# The challenges of proving solvency while preserving privacy.

Tabacaru Robert*, Anghel Florin*, Asandoaiei David*, Simion Emil†

January 2023

**Abstract**

The increasing popularity of blockchain technology has affected the way we view many fields related to computer science, with E-commerce being no exception. The distributed nature and transparency of blockchain-based systems is one of its main perks, but it also raises some issues when it comes to privacy.
Zero-knowledge proofs are very powerful building blocks when it comes to building privacy-preserving protocols, so, naturally, they have attracted a lot of attention in the last years.
Following the recent collapse of the very popular crypto exchange *FTX*, we believe it is important to analyse how such events can be prevented in the future. This paper aims to highlight solutions that use zero-knowledge to prove solvency.

**Keywords:** Zero-knowledge Proofs, Blockchain, Cryptocurrency Exchanges.

# 1 Introduction

## 1.1 Zero-knowledge proofs

*"Zero-knowledge proofs (ZKPs) are protocols in which one party, named the prover, can convince the other party, named the verifier, that some assertion is true without revealing anything other than the fact that the assertion being proven is true."* [1]

The notion of Zero-knowledge proofs was put forward by Goldwasser et al. [2], and since it was introduced it has led to several breakthroughs, especially in secure multi-party computation and blockchain.
The first and most important concept to understand is that we are not talking about deterministic proofs, like those one would use in algebra, but rather probabilistic proofs.
The easiest way to explain is by example, and Quisquater et al. [3] use the story of Ali Baba's cave:

*Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi
†Politehnica University of Bucharest, Email: emil.simion@upb.ro

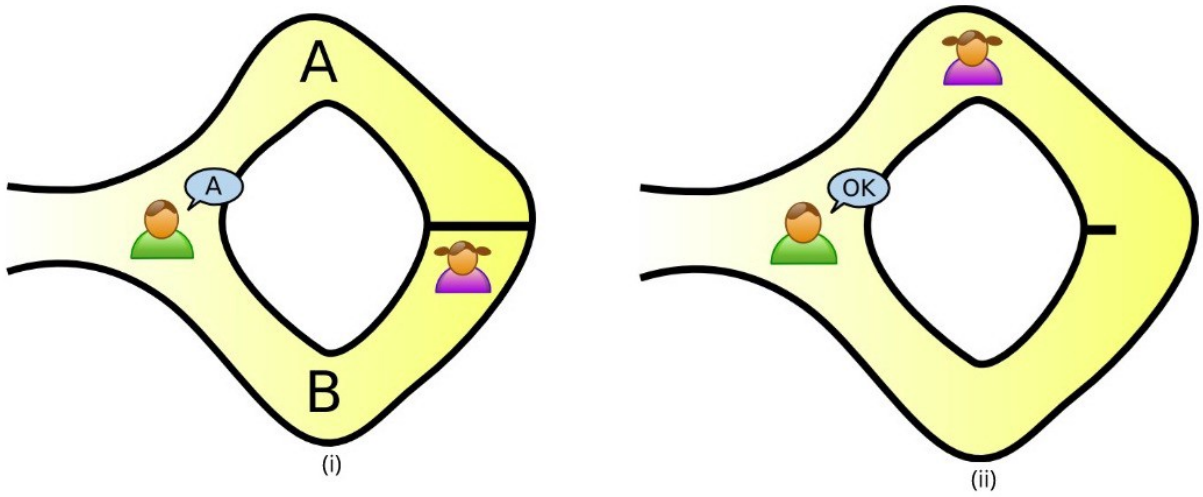


Figure 1: Ali Baba's cave.[4]

Our protagonists are Victor and Peggy. Peggy claims that she knows the magic words used to open a secret door between the two sides of the cave: $A$ and $B$. She wants to prove this to Victor, but she does not want to reveal the magic words to him.
To achieve this, they set up an experiment:

1. Peggy goes into the cave and picks any of the two sides to go through.
2. Victor then walks in to make sure that she cannot go back the way she came in and calls out a side randomly. ($A$ or $B$)
3. Peggy must come out from that side to convince Victor that she knows the magic password.

When repeated once, it is obvious that there is a 50% chance that Peggy had gone in through the exact side Victor asked her to come out of, meaning she did not need to know any magic words to succesfully complete the experiment. If we repeat the experiment enough times, the probability of Peggy scamming Victor becomes increasingly lower: $(\frac{1}{2})^n$, where $n$ is the number of repetitions, to be exact.
So, after an appropriate number of tries, Victor is convinced, but how would Victor and Peggy go about convincing somebody else that Peggy does indeed know the magic words? Victor could record his perspective but no one could really be sure that Victor and Peggy are not collaborating to scam them.
Building on this simple example we can distinguish the two main types of zero-knowledge proofs: **interactive** and **non-interactive**. Ali Baba's cave is the simplest example of an interactive zero-knowledge proof.
The latter type was introduced by Blum et al. in [5], and involves the use of a random string generated by a trusted third party. For some examples of non-interactive protocols, one can consult works such as [6] and [7]. We will also present non-interactive protocols in a more formal manner in later sections as they are fundamental building blocks for solvency proving protocols.

**An interesting implication:**

Blum [8] puts forward the idea of being able to prove any mathematical theorem in zero-knowledge,

but does not specify an exact way of doing it. Since then, Goldreich [9] has proven that all NP statements can be proven in zero-knowledge. This versatility has contributed a lot to their increase in popularity.

### 1.2 Blockchain and e-commerce

The introduction of blockchain technologies has impacted the entire world, however, the field of e-commerce has proven to be a challenging one to adapt to such a distributed model. At its core, the blockchain is a distributed, write-only ledger that provides full transparency when it comes to the transactions added on it.

This sort of decentralised system promises to be secure in the sense that, once added to the

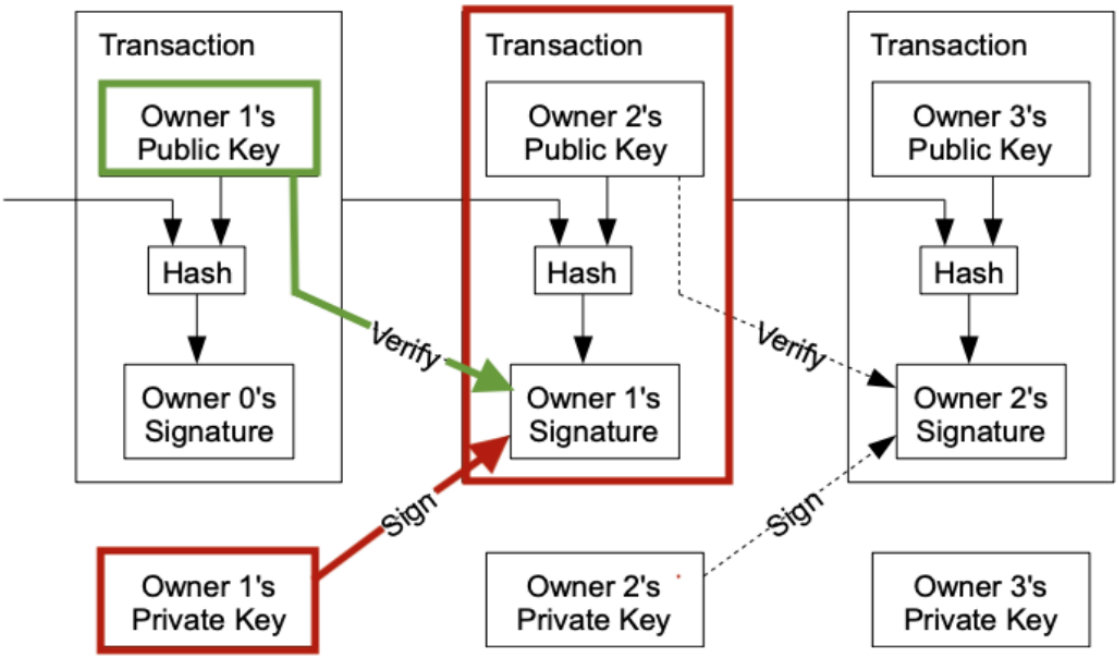


Figure 2: The blockchain. [10]

blockchain, transactions can no longer be tampered with. This could also help prevent corruption, money laundering, and numerous other illegal activites. However, there are some challenges when it comes to scalability and privacy, and the road toward a standardized suite of protocols is still an uncertain one. Works such as [11] and [12] do well to outline some applications as well as some challenges of integrating blockchain technologies in the e-commerce field.

To be able to make trades, a user needs to have a wallet. A wallet is defined with the help of a key pair: a public key used for receiving funds, and a private one for transferring funds to another address.

Storage and management of such keys would be a hassle for the average user, and that's where crypto exchanges come in, allowing user to trade using cryptocurrencies without having to worry about key management. However, there have been cases of such exchanges going bankrupt, thus leading to significant financial losses for their customers.

In the following sections, we are going to present some preliminary notions, review the existing literature on this topic and finally draw some conclusions regarding open issues in this field.

# 2 Preliminaries

## 2.1 Commitment schemes

Commitment schemes are important building blocks for zero-knowledge systems as they allow an entity to commit to a secret value without revealing it right away.
Formally, a commitment scheme is made up of two algorithms:

1. Commit: $c = Commit(m, r)$, where $m$ is the message being committed and $r$ represents a randomising factor
2. Open: $b = Open(c, m, r)$, $b \in \{0, 1\}$, and $b = 1 \iff c = Commit(m, r)$

The two main security requirements of a commitment scheme are:

1. Binding: Given $c = Commit(m, r)$, it is infeasible to find $m'$ and $r'$ such that $Commit(m', r') = c$.
2. Hiding: It is hard to gain any information about $m$ from $c$.

Pedersen [13] introduced a scheme based on the hardness of the discrete logarithm problem that has the very benefic property of being **homomorphic**.
We assume we are operating in the context of a group $\mathbb{Z}_p$ of prime order where the discrete logarithm problem is considered to be difficult.

- $Commit(m, r) = g^m h^r$, where $g, h$ are generators. (and are also available as public parameters)
- $Open(c, m, r) = 1 \iff g^m h^r = c$.
- **Homomorphism:** Given $c_1 = Commit(m_1, r_1)$ and $c_2 = Commit(m_2, r_2)$, we have $c1 \cdot c2 = Commit(m_1 + m_2, r_1 + r_2)$.

## 2.2 Merkle Trees

Introduced by Ralph Merkle in 1987 in [14], Merkle trees (also known as hash trees) represent a tree in which:

- every leaf node is labelled with the cryptographic hash of a data block;
- every internal node is labelled with the cryptographic hash of the labels of its child nodes.

The purpose of Merkle Trees is to allow secure and efficient (logarithmic time) verification of contents of large data structures. They are especially useful because they allow a prover to commit a large dataset by publishing the hash of the root, and subsequently provide membership proofs for each element of the dataset.

### 2.2.1 Summation Merkle Trees

Summation Merkle trees ([15], [16]) are modified Merkle trees. Each node contains, in addition to a hash $h$, a value $v$, which represents:

- a prover's liabilities to the user, in a leaf node;
- the sum of the values in its two child nodes $lch$ and $rch$, i.e. $v = v_{lch} + v_{rch}$, if the respective node is internal.

For the leaf node , $h$ contains $H(v)$, and for internal nodes $H(v||h_{lch}||h_{rch})$. The summation Merkle trees are build in bottom-up manner, the value $v$ of the root node being sum of all leaf node numeric values.
However, summation Merkle trees are vulnerable to attacks as the prover is able to claim smaller liabilities by setting value field as $v = \max(v_{lch}, v_{rch})$, while still generating proofs the pass verifications successfully (as presented in [17]).

### 2.2.2 Sparse Merkle Trees

Sparse Merkle Trees (([15], [16]) have emerged in order to fix the vulnerabilities presented in [17] while also hiding the population count using padding nodes and improving memory use by not constructing the full binary tree.

## 2.3 Non-Interactive Zero Knowledge Proofs (*NIZKs*)

In the previous section we provided an informal introduction to zero-knowledge proofs, but in order to understand practical implementations, it is important to provide a formal definition.

We briefly mentioned the importance of Goldreich's work [9] in defining the impact of *ZKPs* on the complexity class *NP*, more specifically proving statements. So, for a language $L \in NP$ and an element $x \in L$, a prover is able to convince a verifier that $x$ indeed belongs in $L$ through a piece of information called a witness ($w$).

**Example:** If we state that for some graph there exists a Hamiltonian cycle, the witness would be one such cycle.

That being said, Morais et al. [18] formally define a non-interactive zero-knowlege proof scheme using three algorithms:

1. **Setup**, which takes a security parameter $\lambda$ as an input and outputs some parameters required for the protocol.
$$params = Setup(\lambda)$$
2. **Prove**, which takes $x \in L$ and a witness $w$ for $x$, and outputs a zero-knowledge proof.
$$proof = Prove(x, w)$$
3. **Verify**, which takes a proof as an input and outputs the bit 1 if the proof has been accepted, and 0 otherwise.

**Security requirements of a *ZKP*:**

1. **Completeness:** For a witness $w$ that satisfies $x$, we have $Verify(Prove(x, w)) = 1$.
2. **Soundness:** For a witness $w$ that does not satisfy $x$, the probability that $Verify(Prove(x, w)) = 1$ is negligible
3. **Zero-knowledge:** The proof that is published reveals no information about the witness $w$ besides the fact that it satisfies $x$. (or that is does not)

**Converting any interactive scheme to a non-interactive one:**
Fiat and Shamir [19] introduced a technique for identification based on interactive *ZKPs* and identity-based cryptography. Nicknamed the Fiat-Shamir heuristic, this technique allows for any interactive ZKP to be converted into a non-interactive one, but it requires the use of a hash function, thus making the protocol dependant on the random oracle model ($ROM$).

### 2.4 Zero-knowledge range proofs (*ZKRPs*)

*ZKRPs* are a special type of *ZKPs* that allows to prove that a secret value belongs in a specific interval. Morais et al. [18] provide a thorough survey of the most promising constructions for implementing *ZKRPs* and some interesting possible applications, most notably:

1. **Zero-knowledge KYC**: For example *ZKRPs* could be used to prove that somebody is over 18 without revealing her exact age.
2. **Electronic voting and auctions.**
3. **Confidential transactions:** Confidential transactions were proposed in an informal manner by Gregory Maxwell, lead bitcoin developer, back in 2014, and formalized in [20]. This aims to provide privacy to users by not disclosing transaction amounts while also making sure that no coins are created out of thin air or destroyed. The notion is also extended to *confidential assets*, a scheme designed for blockchains that allow transactions with multiple types of items, hiding both the amount transactioned and the asset type.
4. **Proofs of solvency:** This is the topic we are going to be focusing on in the rest of this paper, and range proofs play a key role in such protocols.

### 2.5 Zero-knowledge proofs for set membership

Having defined what a *ZKRP* is, a proof for set membership is just a generalisation. If in the case of *ZKRPs* we published a proof that a secret value was in a specific interval $[a, b]$, where $b > a$, we would now have to prove that the value belongs to a public set.
Because set membership proofs are a generalisation of range proofs, their applications are similar, so we won't go into more detail for now.

### 2.6 Proofs of solvency

For a financial entity to be considered solvent, it needs to have more assets than it has liabilities. To be able to prove such an entity is solvent, we need to combine two categories of proofs:

### 2.6.1 Proof of assets

A proof of assets aims to prove that the amount of assets owned by an entity is above a specific lower bound, or even equal to a specific value, depending on the use-case. Depending on the context it was designed for, a proof of assets can differ in implementation. For example, in *Provisions* [21], the exchange would prove that the sum of balances for the bitcoin addresses he owns is equal to a certain amount without disclosing what the public keys for those addresses are, just that it knows both the public and private keys for that specific address.
It is worth noting that not all works concerning zero-knowledge proofs of solvency provide implementations the proof of assets.
This is because implementation can differ a lot depending on context (supported currencies, etc.), and also because there are less possible exploits for these sort of proofs compared to liability proofs, which are the second part of the puzzle. A good source of information regarding the best practices that should be followed for publishing proofs of assets was published by the *Chamber of Digital Commerce* [22].

### 2.6.2 Proof of liability

Liability proofs can be a bigger challenge to implement, as the risk of leaking individual user information or hiding certain liabilities comes into play.
Depending on the type of auditing that is required, liability proofs can be centralized or distributed. An advantage of distributed proofs is that any user can verify the published proof to confirm that his individual liabilities have been reported correctly. Merkle proofs have been employed extensively in such works.
The work of Chalkias et al.([15] ,[16], [23]) outlines some very important issues regarding existing solvency proofs and how they handle liability proofs, while also starting an initiative for standardizing distributed auditing proofs.

We are now going to analyse some of the most promising constructions for implementing such proofs and the problems observed in practice.

# 3 State-of-the-art

In order for an exchange to prove it is solvent, it would need to publish both a proof of assets and liabilities, however, we will examine the two types of constructions separately, as they differ heavily in implementation and deal with different security issues.

## 3.1 Proof of assets (*PoA*) solutions

Implementation of a proof of assets solution, especially in the context of crypto exchanges, which often support more than one type of digital currency, is often reliant on the underlying blockchain architecture.

Before *Provisions* [21], no proposed solutions would preserve the privacy of the addresses owned by the exchange, and while some exchanges have no problem disclosing that information, others desire privacy.
To preserve privacy, the protocol makes use of the homomorphic property of Pedersen commitments and an anonymity set to hide the individual addresses it controls in the following manner:

The exchange hides the addresses it controls (for which he knows the private keys) inside a larger anonymity set and then publishes a commitment to the total values of assets along with a proof that it did not add assets from addresses it does not now the private key to.

**A problem with privacy:** Providing privacy to the exchange sounds good at first, but the base provisions protocol does not account for the case of collusion between multiple exchanges. They do, however, provide an extension to the base protocol that uses the discrete logarithm problem to allow the exchange to also publish additional information to ensure no two exchanges can use the same address in their proof.
However, despite introducing a very promising concept, the protocol is limited by the fact it is only compatible with addresses that have unhashed public keys.

Since nowadays most blockchain systems make use of such techniques, the protocol is not quite usable in practice, it had, however, a fundamental impact on subsequent literature:

- *MProve* [24] was designed for the *Monero* [25] cryptocurrency since the underlying implementation did not fit in with the restrictions of *Provisions*.
  Their approach was to publish a proof that the key images of one-time addresses owned by the exchange have not appeared on the blockchain before. They also offered a proof of non-collusion. *MProve+* [26] was later introduced to solve a privacy issue that leaked the exchange-owned address when a transaction spending from it was published, while also leveraging *Bulletproofs* [27] in order to decrease proof size, generation and verification times.
- Chatzigiannis and Chalkias provide a construction designed for the *Diem* blockchain [28].
- Baldimtsi et al. [29] introduce *gOTzilla*, an efficient **interactive** zero-knowledge proof protocol based on *Oblivious transfer* [30] and the *MPCitH* paradigm [31] designed with proof of assets in mind. The authors argue that, despite being interactive, the protocol is still suitable for this context as it can be executed between the exchange and a trusted auditor.

As a conclusion on *PoA* protocols, no construction exists that is suitable for all scenarios as blockchain architectures are still evolving and multiple paradigms are explored. However, standards have been set in works such as [22] and [32] regarding the requirements that must be met when designing *PoA* protocols in order to prevent cheating.

**A note on "borrowed" assets:** To protect against collusion, protocols such as *Provisions* can ensure that two addresses are not used in proofs provided by different exchanges. To inflate its assets without using another address, an exchange can borrow funds so that it appears to have more assets than in reality. Cryptographically, such a case would be difficult to handle, especially since privacy is something that is often desired, however, periodic checks and the inherent transaction transparency provided by most blockchain systems can help spotting such irregularities.

### 3.2 Proof of liabilities (*PoL*) solutions

Proving liabilities in a privacy-preserving manner, in contrast to proving assets, is a bit more elaborate as there are more variables to consider.
First of all, for asset proofs, the exchange would not be incentivized in any way to withhold information. When proving liabilities, any malicious exchange will try to leave out information whenever possible to seem solvent.

To prevent such cases, the latest trend in this area of the field is to publish a proof that can be distributively audited, meaning that any user of the exchange can validate that their respective liabilities have been included in the proof.

**Publishing medium:** Unlike asset proofs which have even been implemented using interactive techniques such as [29], it is very important that liability proofs are published on a public bulletin board available to all users. This prevents the exchange from sending different proofs to different users. This also means that any private information leak will be publicly available: number of users, individual liabilities, etc..

Recent efforts have been made to propose a standard of implementation for distributed proof of liabilities, with *DAPOL* [15] and later *DAPOL+* [16] emerging as leading candidates.
*DAPOL+* was introduced by Ji and Chalkias and it proposes the first provably secure implementation standard for liability proofs. It preserves the privacy for the total liabilities, the user population count and also their individual liabilities.
It is heavily inspired from the construction proposed by Camacho [33], which makes use of Pedersen commitments, zero-knowledge range proofs and summation Merkle trees. It is a protocol that had already been used several times in practice, but it does not account for the privacy of the user population.
*DAPOL* was proposed to address a vulnerability related to summation Merkle trees allowing an exchange to claim less liabilities [17] and to ensure the privacy of the number of users. *DAPOL+* subsequently fixes an exploit that allowed an attacker to bound the population number to an interval and addresses certain inefficient aspects of the original construction.

*Provisions* also provides a construction for proving liabilities that also leverages Pedersen commitments and zero-knowledge proofs, but it does not take advantage of Merkle trees. This is because at the time of its proposal solutions such as *Bulletproofs* [27] did not exist, and there were concerns of large proof sizes making the scheme unusable. However, in its current state, the commitment published on the bulletin board is linear in size with the number of users, rendering it unusable for large exchanges. *DAPOL+*, on the other hand, has a constant commitment size.

## 4 Open issues and challenges

A frequently update list of exchanges that have published a proof of assets accompanied by a proof of liabilities (in one form or another) can be found at [34].

Exchange Proof of Reserve Dashboard – Dec. 2022

| | Date of most recent PoR | First PoR undertaken | Ongoing frequency | Assets Covered (most recent PoR) | Estimated share of deposits covered | Covered asset value at time of PoR ($m) | Overseen by | Liability verification | PoR score (out of 6) |
|---|---|---|---|---|---|---|---|---|---|
| BitMEX | 22-Jun | 2021 | Twice weekly | BTC | >95% | 1,320 | None | Anyone | 5 |
| Kraken | 22-Nov | 2014 | Semi-annual | BTC, ETH, USDC, USDT, ADA, DOT, XRP | 63% | 9,100 | Armanino (AUP) | Clients | 5 |
| Deribit | 22-Dec | 2022 | Daily | BTC, ETH, ETHW, SOL, USDC | 100% | 1,600 | None | Anyone | 4.5 |
| Kucoin | 22-Nov | 2022 | Unknown | BTC, ETH, USDC, USDT | 58% | 1,760 | Mazars (AUP) | Clients | 4.5 |
| OKX | 22-Nov | 2014 | Ongoing but unknown | BTC, ETH, USDT | 92% | 6,200 | None | Clients | 4 |
| Gate.io | 22-Oct | 2020 | Unknown | BTC, ETH | 29% | 667 | Armanino (AUP) | Clients | 4 |
| Crypto.com | 22-Dec | 2022 | Unknown | BTC, ETH, USDC, USDT, XRP, DOGE, SHIB, LINK, MANA | 87% | 2,810 | Mazars (AUP) | Clients | 4 |
| Bybit | 22-Dec | 2022 | Unknown | BTC, ETH, USDC, USDT | 87% | 791 | None | Clients | 4 |
| Luno | 22-Nov | 2021 | Quarterly | BTC, ETH, LTC, USDC, BCH, XRP, LINK, UNI, ADA, SOL | 90% | Unknown | Mazars (AUP) | None | 3.5 |
| Binance | 22-Nov | 2022 | Unknown | BTC | 15% | 9,300 | Mazars (AUP) * | Clients | 2 |

* Binance Mazars AUP was subsequently deleted

Figure 3: Recent Proof of Reserves[1]attestations performed by major exchanges [34]

.

This shows that the solutions that are used today in practice are not necessarily indisputable, and can lead to information being leaked or to unreported liabilities.
Even though strong solutions such as *DAPOL+* have been proposed in the literature, they have not been adopted yet. Ji and Chalkias [16] reference a proof-of-concept implementation written in *Rust* that they used to benchmark their construction, but it is no longer publicly available.

A possible direction for future work is to work on implementations for solutions such as *DAPOL+*, performing benchmarks to make sure they are ready for production use.

Aside from that, there are still some other important issues worth noting:

### 4.1 Statistical attacks on liability proofs

As we mentioned before, to ensure that a liability proof correctly includes the liabilities of each user, a public bulletin board is used so that any user can verify that his individual part of the proof was computed correctly.
In a practical scenario it is unlikely that every user will verify his inclusion.
Works such as [35] outline the dangers of having no privacy for the verifiers of the individual proofs. To avoid cases where query probabilities are exploited in order to exclude user liabilities, *Private information retrieval* [36] can be leveraged to protect the users' access to their information. (proposed in [15])

### 4.2 Dispute resolution

The problem of dispute resolution is an issue that is not often discussed, but has been brought up in some works. ([16], [32])
The problematic scenario is that of a user contesting the correctness of his reported individual liabilities.
To be able to resolve this issue, transactions between the exchange and user would have to be signed by both entities. A trusted third party could review this evidence and decide if the published proof is a correct one. [16] provides an example for dispute resolution in the scenario of solvency proofs,

[1]Proof of solvency schemes are also referred to as proofs of reserves.

but such a system for emitting digital "receipts" each transaction in a manner that is also practically deployable remains an open issue.

### 4.3 Challenges of computing and storing proofs

Besides the implementations for the algorithms themselves, deployment of such solutions is still a challenge:

1. Proofs need to be periodically recomputed to ensure "freshness".
2. Users need to opt-in to verify their individual liabilities.
3. The exchange needs to store the generated Merkle trees.

Dahlia Malkhi from Chainlink [37] proposes several ways oracles could be used to secure the process. Three main approaches emerge:

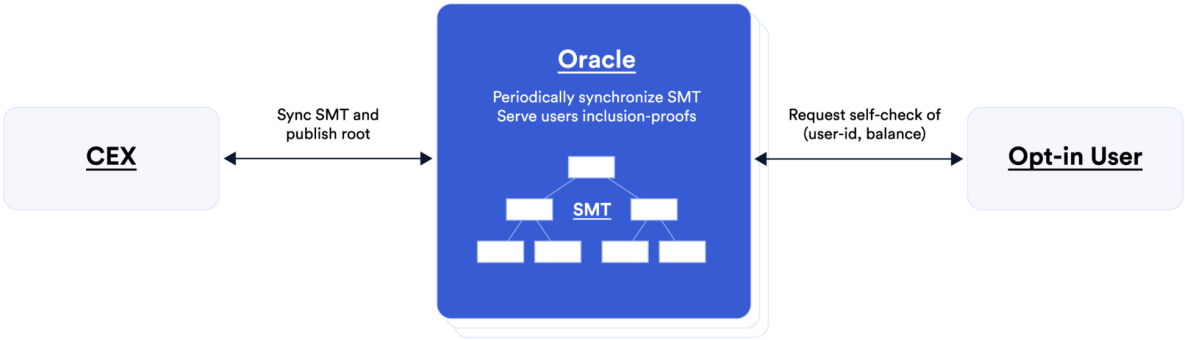


Figure 4: Approach 1 - use an oracle to store proofs and handle user self-checks.[37]

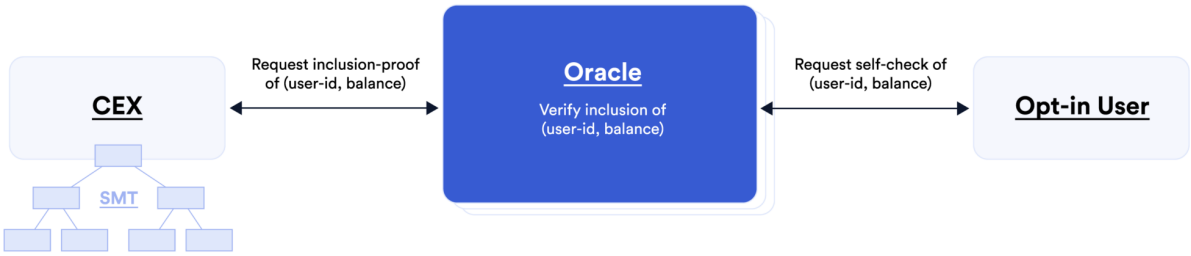


Figure 5: Approach 2 - use an oracle for self-checks on behalf of users.[37]

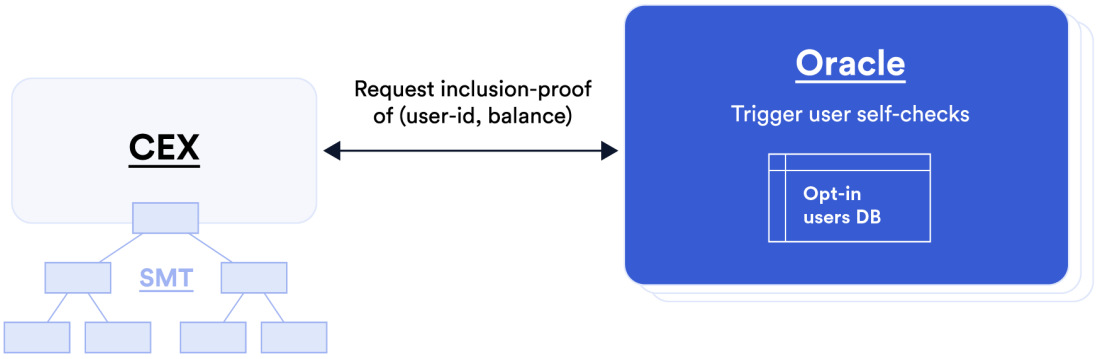


Figure 6: Approach 3 - use an oracle for self-checks on behalf of users, but automated.[37]

These proposed approaches can help standardize the use of solvency proofs in practice. The first approach is very promising as it can be used with constructions such as the one proposed in *DAPOL+* because it is privacy-preserving and it would not leak user information to the oracle, and automating self-checks is definitely a step in the right direction. Statistical attacks still have to be taken into consideration.

## 5 Conclusions

The rapid development of zero-knowledge proofs is opening the gates for many promising privacy-preserving applications. Zero-knowledge solvency proofs are a very good example of that, and in the aftermath of the fall from grace of *FTX*, we can see that solutions have been proposed in order to prevent such things from happening again.
There are still some issues that need to be addressed in order for these proofs to be indisputable, but the progress in this direction is looking very promising.

## 6 Acknowledgements

The work of Chalkias and Chatzigiannis has been especially impactful in this area, with works like [15], [16], [32], [23] providing inisghtful reviews of existing solutions in the field and unresolved problems.

## References

[1] Feng Li and Bruce M. McMillin. A survey on zero-knowledge proofs. *Adv. Comput.*, 94:25–69, 2014.

[2] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[3] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis C. Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig

Guillou, and Thomas A. Berson. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631. Springer, 1989.

[4] Golden wiki. Proof of reserves wall of fame. https://golden.com/wiki/Zero-knowledge_proof-3GRZ5, 2022.

[5] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112. ACM, 1988.

[6] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72. Springer, 1987.

[7] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. ACM, 2019.

[8] Manuel Blum. How to prove a theorem so no one else can claim it, August 1986. Invited 45 minute address to the International Congress of Mathematicians, 1986. To appear in the Proceedings of ICM 86.

[9] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 1986.

[10] Toby Chitty. The mathematics of bitcoin — the blockchain, May 2020. [Online; posted 25-May-2020].

[11] Alex Biryukov, Dmitry Khovratovich, and Sergei Tikhomirov. Privacy-preserving kyc on ethereum, 2018.

[12] Horst Treiblmaier and Christian Sillaber. The impact of blockchain on e-commerce: A framework for salient research topics. *Electron. Commer. Res. Appl.*, 48:101054, 2021.

[13] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[14] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[15] Konstantinos Chalkias, Kevin Lewi, Payman Mohassel, and Valeria Nikolaenko. Distributed auditing proofs of liabilities. *IACR Cryptol. ePrint Arch.*, page 468, 2020.

[16] Yan Ji and Konstantinos Chalkias. Generalized proof of liabilities. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3465–3486. ACM, 2021.

[17] Kexin Hu, Zhenfeng Zhang, and Kaiwen Guo. Breaking the binding: Attacks on the merkle approach to prove liabilities and its applications. *IACR Cryptol. ePrint Arch.*, page 1139, 2018.

[18] Eduardo Morais, Tommy Koens, Cees van Wijk, and Aleksei Koren. A survey on zero knowledge range proofs and applications. *CoRR*, abs/1907.06381, 2019.

[19] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[20] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, volume 10958 of *Lecture Notes in Computer Science*, pages 43–63. Springer, 2018.

[21] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 720–731. ACM, 2015.

[22] Chamber of Digital Commerce. Proof of reserves. the practitioner's guide to an emerging standard for increasing trust and transparenct in digital asset platform services, May 2021.

[23] Konstantinos Chalkias, Panagiotis Chatzigiannis, and Yan Ji. Broken proofs of solvency in blockchain custodial wallets and exchanges. *IACR Cryptol. ePrint Arch.*, page 43, 2022.

[24] Arijit Dutta and Saravanan Vijayakumaran. Mprove: A proof of assets protocol for monero exchanges. *IACR Cryptol. ePrint Arch.*, page 1210, 2018.

[25] Nicolas van Saberhagen. Cryptonote v2.0. https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf, October 2013.

[26] Arijit Dutta, Suyash Bagad, and Saravanan Vijayakumaran. Mprove+: Privacy enhancing proof of reserves protocol for monero. *IEEE Trans. Inf. Forensics Secur.*, 16:3900–3915, 2021.

[27] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.

[28] Panagiotis Chatzigiannis and Konstantinos Chalkias. Proof of assets in the diem blockchain. *IACR Cryptol. ePrint Arch.*, page 598, 2021.

[29] Foteini Baldimtsi, Panagiotis Chatzigiannis, S. Dov Gordon, Phi Hung Le, and Daniel McVicker. gotzilla: Efficient disjunctive zero-knowledge proofs from MPC in the head, with application to proofs of assets in cryptocurrencies. *IACR Cryptol. ePrint Arch.*, page 170, 2022.

[30] Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31. ACM, 1988.

[31] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30. ACM, 2007.

[32] Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. Sok: Auditability and accountability in distributed payment systems. In Kazue Sako and Nils Ole Tippenhauer, editors, *Applied Cryptography and Network Security - 19th International Conference, ACNS 2021, Kamakura, Japan, June 21-24, 2021, Proceedings, Part II*, volume 12727 of *Lecture Notes in Computer Science*, pages 311–337. Springer, 2021.

[33] Philippe Camacho. Protocols for provable solvency. https://www.slideshare.net/philippecamacho/protocols-for-provable-solvency-38501620, 2014.

[34] Nic Carter. Proof of reserves wall of fame. https://niccarter.info/proof-of-reserves/, 2022.

[35] Nihar Shah. Statistical attacks on proof of solvency. https://www.slideshare.net/philippecamacho/protocols-for-provable-solvency-38501620, 2014.

[36] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50. IEEE Computer Society, 1995.

[37] Dahlia Malkhi. Exploring decentralized proof-of-solvency systems. https://blog.chain.link/decentralized-proof-of-solvency-systems, January 2023.