Implementation and preliminary evaluation of an auditable confidentiality mechanism for DeFi

AoXuan Li*‡, Su-Kit Tang*, Gabriele D'Angelo†
*Faculty of Applied Sciences, Macao Polytechnic University, Macao SAR, China
†Department of Computer Science and Engineering, University of Bologna, Italy
‡Mystiko.Network
{aoxuan.li,sktang}@mpu.edu.mo, g.dangelo@unibo.it, douglas@mystiko.network

***Abstract*—A well-known use of the blockchain technology is Decentralized Finance (DeFi). DeFi makes financial information accessible to the public but raises potential privacy and security issues. In this study, we implemented a DeFi protocol that protects privacy, which is based on the Mystiko.Network protocol. As a proxy between the user and DeFi platforms, the Mystiko.Network protocol offers an auditable confidentiality mechanism for blockchain transactions. Via the new system, users may submit anonymous DeFi transactions and get income back into a shielded tokens pool. Moreover, we implemented a rollup approach to handle anonymous DeFi transactions in groups. The evaluation results suggest that the protocol is both practical and affordable, in fact it is able to save around 90% of the cost for DeFi transactions.**

***Index Terms*—DeFi, Privacy, Zero-knowledge Proof**

I. Introduction

The most well-known blockchain application, called Decentralized Finance (DeFi), permits peer-to-peer trading, exchanging, and staking through a collection of smart contracts. As of December 23, 2022, the DeFi sector's total locked value was close to $69.95 billion [1]. DeFi exposes user transactions, including identification and account balance, to the public, which might represent a privacy and security risk. This is different from standard finance apps. Anonymity and confidentiality [2] are the two basic concepts that are frequently utilized when discussing transaction privacy. Anonymity involves keeping the user's identity a secret, whereas confidentiality means keeping the transaction's financial details a secret. The smart-contract-based DeFi apps, which need the specification of a transaction value, make confidentiality impossible [3]. An automatic market maker, for instance, needs the value to decide on the exchange rate. As a result, we anticipate achieving DeFi transactions that protect user privacy and anonymity.

The adoption of privacy-preserving payment protocols gave rise to the idea of decentralized anonymous payment (DAP) systems, which used a shielded asset pool to conceal ownership [4]–[7]. In Zerocash [4], to spend tokens, the owner uses zero-knowledge proof, which exposes nothing other than the truth of the assertion, to demonstrate the ownership and existence of the tokens in the pool. The token cannot be deanonymized by an attacker and the transaction link cannot be restored. In their paper [8], Li et al. extended the Mystiko.Network [6] system and added a proxy that connects the external DeFi platforms to the current DAP systems. The solution enables users to initiate DeFi transactions and receive income back into their pool of protected assets.

Mystiko.Network is an auditable confidentiality protocol for blockchain transactions. Li et al. solved two major hurdles in order to extend the protocol and accommodate DeFi transactions: unpredictable output amounts and high transaction fees. In contrast to Mystiko.Network, which demands that the total output amount be known and equal to the input amount, a DeFi transaction, such as a trade on Uniswap, has no predetermined output amount when the user initializes the transaction. Li et al. developed partial tokens, an intermediate state of shielded tokens with certain ownership but uncertain value, to address this problem. Also, a user must pay a gas fee for each DeFi transaction, which can be costly when there are many transactions. The rollup approach was used by Li et al. to lower the cost of the protocol. The rollup system compiles a number of transactions into a single one and uploads it to a blockchain that is accessible online, such as Ethereum.

In [8], the authors proposed the overview of the system, but they did not describe the concrete design of the system. In this paper, we instantiate the cryptography primitives from [8] with practical algorithms and then evaluate the resulting performance.

A. Previous Works

Zerocash [4] is the first privacy-preserving blockchain employing zero-knowledge proof. The sender conceals the transaction data in a commitment, while the receiver offers a zero-knowledge proof demonstrating that the issuer holds the commitment. A Merkle tree is used to organize the commitments. Using Groth16 architecture, Zexe [9], Zkay [10], and ZeeStar [11] expand the privacy-preserving foundation to any smart contract. Zexe does not offer any development tools, instead requiring users to calculate smart contracts off-chain and then submit a zero-knowledge proof for accuracy. Zkay and its follow-up work, Zeestar, suggested a language for smart contracts that protect privacy. Yet those protocols demand a lot of computing power, such as a lot of CPU cores (Zkay, 12 cores), or a lot of memory (Zexe, 256 GB of RAM).

Another Bitcoin-based blockchain that protects anonymity is Monero [7], although unlike Zerocash, Monero makes use

This work is supported in part by the research grant (No.: RP/ESCA-04/2020) offered by Macao Polytechnic University.

of ring signatures and range-proofs [12]. Certain kinds of zero-knowledge proofs called range-proofs demonstrate that a value is within a range. For instance, in order to prevent overflow, users of Monero must demonstrate that the inputs and outputs of their transactions are inside a legal range. Zether [13] is a project built on Ethereum that is comparable to Monero, which is the first construction to update a user's balance via homomorphic addition. The size of range-proofs is not constant, hence scaling is fundamentally constrained for both Monero and Zether.

It should be noted that earlier projects similar to Zerocash and Monero only supported secret transactions on one chain. Zerocross [14] is a cross-chain solution for Monero that protects privacy and makes use of a sidechain and zero-knowledge proofs for set membership [15]. The sender and the recipient cannot maintain anonymity from one another, and their work only permits cross-chain transactions using Monero. P2DEX [16] is a safe multiparty computation-based, privacy-preserving exchange (MPC) [17]. Due to MPC's high cost as a cryptographic primitive [18], P2DEX is unable to handle high frequency trading. Some efforts [19], [20], which are unfeasible, demand that all users use specialized hardware (TEEs). TEEs can also be seriously vulnerable [21], [22].

All of the previously mentioned suggested solutions lack auditability and are vulnerable to unauthorized usage. Using non-interactive zero-knowledge proof and homomorphic commitments, ZkLedger [23] and Fabzk [24] enable auditability. Nevertheless, because these systems prioritize cross-organizational transactions and their efficiency suffers as user counts rise, they are unsuitable for handling large chains. ZEBRA [25] is a privacy-preserving blockchain for an auditable anonymous credential system, but ZEBRA does not support DeFi applications.

A secure system for blockchain transactions that supports well-known chains is called Mystiko.Network [6]. Mystiko.Network has a shielded tokens pool, like Zerocash, but it also offers single-chain and cross-chain private payment using an inter-chain data bridge. The Groth16 [26] system and a ZK-rollup method were used by Mystiko.Network to process transactions in the batch, which increased throughput and reduced the cost of the protocol. The protocol also permits the audits of private transactions.

Using a layer-2 blockchain and a bridge connecting it to the layer-1 DeFi platform, Aztec [27] addresses the privacy issue with DeFi. The layer-2 blockchain allows for private payments utilizing Plonk [28] and shielded assets. Plonk just needs a single universal setup, as opposed to Groth16, which requires a trusted setup for each assertion to be proved. Yet, Groth16 outperforms Plonk in terms of concrete efficiency. A rollup miner is also used by Aztec to batch DeFi transactions, but Aztec's miner must produce a zero-knowledge proof, adding to the protocol's expense. The rollup procedure in [8], however, does not need zero-knowledge verification. Furthermore, [8] enables users to spend plain tokens directly, unlike Aztec, which requires users to first transfer assets to the shielded tokens pool before using them for DeFi transactions.

Flex [2] is an addition to the ERC20 standard [29], which serves as the common interface for exchanging different Ethereum tokens. Although ERC20 serves as the foundation for DeFi apps, anonymous transfer is not natively supported by the standard at this time. Flex demonstrates how to create composable DeFi apps while protecting privacy by demonstrating how to create an anonymous token standard from pre-existing DAPs [4], [7], [13], [30]–[32]. Flex, however, needs to be modified or it is incompatible with current DeFi apps. Instead, [8] might be implemented in DeFi apps that already exist. Also, [8] has better compatibility, as it works with DeFi apps on blockchains other than Ethereum.

While the previous paper provided an overview of the system, this contribution focuses on instantiating the cryptography primitives from the previous work using practical algorithms. The resulting performance of the system is evaluated, presumably to assess its effectiveness and feasibility in real-world scenarios.

B. *Paper Organization*

This paper is organized as follows. In Section II, we introduce the background on blockchain, zero-knowledge proofs, and the other needed building blocks. Section III provides an overview of the proposed architecture and of the implementation. Section IV details the performance of the protocol, and in Section V, we discuss the conclusion and future works.

II. Background

In this section, we introduce the background on blockchain, zero-knowledge proofs, and the other needed building blocks.

A. *Blockchain*

A distributed ledger shared by a network of computers is referred to as a blockchain. Bitcoin [33], which uses a blockchain to record peer-to-peer transactions, is considered to be the first generation of blockchain technology. Smart contracts were introduced into the blockchain by Ethereum [34]. Decentralized applications on blockchains are made possible via smart contracts, which are computer programs on the blockchain that execute themselves when specific circumstances are met. All nodes in the network must concur on the current status as the blockchain expands and attach the same block. The blockchain defines a group of protocols called consensus algorithms to guarantee consistency among nodes.

B. *Commitment Scheme*

A commitment scheme, a cryptographic primitive, is widely used in various advanced protocols, e.g., zero-knowledge proofs and multiparty computations. The notion of *Commitment* means a sender selects a value from a finite set and commits the selection such that he/she cannot change his mind later. The sender may send the commitment to a receiver. A secure commitment scheme requires that the committed value is hidden from the reviewer, which is called the *hiding property*. Moreover, the sender cannot open the commitment to values other than the original one, which is called the *binding property*. There are many realizations of the scheme, and a simple example of a commitment scheme is based on the RSA cryptosystem [35], [36].

C. Zero-Knowledge Proof

The concept of zero-knowledge proof was first developed in 1985 by Goldwasser, Micali, and Rackoff [37]. It enables a prover to establish the veracity of a claim without disclosing anything other than the claim itself. For example, the prover may prove that two graphs are isomorphic without revealing anything, especially the isomorphism between the two graphs. In addition, Blum, Feldman, and Micali [38] developed the non-interactive zero-knowledge proof, which allows for public verification of the proof without involving the prover. The early zero-knowledge proof protocols, however, are redundant or inefficient in terms of proof size; as a result, those protocols are not practicable. One of the earliest workable protocols is the zk-SNARK algorithm [39], [40], which stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge. The proof size in zk-SNARK is brief and unrelated to the complexity of the statement. We used the Groth16 construction [26], [41] of the zk-SNARK in this study since it offers the best concrete efficiency and shortest proof size. Zk-SNARK generates proofs in three phases:

1) For a language L in NP, Prover and Verifier jointly compute the public parameters and the SNARK.
2) Prover generates a proof π with the instance x and the witness w.
3) Verifier can check the validity of π and x.

D. Merkle Tree

Every non-leaf node in a Merkle tree [42], also known as a hash tree, is labeled with the hash value of each of its child nodes' labels. The root hash is at the top of the tree, and there is an authentication path from each leaf node to the root. It enables effective membership verification in a sizable data set. This 1 is an illustration of a Merkle tree with labels l_1, l_2, l_3, l_4 and a root hash of rt. All that is required to confirm the label l_1's membership is H_2 and H_6.

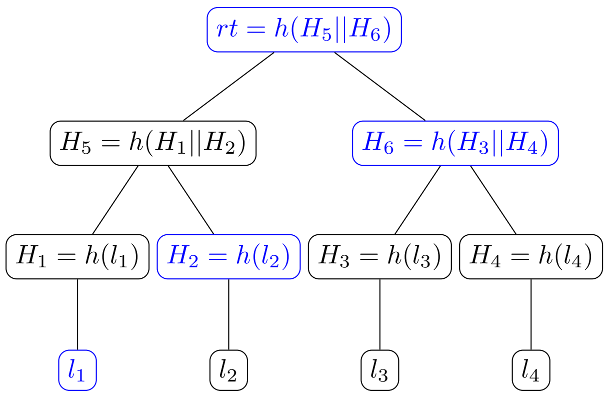

Fig. 1. Merkle Tree.

III. Implementation

In this section, we introduce the proposed architecture as illustrated in Figure 2, and we describe its implementation. We employed the Groth16 proof system provided by ZoKrates [43], a toolbox for zkSNARKs on Ethereum. We developed smart contracts by using the Solidity language and initialized each cryptography primitives. Namely, we build statistically-hiding commitments from **Poseidon** [44] and Merkle tree from **Keccak** [45].

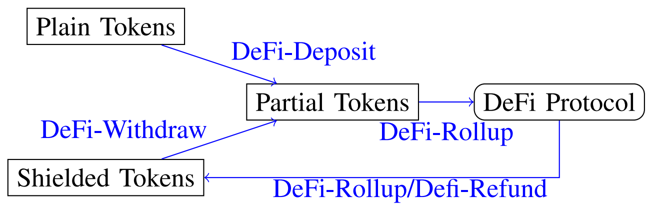

Fig. 2. Technical overview of the proposed approach.

The system provides three different token types: plain tokens, shielded tokens, and partial tokens. A set of smart contracts allows for **DeFi-Deposit**, **DeFi-Withdraw**, **DeFi-Rollup**, and **DeFi-Refund**. Plain tokens are regular cryptocurrencies with public ownership and value, such as Bitcoin and ETH. On the other hand, shielded tokens conceal ownership and value. The commitments of shielded tokens are organized in a Merkle tree, and shielded token owners can use zero-knowledge proof to confirm their ownership and spend the tokens. The value of tokens that are returned during DeFi transactions is not verified until after transactions are complete. The system offers partial tokens with clear ownership, albeit the value won't be known for a while.

Through a process known as **DeFi-Deposit**, a regular user can transfer plain tokens to a DeFi contract and get shielded tokens in return. The user may also utilize shielded tokens as inputs and get shielded tokens as income during **DeFi-Withdraw**. Fees associated with processing each DeFi transaction make them less affordable. **DeFi-Rollup**, in which a DeFi-Rollup miner transmits a batch of DeFi transactions to an external DeFi smart contract, allows the transaction fees to spread out among other users. In unlikely corner scenarios, the DeFi platform might not be available or the protocol could be unable to reach the platform. The money reimbursement in this instance is handled by the miners and is known as a **DeFi-Refund**. In particular, the funds are added to the pool of shielded tokens, and the new shielded token's ownership is the beneficiary of the DeFi transaction.

This system, inherited from Mystiko.Network, allows external auditors to review transactions. The system will not disclose users' transaction data unless a large enough partition of the auditors agrees so, which enables the auditability by threshold secret sharing of commitments and encryption of each shares. We instantiated the algorithms from **Shamir's secret sharing** [46] and **key-private Elliptic-Curve Integrated Encryption Scheme**.

A. DeFi-Deposit

During **DeFi-Deposit**, a user deposits plain tokens to a DeFi platform via their wallet, which creates partial tokens that are packed into a transaction and include partial commitments and zero-knowledge parameters. The protocol waits for DeFi-Rollup before adding the events to the appropriate queue. In particular, a queue is made up of operations to the same DeFi platform. The liquidity pool has sealed up the plain tokens. We implemented the smart contract in Solidity, which takes a DeFi address, the deposit amount, a partial commitment, and other essential information as inputs. If the inputs are valid, the partial commitment is enqueued while waiting for the DeFi-Rollup.

B. DeFi-Withdraw

Users can utilize **DeFi-Withdraw** to remove their assets from the shielded token pool and use them to fund DeFi. Users must employ zero-knowledge proof to demonstrate their ownership to withdraw tokens from the pool. The proof also demonstrates the existence of the commitments in the Merkle tree. Then, the user selects the DeFi platform and creates partial tokens using the wallet, which condenses crucial data into a single transaction. If the proof is legitimate, the protocol serves as the verifier and permits the user to withdraw the token. After that, while awaiting the DeFi-Rollup, the transaction will be enqueued. We implemented the proof and verification schema with the ZoKrates toolbox, which outputted a smart contract acting as the verifier. Inputs for the smart contract include a proof, the Merkle tree root, a serial number, the withdrawal amount, a partial commitment, and other essential information.

C. DeFi-Rollup

DeFi-Rollup can significantly lower transaction expenses. Before transmitting a DeFi transaction to the DeFi platform, the DeFi-Rollup miner combines DeFi transactions of the same kind into a single transaction, and the transaction cost is shared among many users. The rollup miner is in charge of calculating the token value and producing the shielded tokens in accordance with each partial token. The token value is specifically equal to (*user's deposit amount/batched deposit amount*) $\times$ *total revenue*. We implemented and tested the protocol with 1, 2, 4, 8, and 16 input DeFi transactions.

D. DeFi-Refund

A **DeFi-Refund** takes place only when the DeFi transaction fails. When the external DeFi platform is unable to meet the conditions, the DeFi rollup miner returns the money, which is then returned to the shielded tokens pool. The ownership and value are identical to those of the DeFi transaction's beneficiary and the initial input, respectively. We also implemented and tested the protocol with 1, 2, 4, 8, and 16 input DeFi transactions.

IV. Experiments

We implemented the proposed protocol by using Ganache [1], an Ethereum simulator, and then evaluated its performance by means of the Truffle Suite [2]. We employ the Groth16 proof system provided by ZoKrates [43], a toolbox for zkSNARKs on Ethereum. We developed smart contracts with the Solidity language.

There are three main phases in the system to be considered: DeFi-Withdraw, DeFi-Deposit, and DeFi-Rollup/DeFi-Refund. During each phase, the protocol submits a transaction to the blockchain and then executes a corresponding smart contract. During the DeFi-Withdraw phase, the protocol also computes SNARKs for withdrawing coins. As a methodology to collect and report the results from the experiments, each experiment has been repeated forty times and the collected results have been averaged. The reported equivalent USD cost was based on the gas price (about 40 gwei) and ETH price (about 1600 USD) at the time of writing (Feb 2023). The system environment used for running this performance evaluation is described in Table I.

TABLE I
System specification of the testing environment.

OS	CPU	RAM
macOS 13.2.1	Apple M1 Pro	16GB

A. Deposit

During the DeFi-Deposit experiment, a user submitted a defi-deposit transaction $tx_{defi-deposit}$ to the blockchain. The blockchain then locked the token and enqueued the transaction while waiting for the DeFi-Rollup. The experiment results are shown in Table II.

TABLE II
DeFi-Deposit experiment results.

	Gas amount	USD
DeFi-Deposit	25,297	1.61

B. Withdraw

During the withdrawal experiment, a user computed a witness and generated the ZK-snark proof. We evaluated the running time (in seconds) of each stage in the environment as shown in Table I. The user then submitted a DeFi-Withdraw transaction $tx_{defi-withdraw}$ to the blockchain. The time cost of each stage is reported in Table III. We evaluated the case when the user withdrew one shielded token and generated a partial token.

TABLE III
DeFi-Withdraw ZK-Snark experiment results.

	Compute Witness	Generate Proof	Verify
Time	0.19	2.395	< 0.01

We also recorded the required gas amount and the equivalent USD value of on-chain verification in Table IV.

[1]https://trufflesuite.com/ganache/

[2]https://trufflesuite.com/

TABLE IV
DEFI-WITHDRAW ON-CHAIN EXPERIMENT RESULTS.

	Gas amount	USD
DeFi-Withdraw	493,529	31.58

C. DeFi-Rollup

During **DeFi-Rollup**, the rollup miner generated shielded tokens for each partial token and submitted a DeFi-Rollup transaction $tx_{defi-rollup}$ to the blockchain. We evaluated different rollup types with 1, 2, 4, 8, 16 input partial tokens. The obtained results are reported in Table V.

TABLE V
DEFI-ROLLUP EXPERIMENT RESULTS WITH DIFFERENT ROLLUP TYPES.

	Type	Gas amount	USD
DeFi-Rollup	1	45,725	2.92
	2	67,046	4.29
	4	98,856	6.33
	8	162,535	10.4
	16	289,422	18.52

D. DeFi-Refund

When the DeFi transaction failed, the rollup miner refunded the tokens to the beneficiary and then submitted a DeFi-Refund transaction $tx_{defi-refund}$ to the blockchain. Also in this case, we evaluated different rollup types with 1, 2, 4, 8, 16 input partial tokens. The obtained results are reported in Table VI.

TABLE VI
DEFI-REFUND EXPERIMENT RESULTS WITH DIFFERENT ROLLUP TYPES.

	Type	Gas amount	USD
DeFi-Refund	1	51,271	6.08
	2	65,806	4.21
	4	95,030	6.33
	8	155,119	9.93
	16	274,838	17.59

E. Discussion

In this section, we reported the performance and gas consumption of our protocol. Generation of zero-knowledge proofs is the most time-consuming process, which takes up to 3 seconds when withdrawing tokens. The performance is practically acceptable considering other privacy-preserving payment protocols take much more time, e.g., 75 seconds for Zcash [3] and 120 seconds for Monero [4]. A DeFi-Rollup averagely consumes 18,088 (289422/16) gas when ZK-rollup 16 commitments, equivalent to 1.15 USD. As a comparison, a DeFi transaction in Uniswap consumes 149,040 gwi (13.16 USD) in Feb 2023. Therefore, the rollup process is able to save around 90% of the transaction cost.

[3] https://z.cash/support/faq/

[4] https://www.monero.how/how-long-do-monero-transactions-take

V. CONCLUSION

In this paper, we implemented and evaluated a privacy-preserving DeFi protocol. This protocol is based on [8], an auditable confidentiality architecture for DeFi. By means of performance evaluation, we found that the proposed solution is both practical and affordable for privacy-preserving DeFi transactions. In future work, we aim to test complex scenarios when the user withdraws more than one shielded tokens and outputs multiple partial tokens. Moreover, the current implementation will be extended to support also other concrete cryptography primitives.

REFERENCES

[1] S. Datta, "Defi platforms across blockchains hold $69.95b in tvl – 65.6% locked on ethereum," Dec 2022. [Online]. Available: https://cryptoslate.com/defi-platforms-across-blockchains-hold-69-95b-in-tvl-65-6-locked-on-ethereum/

[2] W. Dai, "Flexible anonymous transactions (flax): Towards privacy-preserving and composable decentralized finance," *Cryptology ePrint Archive*, 2021.

[3] G. Angeris, A. Evans, and T. Chitra, "A note on privacy in constant function market makers," *arXiv preprint arXiv:2103.01193*, 2021.

[4] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 459–474.

[5] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," *GitHub: San Francisco, CA, USA*, p. 1, 2016.

[6] A. Li, G. D'Angelo, S.-K. Tang, F. Fang, and B. Gong, "An auditable confidentiality protocol for blockchain transactions," *Cryptology ePrint Archive*, 2022.

[7] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[8] A. Li, G. D'Angelo, S.-K. Tang, F. Fang, and B. Gong, "Privacy-preserving protocol for decentralized finance," *Under Review*, 2023.

[9] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu, "Zexe: Enabling decentralized private computation," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 947–964.

[10] S. Steffen, B. Bichsel, M. Gersbach, N. Melchior, P. Tsankov, and M. Vechev, "zkay: Specifying and enforcing data privacy in smart contracts," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 1759–1776.

[11] S. Steffen, B. Bichsel, R. Baumgartner, and M. Vechev, "Zeestar: Private smart contracts by homomorphic encryption and zero-knowledge proofs," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 179–197.

[12] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 315–334.

[13] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 423–443.

[14] Y. Li, J. Weng, M. Li, W. Wu, J. Weng, J.-N. Liu, and S. Hu, "Zerocross: A sidechain-based privacy-preserving cross-chain solution for monero," *Journal of Parallel and Distributed Computing*, vol. 169, pp. 301–316, 2022.

[15] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: efficient, succinct, modular," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 393–414.

[16] C. Baum, B. David, and T. K. Frederiksen, "P2dex: privacy-preserving decentralized cryptocurrency exchange," in *International Conference on Applied Cryptography and Network Security*. Springer, 2021, pp. 163–194.

[17] C. Baum, B. David, and R. Dowsley, "Insured mpc: Efficient secure computation with financial penalties," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 404–420.

[18] M. Ciampi, M. Ishaq, M. Magdon-Ismail, R. Ostrovsky, and V. Zikas, "Fairmm: a fast and frontrunning-resistant crypto market-maker," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 2022, pp. 428–446.

[19] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1521–1538.

[20] J. Lind, O. Naor, I. Eyal, F. Kelbert, E. G. Sirer, and P. Pietzuch, "Teechain: a secure payment network with asynchronous blockchain access," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 63–79.

[21] J. Van Bulck, D. Oswald, E. Marin, A. Aldoseri, F. D. Garcia, and F. Piessens, "A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1741–1758.

[22] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 142–157.

[23] N. Narula, W. Vasquez, and M. Virza, "zkLedger: Privacy-Preserving auditing for distributed ledgers," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 65–80. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/narula

[24] H. Kang, T. Dai, N. Jean-Louis, S. Tao, and X. Gu, "Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 543–555.

[25] D. Rathee, G. V. Policharla, T. Xie, R. Cottone, and D. Song, "Zebra: Anonymous credentials with practical on-chain verification and applications to kyc in defi," Cryptology ePrint Archive, Paper 2022/1286, 2022, https://eprint.iacr.org/2022/1286. [Online]. Available: https://eprint.iacr.org/2022/1286

[26] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2016, pp. 305–326.

[27] AztecProtocol, "Aztecprotocol/aztec: Public repository for the aztec v1 protocol. for the latest zkrollup release see here https://github.com/aztecprotocol/aztec-2-bug-bounty," Jan 2023. [Online]. Available: https://github.com/AztecProtocol/AZTEC

[28] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *Cryptology ePrint Archive*, 2019.

[29] V. B. Fabian Vogelsteller, "Eip-20: Token standard," Nov 2015. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20

[30] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 456–474.

[31] T. H. Yuen, S.-f. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 464–483.

[32] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "Quisquis: A new design for anonymous cryptocurrencies," in *International conference on the theory and application of cryptology and information security*. Springer, 2019, pp. 649–678.

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

[34] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[35] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[36] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *School organized by the European Educational Forum*. Springer, 1998, pp. 63–86.

[37] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, ser. STOC '85. New York, NY, USA: Association for Computing Machinery, 1985, p. 291–304. [Online]. Available: https://doi.org/10.1145/22145.22178

[38] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 103–112. [Online]. Available: https://doi.org/10.1145/62212.62222

[39] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 321–340.

[40] N. Bitansky, A. Chiesa, Y. Ishai, O. Paneth, and R. Ostrovsky, "Succinct non-interactive arguments via linear interactive proofs," in *Theory of Cryptography Conference*. Springer, 2013, pp. 315–333.

[41] J. Groth and M. Maller, "Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks," in *Annual International Cryptology Conference*. Springer, 2017, pp. 581–612.

[42] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[43] J. Eberhardt and S. Tai, "Zokrates - scalable privacy-preserving off-chain computations," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1084–1091.

[44] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for Zero-Knowledge proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 519–535. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi

[45] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Advances in Cryptology – EUROCRYPT 2013*, T. Johansson and P. Q. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 313–314.

[46] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, nov 1979. [Online]. Available: https://doi.org/10.1145/359168.359176