Commit-Chains Without Smart Contracts for Blockchain Applications in Local Communities

Fadi Barbàra
fadi.barbara@unito.it
University of Turin
Torino, Italy, Italy

Flavia Fredda
flavia.fredda@unicam.it
University of Camerino
Camerino, Italy, Italy

Claudio Schifanella
claudio.schifanella@unito.it
University of Turin
Torino, Italy, Italy

ABSTRACT

Blockchain technology's scalability is a major challenge that hampers its adoption and utility, particularly in resource-constrained local communities. In this paper, we present TokenCards [1], a novel commit-chain tailored to local communities.

Our Layer 2 proposal utilizes an RSA accumulator, selected for its efficiency over Merkle Trees and ability to provide non-membership proofs, and propose a novel blockchain-agnostic time-stamping mechanism called BATS to certify off-chain operations. Our approach allows TokenCards to offer a cost-free service to users while incurring minimal costs for business owners. This strategy supports local economies by digitizing transactions, enhancing privacy.

Results indicate that RSA accumulators excel in performance in terms of costs, computation and time complexity outperforming traditional methods. Moreover, the implementation of TokenCards represents a significant advancement in the practical application of blockchain technology in local communities, providing a user-friendly, scalable, and economically feasible solution. Through this innovation, we contribute to the field by offering a system that imposes minimal costs, primarily on the business owner, and significantly simplifies the user experience in local economic transactions.

CCS CONCEPTS

• **Security and privacy** → *Digital signatures*; **Hash functions and message authentication codes**; • **Applied computing** → **Digital cash**; **Electronic funds transfer**; • **Networks** → Network File System (NFS) protocol; Cross-layer protocols.

KEYWORDS

blockchain, tokens, rsa accumulator, local communities, local economy

ACM Reference Format:
Fadi Barbàra, Flavia Fredda, and Claudio Schifanella. 2024. Commit-Chains Without Smart Contracts for Blockchain Applications in Local Communities. In *International Conference on Information Technology for Social Good (GoodIT '24), September 04–06, 2024, Bremen, Germany.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3677525.3678664

[1] https://github.com/flaviafredda/token-cards

GoodIT '24, September 04–06, 2024, Bremen, Germany

ACM ISBN 979-8-4007-1094-0/24/09
https://doi.org/10.1145/3677525.3678664

1 INTRODUCTION

It is widely known that permissionless and decentralized blockchains "cannot scale", i.e. transaction throughput do not improve as more nodes participate in the network. This is noted both by the industry practitioners (see the ubiquity of Buterin's scalability trilemma [7], a bi-dimensional trade-off asserting that it is possible to pick at most two qualities among decentralization, security and high transaction throughput) and the great body of literature (see e.g. [38]). Scalability is a significant issue impacting the usability of digital applications in local communities, which otherwise could foster a great variety of Grassroots Innovation (GRI). These innovations are typically driven by economically disadvantaged individuals who apply practical and creative solutions using indigenous knowledge to solve localized problems [16]. However, the difficulty in extending and improving everyday user-friendly applications, such as the throttled transaction throughput leading to prohibitively high transaction fees that generally exceed day-to-day transaction values, presents a major obstacle. Therefore, addressing these prevailing technological challenges is crucial to create opportunities for such innovations to flourish.

To solve the scalability problem, a plethora of potential solutions have been proposed, both in academic literature and by the industry. The solutions range from more centralized blockchains such as Solana [36], to permissioned blockchains such as Hyperledger [12], pegged sidechains [2, 11, 29], sharding [35], payment channels [32], state channels [27] or validating bridges [23]. The interested reader can see in the work by Hafid et al. [17] a comprehensive survey on the topic. However, none of them is tailored to the needs of local communities.

The study of the aforementioned solutions let researchers understand the inherent characteristics of the problem and get new insights. Probably as a consequence of the aforementioned scalability trilemma, recent solutions nearly always assume moving the vast majority of the transactions off-chain. This can be seen in both of the most popular permissionless blockchain projects, namely Ethereum [6] and Bitcoin [26]. The first is following a "roll-up approach" (off-chain batches of transactions without with on-chain certification), as expressively announced by the Ethereum de facto leader Vitalik Buterin in a series of blog posts [8, 9]. The latter is mostly focused on its most popular payment channel network, Lighting Network [32], while experiments with the off-chain smart contract system RGB [33] and ZK-based roll-ups such as BitVM [21].

The clear advantage of removing transactions from the main blockchain, also known as Layer 1 or L1, is the decoupling of scalability and security. By achieving a higher transaction throughput outside the main blockchain the security and trust assumptions regarding the L1 stay the same and no ulterior study or redesign of consensus mechanisms is needed. Conversely, a notable drawback of this method is the fragmentation of liquidity, which refers to the reduction in the availability of assets or tokens that can be quickly bought, sold or otherwise used in the network. Typically, distinct L2s necessitate external liquidity for initiation and usage, which is derived from the parent L1. This poses a challenge for the integration of new users. At other times, liquidity is generated (or *issued*) through questionable practices, such as "airdrops," where tokens are distributed to new users based on specific and arbitray rules.

Another issue with current L2 methods is that they are global in nature. Unstated assumption is that the users should interact even if geographically apart and do not know each other. Consequently, every security model treats users as rational, active malicious attackers until proven otherwise, an assumption that goes against everyday experience

We claim that the union of these two facts is the major reason contributing to a lack of adoption in everyday systems of blockchain technology, especially in local communities. This despite the fact that the decentralization and data integrity properties of blockchain based systems and protocols are major advantages in user facing systems.

Contributions: To investigate our claim, we propose a change of attitude. In this paper, we present an off-chain architecture whose assumption is that users are geographically close and does not require any outside liquidity to start participating in the system, while not engaging in speculation-fueled airdrops. We achieve the latter by requiring backing of the token issuance via real physical objects. To show the feasibility of our approach, we also present an implementation, TokenCards, that can be thought of as a fidelity card for local commerce. Starting from our previous work, Locale [3], we extend it by adding token transfer, more efficiency and stronger cryptographic guarantees while still retaining the user-friendly experience necessary for adoption by local businesses and consumers. To accomplish that, we developed a new time-stamping mechanism (named BATS) that is blockchain agnostic, besides being cheaper and more private, which can be of independent interest. The goal is to foster local economic growth but strengthen community bonds through secure and efficient technological interactions.

We also provide the proof of concept implementation of TokenCards[2]. In other words, TokenCards is an efficient and usable extension of Locale (see Sections 2.3 and 6.2 to understand the details of this difference)

Organization: The rest of the paper is structured as follows: Section 2 discusses previous work in the field, Section 3 introduces the basic concepts and technologies used in our work, namely Accumulators (Section 3.1), and Blockchain Certification (Section 3.2). In Section 4, we presents the design of our solution, including the actors involved and the threat model, while in Section 5, we describe our implementation of the proposed solution. Finally, in Section 6, we provide an evaluation of our solution. In Section 7, we conclude the paper and discuss potential future work.

[2]The project repository can be found at: https://github.com/flaviafredda/token-cards

2 RELATED WORKS

In the following we give an overview of current proposals for commit chains. For a general overview on Layer2 systems, the interested reader can see systematization of knowledge by Gudgeon *et al.* [15] and the taxonomy by Jourenko *et al.* [18]. For a comparison between state of the art commmit-chain proposals and our proposal see Table 1.

2.1 NOCUST

The work by Khalil *et al.* [19] introduced the concept of Commit-Chains. The system is based on the interaction between an Operator and a smart contract deployed on an EVM blockchain. The users of NOCUST can transfer coins off-chain, although after a registration through the operator and a deposit of funds. Registration, deposit and withdraw of funds are on-chain operations, so the user has to pay the fees of the transaction in addition to providing external liquidity to the system. Consequently, due to the deposit, the NOCUST commit-chain lets users transfer pegged token already present on the blockchain. If the operator is honest, the total sum of funds in the commit-chain is equal to the totality of deposits of the users.

User are expected to be periodically online and monitor the blockchain to avoid loss in case of malicious behaviour by the operator. In case of malicious actions on the part of the operator, users have a window-period to withdraw the funds. Therefore as long as the user has (periodic) access to the blockchain where the smart contract is deployed, then the user is able to recover funds in case of malicious actions.

2.2 Plasma Cash

G. Konstantopoulos in [20] proposes a Layer 2 scalability solution named *Plasma Cash*. *Plasma Cash* is a Plasma [31] construction. Plasma has been probably the first proposal to introduce *sharding* to the ethereum blockchain. Plasma cash implementation uses NFTs and Sparse Merkle Trees (SMTs) to reduce data storage and bandwidth requirements for users.

NFTs make tokens unique and allow them to have a unique history, so when a user receives a coin, they only need to monitor the history of that specific coin rather than the entire Plasma chain. This reduces the user's checking requirements to the NFTs they possess, in contrast to Plasma, which necessitates that users monitor the entire Plasma chain to ensure transaction validity.

SMTs are used for efficient verification of (non)-inclusion of a transaction involving a NFT in a block.

The overall system scales down the storage and bandwidth requirements with respect to Plasma, but it still requires heavy storage and bandwith burden on the coin's senders, given that the proofs required to send a coin in *Plasma Cash* are linear on the number of blocks.

2.3 Locale

Our current research builds directly upon the Locale protocol introduced by Barbàra *et al.* [3], leveraging its foundational blockchain-based methodologies to further enhance and expand its capabilities.

Table 1: Comparison of different Commit-Chain constructions.

	On-Chain Check	Off-Chain Transfer Possible	Registration Exemption	Security Assumptions	Different Tokens Support	Type of Accumulator	Smart Contract Independence
NOCUST	Periodic	Yes	No	Monitoring	No	Merkle Tree	No
Plasma Cash	Periodic	Yes	No	Monitoring	No	Merkle Tree	No
Locale	Not Needed	No	Yes	N/A	Yes	Merkle Tree	Yes
TokenCards	Not Needed	Yes	Yes	Not needed	Yes	RSA	Yes

It introduces Locale, a novel blockchain-based protocol designed to digitize traditional paper loyalty systems through a new time-stamping mechanism that aids fidelization processes in local communities. The simplicity of the Locale protocol guarantees its further developments to have user-friendly interfaces. It encourages then community participation without demanding unnecessary complexities, such for example the necessity of holding cryptocurrency funds on a Layer 2, thereby lowering the entry barrier for users.

However, while the Locale protocol provides a proof of concept for transitioning from paper-based systems to a digital platform, it is not without limitations. The implementation uses a Merkle Tree as an accumulator (Section 3.1) to maintain the integrity and verifiability of transactions . Although secure, this choice is computationally intensive and poses challenges for dynamic updates which are crucial for businesses experiencing frequent purchases. Additionally, the current design of Locale does not support the transfer of tokens between customers, restricting its functionality and potential for broader adoption within community-based loyalty programs.

3 PRELIMINARIES

3.1 Accumulators

An accumulator is a cryptographic primitive whose goal is to succinctly represent a set of elements [5], with the capability to later prove the inclusion of any element in the set without revealing the entire set. Here *succinctly* means that the space complexity of the accumulator is constant in the number of elements in the set.

Accumulators can be seen in four kinds: (dynamic) RSA accumulators [10], Merkle Trees [24, 25], Bilinear accumulators [28] and Expressive set accumulators [37]. For a general survey on accumulators see the work by Loporchio et al. [22]. In TokenCards we decided to deviate from current state of the art and use RSA accumulators instead of Merkle trees. Advantages of this choice can be seen in Sections 6.2 and 6.3. Therefore the definitions in this section will be done in that direction. We made this choice for reason of efficiency and trustlessness. See Section 6 for more details.

A RSA-accumulator (from now on, simply *accumulator*) **Acc**, is a 7-tuple of functions such that:

- $(\mathcal{A}_0, N) \leftarrow \textbf{Acc.Setup}(1^\lambda)$ is a probabilistic algorithm that initializes the accumulator with a biprime modulus $N = pq$ and the output of the Euler's totient function $\phi = \phi(p, q) = (p-1)(q-1)$, p and q strong prime numbers randomly chosen, and a base $\mathcal{A}_0$. Primes as large as λ bits, the security parameter. Typically each prime is 1024 bits [30]. We stress the fact that $(\mathcal{A}_0, N)$ are *public* parameters, while the strong primes p and q a the value ϕ are kept *private* by the creator of the accumulator. This operation is done only once. We may call the creator of the accumulator the *owner* of the accumulator.
- $\mathcal{A}_{i+1} \leftarrow \textbf{Acc.Add}(\mathcal{A}_i, N, x)$ is a deterministic algorithm that add a new token in the accumulator. It takes as input the current state of the accumulator $\mathcal{A}_i$, the biprime modulus N and an prime number x, and outputs a new state of the accumulator, typically by computing a new value as
$$(\mathcal{A}_i)^x \mod N. \tag{1}$$
Note that this operation can be efficiently done only by who knows p and q and that the current state of the accumulator is public parameter.
- $\mathcal{A}_{i+1} \leftarrow \textbf{Acc.Rm}(\mathcal{A}_i, N, x)$ is an algorithm that modifies the accumulator to remove an element x. In practice that function is instantiated by using $\textbf{Acc.Add}(\mathcal{A}_i, N, x^{-1} \mod \phi)$. It is easy to see the correctness of this operation by looking at Equation (1) and noting that $xx^{-1} \equiv 1 \mod \phi$. Note that this operation can be efficiently done only by who knows p and q, i.e. the owner.
- $\mathcal{W}_i \mid err \leftarrow \textbf{Acc.GenMemProof}(\mathcal{A}_i, N, x, X)$ is an algorithm that generates a proof of membership for an element x if included in $\mathcal{A}_i$ without revealing any other elements, err otherwise. In practice assume $X = \{x_1, x_2, \ldots, x_n\}$ is the list of n elements included in $\mathcal{A}_i$. To prove that $x \in X$ at state i, the owner of the accumulator creates a *membership witness* $\mathcal{W}_i$:
$$\mathcal{W}_i = \mathcal{A}_0^{\prod_{y \in X \setminus \{x\}} y} \mod N \tag{2}$$
where $X \setminus \{x\}$ represents the set X excluding the element x. To accomplish this, the owner has to keep track of all the elements x_k, $k = 1, \ldots, n$ in order to efficiently compute $\prod_{y \in X \setminus \{x\}} y$.
- $\mathcal{N}_i \mid err \leftarrow \textbf{Acc.GenNonMemProof}(\mathcal{A}_i, N, x, X)$ is an algorithm that generates a proof of non-membership for an element $x \notin X$ and x prime and outputs err otherwise. This confirms that x has not been included in the accumulator in a trustless manner; without this capability, the verifier would have to rely on the assumption that the **Acc.GenMemProof** function has returned an error, and would not be able to independently verify that x is indeed absent from the accumulator.
In practice, consider X as before and let $p = \prod_{y \in X} y$. Since x prime, then the Greatest Common Divisor $\textbf{gcd}(p, x) = 1$, or equivalently there exist numbers a and b such that

$a \cdot p + b \cdot x = 1$ (see the Extended Euclidean Algorithm [34]). Then $\mathcal{N}_i = (a, b)$ (see **Acc.VerNonMemProof** on how this proof is verified)

- $0|1 \leftarrow$ **Acc.VerMemProof**$(\mathcal{A}_i, N, x, X, \mathcal{W}_i)$ is a verification algorithm that, given at least an element x, a proof of membership $\mathcal{W}_i$, and the current state of the accumulator $\mathcal{A}_i$, confirms whether x is indeed a member of the accumulator by checking:

$$(\mathcal{W}_i)^x \stackrel{?}{\equiv} \mathcal{A}_i \mod N. \tag{3}$$

  As seen by Equation (3), wponentiation can be done equivalently even if x is the product of t elements, i.e. $x = \prod_{i=1}^{t} x_i$. With this notation, Equation (2) still stands. Verifying many token at once is called *batch* verification.
- $0|1 \leftarrow$ **Acc.VerNonMemProof**$(\mathcal{A}_i, x, \mathcal{N}_i, \mathcal{A}_0)$ is a verification algorithm that, given an element x, a proof of non-membership $\mathcal{N}_i = (a, b)$, and the current state of the accumulator, confirms whether x is indeed not a member of the accumulator by checking:

$$(\mathcal{A}_i)^a (\mathcal{A}_0)^{x \cdot b} \stackrel{?}{\equiv} \mathcal{A}_0 \mod N. \tag{4}$$

Each component of the tuple is designed to support the secure, efficient, and verifiable accumulation of data, ensuring the integrity and confidentiality of the accumulated information.

3.2 Blockchain Certification and BATS

A *timestamp* serves as proof that a document existed in a particular state before a specific moment, thereby assigning a verifiable date to that document (e.g., postmark). Digital data can be securely timestamped by recording its value in a public public board with verifiable integrity, e.g. in a blockchain via a transaction. If the timestamp is obtained using a cryptographically secure hash function, the likelihood of a hash collision is negligible, as defined by the properties of the hash function. This way, the timestamp acts as a unique digital fingerprint of the document, virtually eliminating the possibility of duplication. Securely recording timestamps is also known as *certification*.

For blockchain-based timestamping and data certification, several alternatives exist on platforms like Bitcoin, Ethereum and Solana. The first and most famous if probably OpenTimeStamps[3] for the Bitcoin blockchain. Other alternatives such as OriginStamp[4] work on multiple blockchains via specific mechanisms.

Generally, current solutions use Merkle Trees as accumulators to reduce the number of transactions on the blockchain, and embed the root in blockchain-specific methods. For example, OpenTimeStamps uses the `OP_RETURN` Bitcoin opcode to embed the root in the transaction. Ethereum specific solution rely on smart contract to store the root hash in the blockchain, as done by OriginStamp.

The problem with these methods is that they are not blockchain agnostic: since the owner has to decide which blockchain to use given a service, the owner (and the local community in general) is then tied to one technological solution or project (and therefore one company, in the majority of cases). In the following we present a novel certification mechanism that only relies on elliptic curve cryptography operations, and thus can be used in any blockchain. We extend the *pay-to-contract* protocol by Gerhardt and Hanke [14].

We say that a Blockchain Agnostic Timestamping Solution (BATS) is a tuple of algorithms (**BATS.Cr**, **BATS.Pub**, **BATS.Ver**) such that:

- $ts \leftarrow$ **BATS.Cr**$(C, \mathcal{H}, data)$ is the cryptographic algorithm that given the curve parameters C, the hash function $\mathcal{H}$ and the data to be hashed, produces a timestamp ts.
- $tx \leftarrow$ **BATS.Pub**$(C, ts, sk_{\mathbf{from}}, pk_{\mathbf{orig}})$ is the cryptographic algorithm that given the blockchain curve parameters C, the timestamp ts and the private key $sk_{\mathbf{from}}$, produces the certification public key $pk_{\mathbf{cert}}$ that can be used to verify the timestamp and sends the transaction to it, obtaining hash tx. The public key $pk_{\mathbf{cert}}$ can be obtained doing:

$$pk_{\mathbf{cert}} = pk_{\mathbf{orig}} + (ts \cdot C.G) \tag{5}$$

  where $C.G$ is the generator of the curve C. Note that no block train is needed and curve is passed as parameters so any curve and blockchain can be used. In another words this is blockchain agnostic.
- $0|1 \leftarrow$ **BATS.Ver**$(C, \mathcal{H}, data, pk_{\mathbf{to}}, tx)$ is the cryptographic algorithm that given the curve parameters C, the hash function $\mathcal{H}$ and the data to be hashed, and the transaction tx, verifies the timestamp.

As demonstrated by the preceding functions, no accumulation is performed automatically: the decision to use data resulting from an accumulator for timestamping rests solely with the BATS user. This approach significantly enhances the flexibility available to BATS users.

4 DESIGN

Actors. Actors within the system are identified by their association with the client they operate for. Hence, references to actors utilize neutral pronouns such as "it" or "them".

- *Operator*: Acts as the instantiator and manager of the commit-chain. The operator is distinguished by a unique private-public key pair (sk_O^b, pk_O^b), managing all transactions, including token generation and expenditure. In practice the operator comprises of a server and a client, which can be a smartphone application.
- *Users*: Engage with the system to receive, spend, and transfer tokens issued by the operator. Users do not need to register to TokenCards. They are only required to use the appropriate client, generally a smartphone app.
- *Blockchain*: A distributed ledger that is accessible for monitoring and/or writing by any party.
- *Decentralized File System (DFS)*: A repository where any entity can upload files (e.g., IPFS). Modifications to the files are restricted to the uploader.

We published the code in https://github.com/flaviafredda/tokencards for the project.

Threat Model. We assume that all cryptographic primitives are secure and follow the recommendations. In particular we assume that operator and users use a permissionless blockchain based on ECC using groups where the discrete logarithm is hard and where

[3] https://dgi.io/ots/
[4] https://originstamp.com/blog/what-is-blockchain-based-timestamping

the operator has less than one-third of the resources needed to participate in the consensus algorithm (be that hash rate for PoW blockchains or stake for PoS ones). We assume that users can read the blockchain whenever they want to (even if they are *not* required to do that at any specific time). We assume that the primes p and q chosen by the operator during **Acc**.**Setup** (Section 3.1) are strong so that the Strong RSA Assumption holds and the accumulator can be considered collision free [4].

We assume the operator is able to keep public checkpoints of the current state of the commit-chain. Furthermore we assume any attacker is capable of corrupt the commit-chain operator or any user, can perform double spend attempt or Sybil attack and Denial-of-service. At the same time we assume that at least one between any user or operator stays honest in case of attack (dishonest majority scenario). Under these assumptions, commit-chain operator must be online at least once with a *epoch*, i.e. to perform the checkpoint.

Security Goals. For any system two goals are important: Safety and Liveness. In the following we explain what those two notions mean in TokenCards.

- **Safety**: Unlike other Layer 2 (L2) systems, TokenCards operates independently of external liquidity sources. Each token is fungible and directly represents a verifiable real-world transaction ascertained by the operator, such as a retail purchase (e.g., the sale of a sandwich). Consequently, the arbitrary issuance of new tokens does not detract from the value of existing tokens. For example, the requirement for a user to redeem a perk (e.g., a free sandwich) remains constant at 10 tokens, irrespective of whether there are 100 or 1,000 additional tokens in circulation. On the other hand, any user must be sure no one can spend its token undetected.
- **Liveness**. Essentially, any user can access its funds when needed within a predetermined amount of time. In particular users must be able to i) spend or ii) transfer funds irrespective of the current state of the blockchain or the DFS.

Communication Model. Given the threat model and the security goal, a specific communication model is needed. We assume users have independent, private (from the operator view) non necessarily secure channels between them (e.g. a messaging app), that all messages from and to the blockchain and DFS are bounded following a semi-synchronous model [1]. All users can read the public key of the operator, and all states of the accumulator, in particular the initial, current and previous state of the accumulator. In practice, the users are aware of the DFS link where all of these information are stored.

5 TOKENCARDS

5.1 High Level Functions

It is important to understand which operations are carried out in TokenCards. In particular, it is important to understand which operation are performed by the operator O and which by the user U. Moreover, some operation performed by operator O can be triggered by external parties (i.e. are *public*) while others can only be triggered by O (i.e. are *private*). Finally some operations can be performed by U directly on the client. To differentiate between operations in TokenCards, we prepend **TC** to the name of the operation.

Private Operational Functions. The following are the functions that can be performed by the operator O:

- **TC**.**Setup**: this operation directly calls the **Acc**.**Setup** (Section 3.1) function to initialize the accumulator and creates public and private parameters and keys. It is a private operation and it is performed once. Details are explained in Section 5.2.
- **TC**.**Add** : this operation is responsible for updating the state of the accumulator by incorporating a new token, thereby expanding the set of valid tokens or credentials. Details are explained in Section 5.3.
- **TC**.**Spend**: this operation checks if a given token is valid and part of the current state of the accumulator. Upon successful validation, the token is removed from the active set in the accumulator. Details are explained in Section 5.4.

Public Operational Functions.

- **TC**.**GenPrime**: this operation generates prime number and note c, which is random number. Details are explained in Sections 5.3 and 5.5.
- **TC**.**SecureTransfer**: given a valid note c, this operation performs a secure token transfer. Details are explained in Section 5.5.

5.2 Setup

As part of the initialization of TokenCards, the operator O performs **TC**.**Setup**. This operation differs from **Acc**.**Setup**, even though **Acc**.**Setup** is included in **TC**.**Setup**. At first, the function setups the RSA accumulator to register all the tokenID, using function **Acc**.**Setup**. The RSA accumulator is initialized with a base value $\mathcal{A}_0$. Then, O creates a pair of private-public key (sk_O^b, pk_O^b) related to blockchain funds and a pair of private-public key (sk_O^f, pk_O^f) to upload and sign content in the DFS.

Careful consideration must be given to the difference between public and private parameters. Let $N = pq$ be the RSA modulus and $\phi = (p-1)(q-1)$ the related Euler's totient function, where p and q are distinct large prime numbers, known only to the accumulator manager (i.e. the operator, O in our context). Then the factorization of N (i.e. p and q) and ϕ are kept secret to ensure the security of the accumulator. On the other hand O releases $(\mathcal{A}_0, N, pk_O^b)$ publicly in the DFS. Finally O sends a blockchain transaction to $\text{addr}(pk_O^b)$, the address related to pk_O^b: the value of the transaction should cover fees for the different certifications (see Section 5.6 and Section 6.3 to understand how that operation is very efficient in terms of O's finances).

We require O to publish all the details that ensure the successful setup, namely the address or link to the DFS where $(\mathcal{A}_0, N, pk_O^b)$ are published, the hash of the transaction (or equivalently a link to a block explorer) and the public key pk_O^f of the operator. An easy way to publish this in a local community is by exposing a serialized version of the information in a QR code in the shop, beside sending it to the users clients.

5.3 Add New Token

Upon a customer making a purchase w, the operator calls the **TC**.**Add** function. This function generates a new tokenID $\tilde{t}_w$ using a pseudorandom function. $\tilde{t}_w$ is uniquely associated with the shopping transaction, in another words the token is backed by a real life purchase, and therefore a physical object exchange. Since the element $\tilde{t}_w$ may not be prime, the **Acc**.**Add** function can not take $\tilde{t}_w$ as input (see Section 3.1). For this reason $t_w = \textbf{TC.GenPrime}(\tilde{t}_w)$ is used as the input for **Acc**.**Add**. The function **TC**.**GenPrime**() is a deterministic function that maps any number to a prime number. Multiple efficient algorithms can be used to compute that function, see e.g. [13].

Beside token t_w, O also computes its inverse $(t_w)^{-1} \mod \phi$ and the pair $(t_w, (t_w)^{-1} \mod \phi)$ is presented as a QR code to the user in order to facilitate the experience. The inverse is needed for token transfer, see Section 5.5. The user saves the pair in the smartphone app together with all the other tokenIDs acquired from other stores.

On the other hand O stores only t_w in the RSA accumulator. Specifically, for each new tokenID t_w, O follows Equation (1) and computes $\mathcal{A}_w = (\mathcal{A}_{w-1})^{t_w} \mod N$. The new state of the accumulator $\mathcal{A}_w$ is signed with sk_O^f and stored in the DFS. As of now the current state of the accumulator is visible by all users, but is not certified to the blockchain.

5.4 Spend Token

When a customer wishes to redeem tokens $\{t_1, t_2, \ldots, t_w\}$ for rewards or discounts, the operator calls the **TC**.**Spend** operation. This operation first batch verifies the authenticity and validity of the tokenIDs in question. In practice, after user request, O replies with the couple $(\mathcal{W}_i, \mathcal{N}_i)$ where $\mathcal{W}_i$ is the witness for inclusion of the tokenIDs in the accumulator at state i while $\mathcal{N}_i$ is the witness for non inclusion of the tokenIDs in the accumulator. $\mathcal{W}_i$ is created using Equation (3), while $\mathcal{N}_i$ is created using Equation (4). If *all* tokens $\{t_1, t_2, \ldots, t_w\}$ are in the accumulator, then $\mathcal{N}_i = err$ and the verification is considered to be successful. Otherwise the verification fails and the user must try again.

If verification is successful, $\{t_1, t_2, \ldots, t_w\}$ are deleted from the accumulator and exchanged for the agreed-upon reward or discount. The removal of the token is done as explained in Section 3.1 and uses the **Acc**.**Rm** function. The RSA-accumulator on the operator's server is updated accordingly from $\mathcal{A}_i$ to $\mathcal{A}_{i+1}$, signed with sk_O^f and uploaded to the DFS.

5.5 Transfer Token

We explain how to securely transfer a token between user U_1 and U_2. In principle a simple communication of the token from U_1 to U_2 is enough: O can not stop that since tokens in the accumulator are anonymously added. Yet this is not secure in practice: U_1 could still spend the token after the communication, making the transfer non secure for U_2.

Assume tokenID t_w currently belongs to U_1. Recall that for each t_w there is an inverse $(t_w)^{-1} \mod \phi$ created during the the creation of the token (see Section 5.3). The inverse is needed for transfer.

When U_1 wants to securely transfer a token to U_2, U_1 sends the couple $(t_w, (t_w)^{-1} \mod \phi)$ to U_2. U_2 then calls the **TC**.**GenPrime** function obtaining the couple (t'_w, c). Here t'_w is a prime generated by O and c is a note proving that U_2 has initiated a secure transfer. Finally U_2 sends $(t'_w \cdot (t_w)^{-1} \mod \phi, c)$ to **TC**.**SecureTransfer**

This function facilitates the transfer of a token from U_1 to U_2. Upon acceptance of c, the **TC**.**SecureTransfer** function upon checking the validity of c invokes $\textbf{Acc.Add}(\mathbf{t'_w} \cdot (\mathbf{t_w})^{-1} \mod \phi)$, updating the state of the accumulator. Given the identity

$$t_w \cdot t'_w \cdot (t_w)^{-1} \equiv t'_w \mod \phi,$$

adding $t'_w \cdot (t_w)^{-1} \mod \phi$ to the accumulator effectively removes t_w and adds t'_w. Consequently, this process is equivalent to U_1 spending t_w and U_2 getting t'_w, but achieved in an atomic manner.

5.6 Checkpoints

5.6.1 Creation. In our framework, the operator periodically, or upon reaching a designated threshold, certifies the state of the RSA accumulator using BATS (Section 3.2) at predetermined intervals. In practice, in TokenCards we partition time into discrete intervals termed *epochs*. For example, an epoch can span two weeks. The operator O, holding a private-public keypair (sk_O^b, pk_O^b) is tasked with committing the state of a RSA accumulator $\mathcal{A}_i$ to the blockchain at the closure of each epoch, an action that could be automated for efficiency .

Assuming we are in epoch i, we denote the current state of the accumulator as $\mathcal{A}_i$ and the content identifier in the DFS as cid_i. Then the data to certify is $data_i = (\mathcal{A}_i \parallel cid_i)$.
Setting $pk_0 = pk_O^b$, we can define recursively pk_i as:

$$pk_i := pk_{i-1} + (ts_i \cdot C.G)$$

where $ts_i = \textbf{BATS.Cr}(C, \mathcal{H}, data_i)$, and analogous definition works for sk_i. The certification then is done by O by computing

$$\textbf{BATS.Pub}(C, \mathcal{B}, ts, sk_{i-1}, pk_i)$$

obtaining the transaction hash tx_i.

5.6.2 Verification. Customers can independently verify the certification of their tokens. By accessing the certified data on the blockchain, they can perform necessary calculations to confirm the inclusion of their tokenID.

In particular, after the publication on the blockchain and the sharing of tx_i , users can independently get the public key pk^{i-1} and the content identifier cid_i and the current state of the accumulator $\mathcal{A}_i$. Therefore they can compute $data_i$ and compute $\textbf{BATS.Ver}(C, \mathcal{H}, data_i, pk_{i-1})$

If the computation is succesful, then users can be sure the current state of the accumulator they accessed has been certified. To see if their tokens are in the current accumulator, users can use the batched verification process via function **Acc**.**GenMemProof** described in Section 3.1. If this computation is successful too, then the user is sure its token is included in the accumulator.

6 EVALUATION

In the following, we present a comparison of the performance and qualities of TokenCards with respect to the current state of the art, namely NOCUST and Plasma Cash (see Section 2).

6.1 Guarantees

TokenCards adheres to both security goals defined in Section 4. It ensures safety because token creation is contingent upon purchasing something from the Owner, who is the sole minter of new tokens. Note that exchanging tokens involves atomically removing and adding a new token to the accumulator. For this reason, it is impossible to create new tokens without purchasing products. Therefore, TokenCards is not subject to token inflation.

Note that with the use of non-membership proofs, a cheating operator is always detected. Since the operator can block the spending or transferring of tokens by requiring a membership proof, any legitimate blocking of such operations must be atomically accompanied by a membership proof. Refusal of operations without this proof will be considered cheating and thus detected.

TokenCards also ensures liveness. Since buying and selling products from the owner are not dependent on blockchain operations or interaction with the DFS, TokenCards is always live (as long as the shop is open in the community, in the case of minting and spending tokens). This is because spending tokens and transferring funds require the user and the operator to interact with the accumulator, which is stored on a server, eliminating the need to interact with the blockchain directly.

6.2 Performance

Since performance is closely related to the accumulator used, this part of the comparison essentially contrasts the RSA accumulator used in TokenCards with the Merkle Tree (MTs) used in NOCUST and Plasma Cash. For a summary of the comparison, see Table 2. For a detailed comparison between accumulators, including the RSA accumulator and MT, refer to the work by Loporchio *et al.* . [22]

The primary difference between MTs and RSA accumulators lies in the operations they use. MTs utilize hash functions for element inclusion, while RSA accumulators use modular exponentiation. This distinction offers advantages for TokenCards. Specifically, modular exponentiation allows the accumulator owner to perform batch verification. This is crucial in settings like a fidelity card system, as proposed in this paper, where customers typically spend many tokens at once to receive a discount. If tokens needed to be verified individually, as with MTs, the process would become increasingly inefficient as the number of tokens grows. Batch verification, on the other hand, ensures efficient operations for both users and operators.

Beyond batching, the accumulator used in TokenCards has advantages in proof size, generation, and verification. Let N be the cardinality of the set of elements in the accumulator. RSA membership proofs are constant in size, while those created by MTs grow logarithmically with N. Furthermore, MT proof generation and verification timese both grow logarithmically with N. In contrast, RSA proof generation grows linearly with N, while proof verification remains constant. We consider this a significant advantage in our setting. Membership proof generation is always performed by the accumulator's operator, who is expected to have at least a retail-grade laptop, a device generally more powerful than the mobile applications used by customers. Therefore, the fixed time complexity of verification operations (which are generally done by the users), regardless of the growth of the token set, is a strength of our proposal.

Regarding proofs, RSA accumulators offer the advantage of non-membership proofs, which are not possible with MTs. Consequently, the RSA accumulator operator can prove that an element is *not* in the accumulator. This feature is used in TokenCards to prevent potential misbehavior by the operator, who controls both the shop and the accumulator itself. This prevents the operator from refusing token spending by honest users (see Section 6.1 for a detailed explanation).

6.3 Costs

In the following we present the costs (in gas) of both NOCUST and TokenCards. We could not compare cost estimates with Plasma Cash as it does not provide a cost evaluation.

Conversely, the work on NOCUST includes a cost analysis in terms of gas, the unit of cost in EVM blockchains. For the purpose of comparison, we also provide costs in gas for EVM compatible blockchain, even though TokenCards is blockcahin agnostic (including the ability to be used alongside other L2).

As shown in Table 3, the only costs incurred by the operator in TokenCards are for check-pointing, which are half of those in NOCUST. Additionally, users of TokenCards do not incur any costs, as no deposits or withdrawals are required.

7 CONCLUSIONS

We introduced a new system that provides a Layer2 scaling solution for blockchains, specifically designed for local communities. Our system relies on commit-chains and assumes a moderate but well-defined level of trust, which is typical in local communities. Nonetheless, his level of trust aligns with the threat models of state-of-the-art commit-chains. Notably, our commit-chain proposal enhances the state of the art in terms of cost and performance. Additionally, we have developed a new time-stamping system for the blockchain that is more private and less costly than current methods. It is also blockchain agnostic, setting it apart from other commit systems on the blockchain.

Currently, our token transfer mechanism requires interaction between the user and the owner of the commit-chain. In the future, we plan to develop a system that allows for non-interactive transfers between parties. This would make our proposal more efficient and private, further reducing trust assumptions and improving upon other commit-chain proposals.

REFERENCES

[1] Hagit Attiya and Jennifer L. Welch. 2004. *Distributed computing - fundamentals, simulations, and advanced topics (2. ed.).* Wiley.

[2] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014), 201–224.

[3] Fadi Barbàra, Flavia Fredda, and Claudio Schifanella. 2023. A New Token Management System for Local Communities.. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good, GoodIT 2023, Lisbon, Portugal, September 6-8, 2023*. 237–245. https://doi.org/10.1145/3582515.3609540

[4] Niko Baric and Birgit Pfitzmann. 1997. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees.. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic*

Table 2: Comparison of RSA Accumulators and Merkle Trees

Feature	Description	RSA Accumulators	Merkle Trees
Base Operation	Type of function used for element inclusion	Modular exponentiation	Hash functions
Batch Verification	Efficiency in verifying multiple elements at once	Yes	No
Proof Size	Size of membership proofs	$O(1)$	$O(\log(N))$
Non-membership Proofs	Ability to prove an element is not in the accumulator	Yes	No
Proof Generation Time	Time to generate membership proofs	$O(N)$	$O(\log(N))$
Proof Verification Time	Time to verify membership proofs	$O(1)$	$O(\log(N))$

Table 3: Cost Comparison between NOCUST and TokenCards

Cost Type	NOCUST	TokenCards
Check-pointing	96,073	2,300
Deposits	64,720	None
Withdrawals	169,238	None

Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding. 480–494. https://doi.org/10.1007/3-540-69053-0_33

[5] Josh Cohen Benaloh and Michael de Mare. 1993. One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract). In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings (Lecture Notes in Computer Science, Vol. 765)*, Tor Helleseth (Ed.). Springer, 274–285. https://doi.org/10.1007/3-540-48285-7_24

[6] Vitalik Buterin. 2014. A next generation smart contract and decentralized application platform. https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[7] Vitalik Buterin. 2017. *This sounds like there's some kind of scalability trilemma at play. What is this trilemma and can we break through it?* https://vitalik.eth.limo/general/2017/12/31/sharding_faq.html#this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it

[8] Vitalik Buterin. 2021. *An Incomplete Guide to Rollups.* https://vitalik.eth.limo/general/2021/01/05/rollup.html

[9] Vitalik Buterin. 2023. *Different types of layer 2s.* https://vitalik.eth.limo/general/2023/10/31/l2types.html

[10] Jan Camenisch and Anna Lysyanskaya. 2002. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2442)*, Moti Yung (Ed.). Springer, 61–76. https://doi.org/10.1007/3-540-45708-9_5

[11] Johnny Dilley, Andrew Poelstra, Jonathan Wilkins, Marta Piekarska, Ben Gorlick, and Mark Friedenbach. 2016. Strong federations: An interoperable blockchain solution to centralized third-party risks. *arXiv preprint arXiv:1612.05491* (2016).

[12] Hyperledger Foundation. 2018. An Introduction to Hyperledger Foundation. https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/Hyperledger/Offers/HL_Whitepaper_IntroductiontoHyperledger.pdf.

[13] Pierre-Alain Fouque and Mehdi Tibouchi. 2019. Close to Uniform Prime Number Generation With Fewer Random Bits. 65, 2 (2019), 1307–1317. https://doi.org/10.1109/TIT.2018.2859045

[14] Ilja Gerhardt and Timo Hanke. 2012. Homomorphic Payment Addresses and the Pay-to-Contract Protocol. *CoRR* abs/1212.3257 (2012). arXiv:1212.3257 http://arxiv.org/abs/1212.3257

[15] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. 2020. SoK: Layer-Two Blockchain Protocols.. In *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*. 201–226. https://doi.org/10.1007/978-3-030-51280-4_12

[16] Shaphali Gupta. 2020. Understanding the feasibility and value of grassroots innovation. *Journal of the Academy of Marketing Science* 48 (2020), 941–965. https://doi.org/10.1007/s11747-019-00639-9

[17] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. 2020. Scaling Blockchains: A Comprehensive Survey. 8 (2020), 125244–125262. https://doi.org/10.1109/ACCESS.2020.3007251

[18] Maxim Jourenko, Kanta Kurazumi, Mario Larangeira, and Keisuke Tanaka. 2019. SoK: A Taxonomy for Layer-2 Scalability Related Protocols for Cryptocurrencies. 2019 (2019), 352. https://eprint.iacr.org/2019/352

[19] Rami Khalil and Arthur Gervais. 2018. NOCUST - A Non-Custodial 2nd-Layer Financial Intermediary. 2018 (2018), 642. https://eprint.iacr.org/2018/642

[20] Georgios Konstantopoulos. 2019. Plasma Cash: Towards More Efficient Plasma Constructions. abs/1911.12095 (2019). http://arxiv.org/abs/1911.12095

[21] Robin Linus. 2023. *BitVM: Compute Anything on Bitcoin.* https://bitvm.org/bitvm.pdf

[22] Matteo Loporchio, Anna Bernasconi, Damiano Di Francesco Maesa, and Laura Ricci. 2023-08. A Survey of Set Accumulators for Blockchain Systems. 49 (2023-08), 100570. https://doi.org/10.1016/J.COSREV.2023.100570

[23] Patrick McCorry, Chris Buckland, Bennet Yee, and Dawn Song. 2021. SoK: Validating Bridges as a Scaling Solution for Blockchains. *IACR Cryptol. ePrint Arch.* (2021), 1589. https://eprint.iacr.org/2021/1589

[24] Ralph C. Merkle. 1980. Protocols for Public Key Cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 14-16, 1980*. IEEE Computer Society, 122–134. https://doi.org/10.1109/SP.1980.10006

[25] Ralph C. Merkle. 1987. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings (Lecture Notes in Computer Science, Vol. 293)*, Carl Pomerance (Ed.). Springer, 369–378. https://doi.org/10.1007/3-540-48184-2_32

[26] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[27] Raiden Network. 2018. *Fast, cheap, scalable token transfers for Ethereum.* https://raiden.network/

[28] Lan Nguyen. 2005. Accumulators from Bilinear Pairings and Applications. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3376)*, Alfred Menezes (Ed.). Springer, 275–292. https://doi.org/10.1007/978-3-540-30574-3_19

[29] Jonas Nick, Andrew Poelstra, and Gregory Sanders. 2020. Liquid: A bitcoin sidechain. https://blockstream.com/assets/downloads/pdf/liquid-whitepaper.pdf.

[30] National Institute of Standards and Technology. 2023. *Cryptographic Algorithms and Key Sizes for Personal Identity Verification.* Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 800-78-5. U.S. Department of Commerce, Washington, D.C. https://doi.org/10.6028/NIST.SP.800-78-5.ipd

[31] Joseph Poon and Vitalik Buterin. 2017. Plasma: Scalable Autonomous Smart Contracts. (2017). https://plasma.io/plasma-contracts.html

[32] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments. https://static1.squarespace.com/static/6148a75532281820459770d1/t/61af971f7ee2b432f1733aee/1638 network-paper.pdf.

[33] RGB. 2023. *RGB. Private & scalable smart contracts for Bitcoin and Lightning Network.* https://rgb-org.github.io/
[34] Berk Sunar. [n. d.]. Euclidean Algorithm. In *Encyclopedia of Cryptography and Security*. Springer US, 427–430. https://doi.org/10.1007/978-1-4419-5906-5_27
[35] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. 2019. SoK: Sharding on Blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies* (Zurich, Switzerland) *(AFT '19)*. Association for Computing Machinery, New York, NY, USA, 41–61. https://doi.org/10.1145/3318041.3355457
[36] Anatoly Yakovenko. 2017. Solana: A new architecture for a high performance blockchain v0.8.13. https://coincode-live.github.io/static/whitepaper/source001/10608577.pdf.
[37] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. 2017. An Expressive (Zero-Knowledge) Set Accumulator. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*. IEEE, 158–173. https://doi.org/10.1109/EUROSP.2017.35
[38] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. 2020. Solutions to Scalability of Blockchain: A Survey. *IEEE Access* 8 (2020), 16440–16455. https://doi.org/10.1109/ACCESS.2020.2967218

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009