World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:18, No:9, 2024

Zero-Knowledge Proof-of-Reserve: A Confidential Approach to Cryptocurrency Asset Verification

Sam, Ng, Lewis Leighton, Sam Atkinson, Carson Yan, Landan Hu, Leslie Cheung, Brian Yap, Kent Lung, Ketat Sarakune

Abstract—This paper presents a method for verifying cryptocurrency reserves that balances the need for both transparency and data confidentiality. Our methodology employs cryptographic techniques, including Merkle Trees, Bulletproof, and zkSnark, to verify that total assets equal or exceed total liabilities, represented by customer funds. Notably, this verification is achieved without disclosing sensitive information such as the total asset value, customer count, or cold wallet addresses. We delve into the construction and implementation of this methodology. While the system is robust and scalable, we also identify areas for potential enhancements to improve its efficiency and versatility. As the digital asset landscape continues to evolve, our approach provides a solid foundation for ensuring continued trust and security in digital asset platforms.

Keywords—cryptocurrency, crypto-currency, proof-of-reserve, por, zero-knowledge, zkpor.

I. Introduction

THE emergence of blockchain technology [1], [2] and cryptocurrencies has ushered in a new era in the financial sector, offering decentralized, transparent, and secure transaction platforms. As cryptocurrency adoption proliferates, there is an increasing demand for robust, secure, and transparent mechanisms for auditing. This need is especially pertinent for cryptocurrency exchanges and wallets, which are required to prove their solvency to users while preserving confidentiality about their reserves.

Traditional auditing methods, however, are often ill-suited for this task due to various factors, including regulatory pressures and the time-consuming nature of manual audits [3]–[5]. In response to these challenges, we propose the use of Zero-Knowledge Proof (ZKP) [6]–[10], a cryptographic method that allows a party to prove a statement's truth without revealing any additional information.

This paper delves into the application of ZKP to Proof-of-Reserve [11] for cryptocurrency reserves, with an emphasis on confidentiality. Our approach combines the Merkle Tree [12], Bulletproof [13], and zkSnark [14]–[17] to create a Zero-Knowledge Proof-of-Reserve mechanism. This mechanism verifies that the total asset value in specific cold wallet addresses is at least equal to the required customer funds, while keeping the total amount and address confidential. Refer to Fig. 1 for a high-level overview of our method.

Sam, Ng, Lewis Leighton, Sam Atkinson, Carson Yan, Landan Hu, Leslie Cheung, Brian Yap, Kent Lung, and Ketat Sarakune are with Crypto.com, Hong Kong (e-mail: sam@crypto.com, lewis.leighton@crypto.com, sam.atkinson@crypto.com, carson.yan@crypto.com, landan@crypto.com, leslie@crypto.com, brian.yap@crypto.com, kent.lung@crypto.com and ketat.sarakune@crypto.com).

The remainder of this paper unfolds as follows: In Section II, we delve into the intricacies of constructing the Merkle Tree. Section III then moves on to elaborate on the utilization of zkSnark in the realm of asset verification. Following this, Section IV presents a detailed description of the final proof validating that assets supersede liabilities. We compare our proposed solution and existing Proof-of-Reserve methods in Section V, before discussing potential avenues for future enhancements in Section VI. Finally, Section VII encapsulates our findings and conclusions.

II. Constructing the Merkle Tree for Customer Funds

The construction of the Merkle Tree for representing customer funds is a crucial aspect of our methodology. The Merkle Tree allows us to store and verify large amounts of data efficiently and securely.

A. Design Criteria for the Merkle Tree

Our Merkle Tree is constructed based on the following design criteria:

- Each leaf node represents the funds of a single user and is strictly tied to that specific user. This is achieved by incorporating the user's email address as part of the node data.
- Users have access to the raw data of their individual leaf nodes only. The necessary data from intermediate nodes is supplied to the users, allowing them to verify the Merkle root. However, all values in other nodes, apart from the user's own leaf node, are blinded.
- The root node holds a blinded commitment of the total funds of all users. This is accomplished by summing the values in all the nodes. A range proof is provided in each node to ensure that we are summing positive values.
- Given the lending and margin trading functions, a user's coin balance may be negative. However, the total asset value for a user is always positive since all loans are fully collateralized. Consequently, instead of providing a range proof for each separate coin, we only provide a single range proof to verify the total asset value in USD equivalent for each user.

B. Node Format Details

Each node is a JSON [18] object containing the information outlined in Table I. The relationship between a parent node and its child nodes is illustrated in Fig. 1. Since the child hashes

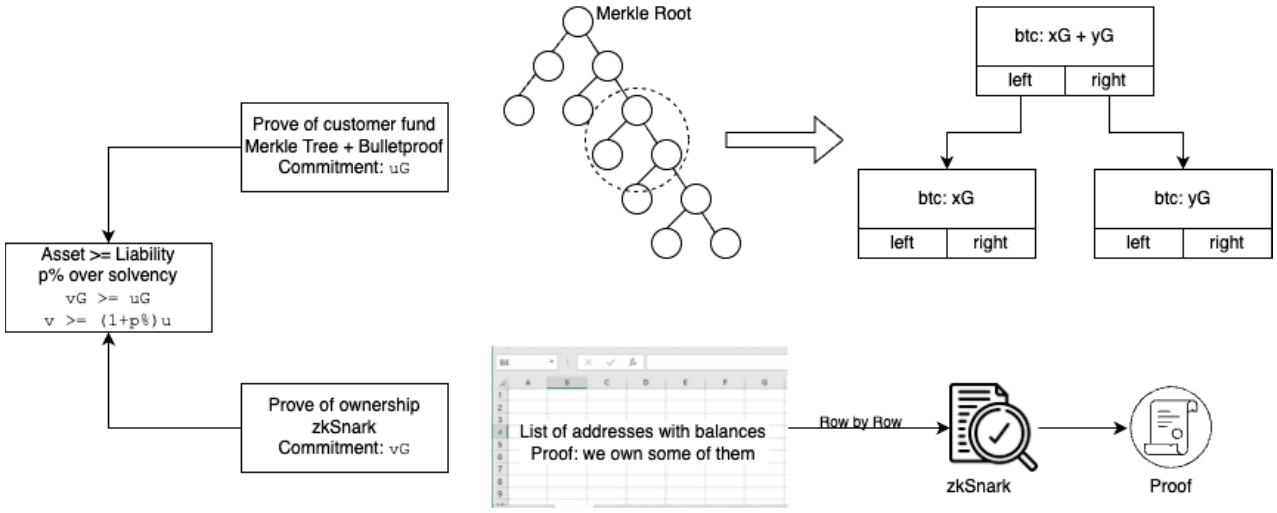

Fig. 1 High Level Overview

TABLE I
NODE DATA FORMAT

JSON fields	Disclosed for user-owned leaf nodes	Intermediate nodes

{
"btc": $c_1 \times \mathcal{G}$
"eth": $c_2 \times \mathcal{G}$
...
"range_proof": $bulletproof(\sum c_i)$
"left": hash_left_child
"right": hash_right_child
}
	x_1 & $r_1 \in ((x_1 << 128) + r_1) \times \mathcal{G} = c_1 \times \mathcal{G}$
	x_2 & $r_2 \in ((x_2 << 128) + r_2) \times \mathcal{G} = c_2 \times \mathcal{G}$
		$c_1 \times \mathcal{G} = c_{1\_left\_child} \times \mathcal{G} + c_{1\_right\_child} \times \mathcal{G}$
		$c_2 \times \mathcal{G} = c_{2\_left\_child} \times \mathcal{G} + c_{2\_right\_child} \times \mathcal{G}$

Note: $\mathcal{G}$ is the secp256k1 generator and x_i represents the USD equivalent of the asset

are already included in the parent's JSON object, the hash of a node is simply computed as `hash(stringify(json node))`.

For a leaf node, $c_i \times \mathcal{G}$ represents a commitment to x_i, which is the USD equivalent value of the user's asset at a specific point in time, for example, `2024-01-01T00:00:00 UTC`. Users should be able to verify the accuracy of this value.

The value of x_i is the USD equivalent of the user's asset in a particular cryptocurrency, truncated to 6 decimal places. Internally, x_i is represented as a 64-bit number, which is large enough to store a value of $18,446 billion USD, a value greater than the total market capitalization of all cryptocurrencies.

The variable r_i is a 128-bit random number used to protect against dictionary attacks.

Consequently, c_i is obtained by left-shifting x_i (a 64-bit number) by 128 bits and then adding r_i (a 128-bit number). A range proof is provided for each node to ensure that $\sum c_i$ is a 192-bit number.

C. Concealing the Total Customer Count

The height of the Merkle Tree can potentially reveal the maximum customer count. To ensure the total customer count remains confidential, the Merkle Tree is set to have 33 layers, equating to around 8 billion leaves. This figure surpasses the total human population, thereby guaranteeing the total customer count remains undisclosed.

D. Verification of User Funds

To facilitate users in verifying their funds, we provide the Merkle path that connects their personal leaf node to the root node. Users can authenticate their fund's integrity by recalculating the hashes along this provided path and comparing the resultant root with the originally provided root. Moreover, by verifying the bulletproof range proof, users can ensure that their funds are accurately incorporated within the total sum.

E. The Root Node and Potential Over-counting

As depicted in Fig. 1, uG represents the total customer fund (our total liability), which is reflected in the root node of the Merkle Tree.

$$uG = \sum (x_i \ll 128) \times \mathcal{G} + \sum r_i \times \mathcal{G} \tag{1}$$

The $\sum (x_i \ll 128)$ is the total customer fund left shifted by 128-bit.

The $\sum r_i$ is the sum of all random blinding factor, and this component can lead to an over-count of uG. This over-count, however, would merely result in placing *extra* funds into the reserve, thus not weakening the overall proof. However, considering a Merkle Tree with 33 layers and assuming support for 100 different cryptocurrencies, the total over-count,

approximately $900k USD, is substantial. Consequently, in our implementation, while r_i is designed to be a 128-bit number, we set the first 16 bits to zero. This adjustment effectively reduces the total over-count to a mere $13 USD.

III. Verifying Ownership of Assets with zkSnark

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, or zkSnark, is a form of zero-knowledge proof that enables one party to prove to another that a given statement is true, without conveying any additional information beyond the authenticity of the statement itself. In the context of our methodology, we use zkSnark to verify the total assets while maintaining the confidentiality of precise reserve amounts and our cold wallet addresses.

A. Primary Design Criteria

Our zkSnark is designed based on the following criteria:

- For security purposes, our Cold Wallet addresses should remain undisclosed. This is achieved by mixing our wallet addresses with, for instance, 100k random addresses. We aim to prove to a verifier that we own some of these addresses.
- Ownership proof is provided through message signing. However, the signature cannot be made public as this would allow the recovery of the public key from the signature, which could then be converted into an address. In fact, this is the primary reason why we use zkSnark.
- Given the use of zkSnark, and to minimize the circuit size, traditional hashing (e.g., SHA256) within a circuit must be avoided. Thus, when incorporating the random mixing addresses, pre-processing is required to recover the public key of the address by utilizing on-chain data and historical transactions. Users are expected to verify the accuracy of the public keys and addresses *outside* of the circuit.
- The commitment in the Merkle Tree, uG, represents the *total* liability in USD for the supported coins. Therefore, in the zkSnark, vG, should also include multiple cryptocurrencies.

B. Circuit Details

TABLE II
Inputs to the zkSnark Circuit 1

Public Inputs	Private Inputs
Coin Name	Claim Ownership ($claim$)
Address	Signature (r, s, pub_s)
Public Key (pub_a)	Blinding Factor (b)
Balance (x)	
Commitment (hG)	
Message (msg)	

The inputs of the circuit are listed in Table II and the pseudo-code is presented in Algorithm 1. Referring to Fig. 1, there is a CVS file with approximately 100k rows, and the zkSnark circuit is run row by row. Some of the public inputs are provided so that users can verify the data *outside* of the circuit. For instance, the balance (x) can be verified by the users through querying the on-chain data corresponding to the address.

Algorithm 1 zkSnark Circuit 1 Pseudo-code

```
validateIsBoolean(claim)
validateRange(b, 128-bit)
if claim then
    confirmEcdsa(r, s, pub_s, msg)
    confirmEqual(pub_s, pub_a)
    k = (x ≪ 128) + b
else
    k = b
end if
kg = eccMul(k, G)
confirmEqual(kG, hG)
generateZkSnarkProof()
```

After verifying the commitments, as depicted in (2), the sum of all commitments equals vG as shown in Fig. 1.

$$\begin{aligned} vG &= \sum hG \\ &= \sum_{i \in claim=1} (x_i \ll 128) \times \mathcal{G} + \sum b_i \times \mathcal{G} \\ &= (\text{Total Asset} \ll 128) \times \mathcal{G} \\ &+ (\text{Random factor}) \times \mathcal{G} \end{aligned} \tag{2}$$

C. Overflow of the Blinding Factor

In (2), the *Random factor* is the summation of approximately 100k 128-bit blinding factors. Clearly, this resulting number exceeds the 128-bit length. This will lead to an overestimation of the total asset value. However, considering a scenario with 100k records and 100 coins, with each number stored to 6 decimal places, the over-count approximates to a mere $10 USD. This amount is negligible within the context of the total reserve.

IV. Final Validation: Asset Surpasses Liability

In Sections II and III, we explained the construction of uG and vG respectively. In this section, we will elaborate on demonstrating $vG \geq uG$ without the need to disclose either u or v. Furthermore, we will prove that v is $p\%$ greater than u.

This proof is also grounded on zkSnark, with inputs as outlined in Table III and circuit pseudo-code as described in Algorithm 2.

TABLE III
Inputs to the zkSnark Circuit 2

Public Inputs	Private Inputs
uG	u
vG	v
p	u_1
	u_2

In summary, we have effectively elucidated the construction of the total liability, represented as uG, in Section II and the formation of the total asset, denoted as vG, in Section

Algorithm 2 zkSnark Circuit 2 Pseudo-code

```
p1 = eccMul(u, G)
p2 = eccMul(v, G)
confirmEqual(p1, uG) // u is the "u" in uG
confirmEqual(p2, vG) // v is the "v" in vG
validateRange(u2, 100) // u2 is less than 100
validateRange(u1, 192-bit) // u1 is 192-bit
t1 = u1*100 + u2
confirmEqual(t1, u) // u = u1*100 + u2
t2 = u1*(100 + p) + u2
confirmGreaterThan(v, t2) // v ≥ u1*(100+p) + u2
generateZkSnarkProof()
```

III. Furthermore, we have demonstrated how to prove that v is p% larger than u, all while preserving the confidentiality of any sensitive information. This successful implementation of cryptographic techniques underscores our commitment to balancing transparency and privacy in the verification of cryptocurrency reserves.

V. Comparison with Previous Approaches

The most straightforward way of establishing a Proof-of-Reserve involves having an auditor examine the financial status of a company and subsequently release a report affirming the sufficiency of funds in reserve. Indeed, this was the prevalent method used by most Crypto Exchanges prior to FTX's collapse in 2022 [19].

Following this event, Crypto Exchanges began employing Zero-Knowledge Proofs (ZKP) to provide Proof-of-Reserve. Several Exchanges already utilize Merkle Trees and zkSnark to verify that the values are not negative. However, these methods do not ensure confidentiality regarding the total amount or the cold wallet addresses. In fact, if the cold wallet addresses are not concealed, the total amount becomes automatically disclosed. Their algorithm resembles our Merkle Tree component, but with the Bulletproof substituted by zkSnark.

The rationale behind our use of Bulletproof is to enable users to directly run the verifier on a browser, a feature that is feasible with Bulletproof but highly complex with zkSnark.

Concerning our zkSnark component, which is responsible for proving the total assets, it theoretically requires verification by only one user. Hence, it is more acceptable for the verifier to be run using a complex setup such as Docker and command line, which may be beyond the capability of a typical user.

VI. Constraints and Future Directions

A. Confidentiality of the Cold Wallet Addresses

In the case of non-anonymous coins, it is theoretically feasible to discover the cold wallet addresses by tracing fund transfers [20]–[22]. Hence, the confidentiality of the cold wallet is best effort only.

As outlined in Section III-A, the recovery of the public key from addresses is essential. Ideally, if *ALL* addresses were incorporated into the zkSnark proof, we could achieve complete zero-knowledge. However, the practical application of this is challenging due to the diversity of address types. Some addresses may be multi-signature, others might be contracts, or unique addresses, making the recovery of the public key from these addresses quite complex. Furthermore, the inclusion of all addresses could significantly increase the run time of the prover and result in substantial costs, presenting an additional challenge to consider.

Alternatively, let's consider a scenario where we use 100k addresses as the mixing addresses. Once the proof is public, any addition of an extra cold wallet address to the list would stand out conspicuously. Therefore, the list of potential cold wallet addresses should remain fixed. However, if there is a legitimate need to include a new cold wallet address, it can be accommodated by adding an additional 100k addresses alongside the new cold wallet address.

B. False Claims of Asset Ownership

This Proof-of-Reserve fails to account for scenarios where the company borrows cryptocurrency from a third party, stores it in the Cold Wallet just before the snapshot time, and then returns the funds after this period. However, large fund movements are easy to detect, making this unlikely to happen. Additionally, the longer the duration for which the company can provide a Proof-of-Reserve, the less likely it is for the funds to be borrowed, thus boosting the credibility of the Proof-of-Reserve.

An "enhanced" version of this attack involves just by telling the third party to produce the necessary signature when required. However, a manual audit would also be susceptible to the same attack, as no central authority can verify the true ownership of a cryptocurrency address. That being said, if Exchange $\mathcal{A}$ uses a cold wallet service provider $\mathcal{B}$, it would be beneficial for an auditor $\mathcal{C}$ to obtain some form of proof from party $\mathcal{B}$ that verifies $\mathcal{A}$'s ownership of these addresses. Given this, we believe a Hybrid Approach – combining a monthly zero-knowledge proof and an annual manual audit with a limited scope (such as simply verifying address ownership), will be the future of Proof-of-Reserve.

C. Limitations of Applying Range Proof to USD Equivalent Value Only

In our current system, the range proof is applied only to the USD equivalent value of assets. This approach, while efficient, may potentially allow for misappropriation of customer assets. For instance, a company could theoretically use all customers' Bitcoin to purchase Ethereum with the expectation of higher returns. While the total USD equivalent value would remain unchanged, the distribution of assets across different cryptocurrencies would be altered.

It is important to note, however, that this issue is not unique to our system. Most, if not all, existing Zero-Knowledge Proof-of-Reserve solutions also have this limitation. Furthermore, our algorithm has been designed to prevent direct manipulation of asset distribution. For instance, even if the company creates an account for itself and engages in margin trading, the system ensures that all lending or margin

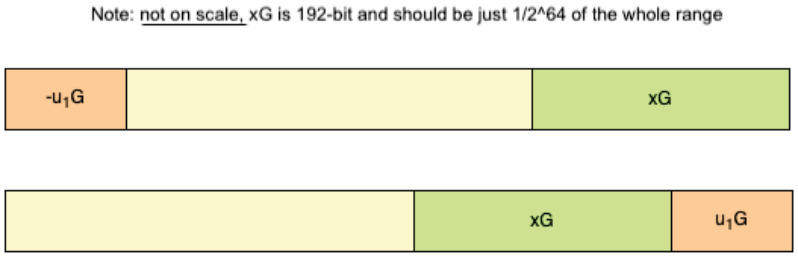

Fig. 2 Offset Range Poof – either a 192-bit number or larger than -u1G

trading activities are fully collateralized, thereby providing a safeguard against potential misuse.

From a corporate risk perspective, stakeholders would likely want to ensure that the lending or margin trading activities of an individual user do not exceed a certain threshold, say 1% of the total fund. For example, if the commitment of the total Bitcoin customer fund is cG and 1% of it is u_1G (which we have the value in Section IV), for a user with a Bitcoin commitment of xG, we would want to prove that xG is either a 192-bit number itself *OR* it is negative and the value is larger than $-u_1G$. In other words, we need to prove that $x+u_1$ falls within a range bounded by $2^{192}+u_1$. This is what we refer to as an offset range proof, as shown in Fig. 2.

If we adopt this approach, we would provide an offset range proof for *each* coin for *each* node, along with a normal range proof for the overall USD equivalent value. However, this approach does have its challenges. For instance, while an individual user may borrow less than 1% of the total value, the combined borrowing of two nodes could exceed this threshold. While we could theoretically arrange the nodes in a way that pairs a long position with a short position to cancel out each other, this may not always be feasible. Therefore, further in-depth study and statistical analysis are needed before we can confirm a concrete solution to this issue.

VII. Conclusion

In conclusion, our proposed system offers a novel solution to the problem of verifying cryptocurrency reserves, striking a delicate balance between necessary transparency and vital confidentiality. By employing advanced cryptographic techniques such as Merkle Trees, Bulletproof, and zkSnark, we have demonstrated that it is possible to verify that a platform's total assets equal or exceed its total liabilities, all without revealing sensitive data.

Our work contributes significantly to the field, providing a robust and scalable system that can enhance trust and security in digital asset platforms. However, we acknowledge that our system is not without limitations. Future research should focus on addressing the identified technical and practical challenges, and exploring ways to improve efficiency and versatility in response to evolving technology and market conditions.

As the digital asset landscape continues to evolve rapidly, maintaining trust and security becomes ever more critical. Our proposed methodology, while not without potential for further refinement, offers a promising starting point for ensuring these crucial aspects in a way that respects the need for privacy and confidentiality in the digital age.

References

[1] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017, pp. 557–564.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 03 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] S. Sharma. (2022) Big four accounting firms 'unwilling' to audit binance despite long history with coinbase. [Online]. Available: https://beincrypto.com/big-four-accounting-firms-unwilling-audit-binance-despite-long-history-coinbase/

[4] (2022) The trouble with auditing crypto firms. [Online]. Available: https://www.icaew.com/insights/viewpoints-on-the-news/2022/dec-2022/the-trouble-with-auditing-crypto-firms

[5] M. Yasmin. (2022) Auditing firm mazars pauses work for crypto clients. [Online]. Available: https://www.reuters.com/technology/auditing-firm-mazars-pauses-work-binance-other-crypto-clients-coindesk-2022-12-16/

[6] 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

[7] M. Blum, A. De Santis, S. Micali, and G. Persiano, "Noninteractive zero-knowledge," *SIAM Journal on Computing*, vol. 20, no. 6, pp. 1084–1118, 1991. [Online]. Available: https://doi.org/10.1137/0220068

[8] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Advances in Cryptology — CRYPTO' 92*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 390–420.

[9] J. Groth, "Short non-interactive zero-knowledge proofs," in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 341–358.

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

[11] S. A. et. al, "Proof of reserves – establishing best practices to build trust in the digital assets industry," Report, 2021. [Online]. Available: https://d3h0qzni6h08fz.cloudfront.net/reports/Proof-of-Reserves-.pdf

[12] R. C. Merkle, "Protocols for public key cryptosystems," in *1980 IEEE Symposium on Security and Privacy*, 1980, pp. 122–122.

[13] 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.

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

[15] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 90–108.

[16] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Advances in Cryptology - EUROCRYPT 2013*, ser. Lecture Notes in Computer Science, vol. 7881. Springer, 2013, pp. 626–645. [Online]. Available: https://www.iacr.org/archive/eurocrypt2013/78810623/78810623.pdf

[17] E. Ben 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*, 2014, pp. 459–474.

[18] "The javascript object notation (json) data interchange format," Internet Requests for Comments, RFC Editor, RFC 8259. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8259.html
[19] (2022) Bankruptcy of ftx. [Online]. Available: https://en.wikipedia.org/wiki/Bankruptcy_of_FTX
[20] J. Herrera-Joancomartí, "Research and challenges on bitcoin anonymity," in *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, J. Garcia-Alfaro, J. Herrera-Joancomartí, E. Lupu, J. Posegga, A. Aldini, F. Martinelli, and N. Suri, Eds. Cham: Springer International Publishing, 2015, pp. 3–16.
[21] F. Reid and M. Harrigan, *An analysis of anonymity in the bitcoin system*. Springer, 2013.
[22] N. Alsalami and B. Zhang, "Sok: A systematic study of anonymity in cryptocurrencies," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, 2019, pp. 1–9.