State of the Art of Zero-Knowledge Proofs in Blockchain

Darko Čapko
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
dcapko@uns.ac.rs

Srđan Vukmirović
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
srdjanvu@uns.ac.rs

Nemanja Nedić
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
nemanja.nedic@uns.ac.rs

2022 30th Telecommunications Forum (TELFOR) | 978-1-6654-7273-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/TELFOR56187.2022.9983760

***Abstract*— Blockchain technology promises a decentralized ecosystem for building apps with decentralized ownership and high security level. One of the most adopted blockchain ecosystems is based around Ethereum blockchain. Ethereum currently has a huge scaling issue, making its use very expensive. Different scaling solutions are proposed based on Optimistic rollups but they have a very long finality time. Zero-Knowledge (ZK) rollup (based on ZK Proof) is another way and offers high security with very fast final resolution. Problem with ZK rollup is that it is very computationally expensive, but some solutions have significant performance improvements. In this paper we presented different solutions that improve performance of ZK rollup.**

Keywords*— *blockchain, zero-knowledge proof, optimistic rollup, zero-knowledge rollup

I. Introduction

Blockchain technology had a meteoric rise in the last decade and more and more applications (called distributed applications - DApps) are emerging almost every day. In the past few years technologies have become more mature and now are ready to accept millions of users and billions of transactions daily.

The blockchain concept was proposed in [1] as a model for executing financial transactions that allows a significant reduction in the cost of transactions from one account to another. He was the basis for the creation of the first blockchain system and cryptocurrency Bitcoin. The following years led to a revolutionary upgrade of blockchain applications and the implementation of smart contracts [2]. Smart contracts are simply programs that are stored on blockchain and execute the actions for transactions with predetermined conditions.

One of the most important ecosystems in blockchain today is based on the Ethereum blockchain and applications that use the Ethereum development stack. Ethereum [3] is the first and by far the most used blockchain that has Turing-complete capabilities. The advantage of Ethereum over Bitcoin is that it allows the execution of smart contracts. Biggest challenge for Ethereum growth is scalability, currently limited scalability results in high prices of transactions[3]. One way of increasing scalability in the process reducing price is the use of ZK proof (ZKP).

There are a number of client applications that use the Ethereum blockchain system. Some of them are client distributed applications for performing simple transactions or smart contracts on the Ethereum blockchain, but most applications that use the Ethereum development stack create their own blockchain and provide services for other client applications. This approach will be explained in more detail in the next chapter.

II. Blockchain architecture

Blockchain is a distributed system where a unique blockchain model is stored in system nodes. This implies the need for all functionalities of efficient distributed systems: data replication, consistency, fault tolerance, communication, time synchronization, etc.

The main purpose of the blockchain system is to execute transactions and smart contracts for which users pay certain financial fees. Costs per transaction depend on the executed operations on the blockchain system and the amount of used resources (primarily memory usage).

An important aspect of this design is checking the validity of a particular transaction (checking the existence of an account, account balance, etc.). Since this is confidential data, cryptographic algorithms are used for both data protection and validation. The basic concepts of these cryptographic algorithms are based on hashing, Merkle trees and consensus algorithms [4],[5].

Ethereum Virtual Machine (EVM) [3] is part of the Ethereum blockchain system that handles smart contract deployments and execution. EVM has a stack-based architecture and it is a computation engine that provides a computation and storage for specific instruction sets (logging operations, execution, memory and storage access, control flow, logging, colling, etc.). The role of EVM is to update the Ethereum state after applying actions of a smart contract.

Since the number of transactions executed on the Ethereum blockchain is limited (~15 transactions per second(TPS)) and the transaction price is relatively high, there is a need to optimize such a system. An efficient solution that speeds up transaction execution (~5000-10000 TPS) and reduces costs is a multilevel system [6]. In that case, the Ethereum blockchain is on the first level - L1, and in its environment there are second level L2 blockchains. Within the L2 system, transactions are inserted into L2 blocks, after which transactions from several L2 blocks (usually 100) are packed in the batch. The batch is forwarded to the L1 chain as one Smart Contract which should be inserted into the L1 block.

In this way, there is a connection between the L1 and L2 chains, where each executes its own transactions. The concept of data protection and validation is based on keeping the balance of all accounts (user accounts from L2 and L1) on the Ethereum L1 blockchain and updating them after executing transactions. After importing the transaction in layer L2, the

Coordinator collects together L2 transactions in a batch. Afterwards, Prover creates the proof for that batch of data and the proof should be validated by the verifier. The speed of execution of transactions in the multilevel blockchain system depends on the efficiency of this process.

Two concepts of transaction validation have been proposed[7]:

- Optimistic rollup,

- Zero-knowledge (ZK) rollup.

Optimistic and ZK rollup use the Ethereum blockchain for data storage instead of computation. Optimistic rollup is based on a rollup contract that keeps state transition history. If any validator discovers that the state transition is incorrect, all transactions in the batch are invalid (Fraud proof). ZK rollup is based on the principle that the validator verifies the proof posted with the batch (Validity proof). By using the ZK rollup mechanism, it is possible to cheaper and efficiently apply batch transactions from L2 chain to the Ethereum (L1) chain with high security. Some examples of Optimistic rollups are Arbitrum[8], Boba Network, Optimism[9], and for ZK rollups are ZKSync, Aztec, ZKSwap, Loopring, StarkEx, zk.money and Hermez.

III. Zero-knowledge proof algorithm

Zero-knowledge proof is a two-party protocol where one party (*Prover*) produces *Proof* that can convince another party (*Verifier*) that some statement is true without revealing any additional information to the verifier. This concept is introduced in the paper [10], and in the beginning is used for the signature scheme[11] and ciphertext security [12]. Authors in [13] demonstrated wide applicability of ZKP, such as cryptography, number theory and graph isomorphism. In addition, these researchers gave a new aspect of the problem in which a verifier can also be malicious.

A zero-knowledge system should satisfy conditions for interactive proof systems [14]: *completeness* and *soundness*. ***Completeness*** means that honest verifier V always accepts a true statement after interacting with honest prover P. ***Soundness***: in case of false statement, the honest verifier V rejects proof of cheating prover with high probability.

In addition, the basic condition is ***zero-knowledge***: Prover P does not provide any additional information to the verifier V other than the fact that the statement is true.

There are basically two types of ZKP algorithms: interactive and non-interactive.

Interactive ZKP requires interaction between the Prover and the Verifier during validation proof.

Non-interactive ZKP algorithms are much more interesting for blockchain applications and it is used for ZK rollups.

A. Non-Interactive ZKP

In general, the non-interactive ZKP can be divided into three phases:

1. Setup
2. Compute
3. Verify

Setup uses computation functions for conversion of security parameters into some common knowledge (available for both Prover and Verifier), usually encoded in a Common Reference String (CRS) [15]. That is the way to compute Proof and verify it with the correct parameters and algorithms.

Compute takes computation functions, inputs and proving keys and gives output of calculation and proof.

In the Verify phase the proof is validated by verification keys.

The most popular non-interactive ZKP are: SNARK, STARK and Bulletproof.

SNARK (Succinct Non-Interactive Arguments of Knowledge) [16] is a non-interactive ZKP that is succinct – proofs are very short and easy to verify. This protocol requires a trusted setup ceremony between two parties to verify transaction validity (proving key k_p to produce proof π and verification key k_v to verify the proof π). It uses elliptic curve pairings cryptography [17].

STARK (Scalable Transparent ARgument of Knowledge)[18] does not require trusted setup and uses simpler cryptographic algorithms (based on hashing and information theory). It uses the FRI[19] prover protocol with strictly linear arithmetic complexity and strictly logarithmic verifier arithmetic complexity. On the other side, proof size for SNARKs is much smaller than STARKs.

Bulletproof [20] is protocol without trusted setup and with very short proof. It improves STARKs by transforming interval proofs into vector inner product computation, which greatly reduces the complexity. However, verifying a bulletproof is more time consuming than verifying a SNARK proof.

Depending on the setup phase of the CPC, three classes of algorithms can be observed:

- Generation I (G1) – require a separate trusted setup for each circuit.
- Generation II (G2) – initially one setup for all circuits.
- Generation III (G3) - proof systems without trusted setups.

Examples of first generation algorithms are Pinocchio[21] and Groth16[22]. Groth16 is an efficient zk-Snark scheme which requires a trusted setup, an elliptic curve pairing and relies on strong assumptions. Growth [23] also proposed an algorithm scheme with a universal structured reference string (SRS) that allows a single setup (G3). Pinocchio is a pairing based proof system that uses Quadratic Arithmetic Program[24] for encoding computation and provides rapid proof verification.

PlonK[25] (Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge) is zk-Snark that the trusted setup should be initialized (as universal SRS) ones for all circuits (G2). Plonk protocol uses a batched version of the Kate polynomial commitment scheme (KZG) [26] similar to that used in [27].

Sonic[27] and Marlin[28] are zkSnarks where SRS is universal and updatable. In addition, SRS has linear size and arguments have constant size.

The most popular algorithms from generation G3 are Bulletproofs[20], STARKs[18], Spartan[29] and RedShift[30]. Bulletproofs have linear verifier times, but rather short proofs[31]. RedShift has large proofs (up to 1 MB for millions of gates).

IV. Improvement Proposals

Optimization of the ZK rollup process can be done in several ways:

- Optimization of cryptographic algorithms calculations.
- Hybrid optimistic and ZK Rollup.
- Development of specialized ZK EVM [32].
- Hardware optimization.

Different solutions for this purpose are shown on Table 1.

TABLE I. ZK BASED SOLUTIONS FOR ETHEREUM SYSTEM IMPROVEMENT

Type of optimization	Solution	Used algorithms/compilers
Algorithm improvement	Polygon Zero[33]	Plonky2[33], Keccek256[35], FRI[19], Plonk[25]
Hybridization	Polygon Nightfall[36]	Nightfall[36]
ZK EVM	AppliedZKP[39]	Halo2[40], KZG[26], BN-254[41].
	zkSync [42]	ultraPlonk[40]
	Polygon Hermez[]	Plonk[25], KZG[26], Groth16[22]
	Sin7Y[38]	Halo2[40], KZG[26] Recursive Plonk[25]
	Polygon Miden[43]	STARK based ZK VM[18]
Hardware	DIZK[44]	Distributed zkSNARK[44]
	PipeZK [45]	POLY[45], MSM[45]
	PipeMSM[46]	PipeMSM[46]
	HardAcc-Groth16[47]	Groth16[22]
	CPU/GPUAcc - Bulletproof[48]	Bulletproof[20][48]

A. Optimization of cryptographic algorithms calculations

Polygon Zero (former MIR) proposed a decentralized ZK rollup – Plonky2 [33]. Plonky2 is a recursive SNARK that is 100x faster than other alternatives compatible with Ethereum. It is a combination of the best characteristics of both STARKs and SNARKs:

- Plonk [25] and FRI [19] - fast proofs and no trusted setup,
- support for recursion [34] and
- low verification cost.

Plonky2 requires keccak-256[35] to verify a proof. Efficiency of Plonky2 is realized by 64-bit recursive FRI in combination with Plonk.

B. Hybrid optimistic and zk rollup

Polygon Nightfall [36] is a hybrid rollup between optimistic and ZK focused on increasing transaction privacy and reducing transfer fees (up to 86%).

C. Development ZK EVM

One of the ways to improve the ZK rollup algorithm is to use a specialized ZK EVM [32],[37]. Several solutions have been proposed [38]:

- AppliedZKP[39] is an open source project funded by the Ethereum foundation which implements ZK for the native opcode of Ethereum EVM. It uses a few cryptographic algorithms: halo2[40], KZG[26] and Barreto-Naehrig (BN-254) elliptic curve pairing[41].
- zkSync [42]- Matter Labs zkEVM is a custom EVM that implements interpreter, compile the contract code into YUL (an intermediate language for Solidity compiler) and compile YUL into custom bytecode supported zkEVM. It uses the extension of Plonk[25] – ultraPlonk[40].
- Polygon Hermez[43] is a custom EVM Compatible decentralized rollup that compiles the contract code into the micro instruction set supported by uVM. It uses Plonk[25], KZG[26] and Groth16[22] proving systems.
- Sin7Y[38] zkEVM implements zk for native opcode of EVM and optimizes specialized opcode. It uses halo2[40], KZG[26] and RecursivePlonk.
- Polygon Miden[44] is a generic STARK-based[18] zero-knowledge virtual machine.

D. Hardware optimization

DIZK (DIstributed Zero Knowledge) [45] is a system that distributes the execution of zkSNARK proof across a compute cluster. In [46] authors propose hardware using an architecture consisting of two subsystems: one for the polynomial computations (POLY) with large-size number theoretic transforms (NTTs), and the other for the multi-scalar multiplications (MSM) that execute vector inner products on elliptic curves (ECs). PipeMSM[47] is pipelined designed MSM algorithm for implementation on FPGA.

In the paper [48] authors propose a design of a hardware accelerator based on FPGA for ZKP. Their proposition consists of:

- multiple FFT (Fast Fourier Transform) units and decomposition of FFT operation,
- multiple MAC (Multiply and Accumulate Circuit) units combined of adders and multipliers,
- multiple ECP (Elliptic Curve Processing) units to reduce the computational overhead,
- design of zk-SNARK based on FPGA that reduced prover time 10x.

Hardware acceleration of the Bulletproof protocol is proposed in [49] by a CPU-GPU collaborative framework and parallel Bulletproofs on the GPU.

V. Conclusion

Trust in the use of the blockchain system depends on the reliability and security of the system, as well as the price and speed of executing transactions. Therefore, investing in the use of ZK algorithms and their optimization is currently considered one of the biggest challenges in the blockchain community. Companies that continuously invest in the research and development of their blockchain ecosystems gain a dominant role in the blockchain world.

Acknowledgment

We would like to thank our associates at Ethernal, Novi Sad, Serbia for their help during our work on this research.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decentralized Business Review , 21260, 2008.

[2] N. Szabo, "Smart contracts: building blocks for digital markets," EXTROPY: The Journal of Transhumanist Thought,(16) 18, no. 2, 1996.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper 151, pp 1-32, 2014.

[4] E. Buchman, K. Jae, Z. Milosevic, "The latest gossip on BFT consensus," arXiv preprint arXiv:1807.04938, 2018.

[5] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," Ruhr-University Bochum, Tech. Report, 2008.

[6] G. Kaempfer, "Fractal Scaling: From L2 to L3," StarkWare, 2021

[7] Buterin, Vitalik. "An incomplete guide to rollups.", 2021., Retrieved from https://vitalik.ca/general/2021/01/05/rollup.html

[8] Arbitrum – Optimistic rollup. 2021. Retrieved from https://github.com/OffchainLabs/arbitrum

[9] Optimism – Optimistic rollup. 2021. Retrieved from https://github.com/ethereum-optimism/optimistic-specs

[10] S. Goldwasser, S. Micali, C. Rackoff, "The knowledge complexity of interactive proof systems," SIAM J. Computing, vol. 18, no. 1, pp. 186–208, 1989.

[11] D. Pointcheval, J. Stern, "Security proofs for signature schemes" In EUROCRYPT, May 1996.

[12] A. Sahai, "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security," In FOCS, Oct. 1999.

[13] O. Goldreich, S. Micali, A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," ACM journal, vol. 38, no.3, pp. 690–728, 1991.

[14] O. Goldreich, "A Short Tutorial of Zero-Knowledge," Secure Multi-party Computation, IOS Press, pp. 28-60, 2013.

[15] Benarroch, D. "Diving into the zk-SNARKs Setup Phase,", 2019., Retrieved from https://medium.com/qed-it/diving-into-the-snarks-setup-phase-b7660242a0d7

[16] E. Ben-Sasson, Eli, A. Chiesa, D. Genkin, E. Tromer, M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge." In Annual cryptology conference, pp. 90-108. Springer, Berlin, Heidelberg, 2013.

[17] K. G. Paterson, "ID-based signatures from pairings on elliptic curves," Electronics Letters , vol. 38, no. 18, pp. 1025-1026, 2002.

[18] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," IACR Cryptol. ePrint Arch. , 2018.

[19] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, "Fast Reed-Solomon interactive oracle proofs of proximity," (2nd). Electronic Colloquium on Computational Complexity (ECCC), vol. 24, no.134, 2017.

[20] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more" In IEEE Symposium on Security and Privacy, IEEE Computer Society, pp. 315–334. 2018.

[21] B. Parno, J. Howell, C. Gentry, M. Raykova, "Pinocchio: Nearly practical verifiable computation," In 2013 IEEE Symposium on Security and Privacy, pp. 238-252, 2013.

[22] J. Groth, "On the size of pairing-based non-interactive arguments," In Annual international conference on the theory and applications of cryptographic techniques, pp. 305-326. Springer, Berlin, Heidelberg, 2016.

[23] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, I. Miers. "Updatable and universal common reference strings with applications to zk-SNARKs," In Annual International Cryptology Conference, pp. 698-728. Springer, Cham, 2018.

[24] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in EUROCRYPT, 2013.as Cryptology ePrint Archive, Report 2012/215.

[25] A. Gabizon, Z. J. Williamson, O. Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge," IACR Cryptol. ePrint Arch, 953. 2019.

[26] K. Aniket, G. M. Zaverucha, I. Goldberg, "Constant-size commitments to polynomials and their applications," In International conference on the theory and application of cryptology and information security, pp. 177-194. Springer, Berlin, Heidelberg, 2010.

[27] M. Maller, S. Bowe, M. Kohlweiss, S. Meiklejohn, "Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings," In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2111-2128. 2019.

[28] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, N. Ward, "Marlin: Preprocessing zkSNARKs with universal and updatable SRS," In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 738-768. Springer, Cham, 2020.

[29] S. Setty, "Spartan: Efficient and general-purpose zkSNARKs without trusted setup," In Annual International Cryptology Conference, pp. 704-737. Springer, Cham, 2020.

[30] A. Kattis, K. Panarin, A. Vlasov, "Redshift: Transparent snarks from list polynomial commitment IOPs," Cryptology ePrint Archive, Report 2019/1400, 2019. Retrieved from https://eprint.iacr.or g/2019/1400.

[31] A. Asher, B. Brennan, Zero-Knowledge Proofs: STARKs vs SNARKs, 2021, Retrieved from https://consensys.net/blog/blockchain-explained/zero-knowledge-proofs-starks-vs-snarks/

[32] O. Bégassat, A. Belling, T. Chapuis-Chkaiban, N. Liochon. "A specification for a ZK-EVM,", 2021. Retrieved from https://ethresear.ch/uploads/short-url/3DM8kjFfIG6PHXu4qpYpmujXgme.pdf

[33] Polygon Zero. 2021, Retrieved from https://github.com/mir-protocol/plonky2

[34] S. Bowe, J. Grigg, D. Hopwood, "Recursive proof composition without a trusted setup," Cryptol. ePrint Arch., Tech. Rep 1021, 2019.

[35] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, "The Keccak SHA-3 submission," Submission to NIST (Round 3), 2011.

[36] Polygon Nightfall. 2021., Retrieved from https://blog.polygon.technology/zk-proofs-protocol-polygon-nightfall-launches-on-testnet-to-provide-low-cost-private-ethereum-transaction/

[37] L. Goldberg, S. Papini, M. Riabzev, "Cairo – a Turing-complete STARK-friendly CPU architecture," Cryptology ePrint Archive, Report 1063, 2021.

[38] Syn7y, "Exploring Popular zkEVM Solutions: AppliedZKP, Matter Labs, Hermez, and Sin7Y," 2021. Retrieved from hackernoon.com

[39] Applied ZKP. 2021, Retrieved from https://github.com/appliedzkp

[40] D. Hopwood, S. Bowe, J. Grigg, K. Nuttycombe, Y. Tong Lai, S. Smith, "The halo2 Book", Retrieved from https://zcash.github.io/halo2/

[41] P. S. L. M. Barreto, M. Naehrig, "Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, Selected Areas in Cryptography," 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers, vol. 3897 of Lecture Notes in Computer Science, pp. 319–331. Springer, 2005.

[42] zkSync - Matter Labs. 2021., Retrieved from https://github.com/matter-labs/zksync

[43] Polygon Hermez. Releasing the Initial Polygon Hermez 2.0 zkEVM Documentation, Polygon Hermez documentation. 2022

[44] Polygon Miden. 2021., Retrieved from https://github.com/maticnetwork/miden

[45] H. Wu, W. Zheng, A. Chiesa, R. Ada Popa, I. Stoica, "DIZK: A distributed zero knowledge proof system," In 27th {USENIX} Security Symposium ({USENIX} Security 18), pp. 675-692. 2018.

[46] Y., Ye, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, G. Sun, "PipeZK: accelerating zero-knowledge proof with a pipelined architecture," In 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), pp. 416-428. IEEE, 2021.

[47] Xavier, Charles F. "PipeMSM: Hardware Acceleration for Multi-Scalar Multiplication." Cryptology ePrint Archive, 2022.

[48] B. O., Peng, Y. Zhu, N. Jing, X. Zheng, Y. Zhou, "Design of a Hardware Accelerator for Zero-Knowledge Proof in Blockchains," In International Conference on Smart Computing and Communication, pp. 136-145. Springer, Cham, 2020.

[49] Y. Huang, X. Zheng, Y. Zhu, X. Kong, X. Jing, "CPU-GPU Collaborative Acceleration of Bulletproofs-A Zero-Knowledge Proof Algorithm," In 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), pp. 674-680. IEEE, 2021.