A peer-to-peer electronic trade system

the Zenotta whitepaper

Zenotta AG

April 28, 2022

Authors:

Roelou Barry
Andrew Kessler
Andreas Furrer
Byron Houwens
Alexander Hobbs
Richard De Moliner
Miles Timpe
Holly Hoch

Zenotta AG
Baarerstrasse 57
CH-6300 Zug
info@zenotta.com
www.zenotta.com

Contents

1 Introduction

Bitcoin gave us cash. Digital, machine-readable and executable, distributed, global cash. While on a social level, Bitcoin has morphed into something more akin to digital gold, the fundamental underpinnings of the technology gave the world practical digital money for the first time. Many developments prior to the Bitcoin protocol were necessary in order for Satoshi Nakamoto to make their peer-to-peer electronic cash system a reality, standing on the shoulders of protocols such as Digicash (Chaum & van Antwerpen, 1990), pricing via processing (Dwork & Naor, 1992), Bitgold (Szabo, 2005) and Hashcash (Back, 2002).

Digital systems and the internet allow the world to be connected like never before. In truth, they allow for a digital world to exist, alongside the real, physical world. For the moment at least (barring the development of self-aware artificial intelligence that may be vastly different from our human consciousness) both of these worlds are inhabited by people. In the physical world, we have a physical presence, and in the digital world, a digital presence; in both cases an 'avatar' with the same wants and desires, the same types of interactions (albeit in the digital world, somewhat restricted) and the same process of thought. In order for people to be truly represented in the digital world, our real-world society has to be reproduced in a digital form.

One quintessential aspect of society in particular is the ability to assign value; to goods, to services, and to assets generally. In the physical world, value is easily evident, albeit often subjective. It just makes sense. We understand, as a society, and as human beings, how physical objects can possess value, and through the related concept of ownership, how this value can be transferred between individuals and groups. This is not the case in the digital world. In order to see why, we must turn to economics. Economic value is largely a function of supply and demand; however, in the digital world, the fundamental mechanics of supply and demand are not preserved because digital goods can be created, duplicated, and manipulated at virtually zero cost. Fundamentally, there are two properties that a thing must possess in order to have economic value:

1. **Rivalry** *(consumption by one party prevents simultaneous consumption by others)*
2. **Excludability** *(consumption of a thing can be prevented for certain parties)*

When we consider legal systems, the discrepancies are even clearer, and this is due to the underlying concept of ownership. Only rival and excludable assets are ownable, while non-rival and non-excludable assets, such as air or water, are public goods available to everyone. Air is everywhere, accessible to all, and in effectively infinite supply, but a tank of compressed air is finite, restricted, containerized, and can be physically transferred in a way that removes it from the ownership of the transferring party. Legal systems must be able to define and protect rights and ownership for digital assets, which they cannot do without a form of ownership that creates such a containerizing concept on the machine level.

Even once you are able to carry the concept of value into the digital world, there is still a crucial part missing: trade. For trade you need assets, and you need money (or some form of a medium of exchange). Blockchains have enabled digital money, through the magic of a ledger-based accounting system combined with distributed trustless consensus. With this approach one can solve the double spend problem[1] (Chohan & Chohan, 2021), and the longer-standing Byzantine Generals' Problem[2] (Lamport et al., 1982), and ensure that money can be sent peer-to-peer with no requirement to trust a third party. The other part of the economic relationship, however, is missing. Transactions are two-way; without an asset received in exchange for payment, the ledger merely tracks a gifting economy rather than a trading economy. This means that any actual trade of money-for-asset must occur on a layer above the ledger, whether through a third-party exchange or through a smart contract ran on a virtual machine.

Ensuring two-way trade sets the stage for a real, ledger-based economy within the digital world and in compliance with legal contractual thinking. The same magic that enables blockchains to manifest digital money allows digital systems to realise digital trade, when the ledger works both ways. Combined with ownership, value can then be transferred between digital identities in a way that preserves the fundamental tenets of distributed, decentralized consensus and cryptographic security.

1.1 Digital economies

The realisation of this goal is particularly important when one considers the latest evolution of the internet. The initial development of the web revolved around the dissemination of information. This first incarnation (web1) largely consisted of one-way access to a repository of information for the average user (**read**). Later this developed into a two-way platform for sharing and creating content (**read-write**).

Since then, the internet has evolved into a social realm where individuals interact, share personal data, and engage in economic activity. This goes far beyond mere social media, and is a fundamental shift towards a digital societal identity, with all of the accompanying societal value and structure, such as rights and privacy.

While many definitions of web3 differ, it is probably described best in this ontology as **read-write-own**. This is not only two-way but also peer-to-peer, with individuals conducting business and personal relationships and transacting, just like in the physical world.

An excellent example of this is in the gaming world. In-game economies built around blockchains have begun to emerge, employing the technology of non-fungible tokens (NFTs) to attempt to embed digital value into digital assets. As an initial experiment, NFTs have been extremely successful, whether you believe that they have achieved this goal or not. The overarching concept of the metaverse harnesses the same technology —

[1] the problem of how to determine whether a given digital asset was spent more than once

[2] fundamentally, a problem describing the difficulty of trusted information transfer without requiring all participants to be trustworthy

the latest project to emerge in the blockchain and cryptocurrency space, the metaverse is one of the most fundamental to recreating the structure of value that exists in the physical world. Communities of identities, with sovereign rights and real estate, exchanging digital goods within a digital economy.

1.2 Money as memory

In order to transition to this digital utopia we need to embed trade at the blockchain level, bringing the same tools that gave us digital cash to effect digital trade. As mentioned previously, blockchains have been able to solve the previously unsolvable problem of digital money due to their ledger-based functionality – instead of a medium of exchange, there is a record, which is immutable and requires no trust in a third party. This ledger approach allows for the solutions to the double spend problem (Chohan & Chohan, 2021), and the Byzantine Generals' Problem (Lamport et al., 1982), ensuring that computation only needs to act on an accounting system rather than on the exchange of specific binary data through communication channels, which could be easily corrupted.

The parallels between taking such an approach for the newest form of money and one of the oldest forms of memory are striking. The 'stone money' on the Micronesian island of Yap (Friedman, 1991) was sufficiently unwieldy to require that each stone (some weighing thousands of pounds) remain in place even when its ownership changed hands. An oral ledger kept track of who owned what, and for all intents and purposes, money existed as a form of collective societal or institutional memory.

Money is an 'instrument of collective memory' before it is a means of exchange, a unit of account or a store of value. – Rachel O'Dwyer, *Cache society: transactional records, electronic money, and cultural resistance*, 2018 (O'Dwyer, 2018)

Blockchain-based distributed ledger entries typically impart ownership to digital money via asymmetric cryptography. A public/private cryptographic key pair creates a digital identity that mirrors an identity in the real world (although the relationship may not be one-to-one). Blockchains are also immutable, which means that the memory of ownership is preserved in a secure and time-resistant fashion.

A layer 1 approach to digital trade is ultimately the only means to effect the transfer of digital value. If money functions as an instrument of collective memory on the most fundamental level, trade must as well. A ledger that allows for the mechanics of only the payment is incomplete. This is particularly important in the digital world, where it is difficult to demonstrate the uniqueness of the asset. A digital ledger entry (with appropriately secure and distributed consensus/cryptography) can impart uniqueness, however, and so incorporating the asset in the ledger in addition to the payment, as an atomic swap, creates native blockchain trade. Such a system is akin to a native over-the-counter (OTC) desk but within a layer 1 blockchain protocol.

1.3 Digital value

A very basic ontology for describing value divides the concept into two types: *intrinsic*, which is a form of value inherent to the asset and its existence, or *instrumental*, which is the utility value of an asset; its purpose, its functionality, or its usefulness as a means to an end. On a practical level, digital systems consist of *files*, which act as containers for knowledge, and *applications*, which act as containers for logic. While applications can read and, if necessary, extend existing knowledge within the system, it is the file itself that contains the knowledge (here used in an abstract, general sense — a file containing images as art may not be knowledge in a strict sense of the word, but it still contains the fruits of a author's creativity, skill, and imagination) and the effort and work that has gone into developing and recording that knowledge.

Files and applications are treated differently in terms of memory management on the protocol layer of most digital systems. Files generally persist, while applications are typically loaded into cache memory and used temporarily for as long as they are needed. This high-level ontology indicates a strong correlative relationship between files and intrinsic value, as well as between applications and instrumental value. Applications realise their value only while loaded in memory, performing their task, while files retain a form of intrinsic value even while inert and existing merely in storage.

The nature of a file as providing infinitely variable content is demonstrably suited to represent all forms of digital value, while an electronic coin, or a token, clearly cannot. As we have discussed, however, while digital coins or tokens can 'live' on a blockchain, files generally cannot — they are more akin to digital objects, and contain binary data, which needs to be preserved across a communication channel. Writing this data to the blockchain would be vastly prohibitively expensive, both in terms of block space and in terms of verification, and would violate privacy.

This realisation seems to suggest that the machinery of a blockchain – with all its benefits of providing proper digital identity, digital ownership, security, efficiency, and of course machine readability – cannot be brought to bear on the thing of ultimate value in a digital system (the file). This would be a shame. The digital utopia offered by blockchain technology would be missing a vital piece of the puzzle without the incorporation of files into the digital realm, each capable of being owned via a digital identity and traded in a digital economy.

1.4 Digital ownership

Thankfully, there is a solution: you *govern* the file from the blockchain. You do this in a way that imparts ownership of files to digital identities, whether they correspond to individuals, or groups, or institutions, and so on. However, you first need a definition of ownership that can be incorporated into legal systems. We define digital ownership as a bundle of rights associated with a digital asset (refer to Section 3) that impart the complete dominion, title or proprietary right in the digital asset.

Practically speaking, we boil this form of ownership (at a high level) down to three key elements:

1. Being able to demonstrate uniqueness
2. Access & control
3. Protection of content (where relevant)

The first two requirements are fulfilled for on-chain assets on standard blockchains, because the ledger accounting ensures that each is tracked and identified across its entire history (regardless of its fungibility or non-fungibility) and the cryptographic public/private key pair allows the holder of said key pair to move or onspend the on-chain asset. The third requirement does not apply to the typical coins or tokens found on a blockchain, because there is no 'content', in any meaningful sense, to protect. However, this third requirement would apply to a file under blockchain governance, particularly if privacy was a key concern.

With these key elements, the Zenotta digital system ensures ownership of the digital asset. In addition, the disposal of the various rights from the bundle of ownership rights (the low-level, in-detail version of the three requirements above) that we introduce in Section 3 is ensured in such a way that real digital contracts can be negotiated, concluded, and executed, and in the event of non-performance or default, dispute resolution can also be handled via the digital system.

This whitepaper is structured as follows. In Section 2 we outline the network protocol, which consists of a layer 1 blockchain ledger & network that facilitates two-way transactions between a Zeno (our native cryptocurrency) and a token asset. The token asset type is a broad definition that can be applied across a variety of asset types, and it is exchanged atomically for Zeno (or for another token asset) through a new, dual double entry form of ledger. We discuss the topology of the network, which consists of a three-tiered architecture for low latency and high throughput, and the coordination between the various node types and users in terms of submitting a transaction, adding it to a block, mining the block, and adding the block to the chain. We describe the design of the transactions in our system along with the transaction types, which includes a brief outline of the economic ('tokenomic') model. We outline the consensus mechanism for each tier of nodes, and introduce a form of balanced mining that aims to make the mining power across network more inclusive and decentralized. Finally in Section 2 we introduce the memory management of the transaction types and describe **Smart Data**, our term for the asset on the other side of the peer-to-peer electronic trade. Section 3 encompasses the legal framework surrounding and enabled by the technology presented in Section 2. Finally, Section 4 presents a few salient conclusions on the technology and how it might benefit global digital systems.

2 Network protocol

Any network protocol for handling electronic coins, promissory notes, rights, claims or digital assets combines at least two basic operations:

1. Users employ public/private key cryptography to take control of, and transact with, their digital value objects.

2. An independent server (or network) timestamps transactions to preserve the globally correct chronology of transaction order.

In any batch processing of transactions we think of transactions as being wrapped in a block, even if the batch is designed to admit only one transaction per block. Blocks are therefore the smallest data structure that can be batch processed by the network.

There are three desirable properties in any distributed network concerned with the transfer of information. These are formalised by Brewer's theorem (Brewer, 2012) via the acronym CAP, which refers to:

1) **C**onsistency – *all nodes in a distributed system have a single, current, and identical copy of the data.*

2) **A**vailability – *nodes in the system are able to accept incoming requests and respond with data, without any failures, as and when required.*

3) **P**artition tolerance – *if a group of nodes is unable to communicate with other nodes due to network failures, the distributed system continues to operate correctly.*

Any given distributed system can only offer two features by sacrificing the third. As partition tolerance is a must have, distributed systems typically offer either consistency and partition tolerance (CP) or availability and partition tolerance (AP).

Blockchains are often seen as a special case whereby all three properties are present (and thus a violation of Brewer's theorem). However, the consistency that blockchains offer is a weaker type known as eventual consistency, which is where consistency is achieved as a result of validation from multiple nodes *over time*, rather than simultaneously with the other two properties. This is where mining comes in, with blocks of transactions further back in time becoming exponentially more established in the shared state that is the ledger.

The other major design challenge that distributed systems face is handling latency and throughput. Blockchain protocols exhibit high latency and low throughput because of the fundamental design choices required in offering Byzantine Fault Tolerance (BFT); most notably, the need for every mining node to execute and store computational tasks. The high latency is in fact a feature rather than a bug, since in Nakamoto consensus (where the longest chain is designated as the valid transaction history) fast block generation increases

the number of unintentional forks[3] that emerge during the mining process of finding new blocks. A higher rate of unintentional forks allow potential attackers to undermine the security of the chain at a lower Byzantine fault percentage.

Our approach makes use of the advantages inherent to a blockchain in terms of trustlessness[4] and distribution while achieving consistency (C) in addition to availability and partition tolerance (AP). This is made possible through the use of *persistent trustless decentralized nodes*, which integrate with the *non-persistent trustless distributed nodes* that make up the mining network. The persistent untrusted decentralized nodes maintain a single version of the blockchain, eliminating forks, and achieve their trustlessness property by taking input from the non-persistent trustless distributed nodes to generate verifiable randomness that directs the decision-making process. The parallelisation and sharing of the computational load made possible with this approach vastly increases throughput and reduces latency compared to a standard mining-only approach.

Our network setup effects the ability to handle two-way trade of payment and asset natively on the ledger, which we outline in Section 2.3.1. This is a vital component of the peer-to-peer electronic trade system, and further enables the governance of files through a trustless notary service.

2.1 Topology

Here we define the network topology of all parties that contribute towards a successful trade between Alice (making a payment but expecting goods or services) and Bob (transferring goods or services and expecting a payment). We refer to the persistent trustless decentralized nodes (two types) as compute nodes & storage nodes, and the non-persistent trustless distributed nodes as mining nodes.

The compute and storage nodes comprise two (CP-type) decentralized compute and storage ring networks that are used to *create* and *write* transaction blocks to the append-only ledger. Blocks are *validated* by ephemeral mining partitions that are set on a block-by-block basis. Each partition is coordinated by a member of the decentralized compute ring network.

The presence of the compute and storage rings provides users of the network with entities akin to blockchain service providers, which are able to offer service levels, monitoring, and efficient auditability. The single copy of the ledger, written to the storage ring, acts as a "single source of truth" and prevents wasted computation.

Figure 1 shows a basic high-level description of the three node types that work to create and maintain the blockchain. In conjunction with the user nodes, these form the three tiers of the network:

1. Compute network – *compute nodes*

[3] a blockchain experiences a fork when a single block has two or more children blocks

[4] this term, which will crop up often, meaning that trust is not needed for the system to work

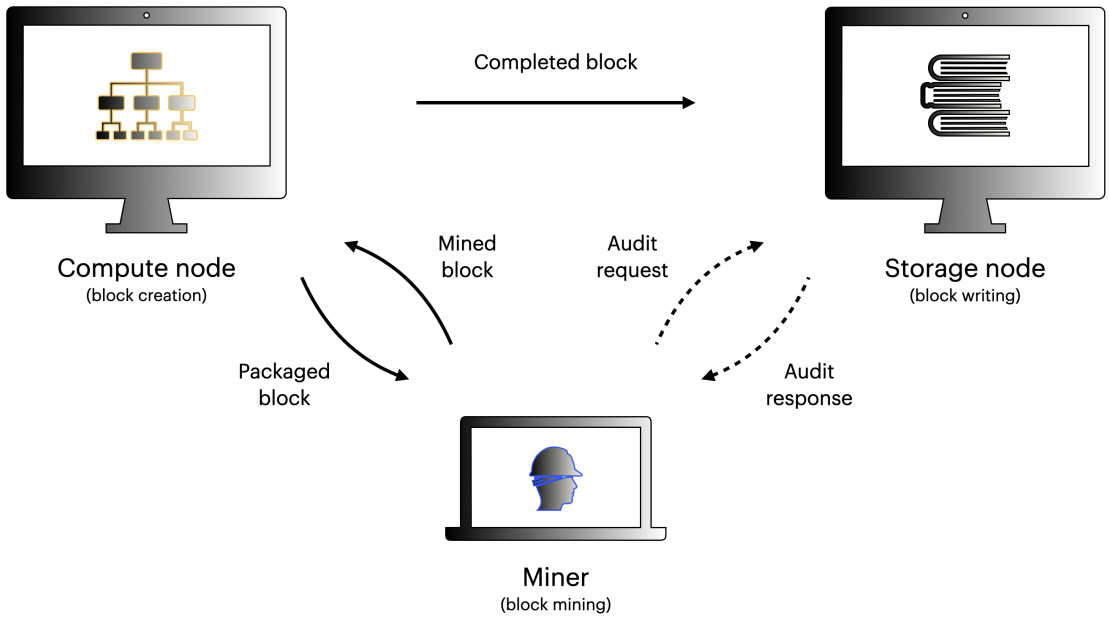

Figure 1: A cartoon depiction of the relationships between the three node types that maintain the blockchain. Compute nodes package randomly-selected transactions into a block and broadcast it to the miners to perform transaction validation and mining consensus. The mined block is sent back to the compute nodes for final checks before it is sent to the storage nodes to be written to the blockchain. Miners can at any and all times perform auditing checks on the data stored in the storage nodes.

2. Storage network – *storage nodes*
3. Contributor network – *mining nodes, user nodes*

User nodes play no part in verification but they are used as contributors to the verifiably fair randomness that keeps the compute nodes trustless and impartial (more details in Section 2.2).

2.2 Coordination

Compute nodes are connected to each other as a decentralized ring network known as the compute ring; similarly, storage nodes are connected to each other as a decentralized ring network called the storage ring. Each compute node is connected to at least one storage node. Mining nodes "apply" to compute nodes for entry into a mining partition. Within a given mining round, miners who win this application process are assigned randomly to a specific compute node and then exist within a specific mining partition, separated from the rest of the active mining network in other partitions. Mining node assignment to a partition is ephemeral, and is recomputed every round.

Network	Node type	Function	Topology	C/A/P
Compute ring	Compute node	Block create	Decentralized	CP
Contributors	User node Mining node	Send tx Block verify	Distributed	AP
Storage ring	Storage node	Block write	Decentralized	CP

Table 1: The components of the three-tiered network.

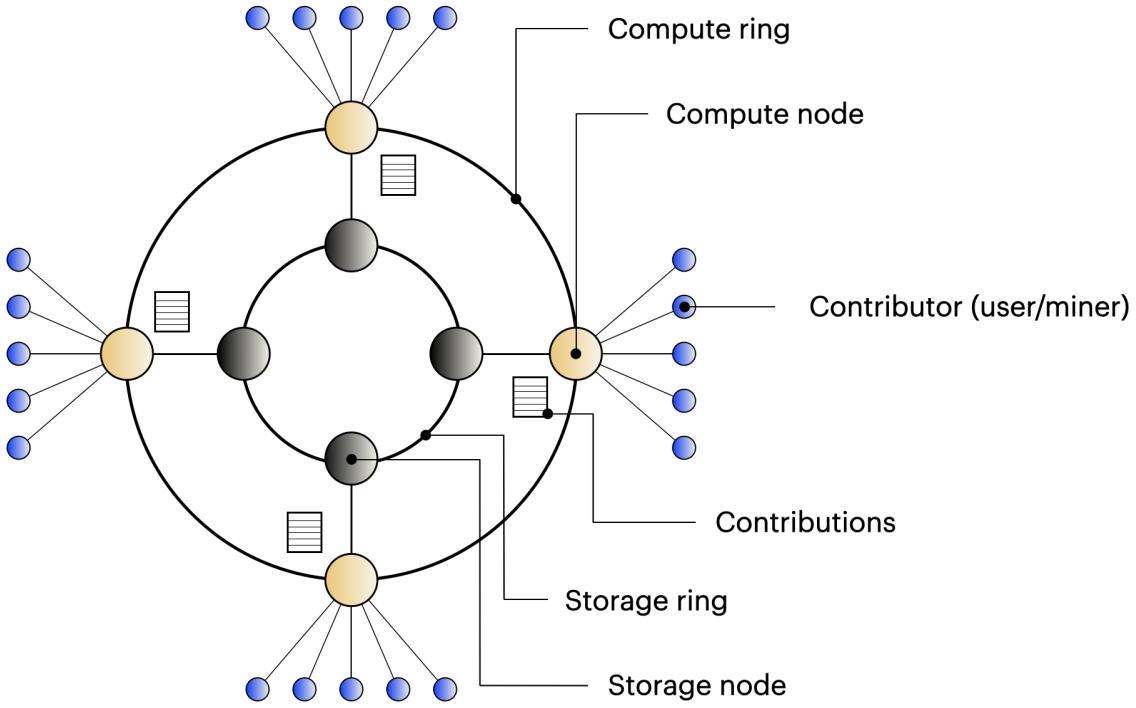

Figure 2: The topology of the network, shown here a small subset of nodes, for all node types. Compute and storage nodes are connected as a decentralized ring network while mining and user nodes form distributed partitions around a compute node and storage node pair.

Partitioned mining nodes and user nodes can send data to a compute node. All mining nodes (whether selected into a partition or not) and user nodes can request data from storage nodes as an optional audit.

Miners signal their proof-of-works to their assigned compute node, which then sends the validated block to the storage node. Once written to storage, other miners signal their acceptance of the block that was written by continuing to mine new blocks that extend

from the previously-written blocks.

It is important to note a difference here between a standard mining approach (in, e.g., Bitcoin) which uses a continuous time framework where the first miner that succeeded in adding a block to the blockchain will get a reward. In contrast to this, we use a discrete time framework where more than one miner may succeed within the same time period. But only one of these successful miners will extend the blockchain and get a reward. The fair (verifiable) selection of this winning miner out of all successful miners is achieved through a form of uncontestable randomness, which we describe in the next section. This verifable form of randomness is also used for other selection processes in order to eliminate the ability of the compute nodes to control any selection.

2.2.1 The UNiCORN

All contributors (in some way) submit data into an algorithm that produces an UNCOntestable Random Number (UNiCORN). This algorithm is running on a designated decentralized node (which can be either a compute or storage node, but not a miner or user node). The algorithm generates a UNiCORN for each block, which is used downstream for any required random selection, such as miner partitioning. The paper:

> Arjen K. Lenstra and Benjamin Wesolowski. *A random zoo: sloth, unicorn, and trx.* Cryptology ePrint Archive, Report 2015/366. `https://eprint.iacr.org/2015/366`, 2015

describes in detail a method to generate uncontestable random numbers. In this section we summarize the relevant steps of this method as shown in Figure 3.

The UNiCORN algorithm starts with a public announcement about a public data gathering phase. During the announced time interval, public contributors (the users and the miners) send their contributions to the compute ring, which concatenates them in the order of their arrival to form a total public contribution. At the end of the time interval the compute ring immediately publishes the public contribution and shortly after that its commitment value, which consists of the double hash of the total public contribution concatenated with an internal contribution from each compute node. The compute ring then computes the slow-timed hash (SLOTH) (Lenstra & Wesolowski, 2015) of the contributions, which produces the uncontestable random number along with a witness value that allows for a fast verification of the correctness of the uncontestable random number. The compute ring publishes the internal contributions, the uncontestable random number, and the witness value, allowing for a public verification of the validity of the UNiCORN.

The UNiCORN enables, crucially, fairness and transparency (and in particular, trustlessness). UNiCORNs themselves are perfectly suited for one-off random decision making, but we employ UNiCORNs for making a series of fair, random decisions on the part of the compute nodes. To do this we feed the UNiCORN as a seed into a suitable pseudo-random number generator and then use the output stream to make the following decisions:

1. Selecting which transactions are packaged into a block.

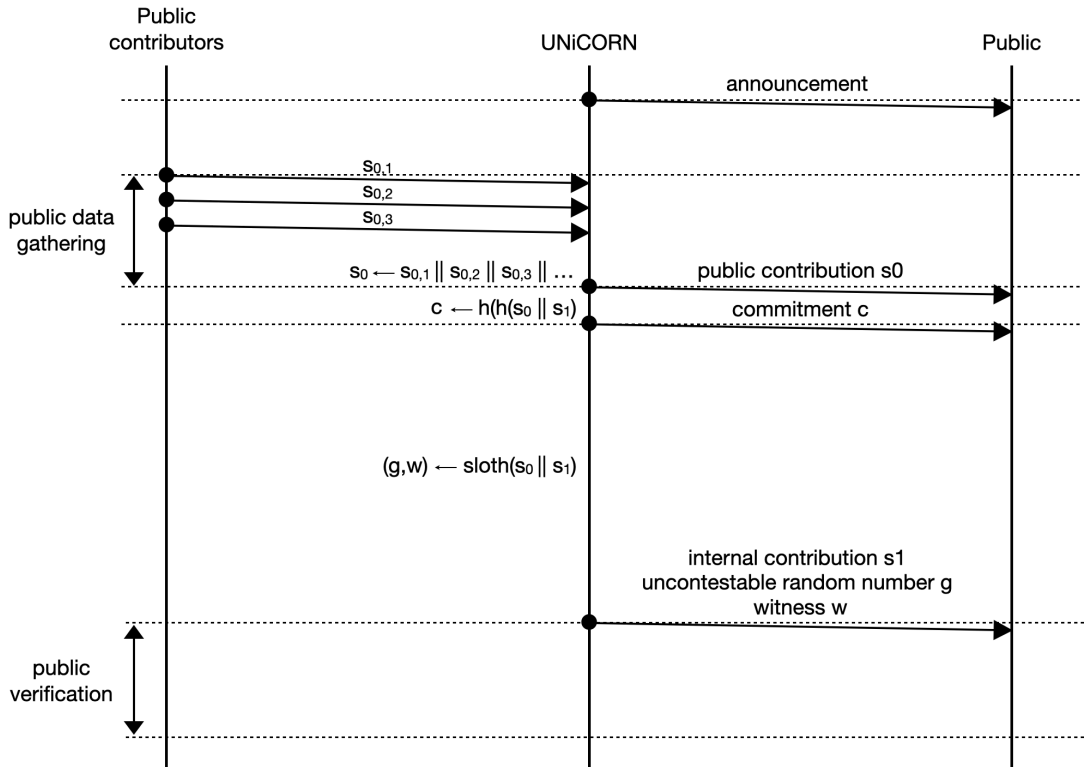

Figure 3: A generalised view of the UNiCORN protocol. It begins with a public announcement that the public data gathering will take place during a given time interval. During that time interval public contributors send their contributions ($s_{0,1}$, $s_{0,2}$, $s_{0,3}$, ...) to the UNiCORN protocol, which concatenates these contributions in the order of their arrival to form the public contribution s_0. At the end of this time interval the UNiCORN protocol immediately publishes the public contribution s_0 and shortly after that its commitment c, which is the double hash of the concatenation of the public contribution s_0 with an internal contribution s1. What follows is the computationally demanding part. The UNiCORN protocol computes the slow-timed hash (SLOTH) of the contributions. The SLOTH produces two outputs: the uncontestable random number g and the witness w. The witness w allows a fast verification of the correctness of g. During the public verification phase everyone interested should be able to perform the verification. The internal contribution s_1 is made by the compute nodes.

2. Selecting which miners will participate in mining during the next time period for a given mining partition.

3. Selecting from the winning miners (assuming more than one miner has found a valid proof-of-work) which miner's block is added to the chain.

The pairwise nature of the ledger in enabling payment-for-asset or asset-for-asset trade means that such a transaction pair receives a single entry in the list of pending transactions, from which the UNiCORN algorithm selects those to become part of the block. With this approach, both parts of a transaction pair are either selected or not selected.

As discussed, the selection process must be fair and transparent. Fairness is achieved by selecting the elements uniformly at random from a list of elements, while transparency is achieved by making the selection process pseudo-random and by publishing the underlying algorithms such that the correctness of the selection process can be verified.

The pseudo-random number generation process built on the UNiCORN algorithm can, if necessary, be weighted according to specific factors – a particularly relevant example is weighting according to the size of the transaction fee, in order to enable the prioritizing of transactions with higher fees (for the users) and incentivising the miners further in terms of the reward obtained through winning the block.

2.3 Transactions

The high-level sequence for transactions across the various network tiers is as follows:

1. Compute nodes receive transaction requests from user nodes and compile blocks of transactions.

2. Once compiled, blocks are given to mining nodes assigned to a partition to be validated and mined.

3. Miners who are part of a valid partition and find a valid proof-of-work submit their solution to their assigned compute node.

4. Compute nodes aggregate all valid solutions into a list and select one winner from the submissions.

5. That winner's mined block is written to the storage node and state replicated across the storage ring.

As described in the previous section, each decision made by the compute nodes in this process is determined via a UNiCORN-driven pseudo-random number compiled from all miner and user contributions.

2.3.1 The DRUID

Both one-way and two-way transactions can be processed over the three-tiered network. Handling a traditional, single signed transfer of coins from Alice to Bob, asynchronously, is relatively straightforward: if Alice knows Bob's public address she can conclude a payment instruction without Bob's involvement. With a two-way trade of coins for goods, however, where both Alice and Bob must co-sign a transaction, a different approach is required. For this we employ what we term a Double Resolution Unique ID (DRUID).

In the event that Alice wants to, e.g., trade a number of Zenos for Bob's data asset, both parties will need to enter into a DRUID-based transaction process. This is laid out in Figure 4. In this process, Alice is attempting to trade her Zenos Z for Bob's data asset A. She will create the first half of a DRUID value (D_1), which can be as simple as a hash value, and send it to Bob in combination with an expectation value E_Z that specifies that Bob expects a certain number of Zenos (Z) from Alice. This ensures that Bob is trading his data asset for the correctly desired value and that Alice has expressed her legally valid intention to enter into this agreement.

Once received, Bob will generate his half of the DRUID value (D_2) and send this back to Alice in addition to his own expectation E_A, in order to specify that Alice expects a certain data asset (A) from Bob (Bob has then also expressed his legally valid intention to enter into this agreement). Once both parties have the two halves of the DRUID they can concatenate them to form the final, common DRUID value (D_F). This value allows the compute node, which will process the transactions of both parties, to understand that these transactions need to be considered as a single trade, and allow the contract to be concluded.

Bob now sends to the compute node the following set of values: A, the data asset he intends to send to Alice, E_Z, the expectation describing the Zenos he expects in exchange for his data asset, and finally the D_F DRUID, which will be the common value that the compute node can match to Alice's transaction. Bob has now fulfilled his contractual obligation.

Alice does the same in her transaction to the compute node, namely: T, the tokens she intends to send to Bob, E_A, the expectation describing the data asset she expects in exchange for her tokens, and finally the D_F DRUID, which will be identical to Bob's. This is Alice's fulfillment of her contractual obligation.

A trade between Alice and Bob proceeds via the sequence below:

i. Alice pre-owns some Zenos Z.

ii. Bob pre-owns some digital asset A.

iii. Jointly, Alice and Bob generate a **D**ouble **R**esolution **U**nique **ID** (DRUID).

iv. Independently, Alice and Bob send their half of the trade with authorization to a compute node.

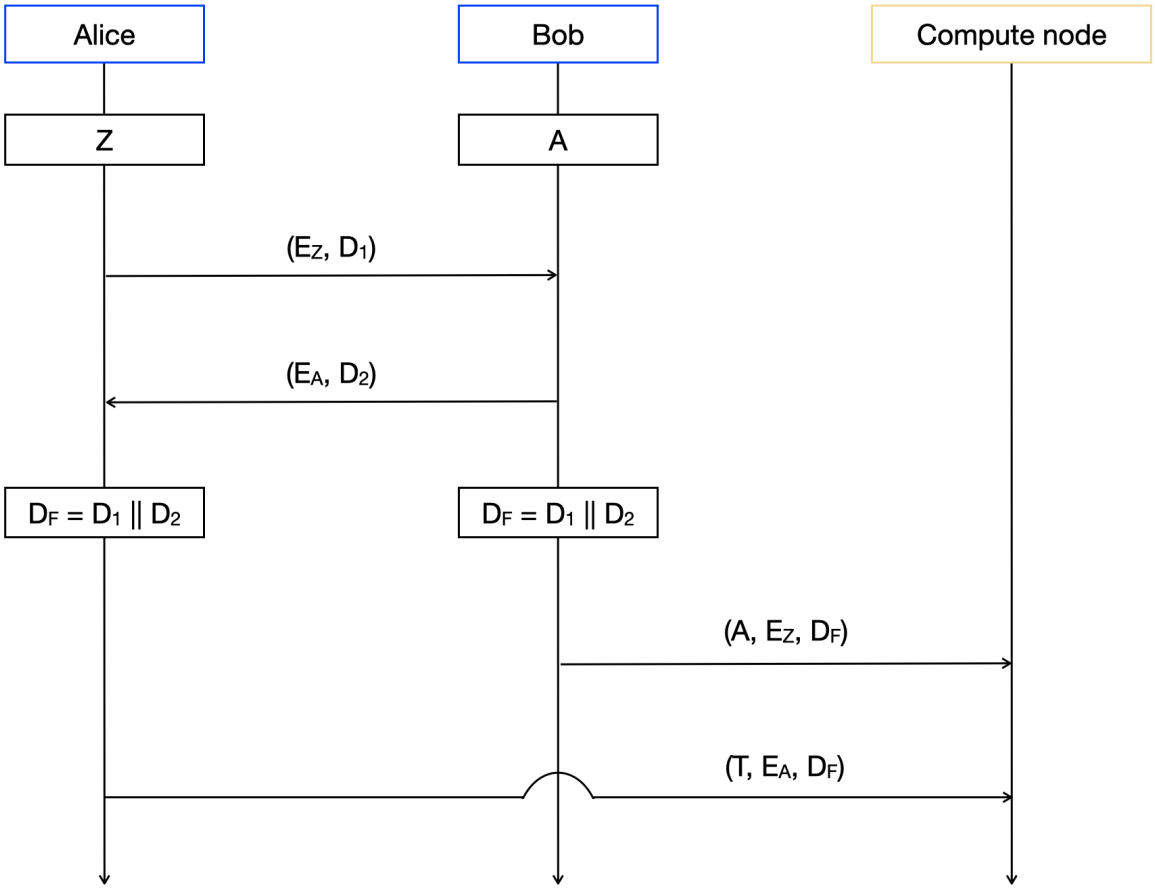

Figure 4: Alice and Bob use DRUIDs to form a dual double entry on the ledger.

v. Trade requests from Alice and Bob are fed into the UNiCORN-generation process as contributions.

vi. The UNiCORN-driven pseudo-random number generator selects which transactions are bundled into block n, while the remaining transactions are set for carry over into future blocks.

vii. The compute node collates transaction halves into valid trades and bundles trades into blocks to be mined by miners.

ix. Miners submit their coinbase transactions as contributions to the UNiCORN generator.

x. The UNiCORN-driven pseudo-random number generator assigns miners to specific compute nodes. This assignment partitions the mining network. Not all miners are admitted to mine a given block.

xi. Each compute node enters into a Raft consensus (Ongaro & Ousterhout, 2014) round to ensure they each share a valid and consistent copy of a block to be mined.

xii. Compute nodes transmit blocks to assigned miners for validation and mining.

xiii. As each block mined was originally packaged by a compute node, rapid miner validation of large blocks is possible through a form of statistical checking combined with a novel data structure (Barry et al., 2021).

xiv. Each miner capable of computing a valid proof-of-work in the allotted time and from a valid partition submits their proof-of-work as a UNiCORN contribution.

xv. A UNiCORN-driven pseudo-random number elects a winning miner from the list of all miners who have submitted valid proof-of-works and who were part of a valid partition. The winning miner's coinbase transaction becomes the winning coinbase.

xvi. Valid mined blocks are written to the storage nodes via the compute nodes.

xvii. Storage nodes use Raft consensus to agree on written blocks.

xviii. A suitable tree of contributions used to compute a UNiCORN is pruned down to a suitable storage size and written to the storage nodes for future validation.

The compute node here is acting as the transaction's *notary*, registering each legally-binding expression of will by all parties. This role could be fulfilled by any node, in theory, or even another user, Charlie – and fundamentally removes the need for a smart contract to be involved in the transfer of an asset for a payment, while remaining trustless, since the notary remains impartial and merely attests to the DRUID value. If the notary does not fulfill their role properly the trade simply doesn't happen, and Alice and Bob need to find a new notary.

The ability for Alice and Bob to set and define expectations that finalise or revert a trade enables many possible utilities. Four noteworthy instances are:

- **Atomic trade:** *The expressiveness of smart contract languages is a double-edged sword, allowing simple code bugs to become serious miscarriages of justice. The UTXO model of a transaction, however, is extremely simple, and allows a payment to be sent in a single atomic step, carried out on the blockchain without the need for a virtual machine equivalent. The DRUID extends this ability to 2 distinct paths for 2 distinct asset types, which can then meet at the 1 point in time that Alice and Bob wish to execute a trade.*

- **Governance without control:** *When the merchant commits to on-spending an asset to Alice's control, Alice can set her payment to Bob against the expectation that the asset is not found on any sanction list. If it is, the expectation is not fulfilled and the trade is not entered into a block for mining.*

- **Domain-based payments:** *A previous entry by Bob might be a digital asset in the form of a certificate. His associated blockchain public key to the certificate can be challenged to produce a new signature for proof of liveness. If such proof can be offered then the certificate can be deemed valid and checked for the domain name, which aliases a payment address to which Alice's incoming payment can be assigned. If all entries match, the payment is included for mining, and if not it is rejected.*

- **Receipt-based payments:** *If Alice requests a receipt from merchant Bob, Bob's countersigned receipt is proof that he willingly and knowingly accepted payment under the programmed terms.*

2.3.2 The Zeno type

The Zeno is the means of payment on the Zenotta network. It is defined as a blockchain-based currency, or coin, that is native to the layer 1 blockchain ledger. It functions as the medium of exchange for digital assets traded on the Zenotta network; namely, for digital goods and/or services.

The Zeno as a transaction object consists of a script, that executes the transfer of the coin on the blockchain, and a signature, that gives the owner of the asymmetric public/private key pair the right to onspend the coin. The scripting language at the low-level compiles to OPs codes while at the high-level is implemented using the BitML process calculus (Bartoletti & Zunino, 2018) designed for Bitcoin smart contracts. The script describes the logic that can be assigned to the resource, e.g. the conditions under which Bob can receive a certain amount of Zeno from Alice. Figure 5 shows a simple representation of the Zeno transaction type, where the script references a transfer of 5 Zeno from Alice to Bob.

The Zeno is a fungible blockchain-based coin, created through the PoW mining process (see Section 2.4). We outline the issuance of the Zeno in terms of mining and allocation below.

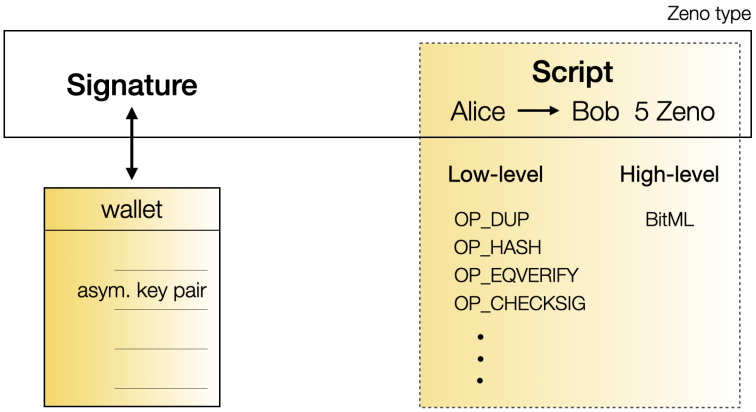

Figure 5: The Zeno transaction type on the ledger, which consists of (i) a signature to assign ownership of the right to onspend to a wallet containing a public/private key pair and (ii) the script to be executed that effects the movement of the coin.

Issuance

Zeno issuance follows a fixed supply cap mechanic according to the properties outlined in the following table:

	Number of Zeno
Total cap	**10 billion**
Treasury (*Zenotta Holdings AG*)	2.5 billion
Mining cap (*Miners, compute nodes, storage nodes*)	7.5 billion

The total number of Zeno coins reached will be 10 billion, with 2.5 billion reserved for a treasury, to be used for stakeholders, a development fund, and an economic activity fund. The remaining 7.5 billion is mined out via a smoothed issuance curve based on the emission approach employed by the CryptoNote protocol. The smoothed issuance is designed to minimize the volatility seen in standard halving mechanics whereby the reward drops suddenly by half on a particular date. The Bitcoin stock-to-flow model (PlanB, 2019) lends support to the idea that this sudden reduction in supply is at least partly responsible for the extreme bull market/bear market cycles.

The sub-unit of the Zeno is the zent. A number with a large number (90) of divisors was chosen as the conversion factor between the zent and the Zeno:

Coin sub-unit	
1 Zeno	25200 zents

This particular conversion factor is the 24th antiprime, or highly composite number, in the sequence of positive integers with a greater number of divisors than any smaller positive integer. This antiprime number has 90 divisors, and 9 prime factors. The use of a number with a large number of divisors facilitates the use of fractional payments without the need for rounding. At the same time we stay within the bounds of the `u64` integer type employed in our codebase for the total cap in zents of 10 billion $\times$ 25200 $= 2.52 \times 10^{14}$.

The block reward in zents is calculated using the total Zeno supply and the current Zeno supply in the market via the following formula:

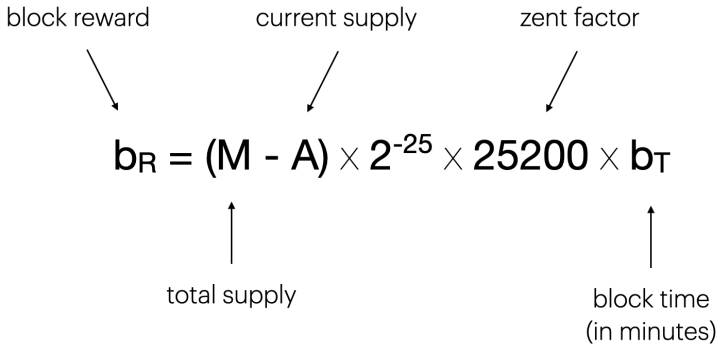

Figure 6: The equation determining Zeno issuance, based on the CryptoNote protocol.

In a practical network protocol, issuance must be implemented using bitwise operators, in order to ensure consistent performance across different hardware types. This allows operations to be performed on the bit level and therefore proceed at the maximum possible speed. Additionally, this approach ensures the consistency of floating point operations across different architecture types. In Bitcoin and bitcoin-like protocols this manifests requiring the block reward to drop by half (the 'halving') every n blocktimes. This is due to the bitshift operator being applied to the block reward (the left-hand side of the equation). We apply it to the recursive right-hand side of the issuance equation, which allows for a smoothed curve without the sudden halving jumps that likely have undesirable economic properties due to sudden supply shocks. Therefore, the block reward in zents (for a 60 second blocktime) is given in practical terms by

$$\text{reward} = (\text{total supply} - \text{current supply}) \gg 25. \tag{1}$$

The mining emission is divided into two parts, with the miners taking the majority and

the compute and storage nodes being allocated a small percentage that varies over time (starting at 15% and reducing smoothly to 5% via a reverse logistic function). This rewards the compute and storage nodes for the larger role that they play in the initial start-up phase of the network, and to recoup capital costs, and to incentivize miners to play a larger role in the network later on.

2.3.3 The Smart Data type

The Smart Data type is the other half of the transaction (refer to Section 2.5). There are two Smart Data types: on-chain and hybrid on/off-chain, which possess different properties. In both cases the creation mechanism for these assets works in a similar way to a standard coinbase transaction, with users constructing a Smart Data creation transaction which spawns a value from nothing. This is submitted to the compute ring for inclusion into the block.

In order to prevent the standard abuse of a asset whose creation is 'free', e.g. DDOS attacks, the requirement to create new Smart Data tokens of both types is to do some work, in the form of auditing a part of the blockchain and the randomness that has contributed to the UNiCORN seed (see Section 2.4.2). This involves an interaction between user nodes and storage nodes, whereby the user nodes will be required to do a very light Proof-of-Work that feeds into a process known as Shuriken auditing. This process employs successive communication with the storage nodes that ramps up the difficulty that each subsequent auditor experiences on the event of a disagreement (see Barry et al. (2021) for more details).

Smart Data type I (on-chain)

The Smart Data type I, on the most basic level, functions as a receipt that constitutes a countersigned acknowledgement that the payment was willingly and knowingly accepted, under the programmed terms of the transfer. However, it can also be used for representing any digital assets on-chain. It is defined as a blockchain-based token that is native to the layer 1 blockchain ledger.

The structure of the Smart Data token type I as a transaction object is identical to that of the Zeno type, as shown in Figure 7. There are certain OPs codes that can be employed in the script for the Smart Data token type I that are not available for the Zeno type, for example `OP_END`, which is used to burn the Smart Data token and remove it from circulation (this is not possible for a Zeno). The issuance model is unlimited, and the creation (as described above) is user-based and resource-limited.

Smart Data type II (hybrid on/off-chain)

The Smart Data type II is a hybrid on/off-chain resource, that constitutes a decentralized file format (described in Section 2.5). The on-chain component is defined as a blockchain-based Smart Data token that is native to the layer 1 blockchain ledger.

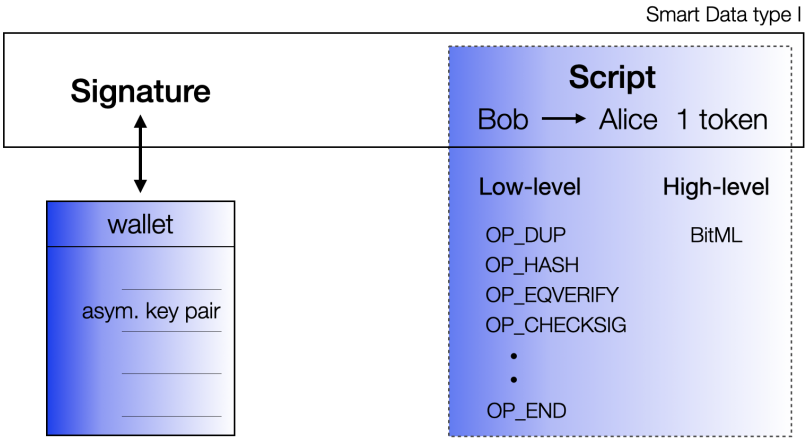

Figure 7: The Smart Data transaction type I on the ledger, which consists of (i) a signature to assign ownership of transfer rights to a wallet containing a public/private key pair and (ii) the script to be executed that effects the movement of the Smart Data token.

The structure of the Smart Data type II as a transaction object includes the signature and script components that are also present in the Smart Data type I and the Zeno, but with the addition of a pointer that connects the on-chain smart data token with the off-chain file component of smart data (extension `*.ta`). This pointer (known as the Data Rights Signature, or DRS) employs a unique encoding and encryption schema, with four hash values and an encoding scheme placed on the blockchain (refer to Figure 16) that ensures digital twin resolution and allows for governance of the off-chain file from the blockchain. The script incorporates on the high-level both the BitML process calculus and the PowerShell scripting language (Jones, 2020) that allows for containerized operating system logic to be executed off-chain from contracts that are on-chain. This pointer mechanism constitutes the 'smart data protocol' that integrates with the network protocol.

Finally, the wallet contains, in addition to the public/private key pair, two encryption keys K_1 and K_2, which encrypt the content of the off-chain file (for more detail on how these keys are used see Figure 16).

Issuance of the Smart Data type II is the same as that of Smart Data type I, which is unlimited in number but limited by resource – users wishing to create Smart Data of type II must perform some validation of the ledger.

2.3.4 The dual double entry ledger

The Zenotta blockchain ledger is based on unspent transaction outputs (UTXOs). This is an accounting approach that relies on outputs of transactions, received by a user, that can be spent at an unspecified date in the future. Transactions therefore consume existing UTXOs and create new UTXOs as they are processed, being combined or divided to reach

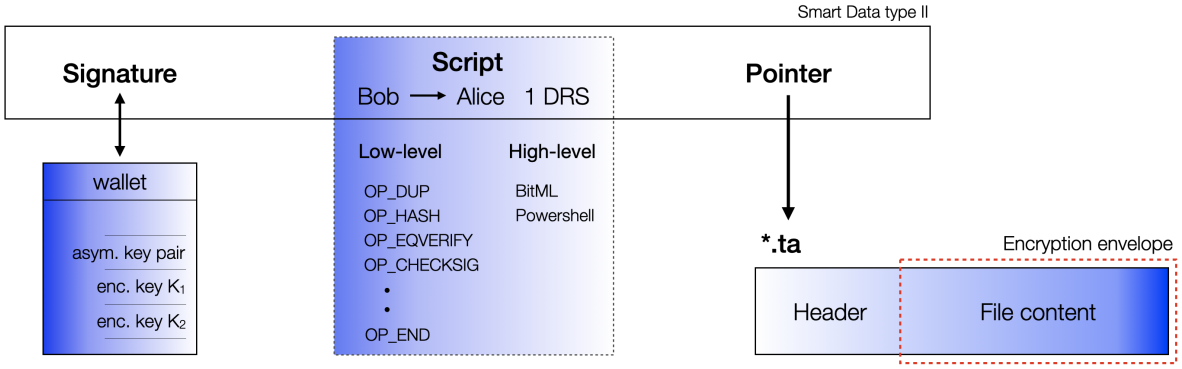

Figure 8: The Smart Data transaction type II on the ledger, which consists of (i) a signature to assign ownership of transfer rights to a wallet containing a public/private key pair (ii) the script to be executed that effects the movement of the Smart Data token (iii) a pointer to an off-chain file (extension `*.ta`). This pointer mechanism is known as the Data Rights Signature (DRS) and consists of four hash values and an encoding scheme, as described in Figure 16. The file is composed of the content, which is protected by encryption, and a file header, which is binary metadata containing information about the file itself, such as the author, the jurisdiction where the file is valid, and the option for programmable logic.

the required amount that needs to be transacted.

The choice of the UTXO accounting approach means that the legal framework surrounding the Zenotta digital system (see Section 3) can be applied *per transaction*, which extends the flexibility of peer-to-peer cash to legally-defended peer-to-peer trade.

The DRUID process outlined in Section 2.3.1 constitutes a UTXO-based approach to the trade of a payment for an asset. Each UTXO-denominated ledger (payment ledger; asset ledger) is handled separately and combined through the dual double entry (DDE) into a single ledger that matches the two histories of the payment and the asset at the moment of trade (see Figure 9). The inherent asychronicity in the two separate histories – the arrows of time that allow for a spend to occur in the future – is handled by folding them together into a single trade through the DRUID.

The employment of a dual double entry ledger gives something akin to a native blockchain over-the-counter (OTC) system. This means that a user can execute an atomic transaction of a payment-for-asset (or payment-for-payment, or asset-for-asset) on the chain without employing a smart contract. Auditing of these transactions can be done in real-time through the UTXO table, and, most importantly, each trade is visible and immutably stored in the blockchain ledger.

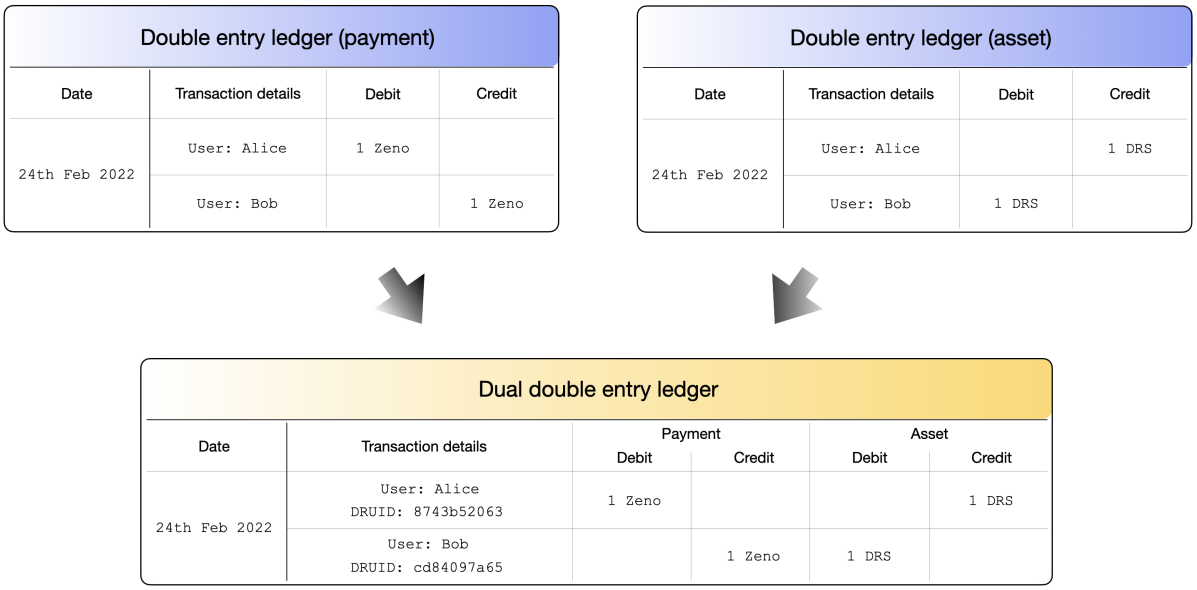

Figure 9: The *dual double entry ledger*, which tracks the payment and the asset separately and combines them atomically through the use of the DRUID. The process of matching the two histories of the payment and the asset, at the moment of trade, enables such an accounting system and allows for asynchronicity between the two halves of a trade. This is made possible by a transaction notary that times the two inputs and folds them into a single trade via the DRUID. Note that previous ledger approaches in cryptocurrencies and blockchains revolve around standard double entry ledgers, which only deal with unidirectional information, limiting the function of machine-executable trade for digital assets.

2.4 Consensus

The fundamental decision that every blockchain network must make is which node will append the next block to the blockchain. This decision is made using a consensus mechanism. A substantial number of consensus mechanisms exist, all possessing different properties in terms of security, complexity, and resource cost.

Without wishing to go into detail on the various consensus mechanisms and the comparisons between them, we will simply say that an approach where the resource cost requires a continued commitment is preferred, rather than a capital one. A continued resource commitment has the advantages of reducing centralizing effects and providing security against the threat of competing alternative histories (of the ledger). With computation as the resource, this approach links the process of consensus with the physical world, since computation is energy, and ensures that the source of truth that is the blockchain ledger is built through a physical proof.

To this end we employ Proof-of-Work (PoW) as our consensus mechanism, but in a role that is altered from the standard approach due to the presence of our three node types.

Fundamentally, in a blockchain, the consensus mechanism is used to select a truly random 'leader' to add the next block to the chain. Since all miners mine different blocks, and they choose the transactions that go into each block, this essentially means that no miner is able to select specific transactions for a sustained period of time, since the likelihood that a particular miner is the leader for many subsequent blocks is low. The resource cost of mining ensures that this process **stays** random, i.e. that it is extremely difficult to amass enough resources to consistently be the leader.

In our system, transactions are added to a block via the verifiably fair randomness provided by the UNiCORN, a process carried out by the compute nodes. This means that every miner receives the same block in a given mining round. Here, the consensus mechanism applied to the miners is used to produce a random seed that (in part) determines which transactions are added to the block, and so the resource cost of mining in this setup ensures that this random seed stays random, i.e. it is extremely difficult to amass enough resources to consistently influence the random seed through the public contributions phase, or indeed to overwhelm the public contributions phase in a similar way to a Sybil or a DDOS attack. The mining consensus process also, as in standard Bitcoin-like blockchains, creates the immutable link between blocks via a hash value that contains the header of the previous block, with computational work backing this immutability.

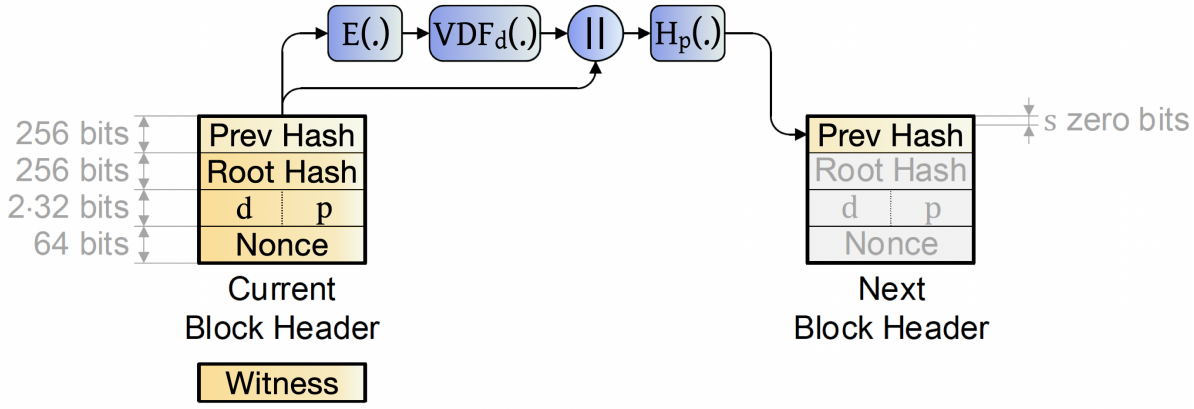

Figure 10: The process carried out by the mining nodes in order to achieve consensus. The block header is first fed through an expansion function that outputs an element of the domain of the verifiable delay function $\mathrm{VDF_d}(.)$, which creates a delay via a set of deterministic sequential steps. The output from $\mathrm{VDF_d}(.)$ is concatenated with the block header as input to the hash function $\mathrm{H_p}(.)$, the output of which becomes a winning hash if the nonce tried results in a hash below a certain threshold.

The mining process is shown in Figure 10. The standard mining process for, e.g., Bitcoin that takes the block header and feeds it into the hash function H(.) to produce (with a valid nonce) the hash for the *prev hash* field of the next block is modified slightly to include an additional verifiable delay function (VDF) along with two parameters d and p.

The purpose of these two parameters is to allow for a variable puzzle difficulty *between miners*, which is separate functionality from the standard difficulty adjustment that the network makes in PoW mining in order to preserve an approximately constant blocktime.

The $\mathrm{VDF_d}(.)$ is described in the paper (Boneh et al., 2018):

> Dan Boneh, Joseph Bonneau, Benedikt Bunz, and Ben Fisch. *Verifiable Delay Functions.* Cryptology ePrint Archive, Report 2018/601. `https://eprint.iacr.org/2018/601`, 2018.

This function requires a specified number of sequential steps to evaluate, yet produces a unique output that can be efficiently and publicly verified. The number of required sequential steps depends on the delay parameter d. Along with the function value a witness parameter is output (referred to as 'the proof' in Boneh et al. (2018)). The witness allows for fast forwards verification of the correctness of the function value, allowing peer miners to verify an extremely long delay regardless of their processing power.

The parameter p alters the number of iterations within our hash function, which employs a modified Keccak sponge construction (Bertoni et al., 2013) as shown in Figure 11. While both d and p have no functional limit, there is a practical limit on p in terms of verification – should the number of iterations be increased past a given threshold, it may be the case that a machine with a low hashrate is unable to verify the hash within the required time. The presence of the VDF is therefore partly to ensure that the individual miner difficulty can be tuned without limit, as well as to provide a second dial with alternative economic properties (see Section 2.4.1 for more details).

The precise form of expansion function and the VDF are detailed in a separate technical paper which can be found at `http://www.zenotta.io`.

The modification of the consensus approach that redirects PoW into generating a UNiCORN for decision making rather than to elect a leader to add the next block to the chain (as in standard PoW) removes the power of the 51% attack in gaining control over the blockchain ledger. The public contributions that determine the UNiCORN seed are rendered verifiably random even if just one participant is honest, and the slow-timed hash (SLOTH) adds a layer of security preventing the compute node from trying several internal contributions (see Figure 3) and selecting from among them the one that results in a random number with particular or desired properties.

2.4.1 Balanced mining

The functionality involving the parameters d and p allows for the assignment of individual mining difficulties as an addition to the global mining difficulty that requires the winning hash to be below a certain threshold. These individual difficulties allow for the effective hashrate among miners to be *balanced* – a term that does not mean that all effective hashrates are made equal but rather refers to the movement of the effective hashrate distribution **towards** equality. A miner with an extremely large amount of processing power can be slowed down by dialing d and p up, while a miner with a very low hashrate

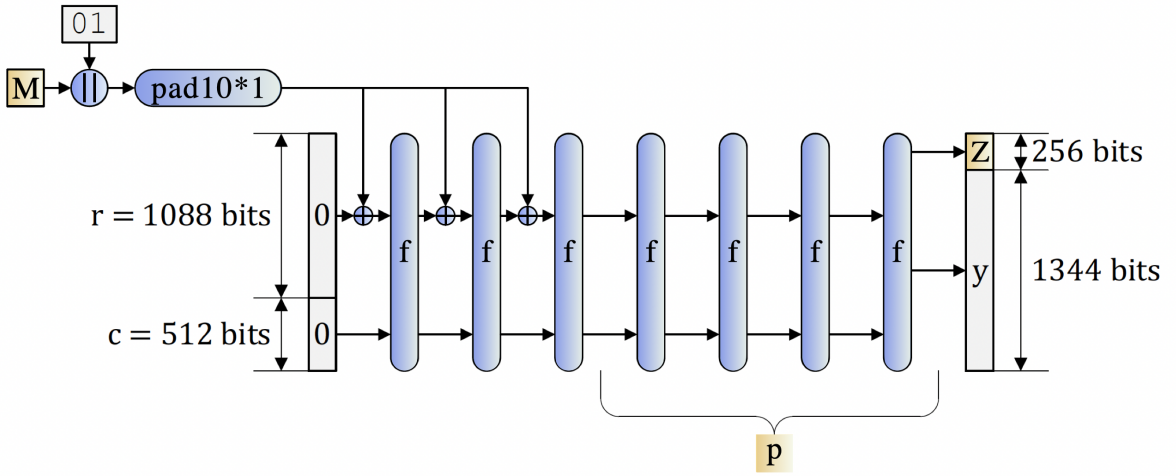

Figure 11: The hash function used in the mining process shown in Figure 10. We employ a modified Keccak sponge construction with a parameter p that refers to the number of additional iterations of the permutation function f (for more details see Bertoni et al. (2013)) beyond the absorbing phase. This modified version of Keccak retains the same security properties as SHA3-256, and is functionally equivalent to SHA3-256 with $p = 0$.

can be 'sped up' by dialing d and p down (or, in a relative sense, simply by dialing their peer miners up).

These dials are set (for each individual miner) through a peer-to-peer algorithm that determines the 'node temperature' of the miner in a given mining round and adjusts it via a simple heat equation, namely

$$u_k(t+1) = u_k(t) + \beta \sum_{i=0}^{n-1} \big(u_{c(k,i)}(t) - u_k(t)\big) \tag{2}$$

where $u_k(t)$ is the node temperature of the miner at time t, and miner k is connected with the miners $c(k,0)$, $c(k,1)$, ..., $c(k,n-1)$.

The node temperature refers to the hashrate of a miner, as determined by the final nonce reached by the end of the mining round. Miners are assumed to start with a nonce of zero and increment by one each time until they reach a value that satisfies the winning hash criterion as per the standard difficulty adjustment. The verification process conducted by peer miners ensures that miners do not gain a consistent advantage by gaming this system since the verification check includes (i) the value of the miner's determined hashrate relative to previous rounds (to prevent lying and/or sandbagging) as well as (ii) a minimum number of rounds that the miner must have participated (to ensure that the balancing/redistribution equation has time to do its job and to prevent a strategy of repeated re-connection leading to an anomalously-high initial hashrate).

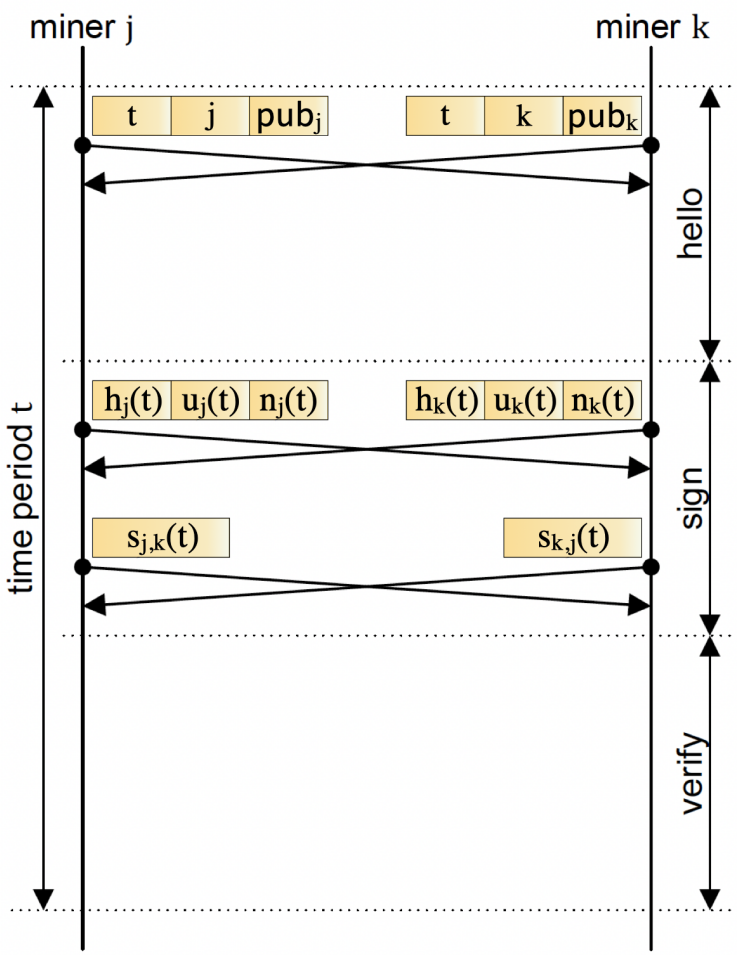

Figure 12: Three phases to compute the miner node temperatures for the next time period. In the first phase the miners check for reachability and exchange basic information. In the second phase the miners exchange temperature information and signatures. In the third phase the miners verify the received signatures.

Figure 12 shows the simple procedure for assessing the miner node temperature and subsequently determining the appropriate adjustment of the dials d and p. In this diagram t refers to the time period during which mining takes place, $\mathrm{pub_k}$ is the public key of miner k during time period t, $\mathrm{u_k}(t)$ is the node temperature of miner k during time period t, $\mathrm{n_k}(t)$ is the number of miners that miner k is connected to during time period t, and $\mathrm{s_{j,k}}(t)$ is the signature that miner j generated for miner k during time period t. Finally, $\mathrm{h_k}(t)$ is the hash of the IDs of the peer miners combined with the hash of the entire set of values already mentioned from the previous round.

The practical implementation of this in a real mining network is naturally more complex than this brief description implies; there are approaches that would constitute a gaming of this algorithm that can be mitigated in a variety of ways but not eliminated completely.

Economic and game theoretic arguments were also incorporated into the design in terms of incentives and return on investment (ROI) of technological advances. Thus, it is impossible to make a prediction for the degree of balancing that would be achieved in the network without specifying a number of assumptions; rather than do this, we present the approach *in potentia* as (i) reducing mining centralisation (ii) increasing the efficiency of the network (iii) reducing the competitive greed that drives extremely high levels of energy consumption. In particular, reducing mining centralization not only increases security but also increases fair distribution of the coin through the mining block reward. For more details and specifics, as well as a discussion on the various attack surfaces and how to mitigate them see Hobbs et al. (2021).

2.4.2 Feather checking

We employ a probabilistic approach to block verification in order to maintain distributed verification with a low barrier-to-entry even in the event of very large blocks. We refer to this probabilistic block verification as "feather checking". Each miner must only verify a subset of the Merkle tree (i.e., a "feather"). A feather represents a tree path from one or more transactions to the Merkle root, and in order to perform the verification a miner must be supplied with the transaction(s), the Merkle root, and any necessary sibling hashes along the Merkle tree path. Miners receive this information from the storage nodes.

The storage nodes can either provide all sibling hashes along the tree path from transaction i to the Merkle tree root, or a limited number. The former case is referred to as 'full sibling hash provision' and the latter case as 'limited sibling hash provision'. With full provision each miner is only required to verify a single transaction during each challenge, which they do by computing the hash of transaction i and the hashes of all nodes along the transaction-root path. The storage node provides therefore all sibling hashes of the nodes along the transaction-root path.

If the Merkle tree is not a perfect binary tree, e.g. if the number of transactions T in the block is $2^{d-1} < T < 2^d$, some sibling nodes along the transaction-root path may not exist. In this case the missing sibling node is replaced by the hash of the existing sibling node.

We briefly describe the limited sibling hash provision case for the storage node and derive the appropriate equations that determine the number of challenges n required for a given certainty C of validation. We assume that the sibling hashes are provided sequentially level-wise from level $\ell = 1$ down through $\ell = d - 1$, where d is the full Merkle tree depth and $d = 0$ at the Merkle tree root level. Thus, each additional sibling hash that is provided will bifurcate the tree at the next available level. When $s < d$ the miners are required to verify all other transactions in the subtree of which transaction i is a member, where the number of unverified transactions in the subtree will be 2^{d-s}.

For a perfect binary tree, for a given number of sibling hashes s provided by the storage nodes, the probability that any transaction gets challenged is,

$$P(i) = \frac{2^{d-s}}{T} \tag{3}$$

where T is the total number of transactions in the block. The depth of the full tree is limited by the number of transactions in the block and is given by $d = \texttt{ceil}(\log_2 T)$. Here, the ceiling function ensures that d is an integer and that the full tree has enough leaves to accommodate all transactions. The probability in equation 3 is always equal to or larger than $P(i) = 1/T$ that applies in the case of full sibling hash provision. In the case where $s \geq d$, equation 3 is equivalent to $P(i) = 1/T$ (however, in practice, s is not allowed to be greater than d).

If the tree is not a perfect binary tree, then incomplete subtrees will skew the probabilities of a subset of transactions being challenged relative to the transactions in the complete subtrees. In order to maintain probabilistic parity the incomplete subtree is filled with dummy transactions and/or duplicate existing transactions within the block. Using duplicate transactions only increases the probability that certain transactions are challenged above the baseline probability, which is not necessarily undesirable.

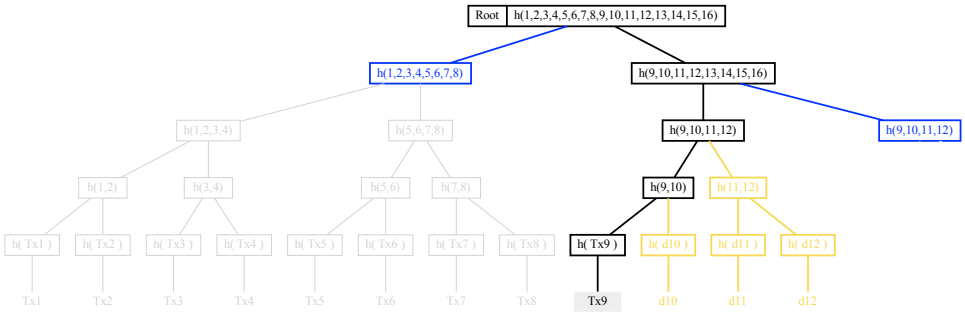

Figure 13: A Merkle tree in the case of limited sibling hash provision with $s = 2$, which is insufficient to compliment the full transaction-root path in a tree of depth $d = 4$. As a result, each miner must compute the hashes for a subtree with depth $d - s$ and a number of transactions equal to 2^{d-s}. The nodes for which the miner must compute hashes are marked in black, the sibling nodes for which the storage node provides the hashes are marked in blue, and the dummy/duplicate transactions and hashes are marked in gold. The challenged transaction in this case is `Tx9`.

Once the incomplete subtree has been filled, the probability of any given transaction (including the dummy or duplicate transactions) being challenged reverts to equation 3, with T now referring to the number of real transactions plus the number of dummy or duplicate transactions.

We derive an equation for the total number of challenges N required to achieve certainty C. In order to do this, we first need to determine the probability that a given transaction is *not* challenged over N total challenges. For any individual challenge in the general case, the probability $P(i')$ that transaction i is *not* selected is,

$$P(i') = 1 - P(i) = 1 - \frac{2^{d-s}}{T'}\,, \tag{4}$$

where T' is the total number of transactions in the block including dummies and duplicates, d is the depth of the Merkle tree, and s is the number of provided sibling hashes. In order to be certain that all transaction have this probability of being challenged,

$$C = \prod_{i=1}^{T'} \left(1 - \prod_{j=1}^{N} P(i')\right) = \left(1 - \left(1 - \frac{2^{d-s}}{T'}\right)^{N}\right)^{T'}\,, \tag{5}$$

where N is the total number of challenges provided to the mining pool, d is the full tree depth (where $d = 0$ at the root level), and s is the number of sibling hashes provided by the storage node. The number of challenges per miner is then,

$$n = \frac{1}{M} \frac{2^{d-s} \ln(1 - C^{1/T'})}{\ln\left(1 - 2^{d-s}/T'\right)}\,, \tag{6}$$

where $T' = T + 2^{d-s} - T \pmod{2^{d-s}}$ and $s \leq d$.

If s can be chosen based on the number of transactions T in the block, then an optimized approach is available. Figure 14 shows the optimal approach and resulting number of challenges per miner when s is a free parameter.

2.4.3 Raft consensus

While the mining consensus is reached through PoW, the compute and storage nodes employ Raft consensus (Ongaro & Ousterhout, 2014) to reach agreement on the block that is added to the chain, in terms of (i) creating the block (selection of transactions via the UNiCORN) (ii) matching the DRUID between Alice and Bob (iii) collecting UNiCORN contributions (iv) assigning block reward to the winner miner (v) monitoring current circulation of Zeno and Smart Data types and (vi) storing the block (adding it to the chain).

Raft is a consensus algorithm based around majority vote to replicate the state of the ledger and/or any decision across all compute and storage nodes. The number of compute and storage nodes is expected to be considerably smaller than the number of miners, and they make up a decentralized ring network with the decision making coming from the distributed mining network via the public contribution to the UNiCORN. For more detail the reader is referred to the original Raft paper (Ongaro & Ousterhout, 2014).

2.5 Memory management

We briefly outline here how the network protocol divides the management of the Zeno type, the Smart Data type, and the off-chain file (see Section 2.3) across the two types of memory architecture, namely stack memory (employed on the blockchain network and only manipulatable through inline assembly; contiguous) and heap memory (employed

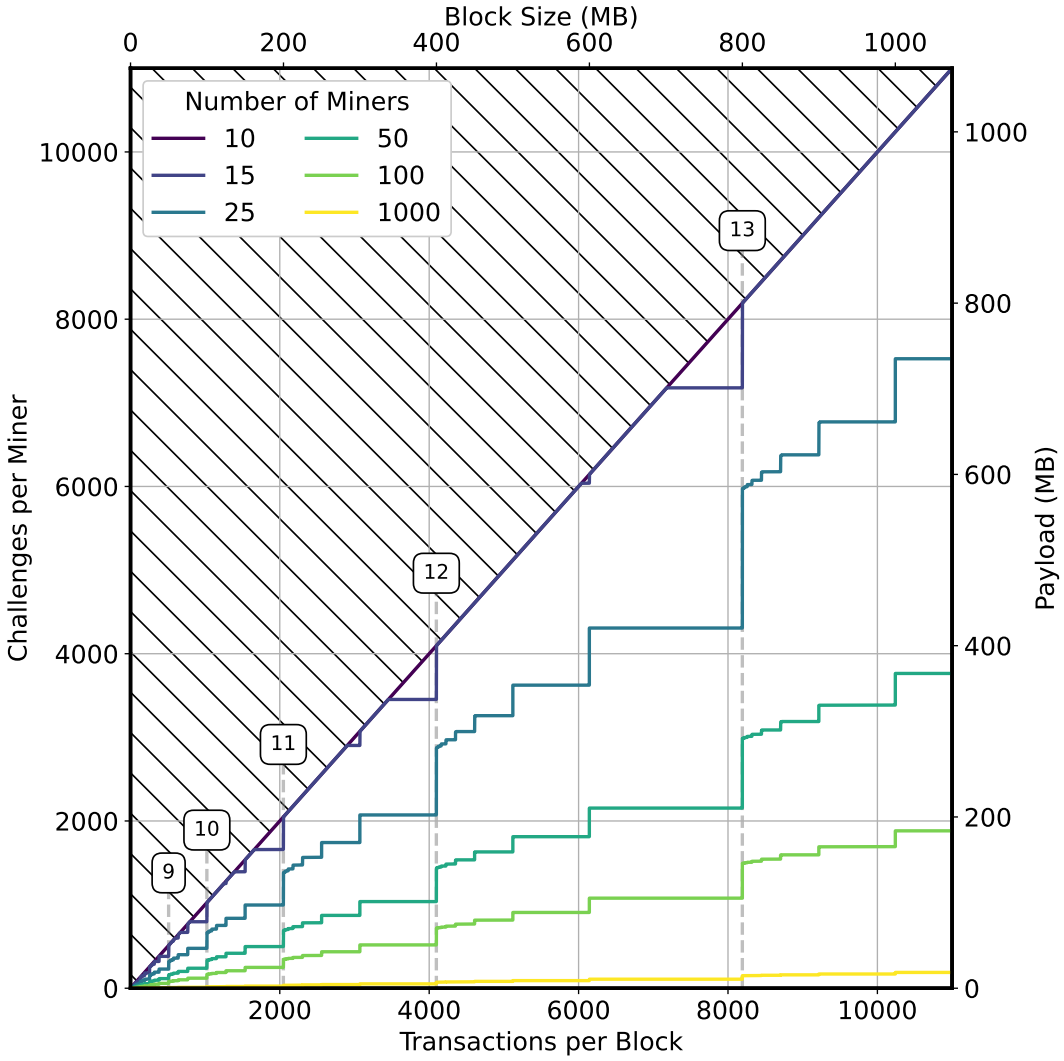

Figure 14: Optimal approach for feather checking. The number boxes above the diagonal mark the number of transactions at which the Merkle tree is a perfect binary tree; the number inside the box is the depth of the full tree (where $d = 0$ at the root level).

on individual machines as the location for the off-chain file, and allocated dynamically; unordered).

Figure 15 shows the design of the transaction objects in terms of their location in the memory architecture. The Zeno type (coin) and the Smart Data type (token) reside in stack memory and are processed through the consensus protocol that detailed in Section 2.4. The off-chain `*.ta` file component resides in heap memory and is managed via a stack-based script employing POWERSHELL OS-level logic in a virtual machine (VM) environment.

This design allows for contracts to reside on the blockchain (and be visible on the

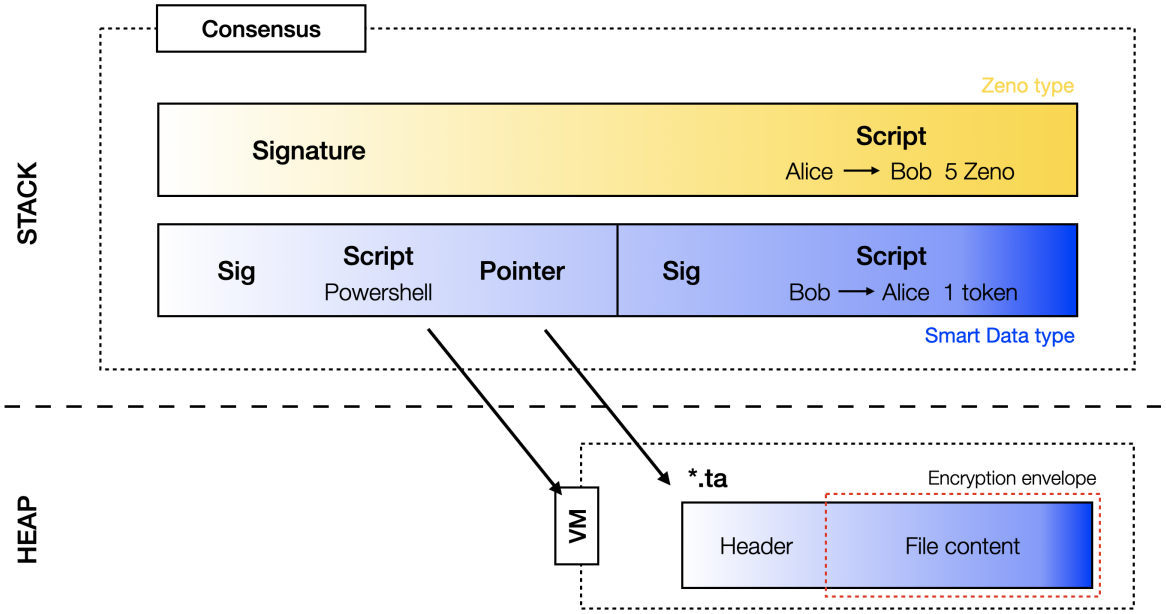

Figure 15: The design of the transaction types in the memory architecture of the blockchain and the user's machine that enables blockchain-based trade between payment and asset types as well as blockchain-based governance of off-chain files. The Zeno type and the Smart Data type I are purely stack-based, and execute in the layer 1 blockchain while the Smart Data type II is stack-based but contains a pointer to an off-chain heap resource via the DRS. This off-chain resource is contained within a VM on the user's machine, and a POWERSHELL script is employed to manage the off-chain resource from the stack.

blockchain) but for files, and the content of files, to be protected by a VM but managed from the blockchain via the stack-based contract. This means public contracts, but private data. In terms of memory management, the scripts in the Zeno type and the Smart Data type I are stack-management scripts while the script in the Smart Data type II constitutes a heap-management script.

2.5.1 Smart data

The setup described above for transaction types I & II constitutes what we term *Smart Data*. The Smart Data transaction type 1 is purely on-chain, while the Smart Data transaction type II is a hybrid of on- and off-chain, with the rights to the content of the off-chain file held on the blockchain but the content held on the user's machine. The term "Smart Data" references the fact that this transaction type is able to contain programmable logic, that can be operated on by a Smart Data contract.

The pointer to the off-chain resource effects blockchain-governance of files. We term it the

data rights signature (DRS), and its composition is described in Figure 16. The original file M is passed through a series of encoding and encryption steps in order to generate four hash values: (i) the hash of the file (ii) the hash of the perceptual hash of the file (iii) the hash of the encryption key and (iv) the hash of the encrypted and encoded file. This four hash structure solves for the problem of *integrity* – how does a machine determine the validity of the asset without seeing the content of the asset? Through the four hash structure the persistence of the integrity of the asset can be confirmed without needing to read the file. The buyer can see the asset has persisted and that it is under blockchain governance.

To these four hashes is added the encoding scheme that imparts a unique 'digital DNA' to the file, ensuring that the copy/edit problem[5] of digital files is circumvented, by imparting a uniqueness to the file at the binary level that can be tracked on the blockchain.

3 Legal assessment

Authors' note: This section contains a legal assessment of the digital ownership concept, developed by our legal division, as the complement to the technological solutions outlined in the rest of this paper.

The Zenotta digital system enables a peer-to-peer electronic trade system that utilizes digital data. We define "trade" as the activity of buying, selling, or exchanging, goods and/or services. From a legal perspective, this requires that digital data be ownable and commoditized within the applicable legal framework(s).

Digital ownability is distinctly realized through several aspects not currently available outside of the Zenotta digital system. On a high level, the elements of ownership can be boiled down to uniqueness, control and protection (see Section 1) which holds true in terms of legally validating and proving ownership via specific, and legal, rights. We define "ownership" as the right to defend the owned asset against any third parties as long as a third party cannot evidence a better right. This right is not one single defensive right, but fans out into a bundle of rights that, taken together, ultimately describe the full scope of ownership. These rights form the basis of ownership more broadly, and while each jurisdiction may define and limit them differently, the basis of such rights can be agreed upon globally.

We define the bundle of rights (shown in graphical form in Figure 17) to consist of the following:

1. *The right to control (possession)* – i.e., the right to be put in exclusive control of a thing and the right to remain in such control;

2. *The right to use* – i.e., the owner's personal use and enjoyment of a thing owned;

[5] the inability to easily distinguish a copy of a file from the original, or to easily identify if a file has been edited

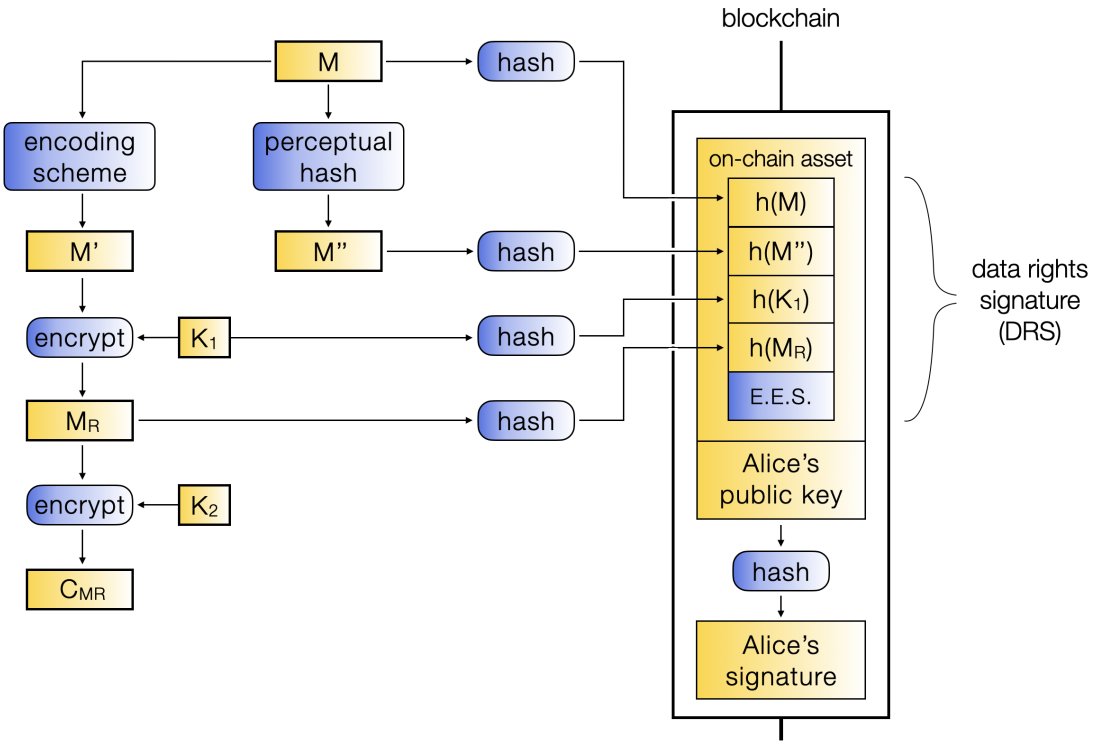

Figure 16: Providing ownership of digital files through blockchain governance. The DRS is the pointer that allows for the script in the Smart Data transaction type II to manage an off-chain resource such as a file. In this diagram M refers to the 'message', namely the binary of the off-chain file, which is passed through an encoding scheme to impart uniqueness into the binary of the file M'. This is subsequently encrypted (output M_R) with a symmetric key K_1 to protect the content from any third-party and encrypted again with key K_2 (output C_{MR}) to further protect the content from any previous owners. As a secondary step M is passed through through a perceptual hash to generate M", which retains enough of the features of the original file to detect piracy of the original file. M, M", K_1 and M_R are hashed and placed on the blockchain, together with the encrypted encoding scheme (E.E.S.) from the first step, making up the DRS that allows for ownership of off-chain assets. The DRS is signed for on the blockchain via the standard public/private key asymmetric cryptography in pay-to-public-key-hash (P2PKH) mode.

3. *The right to manage* – i.e., the right to decide how and by whom the thing owned shall be used;

4. *The right to income* – i.e., to deriving a revenue from a thing from personal use or allowing others to use it for reward;

5. *The right to capitalize* – i.e., the power to gain advantage from the asset and the liberty to consume, waste or destroy the whole or part of it;

6. *The right to legal certainty* – i.e., immunity from expropriation (apart from bankruptcy and execution of debt);

7. *The right to transferability* – i.e., that the interest can be transmitted to the holder's successors ad infinitum;

8. *The right to execute intellectual property rights*;

9. *The absence of a time limit* – i.e., the condition that the holder's interest is not due to determine on a fixed date or on the occurrence of some contingency;

10. *The prohibition of harmful use* – i.e., the condition that uses of the asset harmful to other members of society are forbidden.

This particular list was drawn from the seminal paper "Ownership" (Honoré, 1961), which drew on the bundle of rights concept as introduced by the American Institute of Law in the 1930s in order to compile a list of elements of ownership that can be combined (but are not required in total) to form a bundle of rights that can be utilized in law (initially, real estate law). It is the interoperability of these rights, meaning no solitary right, or particular accumulation of rights, that is the key to ownership. It is how these rights coalescence and operate together that form the basis of ownership and can be separated or kept whole to determine ownership in title as well as how an asset may be owned by one or many subjects all with different, or distinct, rights.

The Zenotta digital system fulfills the requirements necessary for evidencing, and further, transacting this bundle of rights through several technical aspects already discussed. Here we provide a brief legal overview of how these technical aspects specifically impart valid legal conclusions:

Uniqueness:

The realization of Smart Data as well as the dual double entry and ability to provide a receipt of transaction ensures the uniqueness of every single digital asset within the system. Each digital asset can be converted to a Smart Data format, making it unique, rival and traceable via its distinct container, acting as a deed of title for the owner while also evidencing a specific DNA of the file itself. Only the genuine original can be authenticated, along with its history. This is critical in evidencing legal ownership

through rivalry and title at the asset level, currently not possible through current NFT or other blockchain technology.

Further, the immutable blockchain ledger records the Smart Data's history and offers an accounting of its ownership legacy as well as its value via the dual double entry ledger. A receipt is possible for an owner to evidence in case of dispute, challenge, or contest to its transaction, worth, or ownership rights. The fulfillment of this kind of inherent record-keeping and digital DNA within a file itself is what unlocks a legally compliant type of digital, data, and content-based ownership on the blockchain. The assurance of two-way trade, as described above, realizes a ledger-based economy within the digital world in accordance with contractual legal frameworks generally.

Control:

Due to the aforementioned record-keeping aspects and identification of data and files, an owner can truly control and monitor their files and their contents as their own. The immutable ledger does not merely point to a location at which an owner may be subject to 'rug pulls' or the intellectual property (IP) rights of a third party, but can directly hold and control the file itself, including the contents, encompassing every layer of potential IP rights. Significantly, this also means that certain rights can be held back, further sold, or otherwise divided contractually, with a blockchain ledger to record such contracts linked back to a unique and specific asset that is always traceable and protectable.

This control can extend to the farthest limits of the division of an asset and its rights, or to the assignment of rights to other third parties, for example, akin to licensing. The bundle of rights may act separately, or coalescence and operate together to form the basis of ownership. Whether separated or kept whole, the Zenotta digital system is capable of recording and maintaining the suite of rights to evidence how an asset is (or can be) owned by one or many subjects all with different, or distinct, rights.

Protection:

Protection of a digital asset and most meaningfully, its actual contents, can be realized through the development of smart data. Not only can a transaction, and thereby an ownership log, be evidenced by the blockchain and/or a receipt, but the file itself will specifically respond to the owner via its programming logic. An owner may choose to safeguard the file and never let another individual see the contents, utilizing their keys to protect the file and their privacy zealously. Or the owner my enforce their ownership through challenging the use of another by evidencing their specific ownership rights, such as in case of a copyright infringement. The possibilities of protection are uniquely evidenced via the technology identifying both owner and subject, and irrefutably proves a record through time of the bundle of rights and any divisions thereof. The possibility to protect not only an individual file, but its specific contents, is a wholly new solution to enacting digital trade and data ownership.

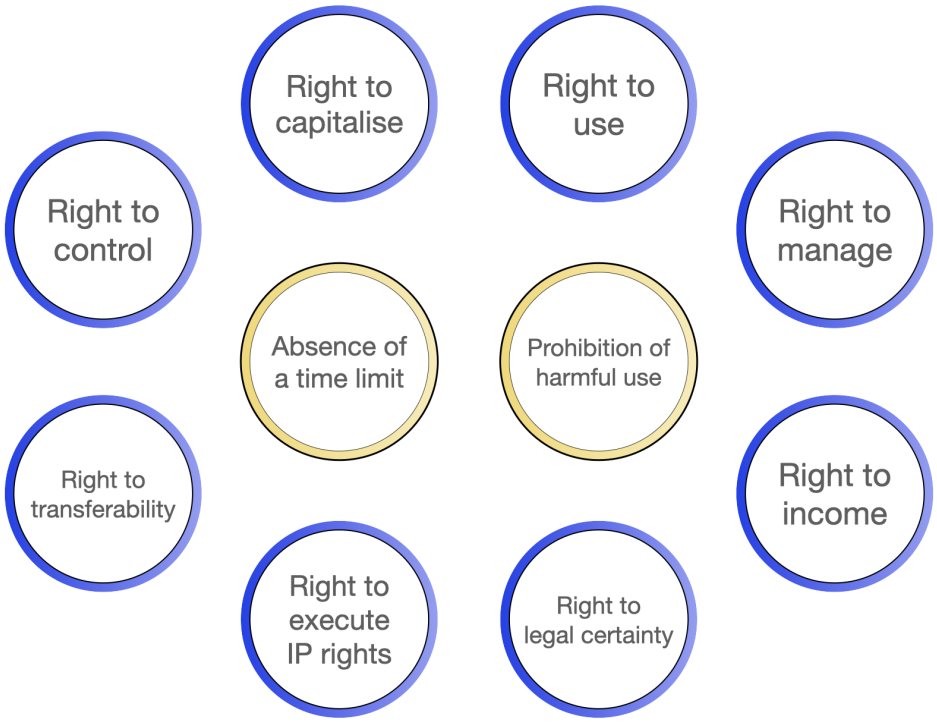

Figure 17: The bundle of rights that should be executable on a digital level to achieve digital ownership, drawn from the seminal paper titled 'Ownership' (Honoré, 1961).

This global understanding of the core concept of ownership, specifically that ownership is a bundle of rights and the protection of such, further gives way to the transactability of digital assets, pending the agreement and evidence of their ownability. Following the summary of how the technical aspects impart legally valid ownership to digital assets, we turn to the concept of digital trade.

The possibility to manage rights goes far beyond the bundle of ownership rights and in particular also covers the contractual rights that are attached to persons and not to property. The legal framework enabling and underlying electronic peer-to-peer digital trade must be grounded in the concept of the freedom of contract, i.e. the ability to execute the common will of the parties. As the Zenotta digital system will run its own version of electronic smart data contracts, the parties shall be free to cover all facets of a legal relationship and transaction. However, if and when controversies, claims outside of the contractual terms, or gaps arise, the Zenotta default legal framework made up of the system's required terms and conditions and the national or local laws each user is subject to shall apply. This legal framework defaults to and respects the mandatory law

applicable to any and every country, based on jurisdiction and/or contract type/subject matter.

Through the Zenotta terms of service, users are subject to "rules of engagement" via the required acceptance to participate in the system. These rules enable the enforceability of different concepts and limits of ownership. The parties are able to define and agree to the contractual terms and the expression of will as the functional equivalent to any current goods & services transactions. Absent the specific expression of critical terms such as choice of law or jurisdiction in which to bring a claim, the default legal framework clarifies and resolves these issues as a gap filler. The safety net provided by the terms of service not only includes general prohibitions on illegal acts, for example human trafficking transactions, but further extends to subjecting each user to the mandatory laws applicable to themselves, their counter party, or their transaction based on jurisdiction and/or industry, subject matter, or transaction type.

While subject to the applicable mandatory law, again the parties maintain their freedom to contract and their expression of will is realized through the required cosigning of a two-way transaction on the dual double entry ledger via the DRUID. As further detailed in Section 2.3.1, both parties in a transaction need to enter into a DRUID-based transaction process. The mutual assent of the parties to enter into, and as such approve of, the transaction is processed by the compute nodes, whereby it is considered as a single trade, allowing the contract to be concluded. Due to the requirement of each party to consent to and take action on behalf of the execution of the contract there can be no discrepancy regarding the mutual assent, or meeting of the minds, of the parties concluding the contract. The compute nodes act as the transaction's notary, registering each legally-binding expression of will by each party. While other ambiguities or claims may still arise, the consent and agreement to the contract itself by the parties is indisputable. Their intent and will to give effect to the contract cannot be challenged as clear and unequivocal actions are required by each party to bring it to execution.

Ensuring recognition of the bundle of ownership rights, contractual rights, and mandatory law, the default legal framework reverts claims and issues outside of the terms of service to the laws and courts of Switzerland, whereby the principle of functional equivalence shall treat the digital transaction as any other transaction in practice. This provides stability, foreseeability and reliability for all users and transactions within the Zenotta digital system. This concept of digital ownership as well as the underlying legal framework for trade are indispensable requirements for improving legal certainty in trade with digital data.

4 Conclusion

Traditional banking payment networks use centralised, trusted, and persistent nodes to function as timestamp servers, and "account-based" data structures to handle public/private key operations – along with certification authorities for the purpose of sending bank countersigned promissory notes from a client to a merchant. Conversely, blockchain-based

protocols use distributed, untrusted, non-persistent nodes employing consensus-based cryptographic operations such as Proof-of-Work (PoW) to timestamp transactions, and unspent-transaction-output (UTXO) data structures to handle ownership, which together with the associated public/private key operations allow for the movement or onspending of on-chain assets.

The network protocol presented in this whitepaper is not a mixture of banking and blockchain systems, but in fact an entirely new protocol that uses a three-tiered network made up of *persistent untrusted decentralized nodes* and *non-persistent untrusted distributed nodes* to deliver a general protocol for the handling of electronic coins and digital assets that has significant benefits over both blockchains and conventional payment networks. Notably, the network protocol can trade objects like electronic coins or receipts for digital assets, goods, or services. The trade is atomic in that either both the merchant receives the digital payment **and** the client receives the rights to the digital asset, in the same moment, or neither do.

New features offered by this protocol include autonomous and programmable service levels, governance without control, the possibility for atomic trade, better transaction throughput, low latency, reduced energy consumption and domain based payments.

The use of a two-way ledger (specifically, dual double entry UTXO blockchain ledger) with a notarized receipt constitutes a "perfect transaction". Goods and services are not merely gifted from a machine-readable and executable perspective, but rather traded. This allows for a pricing mechanism that is native to the blockchain, with the price signaled on the machine layer, in a 'just-in-time' (JIT) manner. The application of a governing on-chain signature, whether for off-chain files or for on-chain assets, that exists as the other side of the two-way trade atomically on the two-way ledger constitutes a form of perfected ownership for NFTs.

An extremely powerful concept that the blockchain world is arguably missing is that of governance. The somewhat fantastical picture of blockchains existing separately from the laws and regulations of the world around us is a compelling narrative, but not practical or even desirable. Integration of blockchain technology with legal systems is vital for it to move beyond fringe uses and hype and make a real difference on a global scale.

At first glance, however, these concepts seem at odds with each other. Decentralization, third-party disintermediation, and trustlessness are the core principles of blockchain, and legal systems would seem to re-introduce undesirable properties that threaten these principles. Framing the problem correctly allows us to reach a middle ground where the principles of blockchain are preserved but participants can wield legal machinery to protect their rights and access legal recourse.

This framing, crucially, must be from the perspective of the user. Ultimately, trade happens as the result of users wishing to exchange or purchase goods & services. In order to keep the ideals listed above for blockchain technology, any legal integration must come primarily from user-centric implementation of e.g., sanction lists. An essential feature here is the UNiCORN approach that allows for the 'service-providing nodes' to remain

fully impartial and prevents central control. Decision making on the part of the compute nodes is generated from (in part) the distributed mining network, as is the validation of transactions as obeying the protocol rules.

The digital world as a parallel global societal structure is inevitable. The only question is whether it will be based on technology that preserves all of the fundamental features of the real world that societies have developed and optimised over centuries — features such as ownership, rights, privacy, governance, self-autonomy, and economics, or on technology that falls short of this vision.

5 Acknowledgments

The authors acknowledge and thank the many advisors, investors, and friends that have been an essential part of the development of Zenotta's thinking and technology. We thank in particular the software development team: Nikita Baksalyar, Jean-Philippe Dufraigne, Anton Troskie, Jean Thysse, Chirag Palsania, Myrin Naidoo, Samay Thakkar, Pradyuman Zala and Robin Plojoux.

References

Back A., 2002, https://www.researchgate.net/publication/2482110_Hashcash_-_A_Denial_of_Service_Counter-Measure

Barry R., Kessler A., Hobbs A., 2021, A method and apparatus for a space-time efficient value transfer network

Bartoletti M., Zunino R., 2018, CCS '18, 83–100, Association for Computing Machinery, New York, NY, USA

Bertoni G., Daemen J., Peeters M., Van Assche G., 2013, in Advances in Cryptology – EUROCRYPT 2013, edited by T. Johansson, P. Q. Nguyen, 313–314, Springer Berlin Heidelberg, Berlin, Heidelberg

Boneh D., Bonneau J., Bünz B., Fisch B., 2018, Verifiable Delay Functions: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19â23, 2018, Proceedings, Part I, 757–788

Brewer E., 2012, Computer, 45, 23

Chaum D., van Antwerpen H., 1990, in Advances in Cryptology — CRYPTO' 89 Proceedings, edited by G. Brassard, 212–216, Springer New York, New York, NY

Chohan U. W., Chohan U. W., 2021, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3090174

Dwork C., Naor M., 1992, in Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '92, 139–147, Springer-Verlag, Berlin, Heidelberg

Friedman M., 1991, Working Papers in Economics, https://miltonfriedman.hoover.org/internal/media/dispatcher/215061/full

Hobbs A., Kessler A., De Moliner R., 2021, https://www.zenotta.io

Honoré A. M., 1961, Oxford University Press, https://www.jstor.org/stable/1119926

Jones D., 2020, Shell of an Idea: The Untold History of PowerShell, Don Gannon-Jones, https://books.google.ch/books?id=o-9bzgEACAAJ

Lamport L., Shostak R., Pease M., 1982, ACM Transactions on Programming Languages and Systems (TOPLAS), 4, 3, 382

Lenstra A. K., Wesolowski B., 2015, A random zoo: sloth, unicorn, and trx, Cryptology ePrint Archive, Report 2015/366, https://ia.cr/2015/366

Ongaro D., Ousterhout J., 2014, USENIX proceedings, https://raft.github.io/raft.pdf

O'Dwyer R., 2018, Journal of Cultural Economy, 12, 1

PlanB, 2019, Modeling Bitcoin Value with Scarcity, Medium article, https://medium.com/@100trillionUSD/modeling-bitcoins-value-with-scarcity-91fa0fc03e25

Szabo N., 2005, Bitgold, Post on blogger website