from one of the authors in his blog with enlightening comments from Eli Ben-Sasson: blog.computationalcomplexity.org/2024/02/sumchecks-and-snarks.html.
 3. Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan, “Algebraic methods for interactive proof systems,” *Journal of the ACM* 39, no. 4 (October 1992): 859–868. doi.org/10.1145/103418.103428
 4. KZG Summoning Ceremony, 2022. ceremony.ethereum.org

proof systems: *succinct non-interactive arguments of knowledge* or SNARKs.

STARKs. A further iteration of ZK systems are *succinct transparent arguments of knowledge*—STARKs—which improve over SNARKs by being *transparent*. Unlike SNARKs, STARKs do not need to use a secret reference value in the system to generate a proof. Without the need for a trusted setup, STARK systems are easier to create and have been popular in open public systems. The trade-off here is transparency requires a larger proof size and slower computation, making STARKs impractical for several use cases.

The scalable or succinct and non-interactive properties of these schemes are what makes them so useful. They allow users to prove information privately and quickly and for others to efficiently verify these proofs. Discussing the theory of zero-knowledge can be abstract but the implications of these protocols are significant.

Here are a few example applications. The most obvious benefit of ZK proofs is better privacy for users. For example, many services with age or location restrictions may require submitting a government ID that leaks all information contained in it. With zero-knowledge, instead of providing their full ID, users can generate a proof of their age, ID issuance location, or both, without providing any additional information. To enable this, the data in IDs needs to be formalized in a ZK protocol. ZK protocols like STARKs or SNARKs are general systems which include specific tools called circuits. A ZK circuit is the encoding of a computer program into constraints, for example, verifying that an ID is issued by a valid issuer and that its country matches a specific list or that it has not yet expired.

A single ZK circuit may be associated with multiple possible proof generation procedures. It also defines how verification works. Anyone with the verification software can then verify proofs for themselves, without requiring the original input data, in this case the ID.

Taking this even further, ID with ZK support can even be used for a voting system. Digitized voting has been a challenge because of privacy and trust assumptions which are more

easily solved with paper balloting where it's easier to avoid a single point of failure. Since ZK proofs are designed to keep information private while proving it in public, they are ideal for democratic voting. Voters can cast a verifiable vote without revealing who they are or what party they voted for, and the integrity of the vote can be independently audited.

This audit is possible by creating a public and open-source proof system with circuits for proving validity of each voter and their vote. Transparency is necessary here because a closed system would be easier to manipulate. The system needs to include multiple circuits for proving authenticity of voters, whether they are registered and what vote they cast. First, the authentication is done by proving that a unique voter ID is in the database of voters without revealing which ID it is. Based on this, voters can vote and generate a proof about vote validity to ensure that it corresponds to valid choices (e.g., specific candidates) without revealing individual vote details.

The result of the election in this case is a system-wide proof that can show that all votes were legitimate and counted correctly. Without disclosing any individual votes, the ZK proof is publicly verifiable and practically impossible to forge if the underlying system is transparent.

While zero-knowledge systems are slowly being adopted, applications are still under active development and not yet sufficiently robust to be used as full-scale replacements for more battle-tested cryptographic protocols. Still, they open up exciting future possibilities. In cases where relying on cutting-edge technology is appropriate, as in blockchain-based applications which already have dependencies on these protocols, ZK proofs can allow novel coordination mechanisms to emerge. Over time, some of these will hopefully prove viable for mainstream usage!

MARIO HAVEL & TIM BEIKO are part of the Ethereum Foundation's Protocol Support team which helps facilitate Ethereum network upgrades as well as other protocol-related initiatives, including Summer of Protocols.

MARIO | github.com/taxmeifyoucan
TIM | warpcast.com/tim

ProtocolKit



ProtocolKit

summerofprotocols.com
hello@summerofprotocols.com

© 2023 Ethereum Foundation. All contributions are the property of their respective authors and are used by Ethereum Foundation under license. All contributions are licensed by their respective authors under CC BY-NC 4.0. After 2026-12-13, all contributions will be licensed by their respective authors under CC BY 4.0.
Learn more at: summerofprotocols.com/ccplus-license-2023

ISBN-13: 978-1-962872-53-9

Printed in the United States of America
Printing history: July 2024

1 3 5 7 9 10 8 6 4 2

ISBN 978-1-962872-53-9

A standard barcode for the ISBN 978-1-962872-53-9.

A standard barcode for the EAN 900000.

9 781962 872539

RETROSPECTUS



NEWSLETTER

