Tech Explainers

ZERO-KNOWLEDGE PROOFS

BUSINESS SUMMARY

Picture the process of opening up a new bank account. You walk into the office and the clerk asks you for all sorts of identification documents, for example, your driver's license. The clerk takes this information, puts it into a system, and runs checks for regulatory compliance, such as ensuring you do not show up on a sanctions list.

Notice two things under this system. One, personal information is revealed that is probably not relevant to the particular things the bank cares about. For example, your driver's license contains your height, but it is not clear how that is needed to run a sanctions check. Second, and more fundamentally, *you* are the index that is searched upon. Your personal information is a key used in further databases of information to answer queries about you—for example, whether you are on a sanctions list. So in actuality, much more information than what is printed on your driver's license is being provided. This data leakage presents an issue for the bank, which now holds personal information it does not necessarily want or need. And unwanted personal information is a liability.

What if there was another way, where questions about you could be answered without revealing anything about who you are? For example, what if you could prove to the bank that you are not on a sanctions list without revealing anything else about you, even your name? The technology that makes this possible already exists, and is called a "zero-knowledge proof" (ZKP). This technology explainer unpacks how ZKPs work and why legal practitioners should understand them to be effective moving forward.

INTRODUCTION

ZKPs are a cryptographic protocol that has recently gained significant traction with a wide variety of industry participants. In brief, the technology allows one party to cryptographically prove to another party that a statement is true without revealing why it is true.

There are three core benefits to the technology. First, it offers a foolproof way to say whether a particular statement is true. Second, it prevents information leakage with respect to facts necessary to prove a statement is true. Finally, it provides these guarantees in a way that conserves significant computational resources.

In this Technology Explainer, we take a first step at demystifying ZKPs so legal practitioners can better understand what this technology makes possible. We do so by walking through how ZKPs work in practice by way of a hypothetical scenario. We conclude by explaining why it is important for practitioners to understand how this technology functions.

ZKPS IN PRACTICE

To demonstrate how ZKPs[1] work, we walk through an example: Prover Penelope and Verifier Valerie want to jointly agree that a statement is true, but do not wish to reveal any more information than absolutely necessary in the process. Specifically, Penelope would like to prove to Valerie that she knows two numbers, or "factors," that multiply to a given number, without revealing what those factors are.

Why are we choosing factoring as our example? Because deriving factors is hard work—and hard work makes the solution valuable and, by extension, worthy of secrecy.

To see why, imagine being asked to calculate 397 times 491. This calculation would be easy: simply plug the numbers into a calculator and receive the solution. By contrast, imagine being asked to identify two integers that multiply to 194,927 (other than 1 and 194,927). No calculator could help you efficiently reach a solution. Attempting to identify these factors effectively requires a trial-and-error process—successively searching for divisors of 194,927 until the factors are identified. The factorization problem is especially difficult when being asked to identify two *prime* integers that multiply to a given very large number[2] (such as 397 and 491). Even today, computing has not been able to solve prime factorization "efficiently."[3]

With this background in mind, we return to Penelope and Valerie. Penelope has done the painstaking work of finding the factors of 194,927, and wants to prove to Valerie that she has found them. Of course, Penelope could tell Valerie the two factors (and then Valerie could multiply them on her calculator to confirm the result), but then Valerie could take credit for Penelope's hard work[4] or otherwise use the information in ways Penelope does not anticipate or does not want. Instead, Penelope could leverage ZKPs to prove to Valerie she has found two factors for a given large number[5] without revealing any information about the factors.

In order to do so, Penelope and Valerie jointly[6] agree to write a program, `checkFactors()`, using ZoKrates, a high-level programming language and toolbox that anyone can use to prove and verify ZKPs.[7] Both Penelope and Valerie have access to the toolbox. Crucially, the toolbox includes a "proving algorithm" and a "verifying algorithm" for participants to use in connection with the program they write (which we demonstrate below). As part of this endeavor, both parties can review the program, verify how it works and agree to its terms.

```
1  def checkFactors(private field factor1, private field factor2, field product) {
2      assert(factor1 != 1);
3      assert(factor2 != 1);
4      assert(factor1 * factor2 == product);
5      return;
6  }
```

FIGURE 1: An expression of Penelope and Valerie's `checkFactors()` program.

For readers that want to understand the program in depth, we walk through how the code works. The `checkFactors`() program is simple, requiring only three variables: `factor1`, `factor2` and `product`. The purpose of this program is to prove knowledge of `factor1` and `factor2` for a publicly (and explicitly) stated `product`. The program first confirms that neither `factor1` nor `factor2` is trivial (because every number has 1 as a factor), in lines 2 and 3, by confirming that neither has a value of 1. The program then confirms that the proposed `factor1` and `factor2` multiply to the `product` figure in line 4. If all these checks pass, the program returns `true` in line 5; if not, it returns `false`.

The key thing to note about this program, as it relates to the principles of ZKPs, is the `private` keyword in front of variables `factor1` and `factor2`. This keyword ensures that the value entered for `factor1` and `factor2` by Penelope will not be revealed to Valerie. The `product` will be, because it is not marked `private`.

After Penelope and Valerie agree on the code for `checkFactors`(), each of them takes an identical copy of the program and installs it on her personal computer.[8] Penelope, far away from and not in communication with Valerie, then runs the proving algorithm included in the ZoKrates toolbox on her personal computer. From Penelope's perspective, the proving algorithm is a black box—she does not know the details of how the proving algorithm works, just that it will prove she knows two factors for a given product if she passes in the program `checkFactors`().[9]

Penelope enters into the proving algorithm the following information: the number 397 for `private factor1`, the number 491 for `private factor2`, the number 194,927 for non-private product and a copy of the program she coded with Valerie (*Figure 1*). The proving algorithm takes this information and returns an unintelligible bundle of data: a "proof" contained within a JSON object (which, for present purposes, is simply a common standardized way to store and transmit data) as depicted in *Figure 2*.[10] Critically, this JSON object, including the proof, does not reveal the private factors (`factor1` and `factor2`) used in its generation.


```
{
  "scheme": "g16",
  "curve": "bn128",
  "proof": {
   "a": [
     "0x279e4a60ba944baca6b9a554b115e9433f1d6a433231474f86de68858bda49ea",
     "0x0b00a21f2c778e24c6fcac76aaaeee1adca823fbb63d412e4b7c177edb5050c9"
   ],
   "b": [
     [
       "0x0c4c00922fa3fa04aa50034b018519b9850b97b82fcc7b73a057acebf7c88acf",
       "0x2f0ba6cd04a7a54e0d2baa9e8d86289356e266be8cdeb287b8fb0ee52147948e"
     ],
     [
       "0x0bf19f8909298a7233961a5ac5ca3637f32eb2e2d93f0b699054a9cea63889f0",
       "0x0faaee822414a141805d21376066b1f1fdfb4f0573dac404254aa2a531dbcc70"
     ]
   ],
   "c": [
     "0x118483269dcf4747737df70487d4c5865289f2ad18b898ae715c60d3ec2d431f",
     "0x1ebf8afe1aaf397cd8dc3796f23f164ae63d13895e7e9ba375f8874f25a906f1"
   ]
  },
  "inputs": [
    "0x000000000000000000000000000000000000000000000000000000000002f96f"
  ]
}
```


FIGURE 2: An expression of the JSON object from Penelope's proving algorithm once she has entered (1) `private factor1` and `factor2`, (2) the non-private `product` and (3) the local copy of the computer program she created with Valerie.

The output states that it was generated for the non-private inputs of "`0x...2f96f`", represented in the JSON object at the "`inputs`" key. This value is a hexadecimal representation of the `product` 194,927. The "`proof`" key contains `a`, `b` and `c`, which represent three points on an elliptic curve. The technical reason this information is included is extraneous—what matters for our purposes is that nothing in these three points reveals anything about `private` `factor1` and `factor2`. To summarize, the `product` is included in the JSON object output, but the `private` `factor1` and `factor2` are not, because we set these two inputs as `private` in our original `checkFactors`() program.

Penelope then communicates to Valerie only this JSON object output generated by her proving algorithm. Recall that in connection with the creation of the program, Valerie was provided a verifying algorithm by the ZoKrates toolbox. Valerie now enters the JSON object, containing the proof and non-private `product` 194,927, along with her local copy of the program into that verifying algorithm.

The verifying algorithm can only return two outputs: `true` or `false`. The verifying algorithm returns `true` if the proof is "valid" for the public statement product, that is, Penelope did in fact enter two factors that multiply to this product when running the proving algorithm.[11] Valerie has no idea what the private components of "`proof`" are that would cause it to return true—she knows only that it does and thus that Penelope has found two factors of the `product`.

Notably, Valerie does not, and need not, run the program itself to verify Penelope's statement of knowledge. The verifier never needing to run the program means that ZKPs come with the further benefit of short and fast verification.[12]

WHY LEGAL PRACTITIONERS SHOULD UNDERSTAND ZKPS

Although our example may appear largely academic, it demonstrates ZKPs' core promise—efficient verification of information without revelation of what underlies that information. In practice, ZKPs may be used to communicate facts that are much more interesting and sensitive than those from our factorization example. For example, consider the current paradigm of age verification. Provers provide a certified government document, such as a driver's license, to verifiers. This scenario has among its many attendant problems unnecessary information leakage—the verifier learns a significant amount of information beyond age, such as name, address, height and even eye color, and becomes liable for the collection and processing of this personally identifiable information under privacy laws. By contrast, ZKPs make it possible to prove age or even identity without revealing any other personal information. To analogize to our hypothetical, the verifier learns only the `product`, that the person is over a certain age or is a specific, identified person. While real-world use cases are certainly more complicated than our hypothetical, at a fundamental level, ZKPs operate as we described.[13]

These obvious advantages have led to ZKPs' mounting adoption by projects trying to solve some of the most existential questions that new technology presents. For example, OpenAI CEO Sam Altman founded Worldcoin as a complement to ChatGPT. Worldcoin, a protocol that aims to use iris scans in order to prove not identity but humanness, is motivated by the principle that as AI increasingly blurs the line between humans and bots, we will need protocols to determine which is which.[14] ZKPs constitute a foundational component of Worldcoin.[15] Specifically, ZKPs

permit Worldcoin users who have undergone a valid iris scan to prove they are a human without allowing Worldcoin—or any verifier— to associate that user with a particular identity. This feature is an essential component to avoiding tracking a person's (immutable) identity across the web.[16]

At the legal level, ZKPs' benefits are obvious for privacy practitioners, but it behooves practitioners across all industries to understand them. ZKPs promise significant improvements with respect to information-sharing practices generally—and information sharing is a core part of every legal relationship. For example, M&A practitioners may wish to leverage ZKPs to prove information in the due diligence phase or to submit a secret bid in an auction, and litigators may wish to deploy ZKPs to protect valuable proprietary information in trade secret litigation.

But specific legal applications are not the only reason—and indeed not the primary reason—why it is essential for practitioners to understand how ZKPs function in practice. We consider three central reasons here.

First, and most obviously, we expect businesses to embrace ZKPs in new and complex ways. Practitioners will need to understand ZKPs to fully grasp their clients' businesses, which is essential to effective advocacy and representation.

Second, only by understanding how ZKPs actually work can practitioners design legal relationships in a way that leverages the technology—for instance, by generally mirroring the protocol Penelope and Valerie followed. Practitioners who understand the nuts and bolts of ZKPs will be uniquely positioned to advise clients on novel information-sharing practices with third parties that were previously thought to be too legally risky, all without compromising privacy principles. Finally, some of ZKPs' most natural use cases occur in contract design. Practitioners will need to understand how ZKPs function in order to draft agreements leveraging them, or to advise clients on how ZKPs function in their agreements. For example, we can easily imagine using ZKPs as an audit function in a license agreement. A licensee could be required to use a ZKP to prove the amount of royalty payments it must make, without revealing to the licensor any particular or sensitive details about its licensing practices.

In short, we predict that ZKPs will soon become a building block, much like encryption, that forms a critical component of our experience on the Internet today. This article presents a first step for legal practitioners to embrace the promise of this technology.

David J. Kappos
+1-212-474-1168
dkappos@cravath.com

Sasha Rosenthal-Larrea
+1-212-474-1967
srosenthal-larrea@cravath.com

Carys J. Webb, *CIPP/US*, *CIPP/E*
+1-212-474-1249
cwebb@cravath.com

Daniel M. Barabander
+1-212-474-1284
dbarabander@cravath.com

1 We focus solely on non-interactive ZKPs, but interactive ZKPs, where many back-and-forth interactions between prover and verifier are required to confirm the veracity of the prover's statements, also exist.

2 Our "large" number, 194,927, is used purely for simplicity of demonstration. In practice, the number would need to be significantly larger than six digits to actually constitute a "hard" problem. For example, the smallest RSA Number (discussed *infra* note 4) is *100* digits.

3 That is, as the input size grows, the running time of state-of-the-art factorization algorithms scale poorly. As a result, current algorithms do not run quickly enough to provide a practical solution for factorizations involving large prime numbers.

4 While finding prime factorizations may seem purely academic, it is such a hard problem that for over 15 years, American network security company RSA Laboratories offered prize money—ranging from $10,000 to $200,000—for anyone able to find factors of such numbers, called "RSA Numbers." Even today, most RSA Numbers remain unfactored. *See* The Physics arXiv Blog, *A 30-Year-Old Cryptographic Challenge Is About to Be Solved*, DISCOVER (Jan. 10, 2023, 10:11 a.m.), https://www.discovermagazine.com/technology/a-30-year-old-cryptographic-challenge-is-about-to-be-solved (stating that, as of January 2023, 31 of 54 RSA numbers remained unfactored).

5 Penelope could prove knowledge of factors for any arbitrary number, but selecting a product of two primes makes finding its factors significantly more challenging (and, by extension, interesting).

6 The parties need not literally write the program together—the parties simply need a copy of the same program, regardless of who wrote it.

7 ZoKrates is one of many programming languages and toolboxes Penelope could use to generate a ZKP. We use ZoKrates because it is designed to facilitate ZKP creation on the Ethereum blockchain, which represents one of the most promising use cases of ZKPs. *See ZoKrates*, GITHUB (last visited May 24, 2023), https://ZoKrates.github.io (discussing how the language bridges certain gaps with respect to ZKPs on the Ethereum blockchain). For a discussion of ZKPs and their Ethereum applications more broadly, *see generally Zero-Knowledge Proofs*, ETHEREUM (Apr. 19, 2023), https://ethereum.org/en/zero-knowledge-proofs.

8 The program takes the form of an "arithmetic circuit," a model into which the source code of the ZoKrates program has been compiled. An arithmetic circuit is a generalizable model used for computation. The circuit is then run through a setup algorithm, which compresses the circuit for efficiency. The setup algorithm "summarizes" the arithmetic circuit, leaving only the components germane to the generation and validation of the proof. We abstract away these concepts for ease of understanding—one of many oversimplifications we use to make the example more understandable.

9 It is important to recognize that the proving/verifying algorithms will provide a proof/verification for any arbitrary program and inputs Penelope/Valerie provide—they are generalizable algorithms, not specific to `checkFactors()`.

10 *See, e.g., Working with JSON,* MDN WEB DOCS (May 10, 2023), https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON.

11 More technically, a "valid" proof demonstrates that Penelope knows two factors, `factor1` and `factor2`, that cause the program to return `true` for a given target `product`.

12 *See* BLOCKCHAIN-WEB3 MOOCS, *ZKP Workshop 2022: Dan Boneh - Constructing Modern SNARKS,* YOUTUBE (Nov. 15, 2022), https://www.youtube.com/watch?v=6psLQv5Hf _ I.

13 That is, non-interactive ZKPs. *See supra* note 1.

14 For a deeper discussion on the challenges synthetic media generated by AI presents, see David J. Kappos, Sasha Rosenthal-Larrea, Daniel M. Barabander and Leslie Liu, *ChatGPT and Text Fakes—Sensible Policy to Balance Growth and Risk*, HARV. J.L. & TECH. DIGEST (Apr. 6, 2023), https://jolt.law.harvard.edu/digest/chatgpt-and-text-fakes-sensible-policy-to-balance-growth-and-risk.

15 *See Privacy at Worldcoin: Technical Deep Dive - Part I*, WORLDCOIN (Feb. 9, 2023), https://worldcoin.org/blog/developers/privacy-deep-dive.

16 Specifically, a user's "identity commitment" (which is unrelated to iris scan data) is stored on a blockchain in a Merkle root, a hashed representation of all identities. A ZKP is used to prove that a given user's identity is a member of the Merkle tree associated with that Merkle root. *See id*.

Cravath, Swaine & Moore LLP

NEW YORK

Worldwide Plaza
825 Eighth Avenue
New York, NY 10019-7475
T+1-212-474-1000
F+1-212-474-3700

LONDON

CityPoint
One Ropemaker Street
London EC2Y 9HR
T+44-20-7453-1000
F+44-20-7860-1150

WASHINGTON, D.C.

1601 K Street NW
Washington, D.C. 20006-1682
T+1-202-869-7700
F+1-202-869-7600