A Walk into Zero-Knowledge Proof

LING Shengchen
Department of Computer Science
City University of Hong Kong
Hong Kong SAR, China
shengling2-c@my.cityu.edu.hk

***Abstract*—As the blockchain technology and cryptocurrency booming up, zero-knowledge proof attracts more and more attention. This article makes a brief introduction about the zero-knowledge proof, including its background, properties, proof process with mathematical theorems and typical examples, drawbacks, and current applications.**

Keywords—cryptography, zero-knowledge, blockchain, privacy

I. Introduction

The concept of zero-knowledge proof was firstly proposed in 1985 by Shafi Goldwasser, Silvio Micali, and Charles Rackoff in their paper "The Knowledge Complexity of Interactive Proof-Systems" [1]. After that, in 1991, Goldreich, Micali and Wigderson showed how to create a zero-knowledge proof system for the NP-complete graph coloring problem with three colors [2]. Since every problem in NP can be efficiently reduced to this problem, which means that all problems in NP have zero-knowledge proofs. Furthermore, as for the scope of popular science, Jean-Jacques Quisquater and his collaborators published 'How to Explain Zero-Knowledge Protocols to Your Children' in 1990 [3].

As a core, a zero-knowledge proof is an interaction between two people, or two parties, known as "prover" and "verifier". In the interaction, one can convince the other one that some statement is true, without revealing any more information except for that result.

Zero-knowledge proof actually goes even further beyond that. For example, the overarching concept applied in multi-party computation (MPC), to accomplish some tasks without knowing anything more than exactly what's needed to accomplish those tasks. In one word, it proves that someone is behaving honestly.

As a matter of fact, zero-knowledge proof is not about making something more efficient, but about doing things that cannot be done before, bringing people who are mutually distrustful together.

The blockchain space is one of the spaces seeing zero-knowledge proof being implemented, which is just a beginning. This article discusses zero-knowledge proof mainly from the blockchain aspect.

II. Background

A. Conflict between Privacy and Blockchain

Imagine a scenario that Alice sends Bob 1 Bitcoin.

Before this transaction is made valid, the network needs to ensure that Alice truly has at least 1 bitcoin (and gas fee, here we omit that for the convenience of description). Because of the decentralized property of Bitcoin system, everyone in the network needs to check that information. It helps to protect the authenticity of transaction for Bob, but with no doubt sacrifices the privacy of Alice.

Actually, they use their Bitcoin address for transactions instead of their names, but it won't make a difference. If only any information in reality is known, etc., some behaviors, others can search through the whole chain to find and track the corresponding address on chain. The process may not be easy, but it's not impossible.

Normally encryption may help to deal with privacy, but not in this case. The blockchain users certainly can encrypt their sensitive data as they want, but still need others to validate the existence of your assets. If the validators can't see, then it won't be validated as true.

As a result, blockchain and privacy seem not to go well with each other. The public ledger system acts as one of the cores of blockchain, naturally opposite to the so-called privacy.

B. Problem of Privacy Leakage

In fact, the conflict of privacy leakage is far beyond the field of blockchain. Traditionally, when proving a claim, "evidence" must be provided, which might lead to the leak of privacy; or obtaining verification by a trusted third party, then the central database it uses might be compromised.

Zero-knowledge proof can solve the privacy problem by using a "witness" to generate the proof of validity without exposing any other information. The "witness" here is not some kind of third party, but a medium to verify the results, which will be explained in detail below.

C. Conflict between Randomness and Determinism

In classic proof process, randomness is specifically against the goal. If thinking about Platonic idea of a proof, there's no randomness, no non-determinism present. It seems so counter-intuitive that randomness can be helpful for proving.

On the other hand, however, the "bad" randomness can become "good". With the help of probability theory, the unpredictability of randomness can be utilized to hide the secret information. That's the point of zero-knowledge proof.

III. In-depth Descussion

A. Key Properties

A zero-knowledge proof requires three key properties to be fully implemented.

1) Completeness: If the statement is true, there must be a way for the prover to prove that, and the verifier will be convinced without any other help.

2) Soundness: If the statement is false, the prover cannot forge the result, and the verifier won't be convinced in any circumstances.

3) Zero-knowledge: The verifier should not get access to any more information except for the result.

B. Proof Process

In a basic proof process, a zero-knowledge proof consists of three elements: witness, challenge, and response. See Fig.1.

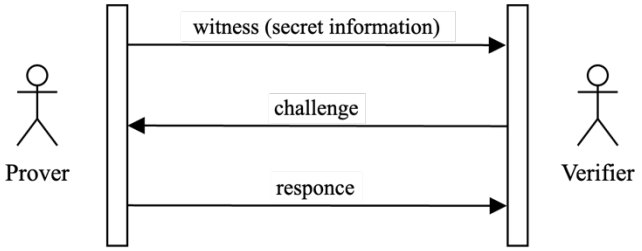

Fig. 1. Zero-knowledge proof process

1) Witness: The secret information is the "witness" of the proof process. The prover established a set of questions that can only be answered by whom owns the so-called "knowledge".

2) Challenge: The verifier randomly chooses a question from the set mentioned above and requires the prover to provide the answer.

3) Responce: The prover receives the question, calculates the result, and sends it back to the verifier for verification.

To ensure that the prover not blindly guesses out the right answer by chance, the verifier can continue to choose more questions. As these interactions repeating, the probability that the prover falsifying the "knowledge" decreases significantly, and the verifier gradually believes that the prover really owns the "knowledge". In fact, the verifier can continuously make requirements until satisfied.

C. Syllogism Examples

As is shown above, normally a syllogism is used to construct a zero-knowledge proof process. The syllogism can be either physical practice or computational mathematics. Here shows the famous Sudoku puzzle as an example of the former one, while discrete logarithm problem as the latter one.

1) Sudoku puzzle

In a Sudoku puzzle the challenge is a 9×9 grid subdivided into nine 3×3 sub grids. The prover needs to prove that he knows the result of the Sudoku puzzle, however, he doesn't want to give away the answers directly [4].

First, the prover writes down the answers of the Sudoku puzzle on a card, puts them in the corresponding position in the Sudoku, and turns the answer card over, with the numbers on the question facing up.

Second, the verifier verifies whether the number on the question is the same as the number on the card facing up, and then randomly assigns a row (or a column).

Third, the prover collapses and scrambles the row (or column) of cards, and then presents them all to the verifier. The verifier verifies that the numbers are exactly 1-9.

2) Discrete logarithm problem

Notation: prover *P* and verifier *V*. Given a function $f(x) = g^x$, where g is a public large integer. Given a number y, the prover proves that he knows the discrete logarithm $\log_g y$, i.e., $y = f(x)$.

The following protocol "f-preimage protocol" [5] can be used for any f generated by a q-one-way generator, acting as a generalization of Schnorr's discrete logarithm protocol [6].

First, *P* chooses *r* at random and sends

$$m = f(r) = g^r \tag{1}$$

to *V*.

Second, *V* chooses a random challenge *e*, so that $0 \le e < q$ and sends it to *P*.

Third, *P* sends

$$z = r \times v^e \tag{2}$$

to *V*, who checks that

$$f(z) = m \times u^e \tag{3}$$

The $f(x) = g^x$ in this example can be replaced with other functions, while the similar processes will still be true. Both in the Partially Homomorphic Encryption and Fully Homomorphic Encryption, the proof problem can be constructed with this method.

Note that in the syllogism, there exists an interaction between the two parties in the whole process. However, in reality, it's relatively expensive. For example, each node in the blockchain network will verify the contents of some zero-knowledge proofs when verifying blocks. In this situation, the node that generates the proof must not be expected to be always online and communicate with all the nodes that verify the proof.

Hopefully that the prover can generate the challenges by itself and give it to the verifier, and then the verifier finds time to look at it when it's free. As a result, non-interactive zero-knowledge proof is needed here.

D. Non-Interactive Zero-Knowledge Proof

Typical zero-knowledge proof requires interactions between peers, the proving and validating process are implemented via the interactions. Despite the fact that this is the more commonly known zero-knowledge proof, it seems to be inadequate when facing more than two parties, when the prover needs to repeat the same process over and over again. As a result, we consider the typical zero-knowledge proof as interactive zero-knowledge proof, and the other considered as non-interactive zero-knowledge proof.

Manuel Blum, Paul Feldman, and Silvio Micali proposed the first non-interactive zero-knowledge proof where the prover and verifier have a shared key [7]. Unlike interactive proofs, non-interactive proofs require only one round of interaction between prover and verifier, using an intermediate algorithm to set the secret information.

Then another problem arises. If the challenge is chosen by the prover at will, then he can know it in advance, which gives the prover room for falsification. Therefore, we must not only allow the prover to generate, but also make it impossible for the prover to predict.

The Fiat-Shamir Heuristic method seems to solve this problem [8]. Its idea is that the prover will make a hash of all the known information including the disclosure material, and use the hash value to generate the proof, together sent to the verifier. The verifier also verifies that the hash value is correct.

E. ZK-SNARK and ZK-STARK

The ZK-SNARK is actually an example technology using non-interactive zero-knowledge proof, short for "Zero-Knowledge Succinct Non-interactive Argument of Knowledge". At the same time, the application of ZK-SNARK on Ethereum is also one of the topics of attention in the blockchain field.

1) Main idea

The first thing needs to know is that any finite program can be transformed into a polynomial problem, which can be univariate or multivariate [9]. After that, ZK-SNARK uses means such as homomorphic commitments to enable the prover to prove that he holds the solution to the polynomial problem without exposing the original data. The proof process is still based on the previous syllogism.

This technology is usually used for the verification of smart contracts in the blockchain, that is, the prover needs to prove that its input can indeed produce a given output through the smart contract.

2) Architecture

a) Key Generator: As its name, the key generator generates a public private key pair based on some trusted sources, and then destories the private key, uses the public key to generate another key pair, one for proving and the other for validating.

b) Prover: Prover uses the generated proving key and some public information to prove the statements.

c) Verifier: Verifier uses the generated validating key and provements to verify the statements.

The ZK-STARK is short for "Zero-Knowledge Scalable Transparent Arguments of Knowledge", which is quite similar with ZK-SNARK, except for some other characteristics [10]. ZK-STARK uses publicly verifiable randomness to create trustless verifiable computation systems, also more scalable in terms of computational speed and size. However, it requires more on proofs, resulting in higher costs when calculating.

IV. Drawbacks

A. Hardware Costs

The basic process of a zero-knowledge proof involves relatively complex computations. It will perform better if using dedicated machines, which may result in costs increase passing on to end users.

B. Gas Costs

The verification process on blockchains also requires complex computations, the difference with hardware costs is that the costs here are on-chain paid in the form of gas fees. For example, ZK-rollups require around 500,000 gas to verify a single ZK-SNARK proof on Ethereum, while ZK-STARK requires even more.

C. Quantum Computing Threats

The ZK-SNARK uses Elliptic Curve Cryptography (ECC) for encryption. It is currently secure anyway, however, future developments in quantum computers could break its security basis.

While ZK-STARK is considered immune to the threat of quantum computing because it uses collision-resistant hashing for encryption. Unlike the public-private key pairs used in ECC, collision-resistant hashes are more difficult to break by quantum computing algorithms.

V. Applications

In recent several years, the academia has "witnessed" the transition of zero-knowledge proof from theoretical to applied. However, when facing practical applications, figuring out where the best place is to use it probably becomes the hardest point [11].

A. Application Scenarios

1) Authentication

Zero-knowledge proof can help with authentication by maintaining an extra security channel, to efficiently avoid data leakage.

2) Messaging

Zero-knowledge proof can be used to encrypt the end-to-end messaging, so that no one can read the private messages except for the client itself, including the server.

3) Data Sharing

There always exists some risks when sharing data via the Internet, no matter how protective the server or platform claims to be, while zero-knowledge proof can deal with that without a third party's eye.

4) Storage Protection

Storage also deserves concerns. With zero-knowledge proof, not only the storage unit, but also the information inside, as well as the related access channels, are under well protection.

B. Notable Cases

1) Zcash

Zcash is an early blockchain-based payment system, also the first project to implement ZK-SNARK, providing an approach for secure authentication of clients to servers [12].

2) ZK-Rollups

Since the birth of blockchain, it has always faced an efficiency problem: low TPS (Transactions Per Second). For example, Bitcoin only supports 7 transactions per second, while for Ethereum it's only around 15. Such a low TPS is far from adequate for large applications. There are two main types of scaling solutions:

a) Layer 1 scaling: directly increases the capacity on the blockchains, common technical solutions are sharding (e.g., Near Protocol) and DAG (e.g., Conflux Network).

b) Layer 2 scaling: transfers a considerable part of the workload from on-chain to off-chain, see Fig.2. Common technologies include Plasma and the popular ZK-Rollup here.

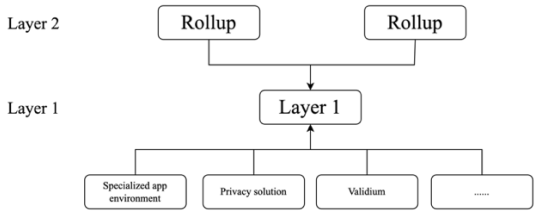

Fig. 2. Structure of layer 2 scaling solutions

The key point of ZK-Rollup is to compress the on-chain user state, store it in a Merkle tree, and transfer the change of the user state to the off-chain. The cost of directly processing on-chain is relatively high, but that of verifying whether a zero-knowledge proof is correct via smart contracts is relatively low. The representative project of ZK-Rollup is zkSync, which launched its version 2.0 network on October 2022.

3) FairTraDEX

On November 2022, in the field of game theory and mechanism design of decentralized finance (DeFi), Conor McMenamin and his colleagues presented FairTraDEX, a decentralized exchange (DEX) protocol based on frequent batch auctions (FBAs), providing these game-theoretic guarantees using zero-knowledge proof [13].

C. Application Bottleneck

The main bottleneck tends to lie on the prover, where large scale of computations is made. It'll be quite interesting if making the prover's job split up into lots of parallel computations [14], exploiting the power of the cloud to enable zero-knowledge proof, in a distributed way.

REFERENCES

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-system", STOC '85: Proceedings of the seventeenth annual ACM Symposium on Theory of Computing, pp. 291–304, December 1985.

[2] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity, or all languages in np have zero-knowledge proof systems", Journal of the ACM, vol. 38, pp. 691-729, July 1991.

[3] J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, and S. Guillou, "How to Explain Zero-Knowledge Protocols to Your Children", Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 628-631, August 1989.

[4] R. Gradwohl, M. Naor, B. Pinkas, and G. N. Rothblum, "Cryptographic and Physical Zero-Knowledge Proof Systems for Solutions of Sudoku Puzzles", 44: 245-268, Feburary 2009.

[5] R. Cramer and I. Damgs, "Zero-Knowledge Proofs for Finite Field Arithmetic, or: Can Zero-Knowledge Be for Free?", Advances in Cryptology - CRYPTO '98, pp. 424-441, August 1998.

[6] C. P. Schnorr, "Efficient Signature Generation by Smart Cards", Journal of Cryptology, 4 (3): 161-174, January 1991.

[7] M. Blum, P. Feldman, and S. Micali, "Non-Interactive Zero-Knowledge and Its Applications", STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing, pp. 103–112, January 1988.

[8] D. Bernhard, O. Pereira, and B. Warinschi, "How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios", Advances in Cryptology - ASIACRYPT 2012, October 2012.

[9] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge", CRYPTO 2013, Part II, LNCS 8043, pp. 90–108, August 2013.

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

[11] O. Goldreich, "Zero-knowledge twenty years after its invention", Electronic Colloquium on Computational Complexity (ECCC), (063), July 2002.

[12] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized Anonymous Payments from Bitcoin", 2014 IEEE Symposium on Security and Privacy, pp. 459-474, May 2014.

[13] C. McMenamin, V. Daza, M. Fitzi, and P. O'Donoghue, "FairTraDEX: A Decentralised Exchange Preventing Value Extraction", DeFi'22: Proceedings of the 2022 ACM CCS Workshop on Decentralized Finance and Security, pp. 39-46, November 2022.

[14] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation", In Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (STOC '07), June 2007.