Accumulate

WHITEPAPER

Version: 1.0
Date: 12/Apr/2022
DOI: 10.6084/m9.figshare.19586515

Accumulate: An identity-based blockchain protocol with cross-chain support, human-readable addresses, and key management capabilities

Owner: Inveniam DeFi Devs, Inc.
Contact: paulsnow@defidevs.io

Kyle Michelson, Anjali Sridharan, Umut Can Çabuk, Ethan Reesor, Ben Stolman, Drew Mailen, Dennis Bunfield, Jay Smith, and Paul Snow.

The Accumulate Protocol ("Accumulate") is an identity-based, Delegated Proof of Stake (DPoS) blockchain designed to power the digital economy through interoperability with Layer-1 blockchains, integration with enterprise tech stacks, and interfacing with the World Wide Web. Accumulate bypasses the trilemma of security, scalability, and decentralization by implementing a chain-of-chains architecture in which digital identities with the ability to manage keys, tokens, data, and other identities are treated as their own independent blockchains. This architecture allows these identities, known as Accumulate Digital Identifiers (ADIs), to be processed and validated in parallel over the Accumulate network. Each ADI also possesses a hierarchical set of keys with different priority levels that allow users to manage their security over time and create complex signature authorization schemes that expand the utility of multi-signature transactions. A two token system provides predictable costs for enterprise users, while anchoring all transactions to Layer-1 blockchains provides enterprise-grade security to everyone.

Contents

1 Introduction

1.1 Motivation & Problem Statement

Designing data storage and management methods that provide a high level of security and efficiency is a challenge of high importance to blockchain protocols. An ideal platform includes desirable characteristics such as fast read/write speeds (low block and transaction times), low transaction costs, scalability, security, pathways for migrations of legacy systems, and easy management of data access and authorizations. These standards have proved difficult to meet for existing blockchain protocols, with computationally-expensive smart contracts often lacking the efficiencies and security required for scaling. Accumulate addresses these requirements through its unique identity-based architecture, with the following key innovations that set it apart from other protocols:

- Use of Decentralized Digital Identity and Identifiers (DDII) as the atomic unit of the blockchain.
- Chain-of-chains architecture for network partitioning and linear scalability.
- Key Books for easy management of credentials and access.
- Synthetic transactions for inter-chain communication.
- Scratch Accounts for efficiently managing consensus.

This document begins by providing a high-level overview of each of these key innovations, followed by a technical description of the overall system architecture. These key innovations will be revisited with a more technical emphasis on how each fits into the overall architecture of the protocol, and how they enable a variety of real-world use cases.

1.2 Inheritance from Factom

The Factom protocol was launched in 2014 as an alternative to the direct use of the Bitcoin blockchain for the management and organization of data, making it one of the earliest Layer-2 blockchains[1]. At the time, blockchain technology was being explored as a means to reliably and securely store data without requiring trust between unfamiliar parties or the involvement of a centralized third party to validate transactions and maintain a record of the transaction history. However, Bitcoin's verification process required an entire ledger to validate individual pieces of data, a computationally expensive method. Factom's founding developers recognized that Bitcoin was inconvenient and unsuitable for enterprise data solutions with its simple linear blockchain format and limited capacity.

Factom was designed to utilize the security and reliability of the Bitcoin protocol while making notable improvements to its efficiency in record storage applications. As a Layer-2 protocol, Factom was able to commit to the Bitcoin blockchain in near real-time, secured by the consensus mechanism of the Factom protocol itself[1]. Factom was designed with a unique chain-of-chains architecture in which a set of chains were linked together into a directory layer. Data for applications built on the protocol was contained in specific data chains, allowing for faster indexing and reduced computational and storage costs for organizations.

The Factom protocol took security one step further by anchoring the results of this consensus to the Bitcoin blockchain every 10 minutes using a data-efficient Merkle tree. Merkle trees[2] were used to organize large amounts of data entered into Factom chains to be secured onto the Bitcoin blockchain with a single data point. This provided users with an immutable history of their transactions and allowed them to track and sync with only a small portion of the Factom Protocol to run their application. Factom also introduced several innovations for more efficient scaling. For example, sharding was simplified due to the separation of token transactions and computation from the data layer. An efficiently sharded data blockchain allows for continuous and real-time securing of data at the level demanded by enterprise customers[1].

With its highly competitive speeds and transaction times, Factom was able to forge several successful partnerships in the public and private sectors. Many of these were groundbreaking implementations of blockchain technology, such as a pilot program with the Department of Energy (DOE) to secure the energy grid of Texas, USA; a grant from the Department of Homeland Security (DHS) protect the authenticity of data collected by Internet of Things (IoT) devices, and an award by the Gates Foundation to store and organize medical records in developing countries[3]. Determined to push the boundaries of the protocol's innovation, Factom's developers began re-envisioning its utility and creating what would eventually become the Accumulate Protocol. With this new design, the founders of Accumulate were committed to preserving Factom's unique and innovative aspects while improving on its ease of use, mass appeal, and scalability.

1.3 Key Innovations

Since Accumulate is a continuation of the Factom protocol, a comparison between Accumulate and Factom is useful. Accumulate introduces many key innovations while carrying over Factom's core features, making the protocol significantly more robust and versatile.

- **The Identity Paradigm:** While most protocols today are tied to the paradigm of public-private keys as a means of storing and accessing records, Accumulate is the first blockchain organized around decentralized, self-sovereign digital identifiers. Creating user-friendly, application-friendly, maintainable identifiers is an important challenge addressed by this protocol. The core organizational structure used in Accumulate is the Accumulate digital identity/identifier/domain (ADI)[4]. The use of ADIs has allowed Accumulate to bypass the 'Scalability Trilemma', a commonly-observed trade-off between decentralization, security, and scalability[5].

  Accumulate overcomes the trilemma by organizing itself around ADIs, where each ADI defines its own state that is independent of other ADIs. ADIs are distributed over a set of Tendermint networks, each of which is referred to as a Block Validator Network (BVN). Each BVN is responsible for a particular set of ADIs, and as the Accumulate Network grows, more BVNs can be added to scale and maintain high throughput.

- **Parallel Processing:** Each ADI is treated as having its own state and set of accounts and chains. Each ADI is updated independently from the state of other ADIs. This highly branched design allows more parallel processing and greater freedom to distribute the responsibilities of processing state across many networks. This allows for fast transaction finalization and short block times.

- **Key Management:** Any organization that handles data has a system of access control for the various levels of authorized users on their system. Accumulate reproduces this capability within a blockchain protocol by providing each ADI with the ability to create Key Books for defining these access controls. Key Books define how decisions are to be made and even how the Key Book itself is managed. The Key Book holds Key Pages that define the keys and authorization schemes to be used in decision making. The Keys and the authorization schemes specified in a Key Page can be managed and modified, eliminating any need to reissue accounts just because an organization must shift responsibilities/privileges over time. This flexibility in key management gives users the tools they need to dynamically model, manage, and maintain the dynamic nature of blockchain applications.

- **Synthetic Transactions:** Since each ADI has its own state, transactions that are routed to an ADI must be processed independently of all other ADIs in the network. This becomes challenging when two or more ADIs are involved in a transaction. Accumulate's solution is to generate an additional transaction that performs settlement within an ADI. Transactions generated by the protocol in response to transactions initiated by a user are referred to as Synthetic Transactions.

  In practice, a Synthetic Transaction is sent to the BVN that manages the destination ADI after the source ADI initiates a transaction. Debits to the source ADI are handled directly by the user's transaction, while credits to the destination account are handled by the Synthetic Transaction. A Synthetic Transaction is sent every time an ADI interacts with another ADI, which provides each BVN with a balance of every ADI that has debited or credited an ADI within its own network. This way, an ADI does not need to query any BVN in the Accumulate network to verify the balance of a source ADI.

- **Scratch Accounts:** Another key vulnerability in many blockchains is the off-blockchain processes that are needed to arrive at an entry on the blockchain. Developers are faced with a trade-off between the expense and difficulty of showing their work on the blockchain and the lack of accountability when the work of coming to consensus is done off-chain. Accumulate provides scratch accounts, which reduce the cost of using the blockchain for consensus building. However, the data availability of scratch accounts is limited. Scratch accounts allow processes to provide cryptographic proof of validation and process without overburdening the blockchain.

1.4 Integrations

Blockchain protocols each tend to exhibit their own niche design and function. Accumulate is first and foremost a system of identity management capable of modeling complex organizational structures and relationships. This makes it ideal for integration with various other protocols, improving the accessibility of their blockchain technologies in a more diverse variety of use cases.

The Accumulate protocol does not natively write smart contracts but it does support various smart contract roll-ups. This allows Accumulate to track the state and validity of contracts stored on third-party chains. An organization is then able to process smart contracts written across a variety of Layer-1 protocols such as Ethereum, Solana, and Tezos. In the near future, when vendors may rely on a preferred Layer-1 protocol to conduct their operations (such as using smart contacts for supply-chain tracking), Accumulate will have the capability to manage this multitude of blockchains.

The Accumulate ecosystem also has room for integration with Layer-0 protocols, which aim to connect the hundreds of existing unique protocols. Known as 'blockchains of blockchains', protocols such as Cosmos and Polkadot[6] have made great improvements towards interoperability, or the ability to make transactions across two different protocols efficiently. Accumulate can then be utilized to manage the transferred asset under the identity or ADI of the buyer and to continue to track the movement of the asset across multiple identities. This allows for the custodial transfer of assets through multiple organizations, using whatever protocols they consider 'native'.

The use of Accumulate to create an ecosystem has several clear benefits. One of Accumulate's most valuable contributions is its security model that allows for rapid response to security breaches. By moving away from the paradigm of a 1:1 relationship between key and asset, Accumulate decouples the concept of identity and security. This ensures that even if a transaction's private keys are compromised, an organization can set its own authority of trusted individuals to prevent an incorrect transfer. This is especially important in the case of assets that are repeatedly transferred across various protocols when there is an increased risk that private keys could become exposed.

2 System Overview

At the systems level, Accumulate is optimized for parallel processing, linear scaling, and efficiency of state. Parallel processing is achieved by partitioning the network into multiple validator networks that process transactions for only a fraction of accounts. Additional validator networks can be added to linearly scale the network as usage increases. Efficiency of state is realized through Accumulate's chain-of-chains architecture that organizes transactions into hierarchies of summary hashes and allows a user to validate transactions on a mobile device. This chapter explains the architectural basis of these features and provides a high-level overview of the Accumulate protocol. A detailed description of the Accumulate architecture is provided in Chapter 6.

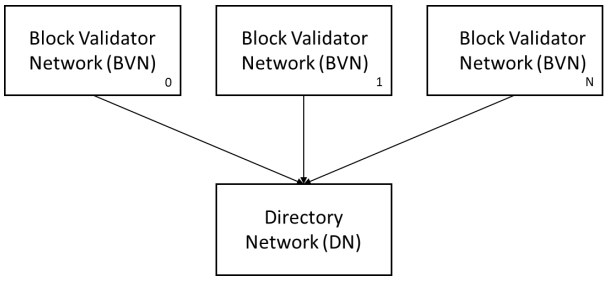

Fig. 1 System overview.

2.1 Introduction

In a traditional blockchain, transactions are hashed using cryptographic algorithms like SHA-256[7] and organized into a data structure called a Merkle tree[2]. Pairs of hashes are concatenated until a single hash remains (the Merkle root). The Merkle tree is stored within a block and connected to other blocks in the network by including the hash of the previous block in the current block header. This data structure creates a block 'chain' that maintains the history of the network. Multiple full nodes, each containing a complete copy of the blockchain, participate in block validation and maintain the state of the network. If a node is compromised and a transaction within a block is modified, then the Merkle roots of all subsequent blocks will change and the network will reject the fraudulent block.

The Accumulate protocol has a fundamentally different architecture that is organized around accounts rather than blocks. Each account is treated as an independent chain and managed as a continuously growing Merkle tree while blocks are treated as synchronization points for all chains in the network. The indexing of these points allows Accumulate to provide 1 second blocks for finalization and 12 hour blocks for synchronization of the historical ledger. This is because blocks are defined by indexing the Merkle Trees of chains rather than building independent Merkle Trees rooted to blocks. Shorter block times are constrained by the speed at which the Tendermint consensus layer can validate transactions. Longer block times require less state and allow mobile devices to work as Lite Nodes that can validate a subset of transactions relevant to the account owner.

2.2 Validation

Networks of Tendermint nodes, called Block Validator Networks (BVNs), validate transactions for an account. Each account is assigned to a particular BVN, and each BVN can process transactions for thousands of accounts. Adding more BVNs allows the Accumulate network to scale linearly as utilization grows. As shown in Figure 1, root hashes from each BVN are fed into a larger network of Tendermint nodes called the Directory Network (DN). The DN produces a final root hash that can be 'anchored' into another blockchain. DN root hashes will be regularly anchored into Layer-1 blockchains like Bitcoin and Ethereum by inserting the DN root hash as data contained within a transaction. Anchoring essentially buys the security of a larger network at a cost that is independent of the number of transactions. For example, a root hash derived from 10,000 transactions can be anchored into Bitcoin at the cost of a single Bitcoin transaction.

2.3 Chain of Chains Architecture

Inside the DN and each BVN is an interconnected network of chains responsible for collecting signatures, synchronizing Tendermint nodes within a BVN, communicating between BVNs, collecting roots, and anchoring roots to other blockchains. These chains and their functions are summarized below and illustrated in Figure 2, which is an expanded view of Figure 1.

- Signature (Sig) Chain: Collects signatures for a period of approximately 2 weeks, after which transactions that have not met the $m-of-n$ signature threshold are discarded.
- Main Chain: Records transactions in both the origin account and any account modified by the transaction once a transaction has met the signature threshold(s) and has been executed.
- Synthetic Transaction Chain: Synthetic transactions are produced when a transaction needs to update accounts across multiple identities. The Synthetic Transaction Chain provides cryptographic proof that a synthetic transaction was actually produced by a particular BVN.
- Binary Patricia Trie (BPT): Collects hashes representing the current state and history of every account in the BVN or DN.
- Root Anchor Chain: Collects an anchor once per block from every account chain and system chain updated during the block. For every major and minor block, the Root Anchor Chain produces a root hash for the BVN or DN. The root hash of the DN is the root hash of the entire Accumulate network.
- Intermediate Anchor Chain: Within the DN, Intermediate Anchor Chains collect an anchor from the Root Anchor Chain of every BVN, once per block. Within a BVN, it collects an anchor from the DN Root Anchor Chain once per block.

Each account has a Signature Chain for collecting signatures, and a Main Chain for recording signed and validated transactions. These are collectively referred to as account chains, as each one is tied to a specific account. As illustrated in Figure 2, each BVN can contain multiple accounts, each of which has a Signature Chain and a Main Chain. Each BVN (and the DN) has a Synthetic Transaction Chain, a Root Anchor Chain, and one or more Intermediate Anchor Chains. These are collectively referred to as system chains, as they are not tied to any specific account and are used by the system as a whole.

Within a BVN or the DN, every chain is anchored into the Root Anchor Chain. Therefore, every chain can be considered a side chain of the Root Anchor Chain. The Root Anchor Chain of each BVN is also anchored into the DN's Intermediate Anchor Chain. Since the DN's Intermediate Anchor Chain is anchored into the DN's Root Anchor Chain, every BVN and thus every account in the network can be considered a side chain of the DN.

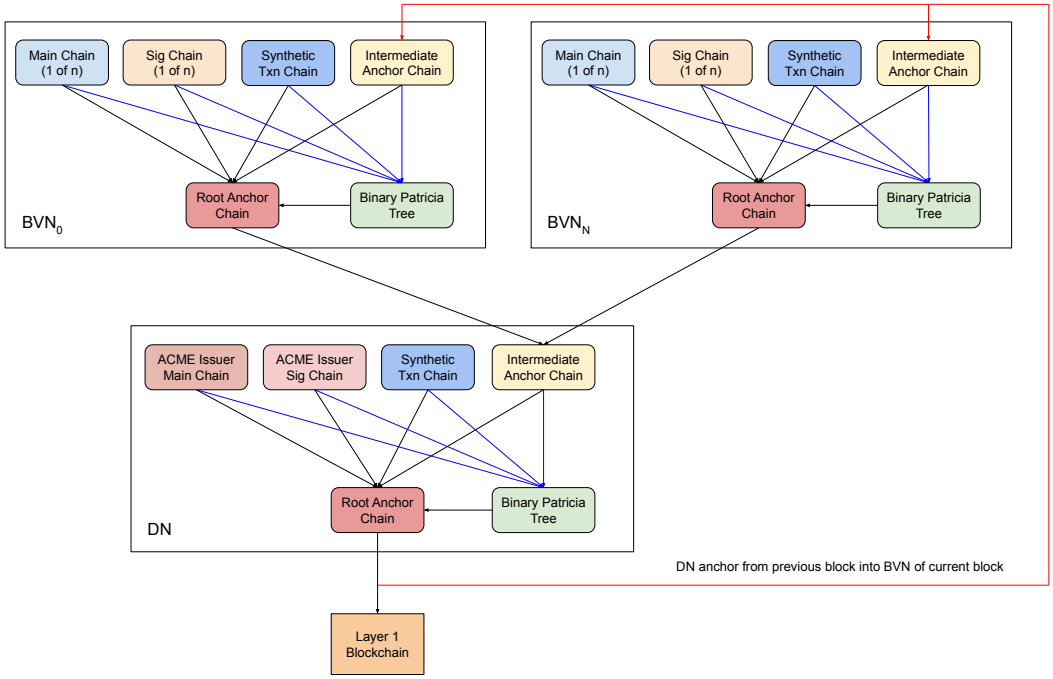

Fig. 2 Chains-of-chains architecture of Accumulate.

Each of these chains is organized as a continuously growing Merkle tree from which a root hash, or anchor, can be produced every block. This creates a chain-of-chains architecture where every chain within a BVN or the DN can be considered a side chain of its respective BVN/DN Root Anchor Chain. A key/value pair representing the state of an account is constructed for each account and added into the Binary Patricia Trie (BPT). Thus, the BPT is responsible for recording the current state of the Accumulate network. This architecture and its applications, such as Scratch Accounts, Managed Transactions, and Delegated Transactions, are discussed in greater detail in Chapter 6.

2.4 Account Architecture

Accumulate accounts are designed to improve the user experience, efficiently organize data, and integrate with web apps, mobile devices, and enterprise tech stacks. Accumulate supports the following types of accounts:

- Lite Token Account: Traditional blockchain address whose URL contains a public key hash and human-readable suffix denoting the token or data type held by the account.
- Lite Data Account: Provides compatibility with Factom. Public data account that can be used for collaboration.
- Accumulate Digital Identifier (ADI): The primary unit of organization within Accumulate. Besides Lite Accounts, all other accounts must belong to an ADI.
- Key Book: Secures accounts and provides advanced key management. Belongs to an ADI. See Chapter 4.
- Key Page: Organizes keys within a Key Book.
- ADI Token Account: Holds and exchanges tokens. Belongs to an ADI.
- ADI Data Account: Records data entries. Belongs to an ADI.
- Scratch Account: An ADI Data or Token Account with limited data availability. Transactions older than approximately two weeks will not be available.

Each account is uniquely identified by a URL. When tokens are sent to a Lite Token Account URL, the account is created if it does not already exist. A new user will get started with Accumulate by creating a key and having a trusted party send tokens to the corresponding Lite Token Account URL. For example, the user may pay USD or BTC to an exchange, which will send tokens from the exchange's Lite Token Account or ADI Token Account to the URL specified by the user. Lite Data Accounts are created through a similar mechanism, but they are only capable of writing data. Since Lite Data accounts do not specify keys, anyone with the address is able to write data to the account. This feature may be useful to an organization but is of little value to a potential hacker. A more detailed description of Lite Accounts is provided in Chapter 3.1.

ADIs are versatile accounts that perform many of the same functions as Lite Accounts in addition to being able to issue their own tokens, stake native ACME tokens, participate in complex authorization schemes, and manage other accounts. Most of these operations are enabled by hierarchical sets of keys, including Key Books and Key Pages, which are introduced in the next section and discussed in detail in Chapter 4. ADIs are purchased using Credits, a non-transferable utility token with a fixed USD value that is created when ACME tokens are burned. Credits are used to pay for all operations on the Accumulate network such as token

issuance, account creation, and updates to keys. An introduction to Accumulate's dual token model is provided in Chapter 7.

2.5 Key Architecture

Accumulate has a hierarchical key management structure where Key Books contain Key Pages and Key Pages contain keys that are authorized to sign transactions. Each Key Page specifies m of n where m is the number of unique, authorized signatures required to approve transactions (i.e., the signature threshold) and n is the total number of keys on the page.

Accounts are linked at creation to a Main Key Book and an optional Manager Key Book. A signature is authorized only if the signing key corresponds to a key in one of the pages of either the Main Key Book or the Manager Key Book (if specified). When creating a transaction, the user selects a Key Page from the Main Key Book of the origin of the transaction to use for the transaction. Signatures of transactions originating from an account are authorized against the account's Main Key Book. Any number of Key Books can belong to an ADI, and each account in the ADI can be linked to any Key Book in the ADI.

The number of Key Pages within a Key Book is specified by the user. Each Key Page is assigned a different priority level where a Key Page with a higher priority can make changes to any Key Page with a lower priority (e.g., change $m-of-n$ requirements). A user can also specify any number of Key Books from external ADIs as the owners of any number of Key Pages within their Key Book. External Key Books can participate in transactions within the Key Book of another ADI and even make changes to the security of Key Pages with lower priority. These are called Delegated Transactions because they are made on behalf of another party. Delegated Transactions and Managed Transactions are formally introduced in Chapters 5.2 and 5.4, respectively. Key management is explained in more detail in Chapter 4.

3 Identity Management

Participation in the Accumulate network occurs through Accumulate Digital Identifiers (ADIs) and Lite Accounts, similar to how participation in other blockchains occurs through a wallet or address. ADIs give users access to the full range of features provided by the Accumulate network, including smart contracts, off-chain consensus building, and dynamic key management. Lite Accounts are a 'lite' version of ADIs that, despite their comparatively limited utility and flexibility, may appeal to users who simply want to send and receive tokens and maintain a record of their transactions.

3.1 Lite Accounts

The Accumulate protocol supports two types of Lite Accounts for basic management of tokens and data. Lite Token Accounts are similar to a traditional blockchain address and are used for sending and receiving tokens. The token type and authorization scheme of a Lite Token Account is specified by the user, but it cannot be changed after the account is created. However, a user can create multiple accounts in their wallet to manage different tokens.

Lite Data Accounts were designed to be compatible with Factom and help its users migrate to Accumulate. For this reason, Lite Data Accounts are constructed a bit differently than other accounts and mirror the way that Factom creates hashes and account IDs. Anyone who knows the URL of a Lite Data Account can append data similar to how anyone who knows the URL of a Lite Token Account can deposit tokens. However, data entries cannot be overwritten. Spamming a Lite Data Account is possible, but this type of attack is disincentivized because every data transaction incurs a small fee. Since Lite Data Accounts have limited utility outside of migration from Factom to Accumulate, they will not be covered in the following sections.

3.1.1 Generating a Lite Token Account

Anyone can generate a Lite Account by arranging for tokens to be sent to a Lite Token Account address corresponding to a public key that they own. ADIs, in contrast, can only be created by a sponsor through the spending of Credits issued through the Accumulate protocol. A user without an ADI to use as a sponsor can buy ACME tokens on an exchange, create a Lite Account for ACME tokens, purchase credits on the network, and sponsor the creation of their own ADI. They can then use their ADI to sponsor the creation of additional ADIs for themselves or others.

A Lite Token Account consists of an identity and token URL. The Accumulate wallet automatically derives the identity by concatenating the SHA-256 hash and checksum of a user's public key. The token URL (e.g., bob/token) is specified by the user and appended to the identity. A Lite Token Account is created by the first transaction sent to its URL. This transaction may originate from an exchange or an external wallet (e.g., Metamask). When Accumulate receives the transaction, it will automatically create a Lite Account if the Lite Account URL is valid and a Lite Account does not already exist for that URL.

To spend tokens in the Lite Token Account, a user will initiate a transaction, hash and sign it with their private key, attach their public key, and submit this to the Accumulate network. The network will hash the public key and compute a checksum, verify that the concatenated hash and checksum are identical to the Lite Token Account provided by the user, then decrypt the signature with their public key and verify the withdrawal.

3.1.2 Lite Token Account URLs

The URL format of a Lite Account is generalized as:

acc://<keyHash><checkSum>/<tokenUrl>.

The <keyHash> is the first 20 bytes of the SHA-256 sum of the account's public key, encoded as hexadecimal. The <checkSum> is the last 4 bytes of the SHA-256 sum of the lower case <keyHash>, also in hexadecimal. The following example illustrates the process of creating a Lite ACME Token Account from a public/private key pair:

PrivK: b270eaaa57e5d4d808a9766f64b340aa655481c298288
238e5b1b3561fb80b27

$PubK$: 023e6165e349c2822089ab042b3a885ca54a0907e237e 8bfb5bd2aa96885966f35

Compute the SHA-256 hash of the public key:

$H(PubK)$: 818D7C1F69E7BEBCE54FE087F44D86D14279100D2 EEA690AC3847AE1B9A14237

Trim this hash to the first 20 bytes (odds of a random match is 1 in 10^{48}) and convert to lowercase:

$H'(PubK)$: 818d7c1f69e7bebce54fe087f44d86d14279100d

Compute checksum (last 4 bytes of the SHA-256 hash of trimmed hash) and convert to lowercase:

$C(H'(PubK))$: 904a336d

Concatenate (20 bytes of) the hash, the checksum, a slash, and the token URL:

URL: acc://818d7c1f69e7bebce54fe087f44d86d14279100d90 4a336d/acme

The length of a Lite Token Account is fixed at 48 characters, where the hash of the public key occupies 40 characters and the appended checksum adds an additional 8 characters. Thus, the length of a Lite Token Account URL is fixed at 49 characters (including 1 character for the slash) plus the length of the token URL. When submitting a transaction to a Lite Token Account, a user can choose whether to append the acc:// prefix. If a prefix is not added by the user, it will be automatically added by the network.

Appending a checksum to the hash of the public key prevents the irreversible loss of tokens if the public key hash is copied incorrectly or derived from another algorithm. When the server receives a transaction, it converts the Lite Token Account identity into bytes, takes the first 20 bytes representing the key hash, recalculates the checksum from the first 40 hexadecimal characters, and verifies that they match. If the checksum is incorrect, the transaction fails and the tokens are returned to the sender. This process functions as a built-in verification mechanism, which is particularly important for Lite Token Accounts since they are not comprised of human-readable text and are therefore easier to mistype.

3.2 ADIs

Accumulate Digital Identifiers (ADIs) are URL-based digital identities with hierarchical key sets that can manage data, tokens, and other identities. They can also be used to govern more complicated operations such as token issuance, off-chain consensus building, and complex authorization schemes (e.g. multi-signature). While key hierarchies are useful for organizing priorities, identity hierarchies are useful for organizing objects. Objects might include different token types held by a user, different departments within an organization, or different data types collected by an array of IoT sensors. How a company organizes its departments, financial records, or data structures can be reproduced in the organization of ADIs, sub-ADIs, their data and token accounts, and their directories as defined below:

- ADI: A digital identifier that governs data, tokens, and identities. An ADI manages these assets using a set of hierarchical keys that it owns or shares with other ADIs.
- Sub-ADI: An nth generation ADI that is created within another ADI.
- ADI Data/Token Account: A terminal Data or Token Account belonging to an ADI.
- ADI Scratch Account: An ADI Token/Data Account with a transaction history that is only retained for 2 weeks.
- ADI Token Issuer: Defines a type of token that can be issued to ADI Token Accounts and Lite Token Accounts.
- Directory: An optional cataloging structure that organizes sub-ADIs and accounts but does not manage their contents.

3.2.1 Design

Any number of accounts or sub-ADIs can be nested within an ADI. Accounts and sub-ADIs are independent of each other and possess their own sets of keys with which they can manage their assets. The parent ADI possesses an administrative key set that can add, delete, transfer, or modify the security of its accounts or sub-ADIs. Data and token accounts are terminal, meaning that they cannot have nested accounts. However, sub-ADIs and accounts can be nested within another sub-ADI.

3.2.2 ADI Data and Token Accounts

ADI Data and Token Account URLs have the general format <prefix>://<ADI>/<directory>/<account> where the prefix *acc* : // specifies the Accumulate blockchain, *ADI* specifies the top-level identity in control of the URL, *directory* specifies a particular type of account, and *account* specifies data or tokens. Several examples of data and token accounts are provided in the table below for the hypothetical ADI 'bank'. Note that both cryptocurrencies and tokenized assets may be considered token accounts.

Table 1 Format for ADI Data and Token accounts.

ADI	**acc://bank**	**acc://bank**
Description	Data	Tokens
Directory	d	t
Account-1	acc://bank/d/accounts	acc://bank/t/loans
Account-2	acc://bank/d/investments	acc://bank/t/securities
Account-3	acc://bank/d/transactions	acc://bank/t/realestate
Account-N	acc://bank/d/nth	acc://bank/t/nth

Directories are optional, and we anticipate that their utility will be mostly confined to enterprise applications where a greater organization of large numbers of accounts and sub-ADIs is needed. The directory label is defined by the user and can be single or multi-character. For the sake of clarity, data, tokens, and identities are given the labels d, t, and i. However, an investment firm

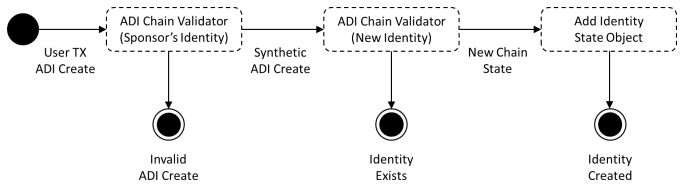

Fig. 3 Creating an ADI.

with ADI 'firm' may create data accounts with directories *r* and *c* for residential and commercial properties, while a cryptocurrency trader and non-fungible token (NFT) art collector may create token accounts with directories *NFT* and *stable* to denote NFTs and stablecoins. Any label can be used so long as the object type is specified within the wallet. However, a directory label cannot be changed after it is created.

3.2.3 Organization of Sub-ADIs

Sub-ADIs are organized differently from data and token accounts in that any number of sub-ADIs can be nested within another sub-ADI, which can extend infinitely in both the vertical and horizontal direction. This allows a user to create an arbitrarily complex directory structure to model anything from a file directory to a corporate hierarchy. In the example below, different departments within a bank are represented by different Sub-ADIs. The human resources department is further subdivided into sub-ADIs that specify different roles within the department such as manager and director. While not shown, any of these sub-ADIs may also have a nested data or token account.

Table 2 Sub-ADI account format.

ADI	**acc://bank**
Description	Identity
Directory	i
Sub1-ADI-1	acc://bank/i/auditing
Sub2-ADI-1	acc://bank/i/auditing/i/director
Sub3-ADI-1	acc://bank/i/auditing/i/director/i/manager
SubN-ADI-1	acc://bank/i/auditing/i/director/i/manager/nth

3.2.4 Combining Accounts and Sub-ADIs

Data and token accounts may also be nested within sub-ADIs. This offers an alternative strategy for organizing large numbers of data and token accounts without needing to create multiple directories. Using the example of an investment firm that manages different types of properties, we can imagine that a team specializing in residential homes may be given a different sub-ADI than a team specializing in commercial real estate. For simplicity, we refer to these combined addresses as URLs below.

Table 3 Format for complex ADIs.

ADI	**acc://firm**
Description	Variable
Directory	Variable
URL-1	acc://firm/i/residential/d/properties
URL-2	acc://firm/i/commercial/i/office/d/properties
URL-3	acc://firm/i/commercial/t/securities
Sub-ADI-N	acc://bank/i/nth

Since accounts are terminal, it is not possible to nest sub-ADIs within accounts. While acc://firm/i/residential/d/properties is permitted, acc://firm/d/properties/i/residential is not.

3.2.5 Creating an ADI

An ADI is created after a sponsor initiates a 'create identity' transaction on the network. The sponsor can be an organization or a user so long as they spend the required number of tokens. The validator on the sponsor's ADI will validate the create identity transaction, produce a synthetic transaction (those created by the protocol), and push that out to the network to be processed in a later block. The network where the new identity resides will receive the synthetic transaction from the Block Validator Network (BVN) and validate it against the current state of the network.

A successful identity creation requires 1) that signatures for the synthetic transaction must be valid and verified against the Directory Network (DN) receipt and 2) that the identity does not already exist on the network. If the above criteria are met, an identity state object is created, as illustrated in Figure 3. For an introduction to BVNs and DNs, please refer to the Litepaper[4]. Synthetic transactions will be covered in more detail in future technical documents.

3.2.6 Routing an ADI

The Accumulate network uses URLs to reference user accounts that are managed by an ADI. The URL consists of the following characteristics:

acc://ADI/Path

Where *ADI* refers to the user's identity on the Accumulate network and *Path* refers to accounts that are managed by an ADI. More accurately, the ADI is a Root Identity that is used to route transactions to BVNs. Each account and sub-ADI within a Root Identity is handled by one BVN that is determined after deriving the SHA-256 hash (represented by H function) of the ADI and hashing this a second time to create an address:

AdiChainId: H(lowerCase(ADI))

Both the ADI and account can be hashed such that the URL acc://RedWagon/AcmeTokens will be internally represented by the following:

H(redwagon) and H(redwagon/AcmeTokens), respectively.

ADIs are evenly distributed among BVNs, similar to how large data sets on social media platforms are evenly distributed to their servers. Network routing is determined by the first eight bytes of the SHA-256 hash of the ADI's account ID. The address corresponding to a particular BVN is determined by the following:

Address: uint64(H(*AdiChainId*) [0:7])

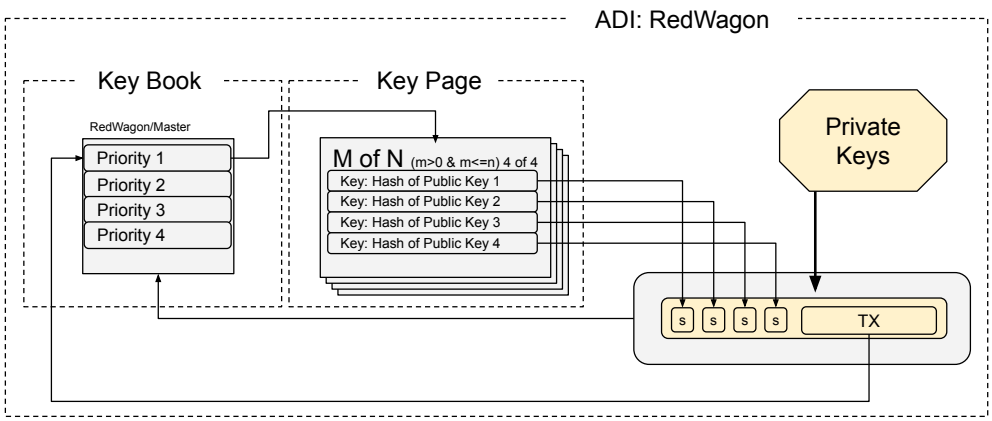

Fig. 4 Key management scheme example.

where $[0:7]$ takes the first 8 bytes, and *int* interprets those bytes as a number. The end result is a random address that can be processed internally and distributed to a particular BVN.

4 Key Management

Security and authorization in Accumulate are implemented with Key Books. Each ADI must contain at least one Key Book but multiple Key Books may be specified. A Key Book specifies a set of public key hashes and Key Book Authorities that are organized and prioritized as Key Pages.

4.1 Organization

The Key Pages within a Key Book are organized in order of priority, where the first Key Page is the highest priority and the last Key Page is the lowest priority. However, it must be noted that the priority of a Key Page is only relevant when managing the Key Book. Each Key Page can have one or more entries, each consisting of a public key hash and/or a Key Book Authority URL.

4.2 Management

The available operations for managing keys are:

- Creating an additional Key Book within an ADI.
- Creating an additional Key Page within a Key Book.
- Updating the public key hash of an entry in a Key Page.
- Updating a Key Page.

The authorization rules for updating the public key hash of an entry in a Key Page are unique. This particular type of transaction is always single-signature and can only be used to update the entry corresponding to the signer. If the transaction is signed with a simple signature, then the only entry that can be updated is the one with a key hash that matches the signer. If the transaction is signed with a Key Book Authority, then the only entry that can be updated is the one owned by that Key Book Authority.

All other modifications of a Key Page are done via updating the Key Page where all of the normal authorization rules apply.

4.3 Securing Accounts

Every account must specify a Main Key Book. The Main Key Book of an account is responsible for securing the account by authorizing signatures of transactions originating from that account. The Main Key Book is set when the account is created and cannot be modified thereafter. When an ADI is created, an initial Key Book is created along with the ADI, which becomes the ADI's Main Key Book. When an account is created, any Key Book within the ADI can be used as its Main Key Book. By default, accounts are created with the same Main Key Book as the ADI to which they belong. While all accounts are secured by a single Key Book by default, the user has the option to create a new Key Book for every account.

Each account may also specify a Manager Key Book, which serves as a second layer of authorization. A Manager Key Book can be set at creation, but it can also be modified after the fact unlike the Main Key Book. If an account specifies a Manager Key Book in addition to its Main Key Book, then transactions originating from that account must have authorized signatures from both Key Books. Managed transactions are introduced in Chapter 5.4.

4.4 Signature Threshold

Each Key Page has a signature threshold. When a Key Page is used to authorize a transaction, whether as part of the origin account's Main Key Book or Manager Key Book, the signature threshold determines how many signatures are required from the keys and/or Key Book Authorities of that Key Page. This is often referred to as $m-of-n$, where m is the signature threshold, and n is the number of entries. This general authorization scheme forms the basis for multi-signature support in Accumulate.

Signature thresholds are illustrated in Figure 4, where ADI 'RedWagon' submits a transaction through one of its Key Books and signs it using a number of keys that are specified in its priority 1 Key Page. In this $m-of-n$ authorization scheme, $n = 4$ entries and m is determined by the signature threshold. If the signatures are valid and the $m-of-n$ threshold is met, then the transaction is executed.

5 Authorization

Transactions are validated on most blockchains once a threshold number of signatures are submitted by accounts with access to the authorized private keys[8]. Authorization schemes are defined by the number of signatures required to validate a transaction, with single signature and multi-signature schemes being the most common. As discussed in the previous chapter, Accumulate provides flexibility in key management and enables a user or group of users to add, remove, or modify the security or ownership of their keys through the use of hierarchical key sets. The ability to manage keys over time not only enhances multi-signature transactions but provides the means to create new authorization schemes that can replicate a variety of complex financial operations. This chapter describes how Accumulate processes single and multi-signature transactions and introduces several new authorization schemes that expand the utility of multi-party transactions.

5.1 Single-signature Transactions

Single signature transactions have an $m-of-n$ signature threshold of 1, meaning that one signature is required to validate a transaction. This authorization scheme is used by the majority of individuals to send and receive tokens due to its simplicity and speed. Most wallets automatically derive a signature from the user's private key and transaction hash, so a transaction only requires a user to input the amount, destination address, and possibly a memo depending on the blockchain and the nature of the transaction. However, a single point of failure can result in a loss of funds, so users must be careful to protect their private keys.

When a user creates a transaction in Accumulate, they must specify the transaction type (e.g., data, tokens) and origin account. Each transaction specifies exactly one Key Page in the Key Book of that account, and all signatures must come from a Key on that page. If the transaction specifies Key Page 1, for example, and Key Page 1 specifies $m = 1$, then the transaction is immediately validated. This is similar to an embedded signature derived from a private key in a traditional blockchain address.

While the concept of Key Books, Key Pages, and Keys may appear complicated to the average user who simply uses their account to send and receive tokens, the authorization process from a user's perspective will not look any different from a transaction executed on another blockchain when $m = 1$. The user will always have the option to upgrade or downgrade their security in the future without having to create a new account.

5.2 Delegated Transactions

In addition to getting transactions approved by account managers within an ADI, it could also be necessary for applications to disperse tokens based on external authorization(s). This is the basic mechanism of smart contracts and is important to many applications of blockchain technology. However, smart contracts have exhibited many inefficiencies, which arise from the fact that they are computationally expensive[9]. Accumulate prevents this issue by limiting all transactions to a static amount of computation, ensuring that they take up a known amount of time, space, and energy on the blockchain. Within these constraints, Delegated Transactions allow for a system of third-party verification analogous to a smart contract while limiting their computational cost. Delegated Transactions allow for the transaction's origin account and the signing Key Book to belong to different Root Identities. As the name suggests, this is done by delegating the signature authority of a key to another Key Book. The Key Book of the origin account contains the URL addresses of the external Key Books authorized to sign transactions against that account. A Delegated Transaction must be sent to the signer's BVN first, so that the signature can be authorized against the signer's Key Book and added to the signer's Signature Chain. It is then sent to the origin's BVN to be executed, at which point synthetic transactions are produced if necessary, for example, to disperse funds.

The Key Book system allows for applications to tailor the number of required signers for a transaction to occur. For example, some transactions may only require the signature of a single authorized party, some require a $2/3$ majority, while some should require signatures of all involved parties. Each Key Page of a Key Book allows for a custom minimum number of signers to be applied to transactions.

5.3 Multi-signature Transactions

Multi-signature transactions have an $n \geq m > 1$ signature requirement, meaning that the number of signatures required to authorize a transaction is less than or equal to the total number of authorized keys specified by the originator, but always more than 1. This authorization scheme is primarily used for adding extra security when transferring valuable assets or coordinating signatures between multiple accounts owned by individuals, groups, businesses, or automated programs with access to a key[10]. The main advantage of multi-signature transactions is the need for multiple points of failure in order for an account to be compromised. Several examples of the real-world utility of multi-signature authorization are provided below:

- 1-of-2: A joint bank account where the signature of either member is sufficient to access the funds.
- 2-of-2: Automated spending control in which a computer program with access to a key will only sign for a transaction below a certain dollar amount.
- 2-of-3: Escrow transactions between two parties with a third party acting as an intermediary in the event of a dispute.
- 1 or 3-of-4: Single sig with distributed account backup from trusted parties in the event the owner's key is lost or stolen.

All of the above scenarios are possible to model on the Accumulate network using multiple Key Pages and Delegated Transactions. For example, an individual ADI may specify $n \geq m > 1$ for a single Key Page and control a multi-signature transaction entirely through their ADI. If one or more authority Key Books from external ADIs are specified in the Key Page of the origin account, then multiple ADIs can participate in a multi-signature transaction through Delegated Transactions. This process is illustrated in Figure 5.

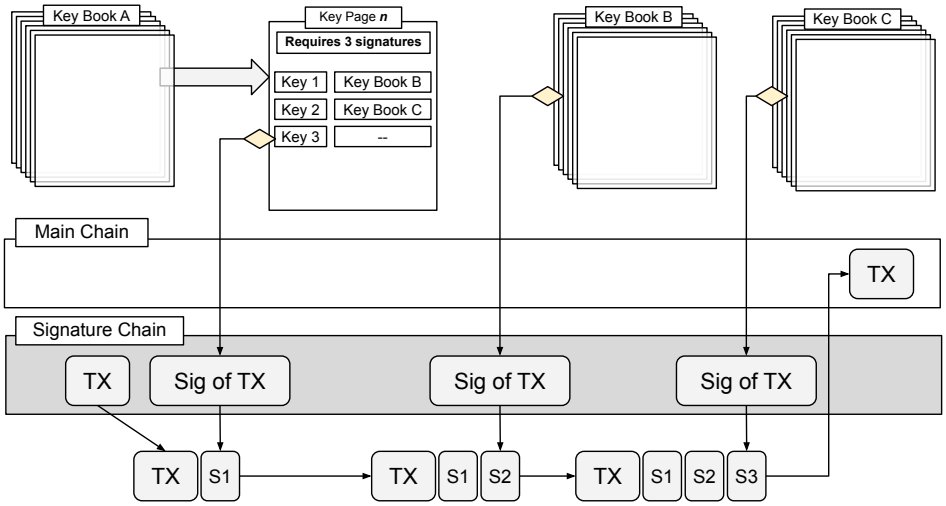

Fig. 5 Overview of multi-signature transactions.

Key Book A is the originator of a multi-signature transaction that requires at least 3 signatures in one of its Key Pages. Key Books B and C are specified as authority Key Books in the Key Page of Key Book A. The transaction is initiated by Key Book A, which signs with Key 3. An envelope is created from the hash of the signature and transaction and put onto the Signature Chain. Key Books B and C then sign the transaction with their keys, creating additional envelopes consisting of the hash of the original transaction and their respective keys. Once 3 signatures are collected on the Signature Chain, the transaction is promoted to the Main Chain. If the signature threshold is not met in approximately 2 weeks, then the transaction will be discarded. Signatures are discarded after 2 weeks regardless of whether the transaction succeeds.

5.4 Managed Transactions

Every account has a Main Key Book and an optional Manager Key Book that users can specify when their account is created. The Main Key Book can initiate single and multi-signature transactions and specify multiple authority Key Books within a Key Page to participate in Delegated Transactions. A Manager Key Book uses its own $m-of-n$ authorization scheme to approve or reject any type of transaction initiated by the Main Key Book. While a manager can have multiple keys, an account can only specify a single manager. A manager is also unable to execute a transaction on their own. This makes managed transactions most useful for automated security without custodial control, as illustrated in the following examples.

- Spending limits: A manager may enforce a spending limit similar to withdrawal limits on ATMs.
- Frequency limits: A manager may allow only a certain number of transactions per day.
- Transaction type limits: A manager may restrict transactions to certain types (e.g., tokens, data).

Transactions with various authorization schemes can be executed depending on whether a Manager Key Book, Main Key Book, or both Key Books are required to sign a transaction. Most users will specify both a Main and Manager Key Book to automate security for a user-generated transaction as an extra layer of security. For example, ADI token account acc://alice/tokens may specify Key Book acc://bob/book as a manager. If a transaction is initiated by ADI Alice, then a threshold number of signatures from the specified Key Page in ADI Alice and any Key Page in ADI Bob is required to validate the transaction. If a manager is specified, but $M = 0$ for a Main Key Book, then a manager can execute transactions that are initiated by an account. This authorization scheme may be useful for automating recurring payments (e.g., utility bills). If $M = 0$ for both a Manager and a Main Key Book, or a Manager is not specified and $M = 0$ for the Main Key Book, then anyone with the account information can execute a transaction. Due to the security risks involved, this authorization scheme may only be useful for data transactions among collaborators. These authorization schemes are formally defined below in terms of their $m-of-n$ signature requirements.

When a manager is not specified:

- If $M = 0$, then anyone in the Accumulate network can execute transactions.
- If $M > 0$, then anyone can suggest a transaction, but at least one Key from the Key Page of the Main Key Book or from any Key Page in an authority Key Book also has to sign it.

When a manager is specified:

- If $M = 0$, then anyone can suggest a transaction, but the manager also has to sign it.
- If $M > 0$, then anyone can suggest a transaction, but the manager and Key Page also have to sign it.

The process of signing managed transactions is illustrated in Figure 6. TX1 requires two Key Page signatures from Key Page 3 of its Main Key Book and two manager signatures from the Manager Key Book. While the Main Key Book signature requirement

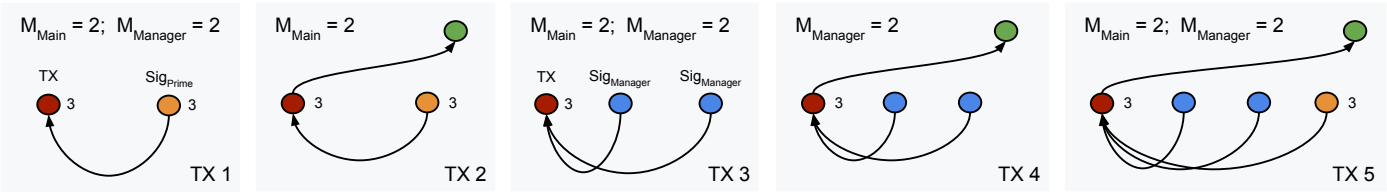

Fig. 6 Managed transactions.

is satisfied due to one embedded signature and one independent signature, the transaction fails because there are no manager signatures. However, TX2 would be promoted to the Main Chain, represented by a green dot, because it does not specify a manager signature. TX3 has the same signature requirements as TX1. While two manager signatures have been provided, the transaction fails because there are no signatures from the Main Key Book. However, TX4 would be promoted to the Main Chain because it does not specify a signature from the Main Key Book. TX5 satisfies the signature requirements for both the Main and Manager Key Books, causing it to be promoted to the Main Chain.

6 Architecture Overview

The Accumulate protocol has a 'chain of chains' architecture in which account chains on a particular BVN are treated as side chains of that BVN's root chain, and the root of each BVN is treated as a side chain of the DN's root chain. This architecture enables massively parallel transactions and virtually limitless scalability as capacity is added in the form of additional BVNs. While the reorganization of the blockchain as a 'chain of chains' built around accounts rather than blocks solves many persistent problems such as the scalability trilemma, it also introduces new challenges, including the efficient communication of receipts between BVNs, the storage of data generated by accounts that function as continuously growing Merkle trees, and the coordination of consensus between multiple BVNs and the DN. This chapter provides a detailed overview of the account structure, data structure, data persistence, and system architecture that allows Accumulate to overcome these challenges and bring new innovations to the blockchain space.

6.1 Data Structure

6.1.1 Stateful Merkle Trees

A Merkle Tree is a balanced binary tree with a single root that can be derived from only a fraction of its hashes[11]. This allows for efficient verification that a particular hash and the transaction data associated with it is a genuine member of the tree. Accounts chains are continuously growing Merkle Trees that append, concatenate, and anchor new transaction hashes into their respective BVN Root Anchor Chains every major and minor block. Due to the irregular frequency at which new hashes are added, accounts are expected to contain a variable number of roots within a block. This data structure is a special form of Merkle Tree, called Stateful Merkle Tree (SMT). SMTs may have multiple roots in different time frames and are constructed in a way that allows them to grow over time, whereas most Merkle tree variations are constructed once and never modified after the root is obtained. This is made possible by keeping a list of root hashes within a tree.

SMTs provide the support for building and maintaining information in accounts across multiple blocks. An SMT can collect hashes of validated data from multiple sources and create a Merkle Tree that orders all these hashes by arrival time to the SMT. The arrival order of entries from all sources is maintained in the tree. SMTs can also be fed into other SMTs to create the superstructure of the Accumulate network in which all accounts that are managed by a particular BVN are anchored into its respective Root Anchor Chain, and all BVN Root Anchor chains are anchored into the DN.

An SMT is append-only, meaning that hashes are added from left to right, and a concatenated hash pair (i.e., parent) is added as soon as two children hashes exist on the same level of the tree. Unlike a traditional Merkle tree, an SMT has multiple roots. Therefore, we need a method to compute a single root from all the roots of every major and minor block. The process of combining roots is illustrated in Figure 7, in which new transaction hashes are appended to a growing SMT. In this diagram, blue dots represent transaction hashes, red dots represent the root of a particular subtree that defines its current state, and orange dots represent the previous roots that defined the previous state. The SMT is constructed on the left, while the column on the right summarizes the current state at each level where Level 1 is considered the entry point of a new transaction hash at the leaves of the tree.

We begin 15 transaction hashes into the SMT, where 4 different root hashes in 4 independent subtrees summarize the current state of the tree. When the next hash is added, the system recognizes that a root hash already exists on Level 1 to the left of the incoming hash. The two children are automatically concatenated to produce a root hash at Level 2. However, the system also recognizes that a root hash already exists on Level 2 to the left of the parent root hash. This process continues until the root hash on Level 4 is concatenated with the parent hash created on Level 4 to produce a final root hash on Level 5. Note that the current state at this point can be represented by a single hash. When the next hash is added on Level 1, it is treated as a second root hash since no other roots exist to the left.

To calculate an anchor at any point in time, the rightmost hash is successively concatenated with the root hash to the left until a single hash remains. For example, the four root hashes in Figure 7 can be concatenated into an anchor without the addition of a new transaction. Since the entire state of the SMT is captured in the root hashes, the previous hashes can be discarded or 'pruned'

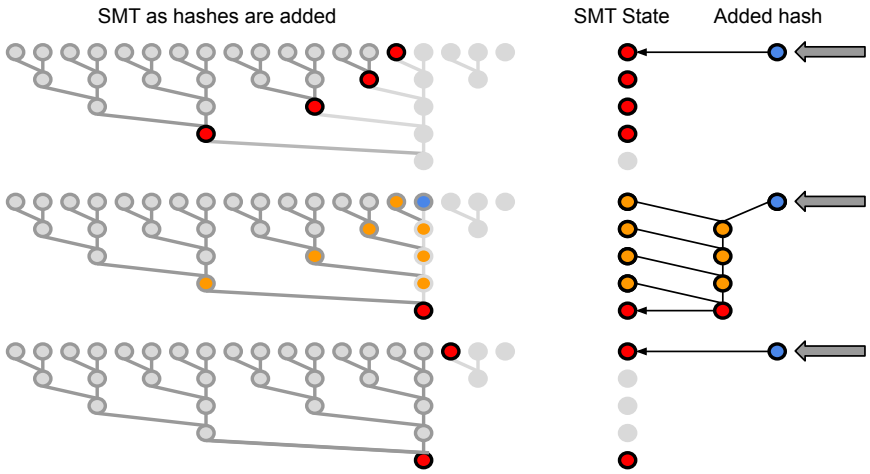

Fig. 7 Adding hashes to a Stateful Merkle Tree.

from the tree to minimize its size. The entry hashes are saved in regular accounts on the Main Chain to maintain a historical record of each entry. However, entry hashes older than 2 weeks are pruned from Signature Chains such that only the root hashes of the current state are recorded. The process of pruning entry and intermediate hashes on the Signature Chain forms the basis for Scratch Accounts, which are introduced in Chapter 6.3.

6.1.2 Binary Patricia Tries

Patricia Tries[12] are generally designed to create small cryptographic proofs about the particular state of values at a particular point in time. On a blockchain, a Patricia Trie can be used to prove the balance of an account (i.e., an address) at a particular block height. Since Accumulate organizes the blockchain into a series of chains under the management of different accounts, we need the ability to prove the state of a chain at a particular block height and index the block where a chain was last modified. The Binary Patricia Trie (BPT) provides these proofs for the state of each chain.

A BPT is similar to a Merkle Tree in that both hash a set of values that are ultimately concatenated into a single root. However, the BPT also adds a key per entry to create a key/value pair where the key refers to the hash of a chain's URL (e.g., acc://RedWagon). While a Merkle Tree organizes entries by the order in which they are added, a Patricia tree organizes records according to the key of each record. The major differentiating feature of a BPT is that the order in which entries are added does not matter. Within Accumulate, the BPT serves as a cryptographic summary of the current state of every chain and account within a BVN. This enables the network to reach consensus at a frequency of one minor block and guarantee that all nodes have the same state. Additionally, the BPT is used to create snapshots of the state of the network, which new nodes can use to get up to date instead of executing all of the transactions.

6.1.3 BPT Implementation

The Accumulate protocol has three entry types in the BPT:

- Node. Node entries are used to organize the tree. They have a left path and a right path and exist at a height in the BPT. The height is used to consider a particular bit in the BPT. The Left path is taken if the key has a zero bit at that point. The Right path is taken if the key has a 1 at that point.
- Value. The key-value pair in the BPT. Value entries have no children, and paths through the BPT from parent to child Node entries must end with either a nil or Value entry.
- Load. Node represents the fact that the next Byte Block is not loaded and needs to be loaded if the search is running through this part of the BPT.

The most important feature of a BPT is its ability to quickly update its state to provide proof of the state of the entire protocol. This ensures that changing the order in which entries are added to the BPT has no effect on the result. The same set of entries will always produce the same structure. Further, subsets of the BPT are independently and provably correct so long as there is a path from the root of the Patricia Tree to one of its subsets.

Many implementations of Patricia Trees are described in the literature[13]. In the Accumulate protocol, keys are randomly distributed from a binary point of view because the keys are the hashes of URLs. Mining these hashes to build some particular pattern of leading bits is not very manageable or possible as the keys are derived from URLs. URLs can be mined to have interesting leading bits, but little incentive exists to do so. Given that the keys created from hashes can be relied on to be numerically random, a BPT will be very well-balanced if this feature is exploited. Note that we use the leading bits to organize entries in the BPT. However, nothing prevents us from using every 3rd bit, the trailing bits, or any other random walk of bits in the key. Should an attack be mounted to create chain IDs that significantly unbalance the BPT, we can refactor the BPT using any of these methods and do so over time by reorganizing only parts of the BPT.

6.1.4 Summary

The organization, membership, and state of chains with an account can be summarized as follows:

- Accounts can act as domains to allow addressing of their Main and Signature Chains as a set of URLs.
- The membership of Chains is proven using Stateful Merkle Trees.
- All entries in a chain are members of one continuously growing Merkle Tree.
- At any point in time, the membership of the Merkle Tree is provable by the Merkle State.
- BPTs can prove the state of every account and chain in a BVN at a particular block height.

The Accumulate protocol is able to maintain proof of its state without requiring all elements of that state to be at hand. This allows the network to efficiently scale and makes it possible for mobile devices to prove the entire state of accounts relevant to the user. Accumulate organizes these proofs as a very large set of chains. Even token transactions are organized as chains where the membership and order of transactions from a particular account exist in its own chain.

Within Accumulate, the BPT captures account proofs in the BPT as key/value pairs, where the 'key' is a hash derived from the account URL, and the 'value' is the final hash of a proof. Account proofs are constructed as transient Merkle Trees that take as inputs the hash of the account state and an anchor from each of the account's chains. The summary of all key/value pairs is a hash that acts like the Merkle Root of a Merkle Tree. Additions to or modifications of this Merkle Tree-like structure only require localized re-computations, making it easy to update the state of the entire network at the frequency of a minor block.

6.2 Account Chains

Each account has a separate Signature Chain and Main Chain. The Signature Chain is responsible for collecting signatures, while the Main Chain is responsible for executing transactions that have met the signature threshold.

6.2.1 Signature Chain

All transactions submitted to the Accumulate network are added to an account's Signature Chain. This includes data and token transactions submitted by Lite Accounts, data and token transactions submitted by ADIs, managed transactions approved by a Manager Key Book, and Delegated Transactions submitted by an external Key Book. Single signature transactions are immediately promoted to the Main Chain if the signature included in the envelope is valid and corresponds to a key in the origin account's Key Page. Multi-signature transactions are not promoted to the Main Chain until the $m-of-n$ signature threshold is reached.

The primary utility of the Signature Chain is the ability to submit signatures independently of transactions. This is accomplished by hashing the transaction once by itself and once again as an envelope that includes the hash of the transaction and the hash of the signatures. Adding the envelope hash to the Signature chain provides cryptographic proof that the signatures were obtained. The first person to submit the multi-signature transaction sends the raw transaction along with their signature to create the first envelope. Once the signature threshold is reached, the transaction is promoted to the Main Chain.

Signatures are collected completely on-chain and over a reasonably long time frame so that signors can be notified, transactions can be modified, and signatures can be collected. Moving these activities on-chain provides an audit trail of the consensus-building process. The lifetime of signatures and transactions submitted to the Signature Chain is approximately 2 weeks. This means that a transaction will fail if the $m-of-n$ signature threshold is not reached within the allotted time.

6.2.2 Main Chain

The Main Chain of an account permanently stores transaction hashes without their signatures because the signatures have already been validated on the Signature chain. The data structure of the Main Chain is similar to that of the Signature chain in that each account's Main Chain is a continuously growing Merkle Tree. After each major block, any entries on the Signature Chain older than 2 weeks are discarded.

The illustration provided in Figure 8 shows the process of collecting signatures on the Signature Chain and promoting the transaction hashes to the Main Chain. The top row represents the Main Chain, while the bottom row represents the Signature Chain. Blue circles represent signatures, and red circles represent envelopes for four Key Pages within a Key Book where:

- Priority 0 specifies 1 key required.
- Priority 1 specifies 2 keys required.
- Priority 2 specifies 3 keys required.
- Priority 3 specifies 6 keys required.

The transaction managed by Priority 0 Key Page only requires a single signature to satisfy the $m-of-n$ requirement. If a transaction is suggested in a previous block, but the final signature is provided in a later block, the transaction is promoted to the Main Chain in the latter block. For example, the transaction managed by the Priority 2 Key Page in Block 0 and signed by the second key in Block 1 is also submitted to the Main Chain in Block 1. Signatures must also be provided from the appropriate Key Page. For example, a signature from the Priority 2 Key Page cannot sign a transaction from the Priority 3 Key Page.

6.3 Scratch Accounts

The architecture of Accumulate allows for the efficiency of the state in validating transactions, collecting signatures, and coordinating consensus. Transactions can be compressed into 12-hour blocks because the Accumulate blockchain is a collection of Merkle trees with arbitrary sync points. Mobile devices can be used as Lite nodes to validate only those transactions that are relevant to an account because each account is treated as an independent blockchain. Signatures can be pruned before a multi-signature transaction is validated because Accumulate has a transient Signature Chain and a permanent Main Chain.

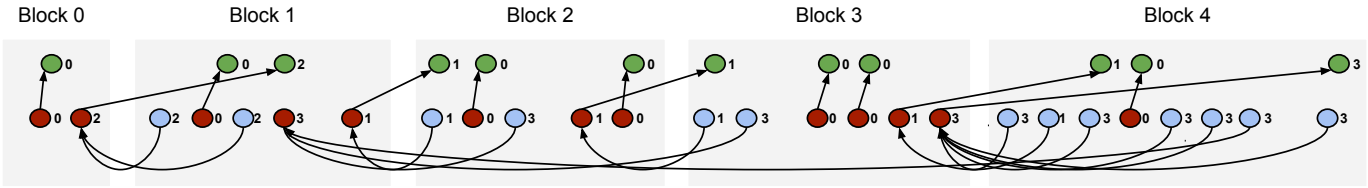

Fig. 8 Collecting signatures on the Signature Chain and promoting the transaction hashes to the Main Chain.

These features can also be combined to create Scratch Accounts, a specialized type of account with limited availability that facilitates cheap on-chain consensus between multiple parties and provides cryptographic proof of validation without overburdening the blockchain.

6.3.1 Implementation of Scratch Accounts

ADI Data Accounts and ADI Token Accounts can be marked as Scratch Data or Scratch Token Accounts when the account is first created. The only difference between an ADI account and a Scratch Account is that the data availability for a Scratch Account expires after approximately 2 weeks, which is similar to the lifetime of transactions and signatures present on the Signature Chain. Once the data expires, the account is compressed, and a proof of the transactions is created. This proof contains less data than the Scratch Account it represents, which allows the blockchain not to be burdened with moving substantial amounts of data over the main network.

To view the current state of data or token transactions in a Scratch Account, a user can query the latest entry on the chain using the following API commands:

$$Query(acc://adi/account\#data))$$
$$Query(acc://adi/account\#transaction))$$

A user can also query the history of data or token transactions in a Scratch Account over a period of time by specifying an arbitrary range:

$$Query(acc://adi/account\#data/0:10))$$
$$Query(acc://adi/account\#transaction/10:50))$$

The main parameter of *query* is a URL, which contains generic labels for *adi* and *account* that are specified by the user. The end of the URL is appended with the *#data* or *#transaction* fragment, which directs the search to the appropriate account type. The range (e.g., 0:10) provides a historical record of entries in sequential order. A query of 0:10 would retrieve records beginning with entry 0 and ending with entry 9. However, any entries older than 2 weeks will be pruned, meaning that they will not be stored by the protocol.

6.3.2 Pruning Data

Pruning is a data compression strategy designed to reduce the size of a data tree by removing unnecessary or redundant data [14]. In a typical blockchain, pruning refers to the removal of intermediate and leaf (transaction) hashes once the transaction outputs on which those hashes rely have been spent. This strategy significantly reduces the data footprint of the blockchain and allows a pruned node to fulfill many of the same functions as a full node. On Accumulate, pruning applies to Signature Chains and Scratch Accounts. All signatures on a Signature Chain and all transactions on a Scratch Account are automatically pruned after approximately 2 weeks.

Unless an account is deliberately purged by a user, the Merkle tree will still contain the historical record of a Scratch Account after its data is pruned. Transaction hashes and intermediate hashes are deleted, but the root hashes that summarize the state of a Scratch Account will remain. This is illustrated in Figure 8, where a Stateful Merkle Tree is being created from its transaction hashes. In a typical account, transaction hashes in the top row of the Merkle tree are stored by the data servers to maintain a record of every entry. Root hashes (red) that define the current state of the account are also stored by the data servers. In a Scratch Account, transaction hashes are pruned, but the root hashes are saved on-chain. This allows any user with access to the transaction hashes to cryptographically prove the data in a Scratch Account by deriving the root.

6.3.3 Utility of Scratch Accounts

Scratch Accounts allow a user or organization to build arbitrarily complex use cases for a low cost without burdening the blockchain. The following list provides several real-world applications that can benefit from Scratch Accounts.

- Isolating events within high-frequency data: IoT networks can collect high-frequency data on Scratch Accounts, prune baseline data, and promote important events (e.g., temperature fluctuations) to the Main chain. Baseline data can be periodically compressed to produce summary information about a system.

- Deriving oracle data: Miners can scrape pricing information from public pricing feeds, mine the pricing data, and submit records on Scratch Accounts for rewards at a fraction of the cost of writing data to the Main Chain. The winning data is put on the Main Chain while the losing data is pruned.

- Proving transactions on exchanges: It is cost-prohibitive for a major exchange to provide cryptographic proof of every transaction. With Scratch Accounts, exchanges could store transaction data off-chain and cheaply derive root hashes from pruned data.

6.4 Anchoring

Anchoring is the process of inserting external data into a blockchain transaction to provide an immutable receipt that can be verified by anyone with access to the original data [15]. This is typically accomplished by hashing the data on or off-chain, constructing a Merkle tree from that data, and periodically inserting its Merkle root into a larger and more secure blockchain. Factom, the predecessor of Accumulate, was one of the first protocols to anchor into Bitcoin and Ethereum. By hashing its transactions into larger directory blocks and combining directory blocks into larger anchors, Factom was able to maintain an immutable record of thousands of hashes at the cost of a single transaction.

Accumulate retains the ability to anchor into other protocols, but it also requires internal anchors as a consequence of its chain-of-chains architecture. Accounts are connected by anchoring their roots to their respective BVNs, BVNs are connected by anchoring their roots to the DN, and blocks are connected by anchoring the root of the DN into each BVN. Anchoring each of these elements allows Accumulate to be a blockchain network as opposed to a collection of independent protocols.

6.4.1 Anchoring Within a Chain

In most blockchains, the beginning and end of a Merkle tree are bounded by the block time. In Accumulate, the end of a block is treated as synchronization points that simply designate when transactions should be anchored. Merkle trees transcend blocks, which means that every chain in Accumulate can be modeled as a continuously growing Merkle tree. However, the freedom to append new hashes on a continuous basis may result in a Merkle tree having multiple roots at any point in time. This bloats the network and adds unnecessary complexity to the protocol when validating transactions.

Accumulate addresses this issue by organizing its hashes with a Stateful Merkle Tree (SMT). Hashes from any account chain (e.g., Main Chain) or system chain (e.g., Synthetic Transaction Chain) that changed since the previous block are concatenated into roots at the end of the current block. Multiple roots are concatenated into a single root (i.e., anchor). This process is illustrated in Figure 9 for a generic account chain where gray rectangles represent blocks, gray circles represent raw transactions, yellow circles represent hashes, and red squares represent anchors. For the sake of clarity, we refer to each row of hashes as a "Level," where Level 0 refers to the top row (i.e., leaf) hash of a raw transaction.

At the end of every block, hashes are concatenated into roots. In Block 0, for example, one root is produced from transactions 1-4, and a second root is produced from transaction 5. These roots are concatenated into a single anchor labeled "Anchor 0". If the root of a previous block is on a higher level than the root of the current block, then these roots must be connected through intermediate anchors before a final anchor is produced. Block 2, for example, has two root hashes on Levels 0 and 1. These are concatenated into an intermediate anchor on Level 2. This intermediate anchor is concatenated with the root on Level 3, Block 1, to produce Anchor 2. This anchor summarizes the entire history of the account chain until this point because it contains every hash from Blocks 0-2.

The number of intermediate anchors required to produce a final anchor that summarizes the history of the chain until that point is equal to the number of roots in previous blocks that are at a lower level than the root in the current block. Block 5, for example, has a single root at Level 0. Block 4 has a single root at Level 1. Block 3 has two roots at Levels 2 and 4 that are higher than Level 0. Therefore, Block 5 needs two intermediate anchors to produce Anchor 5.

6.4.2 Anchoring Within a BVN and the DN

Every chain in a BVN, except for the Binary Patricia Trie (BPT), appends its hashes to an SMT whose anchor represents the current history of its chain. Every chain, with the exception of the Root Anchor Chain, inserts its key/value pairs into the BPT, whose anchor represents the current state of its chain. These anchors are then fed into the Root Anchor Chain, whose anchor represents both the history and the state of every chain within a particular BVN. This process occurs in parallel within every BVN in the network and also within the DN, which contains its own BPT, Synthetic Transaction Chain, and a Main and Signature Chain for the ACME token issuer.

The process of collecting hashes, producing roots, and anchoring roots into other chains can be modeled as a fractal. This is illustrated in Figure 2 for the DN, an arbitrary number of BVNs, and an arbitrary number of account chains within a BVN. This hierarchical organization of hashes is the basis for calling the Accumulate network a chain-of-chains.

6.4.3 Anchoring between a BVN and the DN

Accumulate partitions its network through the use of BVNs, which are independent networks of Tendermint nodes that process transactions in parallel. Scaling is achieved by adding more BVNs. However, partitioning can also lead to inefficiencies in communication between BVNs. Consider the case of 3 BVNs in the absence of synthetic transactions or the DN. An account on BVN-1 sends a transaction to an account on BVN-2, but neither BVN has a complete record of the account balances of the other so queries need to be made to all 3 BVNs. Mathematically, this is represented by $N * N - 1$ total queries. As the number of BVNs increases, the number of queries required to communicate debits and credits will grow exponentially.

Accumulate addresses this issue by sending anchors and synthetic transactions from each BVN to the DN, then sending an anchor from the DN to each BVN. This process can be visualized by once again considering a network of 3 BVNs. An account on BVN-1 sends a transaction to an account on BVN-2, which results in the creation of a synthetic transaction by BVN-1. In order for BVN-2 to trust the synthetic transaction, it must be able to prove that the synthetic transaction was included in a completed DN block. This synthetic transaction is concatenated with other synthetic transactions generated by accounts in BVN-1 that block and sent as an anchor to the DN's Intermediate Anchor Chain. The Intermediate Anchor Chain collects synthetic transaction anchors from BVNs 1, 2, and 3 and combines them into a single anchor within its Root Anchor Chain. This anchor is sent back to BVNs 1, 2, and 3 as a receipt. BVN-1 validates the anchor and creates a

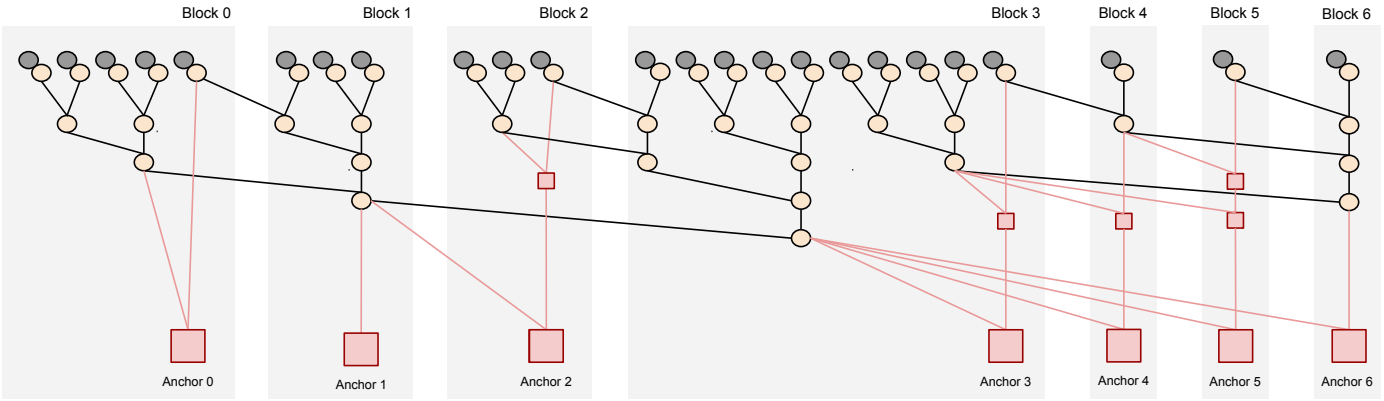

Fig. 9 Anchoring transactions within a Stateful Merkle Tree.

receipt for the synthetic transaction, starting with the hash of the synthetic transaction and ending with an anchor from the DN's Root Anchor Chain. It packages up that receipt as a signature and sends this as a synthetic transaction to BVN-2. However, BVN-2 has also validated this block of the DN, so a simple Merkle proof is all that's needed for BVN-2 to prove this synthetic transaction. This process reduces the number of queries to $2N$, which allows the network to scale linearly as more BVNs are added.

6.4.4 The Utility of Anchoring

Bitcoin was originally designed as a trustless and permissionless alternative to cash that was free from the control of centralized banks and third-party intermediaries. As businesses began to invest in Bitcoin and offer financial products such as Bitcoin ETFs, Bitcoin evolved from a peer-to-peer network for sending digital currency between individuals to a store of value and a hedge against inflation. However, the institutional adoption of Bitcoin for purposes other than investment has been relatively slow to develop. Businesses have little incentive to adopt a trustless solution if they have established relationships with partners that they already trust, and regulation limits the activities that a user can engage in without permission.

Accumulate believes that the business value of Bitcoin lies primarily in its utility as an immutable historical ledger for anchoring external data. This concept gained mainstream attention in 2015 when DocuSign partnered with Visa in a pilot study to anchor DocuSign contracts into Bitcoin transactions using Visa's payment rails. External data can also be anchored into Bitcoin by other blockchain protocols. Factom, for example, regularly anchored its directory blocks into Bitcoin and Ethereum to bolster the credibility of its Proof of Authority (PoA) consensus mechanism.

Accumulate expands the utility of anchoring through its use of internal and external anchors by virtue of its multi-chain architecture and cross-chain capabilities. Internal anchors can be used for cryptographic timestamping within the Accumulate network. External anchors can be exchanged with any blockchain protocol that produces a summary hash, allowing Accumulate to act as a universal bridge between the blockchain and the business world. This bridge enables a variety of business use cases, a few of which are summarized below:

- Tamper-proof records: Accumulate will regularly anchor its DN root hash into Bitcoin transactions to create timestamped receipts. Any customer with access to the original data can derive the anchor and cryptographically prove the validity of the receipt.
- Proof of existence: Internal anchors are compact historical records that can be used to prove the existence and order of transactions at the frequency of a minor block. The valuation of a piece of real estate at a given point in time, the purchase of an insurance policy before a claim is submitted, and the use of a license before a contract expires can all be managed within the Accumulate network.
- Cross-chain bridges: Anchoring allows Accumulate to extend its Merkle proof into another blockchain. For example, data from Tezos can be validated through an Ethereum smart contract using Accumulate as a bridge.

6.5 Major and Minor Blocks

6.5.1 Organization of Blocks

In a traditional blockchain, all transactions submitted by all accounts within a block are concatenated into a root hash inside of a single Merkle tree[16]. A new Merkle tree is generated in the subsequent block and tied to the state of the previous block by including the previous root hash in its header. Synchronization between nodes on the network is handled on a per-block basis, which means that transactions cannot be processed until a new block is created. This typically occurs on the scale of minutes for Proof of Work (PoW) blockchains like Bitcoin[17] and seconds for Proof of Stake (PoS) blockchains like Algorand and Ethereum 2.0.

The Accumulate protocol takes a different approach by organizing itself around accounts that define blocks rather than blocks that contain Merkle trees. Each account is a continuously growing Merkle tree with arbitrary blocks that can be validated at multiple synchronization points to coordinate consensus between BVNs. These synchronization points are defined as Minor and Major blocks, with approximate block times of 1 second and 12 hours, respectively.

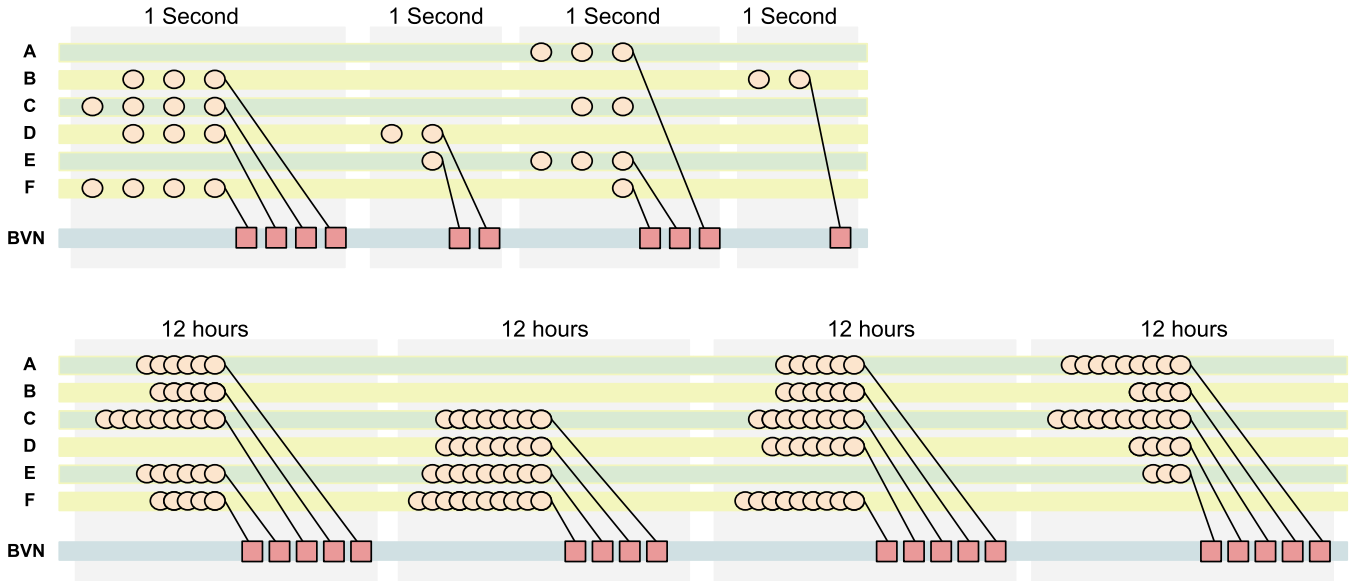

Fig. 10 Major and minor blocks.

6.5.2 Minor Major Block Architecture

Minor blocks are used to establish what set of accounts within a particular BVN changed within the last second. Major Blocks use Minor Blocks to establish what set of accounts changed in the last 12 hours. Changes to the Transaction and Signature Chains of each account and to the Synthetic Transaction Chain of a BVN are recorded as root hashes that are anchored into the Minor and Major Anchor Chains of that particular BVN. This process is illustrated in Figure 10 for hypothetical accounts A-F across several minor and major blocks on a single BVN.

Yellow circles represent transaction hashes submitted by accounts A-F. These transaction hashes are concatenated in the Merkle tree of each account until a root hash is derived. Red squares represent the root hashes that are anchored to the BVN Minor and Major Anchor Chains. The order in which each anchor is added to the Anchor Chain is determined by the account ID, which is derived by hashing the entire account URL and collecting the first 8 bytes of the hash.

6.5.3 Benefits of Multiple Block Times

Most blockchains are committed to a single block time because each block is a part of the entire network with a well-defined beginning and end. However, there is no block that represents the entire Accumulate network because each account is treated as an independent blockchain. Similarly, transactions submitted by an account are processed by one of many BVNs in the network that coordinate consensus but otherwise operate independently. This architecture allows Accumulate to define multiple block times of arbitrary length without disrupting the network.

The use of multiple block times allows the Accumulate network to quickly process transactions at speeds comparable to the fastest PoS protocols while giving users the ability to store the entire state of the network on a mobile device. Indexing on two different time scales allows for the possibility of reducing the required storage space by increasing the granularity of indexing for past transactions. For example, entries on the minor anchor chains may be discarded after 2 weeks to save space. As a result, the granularity of the data is reduced to 12 hours after that point.

6.6 Synthetic Transactions

6.6.1 Database Scaling

How a business chooses to store and process its customer data largely depends on the size of the database, its read/write volume, and the availability requirements of its data. A small business may find it more convenient and economical to store its data in a database handled by a single server and vertically scale its hardware by upgrading its RAM, CPU, and storage capacity. It may also choose to replicate its database across multiple servers to make its data available even if some hardware fails or a server crashes. However, vertical scaling is not a viable option for a larger organization like Facebook, which generates upwards of 4 petabytes of data per day.

Instead, large and data-centric corporations often use a horizontal scaling strategy called sharding that distributes a single dataset across multiple databases and stores these databases on multiple servers or nodes. With this architecture, scalability and throughput are no longer limited by hardware but by capacity. Adding capacity by installing additional servers allows a database to scale while distributing the data across different shards increases throughput by allowing parallel read/write operations.

6.6.2 Scaling Solutions in Blockchain

Blockchain also utilizes a form of horizontal scaling, which relies on a system of rewards for those who do the work of validating transactions and securing the network[18]. Stakers in PoS blockchains and miners in PoW blockchains lend their tokens or computational power to confirm transactions. Full node operators for PoS blockchains store copies of the entire blockchain and secure the network, receiving rewards at the risk of losing their

stake if they engage in bad behavior.

A major difference in database management between traditional enterprises and blockchain protocols is the decentralization of the latter. Unfortunately, this also introduces the Scalability Trilemma, which can be defined as a trade-off between decentralization, security, and scalability. Bitcoin, for example, chose security over scalability when it introduced a delay in the form of blocks. EOS, meanwhile, chose scalability at the cost of decentralization.

6.6.3 Scaling Solutions in Accumulate

Accumulate bypasses the scalability trilemma by organizing itself around digital identities (e.g., ADIs, Lite Accounts) and treating each account as an independent blockchain. All accounts and the data contained within can be operated upon independently by design and validated by a series of BVNs that operate in parallel.

Each BVN is a network of Tendermint nodes that is responsible for validating transactions initiated by an account. Each account is assigned to a particular BVN, and as more identities are created, more BVNs can be added to scale the network and maintain high throughput. The summary hashes of an account's current state are anchored to the BVN. Meanwhile, each BVN is anchored to the DN, which is responsible for coordinating the BVNs and guaranteeing security for the network. Security is further enhanced by anchoring the DN to a secondary blockchain such as Ethereum or Bitcoin to gain the added security provided by those networks.

While each identity acts as its own independent blockchain, transactions between the BVNs that host these identities necessitate a query of the sender's transaction history to determine if they actually possess the assets they wish to send. Since no BVN contains the entire state of the Accumulate network, each BVN must be independently queried. As more BVNs are added to scale the network, the majority of queries will involve BVNs that have no transaction history with the sender. This puts an unnecessary load on each BVN, which increases transaction cost and decreases transaction throughput.

To solve this problem, a third type of node network is introduced, which is called the Data Server Network (DSN). The DSN does not have the requirement of high validation throughput. It will move the transaction data from the BVNs and organize the transaction history into organized chains. This reduces the resource load on the BVN nodes, enabling the BVNs to focus on managing the identity states while allowing the DSN to maintain the transaction history for those states.

6.6.4 Introduction to Synthetic Transactions

To optimize transaction throughput, Accumulate uses synthetic transactions, which can be broadly defined as any transaction that is produced by the protocol rather than by the user[19]. To understand how synthetic transactions work, consider the classic example of Alice, Bob, and Charlie, who will be treated as separate ADIs on the Accumulate network.

Alice's transactions are handled by BVN-1, Bob's by BVN-2, and Charlie's by BVN-3. Bob needs to pay Charlie, but he can only do so once Alice pays him back. Alice sends a transaction to Bob after BVN-1 verifies that Alice has sufficient funds. Bob receives the tokens and sends a transaction to Charlie through BVN-2. However, BVN-2 does not know about the transaction or from which BVN it came since Alice sent her transaction through BVN-1. Therefore, BVN-2 will have to query all other BVNs to verify that Bob has enough money to pay Charlie. This problem can be resolved by creating transaction flows with synthetic transactions.

After BVN-1 has validated Alice's transaction, it sends a second transaction to BVN-2 that says, "Deposit X tokens into Bob's account". The transaction that BVN-1 sends to BVN-2 is called a synthetic transaction. Now both BVNs have a complete record of everything that has happened to their respective chains, and future transactions involving these chains will not require either BVN to query others in the Accumulate network. To ensure that no fake synthetic transactions can be injected into the system by an external player, the DN is used to produce cryptographic receipts for validating synthetic transactions. Thus, only valid synthetic transactions produced by the validators on a BVN can be processed and validated.

6.6.5 Processing Synthetic Transactions

Figure 11 shows individual synthetic transactions between identities A-F that are being collected and sent as a bulk synthetic transaction from BVN-0 to BVN-1, BVN-2, and the Directory Network (DN). The yellow circles in rows A-F represent transactions, while those in the "Synthetic TX" row represent sets of transactions between identities A-F. The quadrangle that connects ADIs A-F, BVN-1, BVN-2, and the DN represents communication between all ADIs hosted on BVN-0 and all identities hosted on other BVNs or the DN. If the box representing BVN-1 were opened, for example, you might find identities G-J.

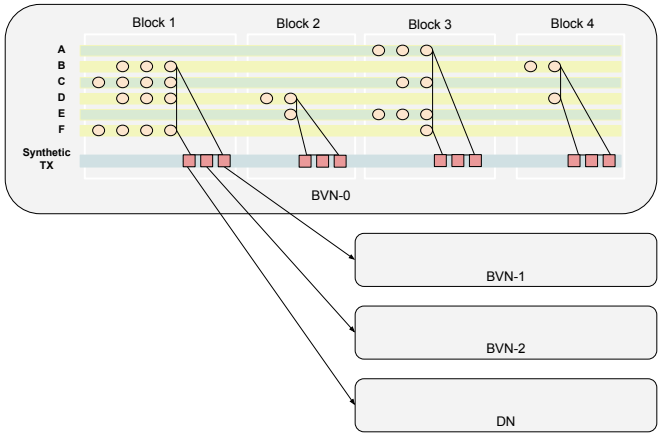

Fig. 11 Synthetic transactions.

When a transaction or set of transactions validates on the Accumulate network, if those transactions need to update accounts in other Root Identities, the protocol produces synthetic transactions that essentially export those updates to other BVNs. Those updates do not require state in BVN-0 to validate what they should do in BVN-1, BVN-2, or the DN. If BVN-0 is sending a transaction to BVN-1 and later on to BVN-2, BVN-0 can operate at full speed and interact with BVN-2 without having to wait on BVN-1 to accept the deposit.

A real-world analogy would be the process of settling a credit card transaction. When a customer makes a purchase with their credit card, the credit card validates the transaction at the point of sale, and the goods are released by the merchant. However, the merchant does not have access to the customer's money until the transaction settles. We will imagine that the credit card platform is hosted on BVN-0, and the merchant's bank account is hosted on BVN-1. The customer can use their credit card for other purchases before the merchant's bank accepts their payment because the customer's credit card platform has sent a synthetic transaction to the bank broadcasting a summary of its transactions that we refer to an account balance.

6.6.6 Summary

For most blockchains, the transaction is the settlement. This is possible because most blockchains store the entire state of their network on full nodes. While this simplifies the process of validation, it also limits scalability. Accumulate is partitioned by design, with each identity acting as an independent blockchain and every transaction on its chain being handled by a particular BVN. This revolutionary architecture allows the network to scale indefinitely through the addition of BVNs, but scalability is limited by increasingly inefficient communication as the network grows. Synthetic transactions provide a means of efficient communication between BVNs and add settlement to the blockchain.

7 Tokenomics Model

Accumulate uses a burn and mint equilibrium (BME) model for its native ACME token that trends deflationary with increasing network demand. A fixed percentage of ACME tokens are periodically minted from the unissued portion of its 500 million maximum supply and distributed to stakers and validators. Depending on transaction volume, a portion of the circulating supply will be burned to create Credits that users need to create Accumulate Digital Identifiers (ADIs) and write data to the blockchain. Credits are non-transferable tokens with a fixed cost and no market value that disincentivize hacking, bypass regulatory requirements, and allow enterprises to budget effectively. Burned ACME tokens are returned to the unissued pool to be reissued in future blocks. Network use incentivizes staking and takes ACME out of circulation while increasing use of the network over time will decrease block rewards and put upwards pressure on the price of ACME. Thus, protocol use rather than speculation will drive the token's value. The goal of increasing the value of ACME is to incentivize users to secure the network in the absence of a traditional fee model. In this report, we will examine the dual token model of the Accumulate protocol and compare its tokenomics and value proposition to that of other blockchains.

7.1 Deflationary Tokenomics Models

The incentive to hold a token creates demand, while the demand of a token relative to its supply will influence its market price[20]. A variety of strategies to incentivize holding and stimulate demand have been developed by blockchain protocols, the majority of which rely upon some variation of a deflationary tokenomics model. These models are briefly summarized below.

- Basic deflationary model: A fixed number of tokens are created. Limited supply is expected to create demand naturally. Examples include Bitcoin (BTC), Ripple (XRP), and Solana (SOL).
- Buy back and burn: Tokens are bought back from holders and burned. This permanently removes them from the supply. Binance Coin (BNB) burns 1% of its supply per quarter.
- Burn on transaction: The protocol's contract specifies a tax on transactions that burns and/or distributes the tax among its holders. Safemoon (SAFEMOON) does both.
- Net deflationary model: The max supply is uncapped, but the burn rate from taxes or buybacks exceeds the issuance rate. Curve (CRV) has become net deflationary at the time of writing.

Buy back and burn operations tend to be executed manually, which gives token issuers flexibility in the timing of buybacks and greater control over the market price. However, this often comes at the cost of decentralization. Protocols adopting this model may minimize centralization by involving a decentralized autonomous organization (DAO) in the decision-making process.

Burn on transaction models are most useful for controlling the inflation of uncapped tokens. Ethereum 2.0 recently added a deflationary mechanism with its London Hard Fork that involves burning its transaction fees. However, these models are entirely dependent on transaction volume. High fees may also encourage hoarding, which could lead to undesirable price swings and excessive speculation.

Tokens with a net deflationary model generally achieve equilibrium in their later stages since buybacks from early-stage companies carry greater financial risk, and transaction rates are not high enough to offset inflation. While this model encourages activity without hoarding, equilibrium is primarily driven by speculation and may be difficult to maintain throughout market cycles.

7.2 The Velocity Problem

The aforementioned tokenomics models are all susceptible to the "velocity problem", which can be loosely defined as the tendency of users in a frictionless market to immediately exchange their tokens for goods and services[21]. Velocity is expressed in the Equation of Exchange, which has been redefined below for cryptocurrencies and tokens:

$$M \times V = P \times Q \tag{1}$$

Where,
M = Market cap of the token
V = frequency in which a token changes hands in a given time period (i.e., velocity)
P = Average price of goods and services purchased within this time period
Q = number of goods and services purchased within this time period

Table 4 Model of ACME price given utility and speculation.

Year	TPS	Total Fees	Avg. % of Tokens burned through Fees			
			4.00%	1.00%	0.25%	0.06%
1	4	$2,399,087	$0.21	$0.85	$3	$14
2	8	$4,798,175	$0.39	$1.53	$6	$24
3	24	$14,394,524	$1.06	$4	$17	$67
4	72	$43,183,573	$3	$12	$47	$189
5	288	$172,734,291	$11	$45	$179	$717
6	1,440	$863,671,457	$54	$215	$858	$3,432
7	7,200	$4,318,357,287	$262	$1,038	$4,143	$16,565
8	14,400	$8,636,714,574	$508	$2,016	$8,049	$32,180

The left and right sides of the equation can be interpreted as the total price of tokens spent and the total price of items bought, respectively, within a given time period. High velocity will cause an asset to be devalued, while low velocity will result in difficulty liquidating the asset. High velocity is often encountered by blockchain platforms with a "medium of exchange" token that is required to access a product or service. While the demand for this product or service may be extremely high, this does not necessarily translate to an increase in the token's value. The solution is to create a token with more economic value to holders outside of its primary utility without incentivizing them to hoard it.

7.3 The Burn and Mint Equilibrium Model

After issuing its "activation block" of 150M ACME tokens and distributing an additional 50 million ACME in a token swap for Factom's FCT holders, 300 million ACME tokens out of its 500 million maximum supply will remain in the unissued pool. Every year, 16% of tokens in the unissued pool will be minted at intervals of approximately 1 month to compensate stakers and validators in the absence of a transaction fee.

ACME tokens are burned to create Credits, and Credits are used to pay for services. When ACME tokens are burned, they return to the unissued pool to be minted in future blocks. As network usage grows, the burn rate of ACME will increase, and fewer ACME tokens will remain in the circulating supply for that minting period. Since Accumulate uses a PoS model[22] to reward its stakers and validators, increased network use will incentivize the lockup of ACME tokens in staking pools and drive Accumulate towards a deflationary model[23].

Equilibrium is achieved due to the inverse relationship between Credits and ACME tokens, as illustrated in the following example. Assume a fixed cost of $0.001 per Credit per kilobyte of data and a minting rate of 1 million ACME tokens. A value of $1 per ACME token is supported with 1 billion transactions:

$$1 \text{ mil. } ACME \times \frac{\$1}{ACME} \times \frac{1 \text{ } credit}{\$0.001} \times \frac{1 \text{ } tx}{1 \text{ } credit} = 1 \text{ bil. } tx \qquad (2)$$

If network usage increases such that 1.5 million ACME tokens are burned each month, then the circulating supply will decrease, and the value of ACME will increase. As the value of ACME increases, the number of Credits issued per ACME token will go up to support the increased demand on the network. Conversely, if network usage decreases, the circulating supply will increase and exert downwards price pressure. Thus, the price of ACME should increase linearly with the usage of the network, which directly addresses the velocity problem.

7.4 Transition from Speculation to Utility

All utility tokens are initially driven by speculation. As network usage grows, speculation will decrease, and the value of the token will be driven by utility. The following table models the transition from speculation to utility over time with different burn rates.

Each color represents the projected price of ACME as a function of burn rate and time, where the dark green band is least likely, and the light green band is most likely based on estimated network use and speculation. The transition from speculation to utility is represented by the leftward movement of each color over time, where utility is expected to dominate as the protocol matures. While not included in the table, a burn rate of 0% represents zero network use, while a burn rate of 100% represents the conversion of all circulating ACME tokens into Credits. The community would likely vote to increase the maximum supply in the latter scenario.

The utility is represented by the number of transactions per second (TPS), which is expected to increase with the protocol's adoption. Note that TPS is a function of protocol use, not the protocol capacity. The current throughput of Accumulate is 70,000 TPS on the Testnet with a projected limit of several million TPS over the next several years as more Block Validator Networks (BVNs) are added to scale the network.

7.5 Value Proposition

The value proposition of Accumulate's BME model is realized for enterprises who 1) want a predictable cost model, 2) cannot legally hold cryptocurrency, and 3) need the flexibility to price their own services.

- Predictable costs: The price of Credits is tied to the USD, and the number of Credits required to write data to the blockchain is fixed per kilobyte of network usage. This allows enterprise users to budget their data use long-term without worrying about market conditions.
- Legal compliance: Some users in both the public and private sectors cannot legally hold cryptocurrency. Since Credits are non-tradeable and non-transferable, they are treated as a product rather than a security. Credits can be purchased from a third party.
- Flexible pricing: The Work Token Model pioneered by Augur is the only other deflationary model that addresses the velocity problem. However, it only works if service providers are offering a pure commodity. The BME model allows service providers to set their own prices and compete with other businesses on marketing, customer service, or other variables.

8 Use Cases

This chapter presents example use cases regarding how Accumulate would serve humanity and bring more efficiency to everyday problems.

8.1 Executing Multi-Party Contracts

Thanks to its unique identity concept and key hierarchies, Accumulate is very good at executing multi-party contracts.

8.1.1 Managing Beneficiaries

Some investments, securities, insurances, and assets may have beneficiaries other than the holder or primary owner of that asset, such as death beneficiaries of a health insurance. For example, the primary owner may decide where their assets are going to be distributed among several people if they pass away. The signers on the will may include family, lawyers, accountants, or business partners. While the person is alive, the signers may come and go, the owner could change lawyers. Such processes can easily be practiced with Accumulate as the owner of the asset can manage identities and accounts with lower priority levels.

In case people lose their keys associated with a protected asset (e.g., a blockchain wallet), having implemented Accumulate may help again. With Accumulate's hierarchy of keys, the owner can backup and recover the keys held in a cold storage from a Key Page at a higher priority level. Those higher priority keys could also add or remove signatures as people enter and leave the related contract. Using a manager Key Book, which basically allows a transaction to go through only when the manager signs a transaction, the owner could assign a legal entity control of the final execution of the will.

Whenever an action is to be taken by one party (e.g., the owner or other signers), the protocol loads the Key Page mentioned in the transaction from the Key Book associated with that party. The Key Page stores the keys that are authorized to sign, particularly for the intended transaction. If the transaction is attempted to be signed by a key, not in that Key Page, the signature is rejected. Another feature of the Key Page is that it states how many keys are required at a minimum to co-sign the prompted transaction. A hypothetical Key Page may require 2 keys out of 3 total keys to sign the transaction before it becomes valid so that a quorum can be enforced for managing the assets. This relates to managing a jointly owned asset (with 3 shareholders), like a joint life insurance. If the Key Page requires all three keys to sign the contract, then a consensus will be required to take action.

Various transactions may point to different Key Pages and may require a different number of signers. In the insurance example, joint insurance owners may agree that a request for cancellation escalated by one of three owners will be enough to void the policy, but all three owners must sign to include another beneficiary (who is not an owner). In the meantime, the new beneficiary may also be requested to sign the contract to state their acknowledgment of the benefits. The beneficiary's signature would have a lower priority so that the owners can change the beneficiary at any point upon consensus. All these are easily possible with Accumulate.

8.2 Modeling Corporate Structures

With its ability to manage multi-party contracts and diverse groups of digital identities, the Accumulate protocol includes all the systems required to replicate entire corporate structures.

One of the first adoptions of blockchain technology in enterprise applications has been the use of smart contracts to automate the exchange of services and funds[24]. However, this groundbreaking innovation has been plagued by a series of large, unauthorized funds transfers that occurred due to external attacks or malfunctioning smart contracts. These incidents highlight gaps in a traditional smart contract's ability to protect an organization's assets and information while also demonstrating the risk of gatekeeping solely through a single layer of code or encryption.

Accumulate's infrastructure includes a level of redundancy and reliability that would be invaluable to enterprise users. For example, delegated networks of validators can have an active role in approving transactions. This allows companies to assign multiple validators to run concurrent algorithms verifying that the conditions of an agreement are met. Synthetic Transactions can be a valuable gatekeeping tool between two entities involved in a transaction, which prevents attackers from directly accessing any entity's funds. Transactions could also be automated via integration of URL-based addresses with external applications.

As organizations incorporate blockchain-based systems into a variety of their processes, it is important that the underlying blockchain protocols can scale as their user base grows. Not only is Accumulate able to scale its own infrastructure inexpensively, but it is also able to help companies manage changes within the dynamics of their organizations. Employee turnover (and the resulting loss of organizational knowledge), restructuring of leadership, merging or termination of departments, and acquisition or loss of customers and vendors can all be modeled and managed via Accumulate's identity and key hierarchies. The design of the Key Book allows permissions to be easily added, revoked, and upgraded while URL-based indexing ensures a clear, easily understandable organization of entities within a company.

8.3 Token Gateways

Wrapped tokens have been a key innovation in the blockchain space[25]; however, many protocols with this capability are still not able to produce and manage the data required to maintain a traded asset over time. One major challenge is managing the validation of the flow of native tokens into wrapped tokens, which is difficult because the set of responsible validators is constantly changing. With a dynamic pool of validators, complications can arise when attempting to maintain a reliable record of users' Know-Your-Customer (KYC) and Anti-Money Laundering (AML) details.

This is especially true when cross-chain KYC is necessary. For example, when a wrapped token is used to buy another kind of token, the acceptor blockchain must get the KYC information of the individual attempting the exchange in order to verify that they rightfully own the wrapped token. However, this verification step is based on the KYC process of another protocol; if the validator

that performed the original KYC for the wrapped token has since dropped out of the protocol, there is a risk that the tokens would be invalid, with no clear proof that they were obtained legally. This kind of multi-chain authorization is beyond the capacity of most smart contracts to handle in a cost-effective manner.

Accumulate can help other protocols address this problem by documenting the gateway processes between blockchains. Accumulate's infrastructure is designed to be able to track changes over time, making it the ideal solution for preserving the audit trail from a token's creation onward. Scratch Accounts can be used to preserve important details related to KYC at low costs off-chain while maintaining a clear, ordered record of essential data points. The reliability of the KYC process can even be improved by assigning multiple parties to the verification process via Delegated Transactions. In addition, approval thresholds can be assigned (ex: approval by $2/3$ of validators) to ensure that the loss of a single validator would not slow down the approval process.

8.4 Internet of Things

8.4.1 The Solutions Provided by Accumulate

Despite the benefits of blockchain technology, its widespread application in IoT devices is still hindered by high costs and transaction fees, limited data storage, and security concerns when data comes from a single source. Transaction fees can become prohibitively expensive when the number of IoT devices is large or the frequency of data generation is high. Long-term storage of all the data generated by an IoT network can still overwhelm a distributed system, limiting its growth. A single bad actor with the permission to access the data can add or delete entries after data has been collected, undermining trust in the network.

Accumulate solves these issues by assigning an identity to each sensor, hashing and pruning the data, and allowing companies to manage their keys over time. On Accumulate, each sensor is assigned a digital identity in the form of an Accumulate Digital Identifier (ADI), which prevents spoofing (e.g., the malicious use of duplicate sensors) and allows the user to monitor and audit data from an individual sensor in the network.

Sensor data is hashed in a Merkle Tree, which creates efficient cryptographic proofs of data validity, while the Patricia Trie creates efficient cryptographic proofs of the current state of the system. This data structure allows an entry to be removed from the Patricia Trie while still maintaining proof that the event existed in the hashes of the Merkle Tree. From a practical point of view, there will probably be some roll-up data that an organization or utility cares about, but not for every device in the network. For example, voltage levels across every device in a power grid may be deleted after 6 months except for devices in particular locations. With Accumulate, you can prune the underlying data, extract the roll-up, delete the unnecessary data, but prove that the data the roll-up depends on existed.

Accumulate also provides a system of robust key management. As explained in Chapter 4, a hierarchy of keys allows a company to assign different levels of security to the manager of an ADI depending on the role of a key holder and the perceived value of the data. The keys to an ADI can be managed just like signers on an account, so new ADIs do not have to be issued with the departure or promotion of an employee or when a company shifts responsibilities over time. In addition, data on the blockchain generated by an ADI can only be validated by someone with key access. An audit trail of their activity establishes confidence in the data across the IoT network.

8.4.2 ESG Scores as a Practical Application

One IoT use case for the Accumulate network is the generation of environmental data to provide an environmental, social, and governance (ESG) score[26] for a building to quantify its environmental friendliness. Temperature, humidity, and sound level sensors may be installed in the building to continuously monitor the parameters that comprise an ESG score. Business partnerships or investments may depend on the ESG score, which is important to a growing class of sustainable investors who seek to maximize the good their investment does for society and the environment. If an investor uses IoT data as part of their process of valuing the property, they need to make sure that the data was not fabricated by a property manager.

Accumulate provides a way to prove that the data is valid by assigning an ADI to each sensor and maintaining an audit trail. Proving that data came from a particular sensor on a particular floor in the building may also alert the property manager to the problem that negatively impacts their ESG score. For example, a poorly insulated property on the ground floor that is generating an unexpected amount of heat. Meanwhile, pruning data after an evaluation but maintaining proof that the data existed may prove the innocence of the property manager if they are later audited.

8.4.3 Industrial IoT

Industrial IoT (IIoT) relates to the use of interconnected and internet-connected sensor devices throughout an industrial plant (also including facilities like farms, ports, airports, etc., in a broader sense). The sensors may continuously monitor production sites, equipment, computerized systems, air and soil parameters (e.g., temperature, humidity, wind, etc.), and the final products or yields. As a result, there may be large amounts of data produced by different departments, units, and entities related to different variables and parameters. Managing, categorizing, and classifying such data is a heavy-duty exercise for businesses. However, not all data collected by an IIoT network needs to be recorded. For example, a food manufacturer may only want to be informed of an extraordinary event that affects the safety of their product. There is also the risk of infiltration and data breaches which may cause undetectable manipulation or loss of crucial data.

How a generic blockchain integration would contribute to this system is quite clear: with a blockchain on one end of the data flow, the data would be kept safer[27]. However, most blockchain systems are very slow (e.g., allowing only several to a few hundred TPS), and the unorganized data remains unorganized, making it even harder to pull when necessary. This is where Accumulate can contribute to IIoT much better than any other blockchain system.

Accumulate allows each department (e.g., production, qual-

ity assurance), unit (e.g., shifts, teams), device (e.g., conveyors, robotic arms), area (e.g., ateliers, hangars), or any physical sector of the facility (e.g., corn farms of a hybrid plantation) to have their own unique human-readable (and internet-readable) addresses within their hierarchical accounts provided by the ADI concept. Therefore, it is possible to categorize and classify every single data packet at the moment that it is produced within its source (e.g., sensor device). Moreover, Accumulate allows removing parts of data that do not need to be kept permanently (such as nominal temperature records) after a preset period, preventing the database from bloating. Scratch Accounts can also be used to store draft measurements that require an authorized staff to confirm (or reject).

Accumulate also allows the creation of complex and flexible authorization schemes, which is beneficial for managing and automating consensus among multiple parties. Imagine that a finished product is subject to a series of quality control tests, and it will be ready to ship only if all the tests are passed. Also imagine that some tests are done by different people or devices belonging to different departments who may change over time. Accumulate can automate consensus building through its Managed Accounts and allow the manufacturer to reassign keys without disrupting their manufacturing process. Therefore, Accumulate can increase the reliability of the production processes, provide better data security, and reduce the operational costs in many IIoT scenarios.

8.5 Sponsored DAO

8.5.1 Origin of Decentralized Autonomous Networks

Satoshi Nakamoto believed that repeated violations of public trust by centralized financial institutions necessitated the creation of a trust-less and decentralized peer-to-peer digital currency based on blockchain technology[28]. By early 2009, during a global financial crisis, many others had echoed Satoshi's sentiment as public faith in financial institutions was at its nadir. The distributed yet collaborative community of blockchain enthusiasts that began to materialize was reflected in the trust-less and decentralized nature of blockchain technology itself. As individuals, however, they lacked the power to challenge traditional financial institutions.

After smart contracts were developed by Ethereum and Proof of Stake was successfully implemented by Peercoin[29], the organization of blockchain enthusiasts into large, distributed groups with shared financial goals became feasible. The bylaws of a decentralized organization could be written into transparent, verifiable, and publicly auditable smart contracts. Members could stake their tokens in exchange for the voting power to make critical decisions about the management of their organization, including its partnerships, technical upgrades, and treasury allocations. Even the bylaws could be changed if a consensus was reached. This collectively governed organization of stakeholders, whose operations are wholly owned and managed by its members in the absence of a central authority, is known as a decentralized autonomous organization (DAO).

8.5.2 Growth and Legal Recognition of DAOs

While some consider Bitcoin to be the first DAO due to the governance mechanism of its mining network, Dash is the first modern DAO[30], launched in 2014, that provided a governance mechanism for its stakeholders and allowed them to vote on proposals that decided the future of the organization. BitShares was launched that same year as an e-commerce platform that connected merchants to customers in the absence of a central authority[31].

Unfortunately, the most widely-recognized DAO was an Ethereum-based organization simply called The DAO that raised $150 million from investors in 2016 before suffering from an exploit in its code that drained one-third of its treasury and ended the project a mere 5 months after launch. The funds were returned, but only at the cost of a controversial hard-fork of Ethereum[32], which resulted in the creation of Ethereum Classic. While the attack on the DAO sent shock waves through the blockchain community and undermined the legitimacy of DAOs for a time, development has continued, and DAOs are now considered a legal entity in the state of Wyoming, USA[33], with more states likely to follow.

Continued innovation of DAOs and their growing recognition as legitimate financial institutions has led to their adoption by nearly every financial sector. For example, MakerDAO provides collateral-backed loans against cryptocurrency and real-world assets, JennyDAO issues fractional shares of NFTs, Nexus Mutual offers insurance, Raid Guild contributes to the gig economy, and Endaoment pools charitable contributions that are distributed by their stakeholders.

8.5.3 Advantages and Disadvantages of DAOs

The most compelling advantage of DAOs is their elimination of the principle-agent dilemma[34]. In traditional finance, there is an inherent conflict in priorities between the principle, who invests in a fund, and the agent, who manages the fund. For example, a trader may take on excessive risk at the expense of their investors by investing in volatile assets or using highly leveraged positions with the goal of securing a large year-end bonus. Here, the agent's goal of a bonus is in potential conflict with the principal's goal of profiting from their investment. In a DAO, however, each member contributes to the purchasing and management of an asset. Therefore, the stakeholders of a DAO are both the agent and the principle whose priorities are aligned.

DAOs also benefit a stakeholder through greater efficiency in business operations, transparency in rules and operations, and autonomy in how their assets are managed. As explained below:

- Efficiency: The lack of hierarchical structures inherent in traditional finance eliminates bureaucratic inefficiencies. In a DAO, decisions are reached by community consensus after a quorum is reached.
- Transparency: The rules of a DAO are embedded in smart contracts that can be publicly audited. Contract details and transactions are permanently recorded on the blockchain.
- Autonomy: A stakeholder has complete ownership of their investments. They can also participate in a project from inception to exit at their own discretion without facing complex regulatory hurdles.

The positive attributes of a DAO can, unfortunately, work against the organization when things go wrong. For example, transparent code gives hackers the opportunity to simulate an attack on a virtual machine before an attack is launched on the actual network. In the case of The DAO, a vulnerability in the code was publicly reported, and the community was called to a vote after a patch had been developed. However, the hack occurred before the voting could be completed, resulting in a loss of 11.5 million Ether[35]. The decentralization of authority means that no one is held accountable for monetary losses or legal action taken against the DAO. The involvement of non-experts in the decision-making process leaves the DAO particularly vulnerable to lawsuits, which may be filed across multiple jurisdictions because of the decentralized nature of the DAO. For example, a DAO may be sued if a stakeholder contributes to a private equity fund without being an accredited investor.

8.5.4 Sponsored DAO

A sponsored DAO (SDAO) is a theoretical organization that is collectively owned by its members but managed by an accredited institution (e.g., a bank or investment firm). The governing institution is responsible for specifying the lending criteria for borrowers, maintaining KYC/AML records for all stakeholders, and providing backstop liquidity for tokenized assets that are locked into Digital Special Purpose Vehicles (DSPVs) to securitize these assets and create liquidity pools for investors.

DSPVs can be multi-tranche, giving investors a choice between a lower risk investment with stable returns and a higher risk investment with variable returns. This model is being implemented by Centrifuge Inc., which recently integrated its Tinlake marketplace with MakerDAO to provide loans that are backed by real-world assets[36]. Tokenized assets may also be integrated with financial oracles that aggregate real-time performance and valuation data. Sponsoring organizations with access to advanced software and financial experts who can interpret this data will help stakeholders make informed decisions about their investments.

8.5.5 How Accumulate Enables the Creation of an SDAO

Financial oracles generate a large volume of data that must be synced to the tokenized asset and credentialed by institutions with legal authority. The Accumulate protocol can process transactions with low cost, high throughput, and minimal storage requirements due to its efficient data structure. Accumulate can also integrate with traditional tech stacks, meaning that financial institutions can use the Accumulate network to manage assets without having to adapt their technology.

The Accumulate protocol's hierarchical identity and key structure enable asset management with greater flexibility and less granularity than traditional DAOs. Attestations given to stakeholders and managed by sponsoring institutions allow stakeholders to invest in different assets depending on their status (e.g., accredited investor). Complex operations like subdividing and selling a mortgaged property are possible on the Accumulate network due to its powerful identity capabilities that allow multiple key holders with different priority levels to create signature groups that can be managed over time.

To understand how this identity framework may be applied in the real world, imagine a sponsoring bank with a senior vice president (VP) in charge of agents who manage investments in the SDAO. The bank may provide the VP with an attestation that authorizes the VP to issue their own attestations to agents who, in turn, can provide attestations to individual stakeholders. Each entity has an identity on the Accumulate network that is capable of assigning and revoking authority to those who are lower on the identity hierarchy. The bank can revoke the VP's authority just as the VP can revoke an agent's authority, and an agent can remove a stakeholder's authority if they lose accredited investor status.

Hierarchical key sets in Accumulate are also useful for managing high-value tokens. Consider an NFT that represents a bundle of real-estate valued at $100 million. In traditional blockchains, the loss or theft of a key can result in an irreversible loss of the NFT. Multi-signature authorization can provide an additional layer of security with flexibility in key management. For example, adding or selling a piece of real estate might take days given time locks in the multi-signature contract and delays in acquiring the threshold number of signatures. Accumulate allows administrative keys to be maintained in cold storage while enabling routine operations with lower priority keys.

8.5.6 Summary

Decentralized autonomous organizations are the natural byproduct of a collective frustration with financial institutions by a decentralized collective of blockchain enthusiasts. Over time, however, DAOs began to associate with these same institutions by outsourcing certain operations, such as KYC/AML, to qualified organizations. With the implementation of an SDAO, stakeholders will be able to minimize their legal exposure, make better-informed decisions, and participate in more complex financial activities while maintaining a high degree of autonomy. This type of organization is uniquely possible on the Accumulate network due to its powerful identity and key management capabilities.

9 Conclusions

Accumulate is a multi-chain protocol with cross-chain support that provides secure and flexible identity and key management solutions. Human-readable ADIs that adhere to W3C standards facilitate the integration of blockchain technology with servers and web-based applications. A hierarchical identity architecture with directory-like indexing makes it possible to categorize and query large datasets on-chain and replicate virtually any organizational structure. A hierarchical key architecture with prioritized Key Pages allows users to create dynamic and arbitrarily complex authorization schemes that mirror but also improve upon many financial operations. In combination, these features allow Accumulate to turn traditional applications into blockchain applications that deliver Web 3.0 technology at the level of convenience expected of Web 2.0 implementations. The engine behind this technology is Accumulate's chain-of-chains architecture that partitions the network and provides linear scalability with unbounded TPS. Scalability is matched by the price stability provided by Accumulate's dual-token model that allows for long-term budgeting independent of network growth.

10 Acronyms and Glossary

10.1 Acronyms

ADI Accumulate digital identifier 1 ff., 5–11, 14 f., 19 f., 23, 26 f.

AML Anti-money laundering 22, 25

BME Burn and mint equilibrium . . . . . 20 f.

BPT Binary Patricia Trie . . . . . 4, 13 f., 16

BVN Block validator network . . . 2 ff., 8, 10, 12 ff., 16–21, 26 f.

BVNN Block validator network node 26

DAO Decentralized autonomous organization . . . . . 20, 24 ff.

DDII Decentralized digital identity and identifier 2

DN Directory network . . . . . 4, 8, 12, 16 f., 19, 26 f.

DPoS Delegated proof of stake 1

DSN Data server network 19

IIoT Industrial Internet of things . . . . . 23 f.

IoT Internet of things 2, 7, 15, 23

KYC Know your customer . . . . . 22 f., 25

NFT Non-fungible token 7, 25, 27

PoS Proof of stake . . . . . 17 f., 21

PoW Proof of work . . . . . 17 f.

SDAO Sponsored decentralized autonomous organization . . 25

SMT Stateful Merkle Tree 12, 16

TPS Transactions per second 21, 23

10.2 Glossary

Accumulate: The Accumulate network is a collection of independent chains. Each chain is managed by a hierarchy of identities known as Accumulate Digital Identifiers (ADIs), and each identity possesses a hierarchy of keys, which allows it to participate in the execution of transactions.

ACME: "ACME" is the symbol of a cryptographic token used in the Accumulate protocol. The ACME token, which is a traditional spendable token, like ETH and BTC and it is issued by the protocol to reward those providing services to the protocol. For example, Validators will be awarded staking fees in ACME.

Accumulate Blocks: The Accumulate protocol defines minor blocks and major blocks. Minor blocks are once per second synchronization points of the Merkle trees. Major blocks are twice per day.

Accumulate Digital Identifier (ADI): An ADI (also called identity) is a human-readable URL that can represent users, institutions, devices, etc. An Identity may have accounts, which hold its assets, whether those are tokens, data, or keys. An identity can be represented as: "acc://Bob". Identities provide structure on the Accumulate Blockchain and allow smart contracts, consensus building, validator networks, and enterprise-level management of digital assets.

ADI Data Account: An ADI Data Account is one of the types of accounts an ADI can control. An ADI data account holds data. An ADI data account can be represented as "acc://Bob/Data". To write data, one would specify this URL.

ADI Token Account: An ADI Token Account is one of the types of accounts an ADI can control. An ADI token account holds tokens and can be represented as "acc://Bob/Tokens/ACME". To send or receive tokens, one would specify this URL.

ADI Scratch Accounts: Scratch Accounts are identical to data accounts. However, after 2-3 weeks, its chain is compressed, and proof of the transactions is created. A proof is a logical argument used to show the truth of the operations made on the chain. This argument or proof contains less data than the Scratch Account it represents. This allows the blockchain not to be burdened with moving substantial amounts of data over the network.

Block Validator Network (BVN) / Block Validator Network Node (BVNN): A BVN executes transactions against records. At the end of each block, the BVN collects Merkle DAG roots (anchors) from the Main Chain of accounts modified by one or more transactions in the block, appending them to the BVN's root chain. The anchor from the root chain is sent to the DN. A BVNN is a node with a BVN.

Chain Validators/Executors: Chain validators can be thought of as transaction executors. The actual code for executors is per-transaction (type), not per-chain. As an example, the creation of an identity would have a specified executor. The validation of the signature, for instance, is handled by the overall validator/executor.

Decentralized Autonomous Organization (DAO): A DAO is an organization represented by rules encoded as a transparent computer program, administered by the organization members, and not influenced by a central government. A DAO's financial transaction record and program rules are maintained on a blockchain.

Credits: Credits are a non-transferable form of payment on the Accumulate Network. Credits are created by converting ACME tokens into credits. Once the user converts ACME tokens to purchase Credits, Credits are then used for actions on the network such as the creation of an ADI, ADI Token Account, or updating Keys in a Key Page.

Directory Network (DN): The DN executes transactions against certain system account, such as the ACME token issuer. The DN also receives root chain anchors sent from BVNs, appending them to the DN's BVN anchor chain. At the end of each block, the DN collects Merkle DAG roots (anchors) from the Main Chain of records modified by one or more transactions in the block (including the DN's BVN anchor chain), appending them to the DN's root chain. The anchor from the root chain is sent to all of the BVNs.

Directory Network Node (DNN): A DNN is a node within a DN.

Keys: Keys are defined as the hash of the public key for a signature.

Key Book: Key Books store the Hierarchical Key Management Structure of an ADI using Key Pages and Keys within those pages. A Key Book is an ordered set of Key Pages by priority where any Key Page can modify itself or a Page of lower priority. Modifications include adding, updating, or removing keys. An account specifies what Key Book applies to any transaction within that account.

Key Page: A Key Page defines the set of Keys required to validate a transaction. A Key Page specifies one or more Keys possible and how many such Keys are required to validate a transaction. Key Pages store credits. A Key Page can be represented as so: acc://Bob/KeyPage

Layer-1 Anchoring: An anchor is a root hash created from the latest Directory Network major block, which is then anchored into a Layer-1 blockchain such as Bitcoin.

Lite Data Chain: A lite data chain is a chain that anyone can write data to, as opposed to ADI Data Accounts which requires specified keys.

Lite Token Account: A lite token account is an unstructured account that is similar to an address such as Bitcoin, a string of non-human readable characters. They are represented as "acc://<keyHash><checkSum>/<tokenUrl>". Lite Token Accounts are not associated with an identity and are not acknowledged by the Accumulate network until they have ACME tokens.

Major Blocks: Every 12 hours, the transactions executed in the last 12 hours (since the previous major block) are collected into a new major block. For each record modified by a transaction in the block, a Merkle DAG root (anchor) is taken from the Transaction Chain of the account and appended to the DN or BVN's major root chain.

Managed Transaction: When creating an ADI account, you can specify an additional Key Book within the same ADI that manages a Key Book. A manager can approve or reject a transaction submitted by an ADI. If a transaction is rejected, a manager can suggest a new transaction on the ADI's Signature chain.

Minor Blocks: All transactions executed in the last second are collected into a new minor block. At the end of each minor block, for each record that was modified by a transaction within the block, a Merkle DAG root (anchor) is taken from the Main Chain of the record and appended to the DN or BVN's minor root chain.

Non-fungible Token (NFT): An NFT is a non-interchangeable unit of data stored on a blockchain. Types of NFT data units may include digital files such as photos, videos, and audio. Since each token is uniquely identifiable, NFTs differ from cryptocurrencies.

Signature Chains: A Signature chain tracks transactions that have not been promoted to the Main Chain. The data that is produced in the Signature chain is pruned approximately every 2 weeks.

Scratch Space: Accumulate provides scratch space on the blockchain that can be used by parties to come to a consensus but whose data availability is not retained by Accumulate forever. Scratch space allows processes to provide cryptographic proof of validation and process transactions without overburdening the blockchain.

sub-ADI: An ADI can contain another ADI. We refer to this as a sub-ADI.

Synthetic Transactions: Synthetic transactions are generated by the protocol to communicate debits and credits between BVNs. Synthetic transactions are any transactions that are generated by the protocol. These include transactions that deposit tokens into other token accounts and refund amounts for failed transactions.

Token Issuance: In Accumulate, users can create custom tokens that can be used within the Accumulate protocol.

11 References

1 P. Snow, B. Deery, J. Lu, D. Johnston and P. Kirby, *Factom: Business Processes Secured by Immutable Audit Trails on the Blockchain*, 2018, https://assets.website-files.com/5bca6108bae718b9ad49a5f9/5bd02670d8a1981ea62cb11f_Factom_Whitepaper_v1.2.pdf.

2 R. C. Merkle, *Method of providing digital signatures*, 1979, https://patents.google.com/patent/US4309569.

3 M. Scottnov, *Gates Foundation Grant Boosts Factom's Blockchain-Based Medical Record Development - Bitcoin Magazine: Bitcoin News, Articles, Charts, and Guides*, 2016, https://bitcoinmagazine.com/business/gates-foundation-grant-boosts-factom-s-blockchain-based-medical-record-development-1479492383.

4 *ACCUMULATE Protocol Litepaper v1.0: A Universal Layer 2 Blockchain for DApps and DeFi*, 2021, https://accumulatenetwork.io/Accumulate-Protocol-Litepaper-V1.0.pdf.

5 G. D. Monte, D. Pennino and M. Pizzonia, Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems, New York, NY, USA, 2020, p. 71–76.

6 G. Wood, *Polkadot: Vision for a Heterogeneous Multi-Chain Framework (Draft I)*, 2016, https://polkadot.network/PolkaDotPaper.pdf.

7 A. W. Appel, *ACM Transactions on Programming Languages and Systems*, 2015, **37**, 1–31.

8 S. M. H. Bamakan, A. Motavali and A. Babaei Bondarti, *Expert Systems with Applications*, 2020, **154**, 113385.

9 T. Chen, X. Li, X. Luo and X. Zhang, 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 442–446.

10 Y. Xiao, P. Zhang and Y. Liu, *IEEE Transactions on Information Forensics and Security*, 2021, **16**, 1782–1794.

11 S. Dhumwad, M. Sukhadeve, C. Naik, M. K.N. and S. Prabhu, 2017 23RD Annual International Conference in Advanced Computing and Communications (ADCOM), 2017, pp. 40–43.

12 M. Jung, M. Shishibori, Y. Tanaka and J. ichi Aoe, *Information Processing & Management*, 2002, **38**, 221–236.

13 W. Szpankowski, *J. ACM*, 1990, **37**, 691–711.

14 E. Palm, O. Schelén and U. Bodin, 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), 2018, pp. 31–40.

15 W. Wang, X. Li and H. Zhao, *The Computer Journal*, 2021.

16 H. T. M. Gamage, H. D. Weerasinghe and N. G. J. Dias, *SN Computer Science*, 2020, **1**, year.

17 L. M. Bach, B. Mihaljevic and M. Zagar, 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 1545–1550.

18 Q. Zhou, H. Huang, Z. Zheng and J. Bian, *IEEE Access*, 2020, **8**, 16440–16455.

19 K. Michelson, *Synthetic Transactions: Technical Guide*, 2021, https://accumulatenetwork.io/2021/11/synthetic-transactions-technical-guide/.

20 P. Freni, E. Ferro and R. Moncada, 2020 IEEE Symposium on Computers and Communications (ISCC), 2020, pp. 1–6.

21 R. Holden and A. Malani, *The ICO Paradox: Transactions Costs, Token Velocity, and Token Value*, National Bureau of Economic Research, 2019.

22 C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen and E. Dutkiewicz, *IEEE Access*, 2019, **7**, 85727–85745.

23 T. H. Austin, P. Merrill and J. Rietz, *Business Information Systems Workshops*, Springer International Publishing, 2020, pp. 73–85.

24 P. A. Ryan, *Technology Innovation Management Review*, 2017, **7**, 10–17.

25 G. Caldarelli, *Information*, 2022, **13**, year.

26 G.-Y. Jang, H.-G. Kang, J.-Y. Lee and K. Bae, *Sustainability*, 2020, **12**, year.

27 G. D. Hakan Altas, Umut Can Cabuk, *TURK. J. OF ELECTR. ENG. COMPUT. SCI.*, 2021.

28 S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2009, http://www.bitcoin.org/bitcoin.pdf.

29 W. Zhao, S. Yang, X. Luo and J. Zhou, 2021 The 3rd International Conference on Blockchain Technology, New York, NY, USA, 2021, p. 129–134.

30 I. Chistiakov and Y. Yanovich, author, New York, NY, USA, 2020, p. 67–72.

31 F. Schuh and D. Larimer, *BitShares 2.0: General Overview*, 2017, https://cryptorating.eu/whitepapers/BitShares/bitshares-general.pdf.

32 V. Buterin, *Hard Fork Completed | Ethereum Foundation Blog*, 2016, https://blog.ethereum.org/2016/07/20/hard-fork-completed/.

33 N. Ladani, *DAOs Are Taking Over, With New Wyoming Law*, 2021, https://finance.yahoo.com/news/daos-taking-over-wyoming-law-194516224.html.

34 R. Beck, , C. Müller-Bloch, J. L. King and and, *Journal of the Association for Information Systems*, 2018, 1020–1034.

35 S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko and Y. Alexandrov, Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, New York, NY, USA, 2018, p. 9–16.

36 R. Senner and M. Senner, *Stablecoins' quest for money: who is afraid of credit?*, 2021, https://doi.org/10.2139/ssrn.3940320.