UBIC—A Blockchain-less Cryptocurrency

Maurantonio Caprolu
College of Science and Engineering
Hamad Bin Khalifa University (HBKU)
Doha, Qatar
macaprolu@hbku.edu.qa

Elmahdi Bentafat
Information Systems Department
Ahmed Bin Mohammed College
Doha, Qatar
mahdi@abmmc.edu.qa

Spiridon Bakiras
Infocomm Technology Cluster
Singapore Institute of Technology
Singapore
spiridon.bakiras@singaporetech.edu.sg

Roberto Di Pietro, *IEEE Fellow*
RC3 Center, CEMSE Division
King Abdullah University of Science and Technology (KAUST)
Thuwal 23955, Saudi Arabia
roberto.dipietro@kaust.edu.sa

2023 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) | 979-8-3503-3535-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/DAPPS57946.2023.00013

***Abstract*—In this paper we propose UBIC, a novel blockchain-less architecture that preserves the main advantages of classic cryptocurrencies while avoiding their pitfalls. The proposed construction is general—that is, UBIC can be adopted at par with any other cryptocurrency—though UBIC also satisfies the requirements to support state-sponsored financial services, like Universal Basic Income and Central Bank Digital Currency. Indeed, UBIC stands for *Universal Basic Income Coin*, to highlight one of its most straightforward use-cases. One of the key features of UBIC is that every user participating in the protocol gets fair and equal access to the rewards, regardless of the available resources, e.g., computational power or financial stake. Moreover, by leveraging standard cryptographic techniques, such as homomorphic encryption and verifiable random functions, UBIC ensures full user privacy and trust in the network, while enjoying a highly scalable architecture. Our experimental results confirm the feasibility of the proposed architecture and demonstrate that UBIC is very efficient in terms of transaction verification time.**

To the best of our knowledge, this is the first blockchain-less cryptocurrency proposal. Other than being interesting on its own, and being particularly fit to support UBI and Central Bank Digital Currency, the architectural solutions and the technical choices discussed in this contribution have the potential to generate high impact and further research in the field.

***Index Terms*—Cryptocurrency, Central Bank Digital Currency, Universal Basic Income, DeFi, Distributed systems, Security, Privacy**

I. Introduction

Cryptocurrencies are digital assets introduced to provide an alternative medium of exchange that differs from classical banking systems. Enabled by blockchain technology, existing cryptocurrencies implement a distributed ledger that does not need a trusted third party to keep the system secure and consistent. Since Bitcoin, different types of cryptocurrencies have emerged, including Ethereum, Cardano, Solana, Polkadot, and many others. From a high-level perspective, all existing cryptocurrencies share the same objectives, strengths, and weaknesses, while differing greatly in architecture, consensus mechanism, and capitalization [1]. Cryptocurrencies were born as a new electronic payment tool, meant to be an alternative to the institutional bank payment channels. However, several shortcomings have limited the large-scale diffusion of cryptocurrencies as a means of payment, leaving room for other use-cases that have become predominant, e.g., speculative investments [2].

Their significant limitations stem mainly from the technological and legal domains. From the technical perspective, common shortcomings include scalability issues, which restrict the system's transaction throughput [3]. This is currently preventing existing cryptocurrencies from competing with traditional electronic payment systems, which have higher throughput and faster transaction confirmation times [4], though recent proposals have reduced the gap. From the legal perspective, the privacy-preserving nature of existing cryptocurrencies makes it impossible to implement Anti-Money Laundering (AML) policies, such as Know Your Customer (KYC) and Enhanced Due Diligence (EDD) [5]. This prevents existing cryptocurrencies from being adopted for legal tender, with just a couple of (marginal) exceptions (El Salvador and Central African Republic). Consequently, many users refuse to use cryptocurrencies, as they operate in an unregulated market, without a legal framework that protects them from market manipulations [6].

However, some common characteristics of existing cryptocurrencies, such as being permissionless, distributed, self-organized, and censorship-resistant, have opened an interesting debate on their use in specific, compelling use-cases. One of those, is for implementing Universal Basic Income (UBI) [7]. UBI is a radical policy proposal of an unconditional periodic cash income granted to people satisfying a few requirements (e.g., living in a particular geographical area or having a given citizenship). Although controversial, UBI is seen mainly as a viable solution to financial inequality, as it fairly distributes wealth to all members of a community [8]. However, existing implementation proposals are not very promising; they accentuate the distinction between makers and takers and, while providing a minimal income, they do little to address inequality [9]. The main challenges in UBI implementation include the following: (i) identify those within the population who meet the eligibility criteria for receiving the grant; (ii)

979-8-3503-3535-4/23/$31.00 ©2023 IEEE
DOI 10.1109/DAPPS57946.2023.00013

ensure that each eligible person effectively receives the grant; and, (iii) design an effective oversight mechanism [10]. Unfortunately, existing blockchain-based platforms do not fully support the development of UBI solutions that solve these challenges, as demonstrated by a few projects currently under evaluation [11].

On the one hand, the current decentralized finance (DeFi) landscape, enabled by disruptive new technologies such as blockchain, provides a launch platform for new digital financial services that will eventually force governments and central banks to choose between banning, tolerating, or co-opting its innovations [12]. On the other hand, the intrinsic limitations of existing platforms make it challenging to adopt this technology for establishing legal tender digital currencies or implementing complex economic proposals, such as UBI [2].

To fill the above highlighted gaps, in this paper, we design a novel cryptocurrency with the ambition to keep most of the advantages of traditional cryptocurrencies while, at the same time, proving the necessary characteristics for implementing a Central Bank Digital Currency (CBDC). With this aim, we propose UBIC, a blockchain-less digital payment system. Unlike traditional cryptocurrencies, UBIC does not use blockchain technology to store the system's data. Instead, a set of semi-trusted nodes, called *bookkeepers*, manage and store transaction data in a decentralized way, i.e., by leveraging a Distributed Hash Table (DHT), giving high scalability to the system. As such, the transaction throughput is not limited by parameters related to blocks, such as size and creation time, making UBIC very efficient in terms of transaction verification time. In addition, the use of DHTs prevents the replication of all system data on every node (as in standard cryptocurrencies) while maintaining good performance and a tunable high level of security and resilience.

The transactions' validity is verified by a set of *validators*, coordinated by bookkeepers and organized in verification committees. The validation mechanism is designed to give each validator an equal probability of being selected for transaction verification. In addition, the computational effort needed to validate transactions is exceptionally lightweight and easily executable on resource-constraint devices, such as smartphones and consumer-level laptops. Furthermore, UBIC protects user privacy while ensuring the application of AML policies. All the above features make UBIC a cryptocurrency at par, if not superior to, existing cryptocurrencies. Though, its idiosyncratic features enable its adoption also as a CBDC, with the possibility to provide key services, such as Universal Basic Income. This latter use-case will be developed during the exposition of our proposal.

Roadmap. The rest of this paper is organized as follows. Section II discusses previous work related to competing cryptocurrencies. Section III presents UBIC's architecture from a high-level perspective, while Section IV introduces the adversarial model. Section V discusses in detail our transaction processing protocol. Finally, Section VI summarizes our contributions and concludes with some insights for future research.

II. Related Work

To the best of our knowledge, we are the first to propose a cryptocurrency that is not blockchain-based. However, to better highlight the advantages of our proposal with respect to competing solutions, in this section, we discuss a subset of the most representative cryptocurrencies on the market, selected in terms of capitalization, performance, and privacy properties. Bitcoin [13] and Ethereum [14] are, by far, the most known and capitalized cryptocurrencies in the world. At the time of writing, Bitcoin is based on Proof-of-work (PoW), a consensus mechanism that requires solving a cryptographic puzzle [15] in order to gain the right to add a new block to the blockchain. Ethereum, instead, switched to its Proof-of-Stake (PoS) mechanism in September 2022 to implement a more scalable and less energy-intensive architecture. Bitcoin is based on Unspent Transaction Output (UTXO), while Ethereum is account-based. Bitcoin only supports basic transactions aimed at transferring a certain amount of digital coins from one user to another. Instead, Ethereum also supports digital contracts, enabling a wide range of complex use-cases.

Despite being the most famous and capitalized cryptocurrency, Bitcoin has several limitations. First, the mining process is computationally intensive, causing the Bitcoin network to consume vast amounts of electricity. Then, newly minted coins are not distributed fairly, as the PoW protocol inherently favors miners with large computational resources. Last but not least, the network scalability is severely limited. Modern cryptocurrencies, such as Cardano, Polkadot, Solana, Avalanche, and Algorand [16], to cite a few, have tried to mitigate these problems, also improving network performance [17]. Proof-of-stake-based cryptocurrencies, for example, use a less energy-intensive consensus protocol, also aiming at increasing reward fairness. However, this goal is not always achieved. For instance, in the case of Polkadot, although the energy problem is well mitigated, the resulting network is far less democratic than expected [18].

Monero[1] and Zcash[2] are considered the most privacy-preserving cryptocurrencies on the market [19]. Monero was launched in April 2014 with the aim of solving privacy issues faced by other virtual currencies, such as Bitcoin. Based on the CryptoNote V 2.0 protocol [20], Monero is designed to ensure *unlinkability* and *untraceability*. In addition, it is able to hide the transaction amount [21]. Though, even such simple UTXO transactions can be exploited for some non protocol-compliant purposes [22]. Zcash is a cryptocurrency based on the Zerocash protocol, officially launched in October 2016. Zcash hides both the origin and destination addresses of every transaction, as well as its amount. Although giving its user a strong level of privacy, the Zerocash protocol requires a trusted party in order to set up some public parameters needed for the transaction. Consequently, any successful attack on the

[1]https://www.getmonero.org/

[2]https://z.cash/

trusted party (including the party itself misbehaving) results in a complete compromise of the coins' security.

III. System Overview

UBIC is a digital payment platform that bases its security and privacy properties on solid and well-known cryptographic techniques. Like any other cryptocurrency, its primary goal is to allow regular users to issue new transactions, verify their legitimacy and correctness, and keep track of users' transactions and account balances. UBIC's architecture, depicted in Figure 1, includes several actors. Specialized users, called **bookkeepers**, coordinate the transaction verification phase and store the system's data, e.g., transaction history and account balance, while **validators** verify the correctness of every transaction made by regular **users**—for these latter ones, their only role is to issue new transactions according to their needs. All the validators registered in the system are randomly divided into small groups—one group per bookkeeper. The groups are periodically reshuffled to avoid collusion among bookkeepers and validators.

In the following, we list the desired properties of our system, and how we achieve them:

- **Computationally lightweight and energy efficient Tx verification protocol:** Unlike PoW-based cryptocurrencies, UBIC does not require validators to solve a difficult cryptographic challenge to verify transactions. Instead, the computation power of the validators is fully dedicated to verify the legitimacy of the encrypted transactions, which reduces the overhead to the bare minimum. As a result, there is no mining involved, i.e., no energy waste. In addition, the validation protocol is lightweight, allowing its execution on resource-constraint devices, such as smartphones and Raspberry Pi devices.
- **Fair access to new coins:** Traditional cryptocurrencies use one of the two major classes of consensus algorithms: Proof of Work (PoW) or Proof of Stake (PoS). Inevitably, both end up benefiting users according to their availability of resources, be they computational or monetary. To avoid such an unfair mechanism, UBIC selects users (validators) randomly to verify new payment transactions and, once completed, they are compensated with transaction fees that are paid by the system itself. Consequently, all users have equal probability of being selected, and every user who stays online to verify transactions will receive their fair share of payments.
- **Decentralized ledger:** In standard cryptocurrencies, every mining node must store and maintain a global ledger. This solution can pose scalability issues when the number of users is in the order of millions or billions. Instead, UBIC relies on a number of *semi-trusted* parties to store and update, in a decentralized manner, the *account balance* and *transaction history* for every registered user.
- **Short verification time:** Once a bookkeeper receives a new transaction, it broadcasts it to the verification committee assigned to it. Then, the σ qualified validators will verify the transaction and send the verification proof to the bookkeeper. In order to consider a transaction as valid, the bookkeeper in charge must receive σ valid proofs, in case of *veto* voting settings, or at least $(\lfloor \sigma/2 \rfloor + 1)$ in case of *majority* voting. Since the required computation overhead is only limited to approving the legitimacy of the encrypted transactions, the required computation time is very low, i.e., a single validator takes, on average, around 80 ms to verify a transaction. The distributed nature of the validators and the selection technique used by UBIC guarantees that the overall verification operation is performed in real-time.

The system is built on top of a distributed hash table ($DHT_{Balance}$) that stores the balance of each registered account in the system. The balances are stored in an encrypted manner, where only the user having the correct secret key is able to decrypt. When two users want to perform a new transaction, they send the transaction to one of the bookkeepers and the latter broadcasts it to its own verification committee that is serving at that moment. The *qualified* validators exchange some messages with the users, in order to get the necessary proofs that the transaction is valid. This entails: (i) the proof that both balances are sufficient to perform the transaction; and, (ii) the sender and the receiver are debited/credited with the same amount, respectively. Note that the validators perform the verification step blindly without decrypting either of the balances. Moreover, the validators are oblivious to the *role* of each user (sender or receiver). It is important to note that the communication between the users and the validators is done through one of the bookkeepers, which acts as a proxy. Consequently, the users' identities are not disclosed to the validators. In addition to the DHT that stores the account balances, the system stores all the transactions generated and verified by the system in another DHT, called DHT_{Tx}.

A. Use-cases

The versatility and scalability of our solution allow for the implementation of a wide range of possible use-cases. In the following, we discuss two possible applications of UBIC.

Central Bank Digital Currency: This use-case can be implemented at different scales, i.e., national, federal, etc. On a national scale, the national central bank represents the higher entity responsible for registering new users after verifying their identities. Any other bank, legally operating inside the national borders, will run one or more bookkeepers. The validators are all citizens who are recipients of the income, while regular users are the end-users users of the digital currency. On a federal scale, the architecture will remain the same, with a small change in the roles. In the case of the European Union, for example, the higher authority would be the European Central Bank, while bookkeepers will be hosted by the national central bank of every member state.

University Cryptocurrency: UBIC can be used to implement a cryptocurrency sponsored by a university. In this case, the higher authority responsible for registering new users

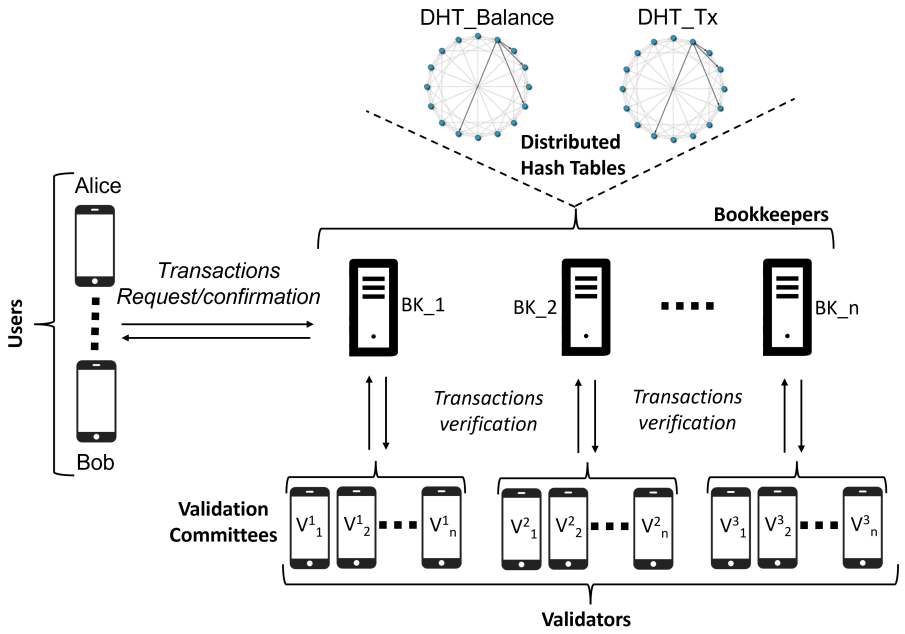

Fig. 1. System architecture

and verifying their identities would be the university itself, while each college will run one bookkeeper. The validators, recipients of the income, would be every student regularly enrolled in a degree program.

B. Privacy configurations

With respect to user privacy, our architecture can be modeled in different ways according to the specific use-case that is implemented. In the most stringent privacy policy, only the higher authority knows the user's real identity. It will first identify every new actor, e.g., user/bookkeeper/validator, by verifying their documents. Then, the higher authority registers the actor's public key into the system, authorizing him/her to operate on it. Bookkeepers only know other actor's public keys, while validators and users only know bookkeeper's public keys. Consequently, UBIC provides its final users with full anonymity on validators and other regular users, and pseudo-anonymity on bookkeepers. Furthermore, only the two users issuing a transaction know the amount of that transaction. Similarly, account balances are only known to the underlying account owners. This setting is sufficient to implement a KYC policy required for state-sponsored use cases, since the higher authority identifies every user registered into the system. However, this policy can be relaxed according to the specific use-case that is implemented. For example, the knowledge of the user's identity can be extended to bookkeepers, which represent the banks in the legal tender digital currency use-case.

IV. Adversarial Model and Assumptions

The objective of the adversary is to deliberately attempt to harm the system, for example, by tampering with and changing account balances or by having an illegitimate transaction being verified. In line with the literature, the adversary has bounded computational power and, therefore, it cannot compromise the underlying cryptographic primitives, except with negligible probability. The adversary has the ability to compromise a certain fraction of bookkeepers and validators, turning them *malicious*. At that point, the adversary has complete control over the compromised entities, for example, it can sign messages with their private keys, send messages on their behalf, and deviate arbitrarily from the protocol specification. In the following we list the general assumptions in our model:

- Bookkeepers and validators are semi-trusted entities, i.e., they behave in an *honest-but-curious* manner (they follow the protocol's specifications but attempt to derive additional information via the communication transcripts). However, they can be compromised, according to the adversary's capabilities, and become malicious.
- Each actor (user, validator, bookkeeper) is registered by a higher authority, which verifies their identity before authorizing them to operate in the system. The registration of a new user/validator account (public key) may involve an official document, e.g., a passport and, potentially, some sort of biometric verification.
- We assume the use of a certification authority (CA) that allows each user to verify: (i) the identity of the other participants; and (ii) the authenticity of the messages they receive through the communication channel (via the use of digital signatures).

V. Transaction Processing

This section describes UBIC's transaction processing protocol. Note that, in existing cryptocurrencies, such as Bitcoin

and Ethereum, the transaction verification process is global, i.e., it includes all the miners connected to the system at that moment. Instead, UBIC verifies each transaction by involving one bookkeeper and a number of validators. Furthermore, each bookkeeper can receive transaction requests and start the verification process at any time, independently from others. Each transaction is then verified by a subset of validators, called the *verification committee*.

A. Public key cryptosystem

Before delving into the details, we briefly describe the public key cryptosystem that encrypts the users' account balances. Our cryptosystem of choice is ElGamal, because it can be efficiently implemented using elliptic curve cryptography. Assume a multiplicative group $\mathbb{G}$ of prime order q with generator g. ElGamal's cryptosystem consists of a tuple $\Pi = (\mathsf{GenKey}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms, constructed as follows:

- GenKey: On input a group $\mathbb{G}$ (as described above), choose a uniformly random $x \in \mathbb{Z}_q^*$ and output a secret key x and a public key $h = g^x$.
- Enc: On input a public key h and a message $m < q$, choose a uniformly random $r \in \mathbb{Z}_q^*$ and output ciphertext $c = (g^r, h^{r+m})$.
- Dec: On input a secret key x and a ciphertext $c = (c_1, c_2)$, compute $c_1^{-x} \cdot c_2 = h^m$ and solve the discrete log problem to recover m.

B. Transaction request

Let us assume that Alice and Bob have agreed on a transaction where Alice is paying Bob an amount of a UBICs. The transaction request involves the following steps:

1) Alice generates the ciphertext related to her transaction amount as $\mathsf{Enc}(PK_A, a_A)$, and Bob will do the same with $\mathsf{Enc}(PK_B, a_B)$. In our example, $a_A = -a$ and $a_B = a$. They then agree on a bookkeeper to handle their request and separately send the transaction data to it. Alice will send $req_A = (PK_A, PK_B, \mathsf{Enc}(PK_A, a_A))$, while $req_B = (PK_A, PK_B, \mathsf{Enc}(PK_B, a_B))$ will be sent by Bob.
2) The bookkeeper that receives the requests merges req_A and req_B and completes the transaction data with Alice's and Bob's (encrypted) balances, after retrieving them from $DHT_{Balance}$. Therefore, the generated request is $Req = (req_A, req_B, \mathsf{Enc}(PK_A, b_A), \mathsf{Enc}(PK_B, b_B))$. Then, the bookkeeper broadcasts (Tx_{ID}, BK_i) to its attached validators, where $Tx_{ID} = H(r_t, Req)$ is the transaction ID and BK_i is the bookkeeper's own ID. Note that, r_t is a random seed that represents the system's current epoch. It will be discussed in more detail in Section V-D.
3) *Qualified* validators contact BK_i to receive the information necessary to verify the transaction. During the transaction verification phase, BK_i serves as a communication proxy between users and validators. In this way, the users who requested the transaction cannot identify the validators, thus preventing collusion among them.
4) Once all (or a majority of) validators confirm the validity of the transaction, BK_i requests $DHT_{Balance}$ to update the account balances for Alice and Bob.
5) When the balance update is completed successfully, BK_i adds Tx_{ID} to DHT_{Tx}, and notifies Alice and Bob. At this point, the transaction is officially verified and the two users may initiate a new transaction.

C. Transaction verification

While Req is being flooded through UBIC's P2P network, every potential verifier will check whether they are eligible to serve in the verification committee for transaction ID Tx_{ID}. This is done through a *cryptographic sortition* protocol, that we detail in Section V-D. The important thing to note here, is that it is infeasible to determine in advance the nodes that qualify for a specific transaction ID; indeed, each node makes that determination based on its private key, and must also provide the necessary *proof* for its qualification. Every validator (Victor) that qualifies for the committee will contact Alice and Bob through the bookkeeper who created the transaction, in order to receive the information required for the verification process. They will then invoke two separate Zero Knowledge Protocols (ZKP)—that can run concurrently—to verify that: (i) $a_A + a_B = 0$; and, (ii) the updated account balances are both positive.

The first protocol involves all three parties (Alice, Bob, Victor) and is executed as follows. Note that, all exchanged messages (padded with the current random seed r_t) should be signed by the sender's private key. Let (x_A, h_A), (x_B, h_B) be Alice's and Bob's key pair, respectively.

1) Victor chooses uniformly random numbers $r, r_A, r_B \in \mathbb{Z}_q^*$ and computes $\mathsf{Enc}(PK_A, r \cdot a_A + r_A) = (g^{r_1}, h_A^{r_1 + r \cdot a_A + r_A})$ and $\mathsf{Enc}(PK_B, r \cdot a_B + r_B) = (g^{r_2}, h_B^{r_2 + r \cdot a_B + r_B})$. Then it sends the first ciphertext to Alice and the second to Bob. He also sends to both parties a commitment $H(r||r_A||r_B)$ of his random values.
2) Alice computes $(g^{r_1})^{-1} \cdot (h_A^{r_1 + r \cdot a_A + r_A})^{x_A^{-1}} = g^{r \cdot a_A + r_A}$, and sends the result to Bob.
3) Bob computes his share $g^{r \cdot a_B + r_B}$ accordingly, and multiplies the two values. He sends the result to Victor, i.e., $g^{r \cdot (a_A + a_B) + r_A + r_B}$.
4) Victor verifies that the result is $g^{r_A + r_B}$, which implies that $a_A + a_B = 0$. If the verification is successful, it reveals r, r_A, r_B to both Alice and Bob, who then verify Victor's initial commitment $H(r||r_A||r_B)$.

The second protocol is invoked by both Alice and Bob that interact independently with Victor. They have to prove that their *new* account balances are positive, i.e., there was no payment without the sufficient funds. Let us consider Alice, and assume her new account balance b_A, in binary, is $x_{k-1}x_{k-2}\cdots x_1 x_0$, where k is sufficiently large to store any realistic amount (e.g., $k = 64$ bits). The protocol works as follows:

1) Alice encrypts all bits with her public key, i.e., $\mathsf{Enc}(PK_A, x_i), \forall i$, and sends them to Victor.
2) Victor reconstructs Alice's alleged account balance, $\mathsf{Enc}(PK_A, \hat{b}_A)$, and subtracts it from the actual one that is computed from the previous account and update ciphertexts, i.e., it produces $\mathsf{Enc}(PK_A, b_A - \hat{b}_A)$. Note that Victor also randomizes the result, by adding an encryption of zero. Given that the output is supposed to be an encryption of zero, i.e., it should be equal to (g^r, h_A^r), for some unknown random r. Now Alice has to prove that she knows the discrete log of h_A^r base g^r (which is her secret key x_A). This is done with Schnorr's non-interactive identification protocol [23], as shown in the following steps. First, Victor sends (g^r, h_A^r) to Alice.
3) Alice selects a random $v \in \mathbb{Z}_q^*$ and computes $s = (g^r)^v$.
4) Alice computes $c = H(s||g^r||h_A^r)$ and then computes $t = v - c \cdot x_A \bmod q$. She sends (s, t, c) to Victor.
5) Victor verifies that $s = (g^r)^t \cdot (h_A^r)^c$.

One potential problem is that Alice might cheat and send much larger values for x_i instead of 0s and 1s. Therefore, Victor has to verify that each one of Alice's ciphertexts corresponds to an encryption of 0 or 1. For that purpose, we will use the disjunction of two ZKPs for the discrete logarithm (similar to Schnorr), where the prover can convince the validator that he knows one out of two discrete logarithms (but not which one). This is done as follows:

1) For every bit x_i, Victor sends to Alice $(g^{r_i}, h_A^{r_i+x_i}, h_A^{r_i+x_i-1})$, i.e., the encryption of x_i (that Alice herself sent earlier) and $(x_i - 1)$. Alice will know only one of the discrete logs, corresponding to the real value of x_i. For example, if $x_i = 0$, she will know the discrete log of $h_A^{r_i+x_i}$ base g^{r_i}.
2) Let us denote the triplet above as (g, h_1, h_2), i.e., the values sent by Victor for one of Alice's encrypted bits x_i. Alice sends to Victor the tuple $(y_1, y_2, c, c_1, c_2, s_1, s_2)$ that is computed as follows:
   - If $x_i = 0$: Choose random $s_2, c_2, t \in \mathbb{Z}_q^*$. Set $y_1 = g^t$ and $y_2 = g^{s_2} h_2^{-c_2}$. Compute $c = H(y_1||y_2||g||h_1||h_2)$, and set $c_1 = c - c_2 \bmod q$ and $s_1 = t + c_1 \cdot x_A \bmod q$.
   - If $x_i = 1$: Choose random $s_1, c_1, t \in \mathbb{Z}_q^*$. Set $y_1 = g^{s_1} h_1^{-c_1}$ and $y_2 = g^t$. Compute $c = H(y_1||y_2||g||h_1||h_2)$, and set $c_2 = c - c_1 \bmod q$ and $s_2 = t + c_2 \cdot x_A \bmod q$.
3) For every tuple received by Alice, Victor verifies that:
   - $c = c_1 + c_2 \bmod q$.
   - $g^{s_1} = y_1 \cdot h_1^{c_1}$.
   - $g^{s_2} = y_2 \cdot h_2^{c_2}$.

The entire transaction verification process is depicted in Fig. 2. When BK_1 retrieves the encrypted balances from the DHT, the corresponding nodes will lock them until the transaction is complete, in order to avoid inconsistencies generated by concurrent transactions involving either user. BK_1 generates the transaction and broadcasts it to its validation committee. If BK_1 receives the minimum required number of proofs—and all (or the majority) of them, depending on the system configuration, are correct—it sends the transaction along with the verification transcripts to the DHT for updating the account balances. Finally, every node on the DHT responsible for storing the balances of either user will verify the data received from BK_1. If the data are consistent with the old balances stored on the DHT and the proofs are correct, the corresponding DHT nodes will update and unlock the corresponding balances. At this point, BK_1 sends a confirmation message to the two users. A transaction can fail in three cases: (i) if at least one of the two balances is already locked when BK_1 attempts to read the balances; (ii) if BK_1 does not receive all the required proofs (according to the system setup) from the verification committee; and (iii) if the validation data sent to the DHT to update the balances are not correct. In a real implementation, however, the first case, i.e., when a balance is already locked, can be handled by implementing a queue on the DHT side.

It is important to note that, even if represented as a single entity in Fig. 2, the DHT is a distributed storage system, where every bookkeeper is a node and data redundancy is enabled. Consequently, data consistency is preserved, and any concurrent requests are handled properly. In addition, in our implementation, we forced the DHT update function to check the correctness of the transaction validation data. Therefore, in order to update a user's balance, every bookkeeper (responsible for that balance in the DHT network) must verify and agree on the transaction validation data, i.e., proof of qualification and transaction verification proof.

D. Sortition protocol

As in Algorand [16], our sortition protocol is implemented with a verifiable random function (VRF) [24]. Sortition allows us to select a random verification committee for each transaction, without the ability to predict the committee members beforehand. We opted for an elliptic curve-based implementation of the VRF, where the input is (Tx_{ID}, r_t). In particular, consider a validator with a key pair (x, h). The input (Tx_{ID}, r_t) is first mapped to an elliptic curve point y and the validator outputs (γ, π), where $\gamma = y^x$ and π is a ZKP for the discrete log. Upon receiving the output, the bookkeeper uses public key h and π to verify the correctness of γ. If the verification is successful, it computes $H(\gamma)$ and determines that the owner of h is a qualified validator *iff* the leading number of zero bits in $H(\gamma)$ is larger than a threshold.

As mentioned previously, r_t is a random seed that is jointly computed by the bookkeepers via a ring signature algorithm [25]. The random seed is periodically updated (i.e., it represents the current epoch) and is leveraged to provide randomness in several aspects of our protocol, including the verification committee selection and the periodic re-shuffling of the validators among the bookkeepers.

Any of the bookkeepers can initiate the ring signature round—typically, after reaching an agreed upon number of verified transactions. In the first round, the bookkeepers jointly generate the random seed and, in the second round, the new seed is broadcasted and verified by each one of the

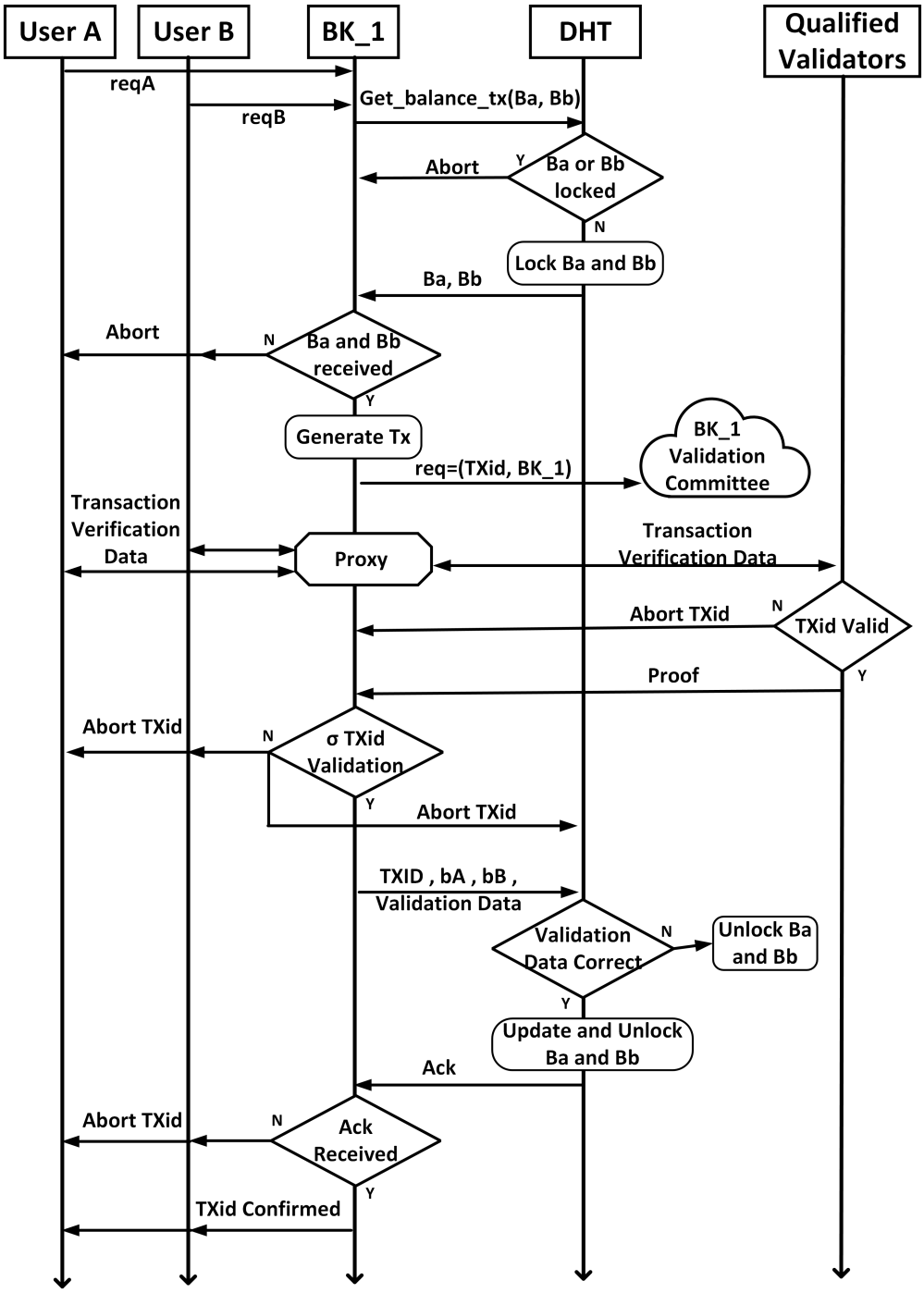

Fig. 2. Transaction verification process

bookkeepers. Once r_t is updated, the version state of the system parameters is incremented. All other parties (users and validators) must ensure that they are using the latest version state before issuing or validating a transaction. Note that the latest random seed is completely unpredictable by any party of the system, including the initiator of the ring signature round. Moreover, the new random seed can be verified by any party, using only the previous seed value and the public keys of the bookkeepers (which are publicly known).

VI. Conclusions

To the best of our knowledge, UBIC is the first blockchain-less cryptocurrency proposal. Our solution avoids the classic weaknesses of existing blockchain-based cryptocurrencies, such as limited scalability and unfair reward distribution, while keeping their advantages. Moreover, given its breakthrough architecture, it removes some of the classical attacks mainstream solutions are subject to, such as 50%-, Sybil-, and eclipse-attack, to cite a few. Additionally, UBIC provides a high level of security and privacy for users.

UBIC stores transaction data in a decentralized way, using DHTs, that are maintained by semi-trusted users, called bookkeepers. The transaction verification process is distributed among independent users, called validators, who receive a fairly shared reward as compensation. The user privacy level is flexible, allowing the application of AML policies, which makes UBIC versatile enough to support a wide range of use-cases, including central bank digital currencies.

To summarize, UBIC enjoys clear advantages with respect to traditional cryptocurrencies, and is perfectly suited to support high-impact use-cases, such as UBI and CBDC.

Acknowledgments

This publication was partially supported by the Qatar National Research Fund (QNRF), a member of The Qatar Foundation, through the awards [NPRP-S-11-0109-180242] and [NPRP11C-1229-170007]. The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of the QNRF.
Dr. Roberto Di Pietro and Dr. Elmahdi Bentafat produced part of their contributions while they were at HBKU-CSE.

References

[1] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 104–121.

[2] R. Di Pietro, S. Raponi, M. Caprolu, and S. Cresci, *New Dimensions of Information Warfare*. Springer International Publishing, 2021, vol. 84, part of the Advances in Information Security book series. [Online]. Available: https://doi.org/10.1007/978-3-030-60618-3_1

[3] B. P. Rankhambe and H. Kaur Khanuja, "A comparative analysis of blockchain platforms – bitcoin and ethereum," in *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 2019, pp. 1–7.

[4] S. Alzahrani and T. U. Daim, "Analysis of the cryptocurrency adoption decision: Literature review," in *2019 Portland International Conference on Management of Engineering and Technology (PICMET)*, 2019, pp. 1–11.

[5] K. Kolachala, E. Simsek, M. Ababneh, and R. Vishwanathan, "Sok: Money laundering in cryptocurrencies," ser. ARES 21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3465481.3465774

[6] D. Arli, P. van Esch, M. Bakpayev, and A. Laurence, "Do consumers really trust cryptocurrencies?" *Marketing Intelligence & Planning*, vol. 39, no. 1, pp. 74–90, 2020.

[7] J. L. Moraes and C. Freire, "Locally universal: Universal basic income policies in the post-pandemic world-order," *GLOCALISM: Journal Of Culture, Politics And Innovation*, vol. 2, 2020. [Online]. Available: https://glocalismjournal.org/wp-content/uploads/2020/10/Lucchesi-Moraes_Freire_gjcpi_2020_2-1.pdf

[8] J. U. Bidadanure, "The political theory of universal basic income," *Annual Review of Political Science*, vol. 22, no. 1, pp. 481–501, 2019. [Online]. Available: https://doi.org/10.1146/annurev-polisci-050317-070954

[9] N. Berggruen, "Here's how blockchain can reduce inequality," The Washington Post. Retrieved, 2018. [Online]. Available: https://www.washingtonpost.com/news/theworldpost/wp/2018/01/29/blockchain/?noredirect=on

[10] J. De Wispelaere and L. Stirton, "A disarmingly simple idea? practical bottlenecks in the implementation of a universal basic income," *International Social Security Review*, vol. 65, no. 2, pp. 103–121, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-246X.2012.01430.x

[11] P. Howson, "Distributed degrowth technology: Challenges for blockchain beyond the green economy," *Ecological Economics*, vol. 184, p. 107020, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921800921000781

[12] M. Raskin and D. Yermack, "Digital currencies, decentralized ledgers, and the future of central banking," National Bureau of Economic Research, Cambridge, USA, Working Paper 22238, May 2016. [Online]. Available: http://www.nber.org/papers/w22238

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

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

[15] I. M. Ali, M. Caprolu, and R. Di Pietro, "Foundations, properties, and security applications of puzzles: A survey," *ACM Comput. Surv.*, vol. 53, no. 4, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3396374

[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[17] B. Nasrulin, M. De Vos, G. Ishmaev, and J. Pouwelse, "Gromit: Benchmarking the performance and scalability of blockchain systems," in *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 2022, pp. 56–63.

[18] H. Abbas, M. Caprolu, and R. Di Pietro, "Analysis of polkadot: Architecture, internals, and contradictions," in *2022 IEEE International Conference on Blockchain (Blockchain)*, 2022, pp. 61–70. [Online]. Available: https://doi.org/10.1109/Blockchain55522.2022.00019

[19] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in zcash," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 463–477.

[20] N. van Saberhagen, "Cryptonote v 2.0," 2013, (Last accessed: 20/02/2023).

[21] D. A. Wijaya, J. K. Liu, R. Steinfeld, D. Liu, and J. Yu, "On the unforkability of monero," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 621–632.

[22] M. Caprolu, M. Pontecorvi, M. Signorini, C. Segarra, and R. Di Pietro, "Analysis and patterns of unknown transactions in bitcoin," in *2021 IEEE International Conference on Blockchain (Blockchain)*, 2021, pp. 170–179.

[23] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[24] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[25] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *International conference on the theory and application of cryptology and information security*. Springer, 2001, pp. 552–565.