IEEE Access
Multidisciplinary Rapid Review Open Access Journal

Received 27 June 2022, accepted 15 July 2022, date of publication 21 July 2022, date of current version 27 July 2022.
Digital Object Identifier 10.1109/ACCESS.2022.3192837

RESEARCH ARTICLE

An Ultra-Scalable Blockchain Platform for Universal Asset Tokenization: Design and Implementation

AHTO BULDAS[1,2], (Member, IEEE), DIRK DRAHEIM[3], (Member, IEEE), MIKE GAULT[4], RISTO LAANOJA[1,2], TAKEHIKO NAGUMO[5,6], MÄRT SAAREPERA[2,7], SYED ATTIQUE SHAH[8], (Member, IEEE), JOOSEP SIMM[2], JAMIE STEINER[2], TANEL TAMMET[9], AND AHTO TRUU[2]

[1]Centre for Digital Forensics and Cyber Security, Tallinn University of Technology, 12618 Tallinn, Estonia
[2]Guardtime, 11316 Tallinn, Estonia
[3]Information Systems Group, Tallinn University of Technology, 12618 Tallinn, Estonia
[4]Guardtime, 1006 Lausanne, Switzerland
[5]Mitsubishi UFJ Research and Consulting, Minato, Tokyo 105-8501, Japan
[6]Graduate School of Management, Kyoto University, Kyoto 606-8501, Japan
[7]Martest Research, 11317 Tallinn, Estonia
[8]School of Computing and Digital Technology, Birmingham City University, Birmingham B4 7XG, U.K.
[9]Applied Artificial Intelligence Group, Tallinn University of Technology, 12618 Tallinn, Estonia

Corresponding authors: Ahto Buldas (ahto.buldas@taltech.ee) and Dirk Draheim (dirk.draheim@taltech.ee)

ABSTRACT In this paper, we explain the Alphabill family of technologies that addresses both unlimited scalability and unrestricted adaptivity. We deliver a sharded blockchain technology with unlimited scalability, called KSI Cash, which is based on a new form of electronic money scheme, the bill scheme. We present performance tests of KSI Cash that we have conducted with the European Central Bank in order to assess the feasibility of a digital euro. We show the system operating with 100 million wallets and 15,000 transactions per second (under simulation of realistic usage); having an estimated carbon footprint of 0.0001g CO2 per transaction. Furthermore, we show the system operating with up to 2 million payment orders per second, an equivalent of more than 300,000 transactions per second (in a laboratory setting with the central components of KSI Cash), scaling linearly in the number of shards. We explain, in detail, the key concepts that unlock this performance (i.e., the concepts of the bill money scheme). The results provide evidence that the scalability of our technology is unlimited in both permissioned and permissionless scenarios, resulting into the Alphabill Money technology. Next, we contribute the architecture of a universal tokenization platform that allows for universal asset tokenization, transfer and exchange as a global medium of exchange, called Alphabill platform. We reveal the crucial conceptual and technical contributions of the platform's architecture, including the data structures of KSI Cash and Alphabill Money, the dust collection solution of Alphabill Money, and the atomic swap solution of the Alphabill platform.

INDEX TERMS Blockchain, Bitcoin, digital euro, decentralized finance, DeFi, Web3, Alphabill.

I. INTRODUCTION

Blockchain technology has received tremendous attention from academia, industry, politics and media alike in the last decade. Since the introduction of blockchain technology with the cryptocurrency Bitcoin [1] in 2009, we have seen a plethora of cryptocurrencies and blockchain-based solutions. In connection with that, we have seen the emergence of a series of extended blockchain-based visions such as smart contracts [2], decentralized finance (DeFi) [3], and, most recently, Web3.

The prerequisite for any valuable blockchain-based vision to be turned into reality is uncapped scalability. With an average transaction rate of 7 transactions per second, Bitcoin

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

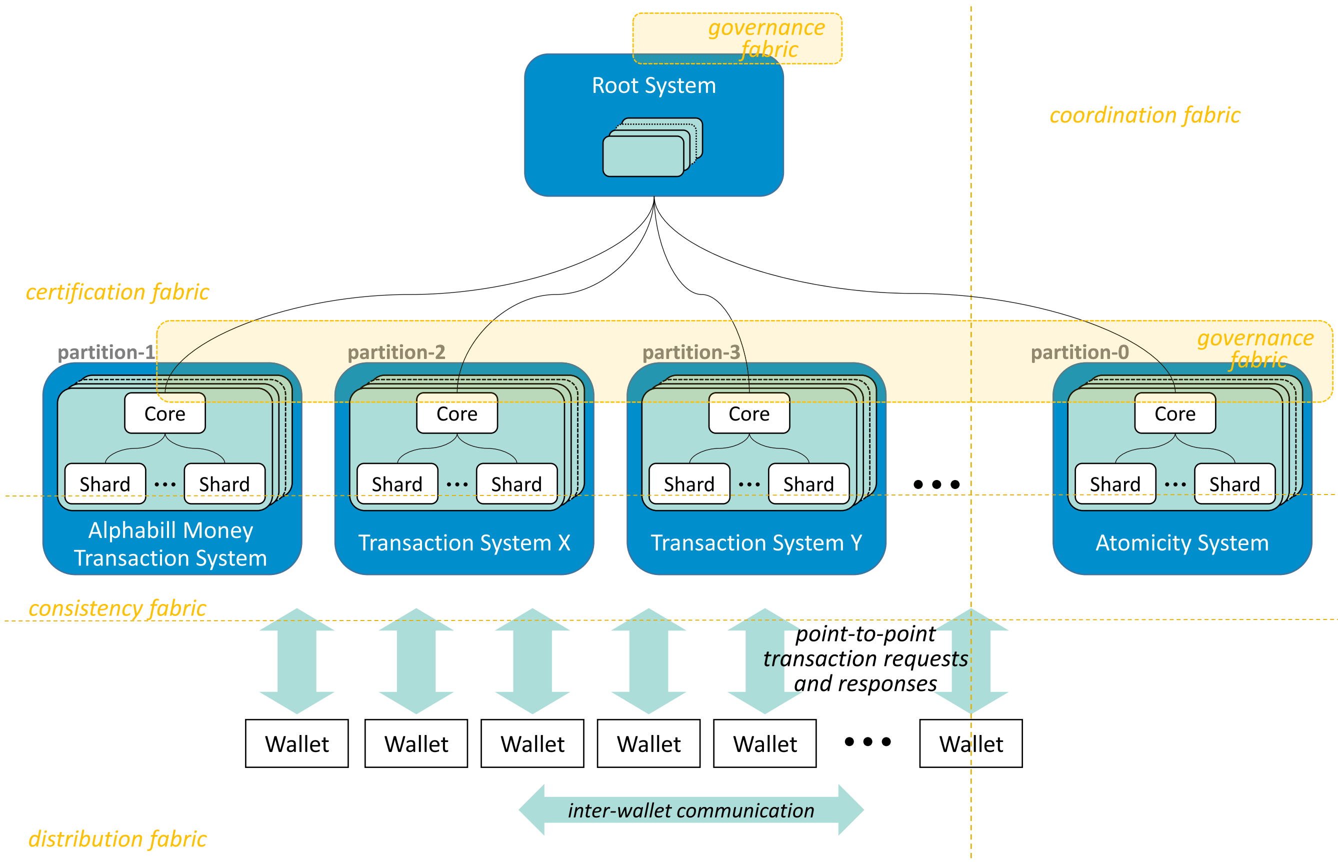


FIGURE 1. **Alphabill high-level architecture.**

is far from the transaction rates of today's established payment systems such as VisaNet, which has a peak performance of 24,000 transactions per second according to [4]. According to [5], there has been a total of approx. 435 billion annual USD transactions in 2017 via the three biggest players in the field alone, i.e., Visa (300 billion), MasterCard (75 billion) and UnionPay (60 billion), which amounts to an averaged transaction rate of approx. 13,800 transactions per second. Therefore, blockchain technologies also need to be capable of thousands of transactions per second. If micropayment visions and related visions such as the Internet of Things (IoT) are to be enabled by blockchain technology [6]–[12], much higher transaction rates will be needed already in the near future, i.e., in the range of hundreds of thousands of transactions per second and even way beyond. Top-level research projects have been launched to study ultra-scalable blockchain technology [13]–[15]. Also, it is often overlooked that shorter and shorter transaction durations (or at least maintaining transaction durations in regard to current transaction duration expectations) will become increasingly important against the background of these developments.

In addition, existing tokenization platforms are limited as they either oversimplify today's established monetary system [16] or neglect the reality of today's institutional stack [16], [17] and (inter-)/organizational settings [18] or both. Typically, new tokenization solutions proclaim disruptiveness in their field (at different levels: organizations, business domains, the whole monetary system, or the whole society). Each of the single tokenization solutions is, however, typically not prepared for adoption, i.e., vulnerable to changes in its environment. Therefore, having more and more specialized tokenization solutions will no longer advance the field significantly. What is needed instead, is an adaptive, heterogeneous tokenization platform that allows for launching and coordinating individual blockchain technologies in a systematic and at the same time innovation-friendly manner, i.e., in way that allows the hosted blockchain technologies to show not only business-related innovations but also technological innovations.

In this paper, we present the Alphabill family of technologies that addresses both *unlimited scalability* as well as *unrestricted adaptivity* by posing the following research questions:

1) How to design a blockchain technology that completely fulfills today's as well as future performance demands?
2) How to design a universal asset tokenization platform that allows for a maximum of both business-related and technological innovations?

In order to solve these questions, we make the following contributions:

- *KSI Cash.* We deliver a blockchain technology with essentially unlimited scalability and performance, called KSI Cash.[1] KSI Cash has been developed for experiments of the European Central Bank and a group of eight central banks from the Eurosystem[2] (Estonia, Germany, Greece, Ireland, Italy, Latvia, Spain, the Netherlands) to assess the feasibility of a digital euro [19]. KSI Cash has been developed as a joint design science effort of the Estonian Central Bank (Eesti Pank) and Guardtime[3] [20], [21]. The scalability of the Alphabill platform is founded on the scalability of its individual partitions. KSI Cash is part of this foundation, and a part of the Alphabill family of technologies.
- *Exhaustive KSI Cash Performance Evaluation.* We present diverse performance tests of KSI Cash that we have conducted with the European Central Bank and the said group of eight central banks [19]. We show the system operating with 100 million wallets and 15 thousand transactions per second under simulation of realistic usage. Furthermore, we show the system operating with up to 2 million payment orders per second (an equivalent of more than 300,000 transactions per second in our technology) in a laboratory setting that runs with the central components of KSI Cash. These tests show that the technology scales linearly in terms of the number of deployed shards. Furthermore, we are able to estimate the carbon footprint of KSI Cash as 0.0001g CO2 per transaction, again under the realistic usage scenario (as compared to Bitcoin, which amounts to much more than 100 kg CO2 per payment transaction according to several existing studies).
- *Bill Scheme.* We explain, in detail, the key concepts that unlock the performance of KSI Cash, i.e., a new form of electronic money scheme, the *bill scheme* [21], [22], which achieves full decomposability by circumvention of monetary splits and joins. The analysis indicates that the performance of our technology is unlimited for practical purposes, in both permissioned and permissionless scenarios.
- *Alphabill Platform Architecture.* We provide the architecture of the Alphabill platform, see Fig. 1. The Alphabill platform enables universal asset tokenization, transfer and exchange as a global medium of exchange. The platform allows users to launch an arbitrary number of new so-called *partitions*. Each partition implements an individual token and corresponding transaction system. The Alphabill platform provides the necessary innovations and protocols, along with respective languages, libraries and toolkits, to implement and launch partitions in such a way that individual partitions as well as their coordination and interaction show unlimited scalability. We specify: the platform's elements, asset presentations, ledger certificates, transaction orders, sharding schemes and transaction system specifications.
- *Alphabill Money.* We introduce the Alphabill Money technology, which forms the native money of the Alphabill platform. With Alphabill Money, we contribute an *extended bill scheme* that balances monetary splits and limited forms of monetary joins with a *dust collection* solution so that scalability is preserved. We specify the data structures of Alphabill Money.
- *Alphabill Money Dust Collection.* We specify the dust collection solution of Alphabill Money. We introduce and elaborate the notion of bill swap scenario and specify the dust collection process.
- *Alphabill Platform Atomicity Partition.* A key contribution is the design of a scalable *multi-asset atomic swap* solution (the atomicity system in Fig. 1) that enables cross-partition transactions that are needed, e.g., for financial transactions (parallel synchronized transfers of monetary and non-monetary assets). We specify the solution by defining the data structures of the atomicity system and defining a novel 3-phase-commit protocol.

The Alphabill platform will be published as open-source software on GitHub.

We proceed as follows. In Section II, we discuss related work. In Section III, we explain the architecture of the Alphabill platform and its design principles. In Section IV, we describe the technical principles and data structures of the central bank digital currency KSI Cash. In Section V, we explain the design of the KSI Cash test bench and present the outcomes of a series of diverse test runs. In Section VI, we delve into the details of a series of essential Alphabill platform components, i.e., fundamental design elements, the Alphabill Money partition, and the Alphabill atomicity partition. In Section VII, we discuss future directions and future work. We finish the paper with a conclusion in Section VIII.

II. RELATED WORK

Scalability is a central concern for blockchain technology. In [23], the authors provide a survey on relevant approaches to improve the scalability of the Bitcoin network, including tuning Bitcoin protocol parameters, basic off-chain payment channels, duplex micropayment channels [24], Bitcoin lightning channels [25], and off-chain payment networks. In [26], the authors provide an overview of scalability of blockchain systems and technologies for scalable blockchain systems. They discuss tuning of blockchain parameters, off-chain transactions, decoupling blockchain management from execution, and sharding. They identify Elastico [27] and OmniLedger [28], [29] as sharding blockchain technologies.

[1] KSI in KSI Cash is not an acronym and stands for nothing. The name KSI Cash has been chosen by Guardtime for internal reasons.

[2] https://www.ecb.europa.eu/ecb/orga/escb/

[3] https://guardtime.com/ksi-cash

A. BLOCKCHAIN SHARDING APPROACHES

1) SURVEYS RELATED TO SHARDED BLOCKCHAIN TECHNOLOGY

Two most recent surveys on sharding in blockchains are provided by [30] and [31]. The survey in [30] provides an analysis of sharded blockchain technologies including aspects such as intra-consensus settings (measures to reach blockchain consensus [32], [33] within a single shard), design of cross-shard atomicity (i.e., measures to guarantee that transactions that span more than one shard are conducted either as a whole or not at all [34]), corresponding overhead, latency and theoretical throughput. The paper reviews Elastico [27], Chainspace [35], [36], OmniLedger [28], [29], RapidChain [37], and Monoxide [38] and investigates Ethereum 2.0 [39]. The survey in [31] is a survey on scaling blockchains, however, it focuses on sharding [31]: ''In particular, we focus on sharding as a promising first layer solution to the scalability issue.'' The survey in [31] analyses technologies in terms of committee formation and intra-committee consensus. In addition to the work analyzed by [30], the survey in [31] reviews [40], SSChain [41] and Ostraka [42] and, furthermore, considers the technology proposals Zilliqa,[4] Harmony,[5] Logos,[6] and Stegos.[7]

2) SHARDED BLOCKCHAIN TECHNOLOGIES

Elastico [27] is an early sharded, proof-of-work-based blockchain technology that did not yet provide cross-shard atomicity control. Elastico is based on UTXOs (unspent transaction outputs), which form the electronic money scheme that has been introduced originally by the Bitcoin blockchain [1] (UTXOs are smallest units of account consumed as input to transactions and produced as output of transactions [22], [44]).

OmniLedger [28], [29] is a permissionless, sharded, UTXO-based blockchain technology that relies on a cross-shard, client-driven two-phase commit protocol (called Atoms) for the needed atomicity control. OmniLedger builds on the consensus protocol ByzCoin [45] by introducing sharding to it, resulting into the OmniLedger's consensus protocol ByzCoinX. Furthermore, OmniLedger builds on Hybrid Consensus [46], [47] and Elastico [27]. Hybrid consensus [46], [47] relies on proof of work and consumes a permissioned BFT (Byzantine fault tolerance) [48] protocol as a building block. To support power efficiency by proof-of-stake rather than proof-of-work consensus, OmniLedger builds on Ouroboros [49] and Algorand [50]. OmniLedger uses the RandHound protocol [51] targeting a scalable and at the same time bias-resistant sampling of representative validators.

Chainspace [35], [36] is a sharded blockchain technology that supports user-defined smart contracts ''operating on controlled objects'' [35], [36]. The system exploits an implementation of PBFT (Practical Byzantine Fault Tolerance) [52], i.e., MOD-SMART (Modular State Machine Replication) [53], for intra-shard consensus. Cross-shard consistency is guaranteed by a distributed commit protocol called S-BAC (Sharded Byzantine Atomic Commit). As opposed to OmniLedger's client-driven commit protocol, in S-BAC, the entire shard acts as a coordinator – rather than a single untrusted client.

'Ethereum upgrades'[8] represents the current roadmap of Ethereum [54] as the successor of Ethereum 2.0 [39]. From its beginning, Ethereum was a proof-of-work blockchain. The roadmap aims to bring a proof-of-stake consensus and sharding to Ethereum via a component called *beacon chain*,[9] which has already been described as part of Ethereum 2.0 [39]. ''The Eth2 execution sharding (formerly known as Eth2 Phase 2) consists of one *beacon chain* and multiple *shard chains*. A shard chain is a sharded blockchain, and a beacon chain is a blockchain that manages the shard chain. The beacon chain mediates cross-shard communications.'' [55] As Ethereum, 'Ethereum upgrades' will be based on so-called *balances*, which imposes an account money scheme.

RapidChain [37] is a permissionless, sharded, UTXO-based blockchain. ''RapidChain partitions the set of nodes into multiple smaller groups of nodes called *committees*.'' [37] Intra-committee consensus builds on an efficient, synchronous version of Byzantine fault tolerance [56], [57] to reduce communication overhead and latency in each committee. In order to achieve efficient cross-shard transaction verifications, RapidChain employs a routing mechanism inspired by Kademlia [58], a peer-to-peer distributed hash table, for committees to discover each other.

Monoxide [38] is a sharded blockchain technology that is based on an account money scheme, i.e., an account/balance model, compare with [22]. Shards are called *zones* in Monoxide, which are defined as ''multiple independent and parallel instances of single-chain consensus systems'' [38]. For cross-shard transactions, Monoxide introduces a notion of *eventual atomicity*. Eventual atomicity allows ''the withdraw operation to execute first, interleaving with other transactions then the corresponding deposit operation to be settled later. What is achieved is that once the withdraw operation is confirmed, the deposit operation will be executed eventually.'' [38] In order to achieve eventual atomicity, [38] suggests a lock-free cross-shard transaction design, which aims at saving the overhead of a usual two-phase commit protocol needed to achieve atomicity immediately.

In [40], Dang *et al.* design a permissioned, sharded, UTXO-based blockchain technology that aims at performance optimizations of the consensus protocol running within each individual shard. They suggest an efficient shard formation protocol, and a secure distributed (Byzantine

[4] https://docs.zilliqa.com/whitepaper.pdf
[5] https://harmony.one/whitepaper.pdf
[6] https://logos.network/whitepaper.pdf
[7] https://stegos.com/docs/stegos-whitepaper.pdf
[8] https://ethereum.org/en/upgrades/
[9] https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/beacon-chain.md

TABLE 1. Comparison of sharded blockchain technologies in regard to implementing money scheme and utilized consensus protocol.

Blockchain	Money Scheme	Consensus Protocol
Elastico	UTXOs	proof of work
SSChain	UTXOs	proof of work
OmniLedger	UTXOs	hybrid proof of work / Byzantine fault tolerance; proof of stake
RapidChain	UTXOs	Byzantine fault tolerance
Dang et al. [41]	UTXOs	Byzantine fault tolerance
Ostraka	UTXOs	generic
Ethereum	account scheme	proof of work
Ethereum upgrades	account scheme	proof of stake
Monoxide	account scheme	generic
Chainspace	object-based	Byzantine fault tolerance
ZyConChain	not specified	hybrid proof of work / Byzantine fault tolerance
Du et al. [44]	not specified	Byzantine fault tolerance
Alphabill	bill scheme	generic

fault-tolerant) transaction protocol for cross-shard, distributed transactions (cross-shard commit protocol). The paper [40] considers technology in the context of trusted execution environments (TEEs).

SSChain [41] is a sharding-based protocol that addresses the problem of extensive data migration due to reshuffling the network. Therefore, SSChain suggests a non-reshuffling structure in service of transaction sharding and state sharding. In order to achieve the elimination of data migration overhead, SSChain allows nodes to "freely join in one or more shards without reshuffling" [41]. The implementation of SSChain is based on the Bitcoin source code, therefore, SSChain uses the UTXO money scheme and runs a proof-of-work consensus protocol.

Ostraka [42] is different from sharding-based blockchain protocols that split the network into shards, i.e., Ostraka shards the nodes themselves. The rationale behind the design of Ostraka is to prevent such denial-of-service attacks where a single shard is overloaded with the goal to bring down the complete transaction system. 'Ostraka utilizes the UTXO model." [42] In regard to the consensus protocol, Ostraka is generic, i.e.: "Ostraka is a node architecture and is thus independent of the underlying consensus protocol." [42]

ZyConChain [59] is a sharded blockchain technology that follows the approach of a general data model, i.e., aims at being not restricted to a certain scheme such as UTXO. ZyConChain's approach introduces three kinds of blocks in each shard including parent blocks, side blocks and state blocks, where each type of block is maintained via its own consensus mechanism. Parent blocks are maintained via proof-of-work consensus, whereas for side blocks, a new consensus mechanism based on speculative Byzantine fault tolerance [60] is suggested.

Alphabill is a partitioned, sharded blockchain technology, i.e., it introduces two tiers of decomposition, see Fig. 1, which makes it different from all technologies [27]–[29], [35]–[41], [59] discussed so far in Section II-A2, which utilize only one level of decomposition into shards. In Alphabill, the partitions host implementations of distinct tokens and each partition stands for a distinct transaction system. Each tier of decomposition can have its one, specific optimizations, which is behind the reasons of Alphabill's ultra-scalability. Alphabill utilizes a new form of money scheme, the bill scheme [22], whereas all other technologies in Section II-A2 are either based on a variation of one of the two established money schemes, i.e., UTXOs and account schemes, or – in case of ZyConChain [59] and Du *et al.* [43] – do not specify details of their transactions. Only the bill scheme [22] is fully decomposable, which is the key enabler for Alphabill's ultra-scalability, since cross-shard transactions become obsolete in Alphabill. Where the other technologies have to deal with cross-shard transactions at highest granularity, Alphabill has to deal only with cross-partition transactions at a coarse-grained level, for which it innovates its own swap-partition-based 3-phase commit protocol. Alphabill does not prescribe a particular consensus protocol, i.e., new transaction systems can join the Alphabill platform as long as their consensus protocol adheres to certain minimal conditions, which can be best described as deterministic finality. As opposed to that, all of the technologies in Section II-A2 (despite of one, i.e., Monoxide [38]) rely on their own specific suggestions to consensus protocols, which advance or combine elements of established consensus protocols such as proof of work, proof of stake and Byzantine fault tolerance. In Table 1, we compare all technologies discussed in Section II-A2, including Alphabill, with respect to their money schemes and utilized consensus protocols.

In [43], Du *et al.* suggest the Mixed Byzantine Fault Tolerance (MBFT) protocol that uses so-called layering to separate specific node functions, i.e., separating the verification function from the demodulation process. Layering aims at reducing the load of nodes and this way improving the efficiency of the overall consensus mechanism. Layering is combined with sharding to further improve the protocol efficiency. In regard to the money scheme, [43] is generic, i.e., it does not further specify the content of transactions.

Alphabill has been designed, originally, as a universal asset tokenization platform. The objective of universal asset tokenization has directed Alphabill's design efforts and can be detected in many of its design decisions and innovations. At least to a certain extent, each blockchain technology that possesses a sufficiently flexible mechanism for locking and unlocking transactions, can be utilized for launching new tokens or wrapping tokens of existing blockchains. However, that does not automatically mean, that such a technology is also well-suited for tokenization. What is needed are (i) appropriate support for joining a transaction system to the platform, (ii) systematic features for interaction of hosted tokens, and last but not least, (iii) sufficient scalability, actually, uncapped scalability. All of these issues are explicitly addressed by Alphabill, but usually not by other blockchain technologies.

However, there is yet another crucial, and even more relevant difference. New tokens can be launched on top of a blockchain technology, but technologically, they are limited by the underlying blockchain, i.e., they cannot bring technological innovation to it. This is different in Alphabill. Here, partitions have their own implementations and therefore allow for technological innovations.

To explain the difference, let us take Ethereum as an example, which is widely used to create new tokens. In an ICO (initial coin offering), the Ethereum development standard ERC-20[10] can be utilized to create new tokens. These new tokens represent the coins of a newly offered cryptocurrency. However, a typical scenario is that this new cryptocurrency is then build as a new, separate blockchain technology. This technology would implement the innovations outlined in the white paper of the respective ICO. These innovations can be about improving (or disrupting) the established processes in a business domain or about technological advancements such as performance, security, decentralization, and integration of emerging technologies. The new cryptocurrency now exists in the form of two kinds of tokens, i.e., its ERC-20-based tokens in Ethereum and the genuine coins of the separately implemented blockchain. Both coins can be traded, and investors can exchange the ERC-20-based tokens to the genuine token via a blockchain bridge.[11] In such an ICO lifecycle, Ethereum has merely the role of a crowdfunding platform, but the new token is not genuinely ''build on top of Ethereum'' [61]. In principle, it is possible to build the new cryptocurrency entirely as an Ethereum token, i.e., without separate implementation and without blockchain bridging. In such case, all innovations of the new cryptocurrency are reflected in Ethereum smart contracts. These innovations can be business domain innovations but no blockchain technology innovations, and the new cryptocurrency is limited by the technological capabilities of the hosting Ethereum platform.

3) OPTIMIZATIONS FOR SHARDED BLOCKCHAIN TECHNOLOGY

In [62], it is suggested to analyze the security of sharded blockchains (such as OmniLedger [28], [29] and RapidChain [37]) by estimating the failure probabilities of single sharding rounds on the basis of failure probabilities of all shards. For this purpose, a hypergeometric distribution model is assumed. The proposed methodology is tried out and numerically analyzed on the basis of a large-scale trial.

In [63], an optimized data storage model for sharding-based blockchain technology is suggested. The approach utilizes an Extreme Learning Machine (ELM) algorithm, a type of single-layer feedforward neural network, to identify hot blocks to be stored and queried locally according to several objective criteria, including historical popularity, hidden popularity and storage requirements of the block.

In [64], an adaptive resource-allocation algorithm is designed for optimizing transaction assignments in sharded, permissioned blockchains. The algorithm is designed on the basis of the drift-plus-penalty (DPP) technique [65] in the framework of stochastic network optimization [65], [66].

B. CENTRAL BANK DIGITAL CURRENCIES

RSCoin [67] is a central bank digital currency. Its architecture is based on shards (called *mintettes*) and a trusted central component (called central bank). Each mintette serves a subset of public client addresses, with a certain level of replication needed for majority voting. The mintettes are authorized by the central component and work together in creating consensus on a next block to be certified by the central component. The necessary communication is implemented indirectly by the wallets via a two-phase-commit protocol, where each wallet makes its decision individually based on incoming majority votes.

The Hamilton Project [68], conducted by the Federal Reserve Bank of Boston together with the Massachusetts Institute of Technology Digital Currency Initiative, is a concept study on implementing central bank digital currency. In the project, they investigate two different architectures. The first one, called the *atomizer architecture*, is a blockchain solution that relies on sharded transaction verification. The architecture follows the UTXO scheme. The central component of the architecture is the *atomizer*, which is responsible for collecting verified payments from shards and creating blocks, introducing an essential bottleneck to the overall system. The study proceeds with comparing the atomizer architecture with the so-called *2PC architecture* (2-phase-commit architecture), which represents an instance of established (non-blockchain) transaction system technologies as found in today's banking domain.

RSCoin as well as the Hamilton project are not designed as universal asset tokenization platforms such as Alphabill, instead they represent central bank digital currencies such as KSI Cash. The crucial difference to KSI Cash is in the utilized money schemes which follow the UTXO model in both the case of RSCoin and the Hamilton project, as opposed to the bill money scheme in case of Alphabill. Therefore, RSCoin and the Hamilton project need to deal with the severe issue of cross-shard transactions, whereas KSI Cash can instead deal with the comparatively moderate issue of increasing amounts of smaller money denominations.

[10] https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

[11] https://ethereum.org/en/bridges/

C. A LARGE-SCALE OFF-CHAIN PAYMENT APPROACH

An example of a large-scale off-chain payment solution is SLIP (Secure Large-Scale Instant Payment) [69]. SLIP uses an aggregate signature scheme to connect off-chain channels that enables locked-in tokens to circulate. This way, SLIP aims at improving the negotiability of locked-in tokens in off-chain channels. The generic construction of SLIP aims at defending against double-spending, over-spending, and malicious settlement attacks. When compared to Alphabill, SLIP is based on an account money scheme, whereas Alphabill is based on the bill money scheme. SLIP is not a sharded approach, but uses off-chain payments to improve scalability.

D. A BILL-SCHEME-BASED BITCOIN SIDECHAIN

CoinCash [70] is a Bitcoin sidechain [71] that aims at enhancing privacy by adding an overlay of transaction anonymization [72]. CoinCash uses the CoinJoin protocol introduced by George Maxwell [73]. The CoinCash blockchain is an example of implementing a bill money scheme, but does not solve the problem of payments with fractional amounts, i.e., "transactions involving fractionation amount are rounded up to the nearest available denomination" [70]. According to [70], CoinCash is neither a universal asset tokenization platform nor a sharded blockchain technology as Alphabill.

E. A HETEROGENEOUS MULTI-CHAIN APPROACH

Polkadot[12] is designed as a "heterogeneous multi-chain" [74]. Polkadot distinguishes between the *relay chain* and *parachains* ("parallelised" chains) [74]. Polkadot itself provides the relay chain, which may host "validatable, globally-coherent dynamic data-structures" [74] called parachains, which do not necessarily have to be blockchains. With the parachains, Polkadot aims at blockchain scalability: "In principle, a problem to be deployed on Polkadot may be substantially parallelised – scaled out – over a large number of parachains." [74] At the same time, parachains aim at blockchain heterogeneity: "In other words, Polkadot may be considered equivalent to a set of independent chains (e.g. the set containing Ethereum, Ethereum Classic, Namecoin and Bitcoin) ..." [74]. In [74], Polkadot suggests a fully asynchronous interchain communication, i.e., parachains can dispatch transactions in other parachains or the relay chain. The dispatched transactions are fully asynchronous, i.e., "there is no intrinsic ability for them to return any kind of information back to its origin" [74]. The communication mechanism is based on queuing: "Interchain transactions are resolved using a simple queuing mechanism based around a Merkle tree to ensure fidelity. It is the task of the relay-chain maintainers to move transactions on the output queue of one parachain into the input queue of the destination parachain." [74]

A key difference between Polkadot and Alphabill is in their approach to decomposition. Polkadot is a federation of multiple blockchains, whereas Alphabill is a single partitioned blockchain. The other key difference is in the communication model. Polkadot chooses asynchronous messaging for its interchain communication, whereas Alphabill establishes rigorous atomicity control for each cross-partition swap.

F. RELATED STANDARDS

With respect to usage of blockchain terminology, we refer to the ISO standard ISO 22739 [32] and the NIST report NISTIR 8202 [33]. Given their status, both documents have a high international visibility and standing, where ISO 22739 is a published international standard of ISO (International Organization for Standardization), and NISTIR 8202 is an internal report of NIST (National Institute of Standards and Technology Internal Report), which received 799 citations (according to Google scholar, as of 17 June 2022) via its publication in the open-access archive arXiv [75]. Despite their standing, the two documents are currently not fully applicable. In its published version ISO 22739:2020 [32], the standard is currently (as of June 2022) to be revised and therefore exists also as an ISO committee draft, version ISO CD 22739,[13] see Table 2. Similarly, despite its visibility, NISTIR 8202 still has the status of a NIST internal report.

ISO has published a series of standards related to blockchain technology and a series of further standards under development, see Table 2, which lists the standards together with their standards development stages. The standard that comes closest to the research questions posed in that paper is ISO standard ISO/AWI TS 23516 'Blockchain and distributed ledger technology – Interoperability framework'. ISO/AWI TS 23516 aims at specifying "a framework, recommendations and requirements for interoperability between DLT systems, between DLT and entities outside the DLT system, the relationship and interactions between these and cross-cutting aspects".[14] The standard is in a very early stage of development, i.e., it has become an officially registered new ISO standards development project. It is therefore too early to compare it systematically with the contributions made in this paper. What can be said in any case is that interoperability of blockchains is definitely a pressing issue – we have explained, in Section II-A2, the role of bridging between blockchains through the example of a typical ICO lifecycle utilizing Ethereum as a token sale platform. However, we have also explained in Section II-E, when comparing Alphabill with Polkadot, that Alphabill is not about federation of blockchains. Instead, Alphabill is a partitioned blockchain.

III. THE ALPHABILL PLATFORM

Alphabill is a system for trading with various kinds of digitalized (tokenized) assets. Alphabill enables swap transactions between different kinds of assets without trusted

[12] https://polkadot.network/

[13] https://www.iso.org/standard/82208.html

[14] https://www.iso.org/standard/82098.html

TABLE 2. Published and planned ISO standards related to blockchain technology and distributed ledger technologies (DLTs), including approved work items (AWIs), working drafts (WDs), technical specifications (TSs), technical reports (TRs), draft technical reports (DTSs), committee drafts (CDs), and proofs of a new international standard (PRFs); together with their current standards development stages as of June 2022.

ISO No.	Standard	Stage
ISO/AWI 7603	Decentralized identity standard for the identification of subjects and objects	New project registered
ISO/AWI TS 23516	Blockchain and distributed ledger technology – Interoperability framework	New project registered
ISO/WD TR 23642	Blockchain and DLTs – Overview of smart contract security good practice and issues	Working draft study initiated
ISO/WD TR 6277	Blockchain and DLTs – Data flow model for blockchain and DLT use cases	Working draft study initiated
ISO/DTR 6039	Blockchain and DLTs – Identifiers of subjects and objects for the design of blockchain systems	Committee draft study initiated
ISO/CD 22739	Blockchain and DLTs – Vocabulary	Close of comment period
ISO/DTR 23644	Blockchain and DLTs – Overview of trust anchors for DLT-based identity management	Close of comment period
ISO/PRF TR 3242	Blockchain and DLTs – Use cases	Final text received
ISO/TR 23455:2019	Blockchain and DLTs – Overview of and interactions between smart contracts	Published
ISO/TR 23576:2020	Blockchain and DLTs – Security management of digital asset custodians	Published
ISO/TR 23244:2020	Blockchain and DLTs – Privacy and personally identifiable information protection considerations	Published
ISO/TS 23258:2021	Blockchain and DLTs – Taxonomy and ontology	Published
ISO/TS 23635:2022	Blockchain and DLTs – Guidelines for governance	Published
ISO/TR 23249:2022	Blockchain and DLTs – Overview of existing DLT systems for identity management	Published
ISO 23257:2022	Blockchain and DLTs – Reference architecture	Published
ISO 22739:2020	Blockchain and DLTs – Vocabulary	To be revised
ISO/DTS 23259	Blockchain and DLTs – Legally binding smart contracts	Deleted

intermediaries, and we call those transactions *multi-asset swap transactions* (similar to multi-asset strategies, which also consist of various kinds of assets[15]).

Definition 1 (Multi-Asset Swap Transaction): A *multi-asset swap transaction* is a swap transaction between different kinds of (tokenized) assets.

The Alphabill system has its native electronic currency called Alphabill Money. With multi-asset swap transactions, digitized assets can be bought and sold for the native currency or swapped directly without involving the native currency. Multi-asset swap transactions are executed automatically – defined by blockchain rules, so that no nobody can break their trading contract agreements.

Each kind of asset is implemented by a corresponding transaction system that is deployed to the Alphabill platform as so called *partition*. Therefore, multi-asset swap transaction are also called cross-partition transactions in this paper.

A. THE ALPHABILL PLATFORM ARCHITECTURE

The Alphabill platform, see Fig. 1, allows for the deployment of arbitrarily many partitions, where each partition is a blockchain implementation of a transaction system. There are three specialized partitions that genuinely belong to the platform:

- *Root partition*. The root partition (or root system) keeps track of the registered transaction systems. The entities of the root partition are the registered transaction systems, each described by the system identifier α and the state tree type (see Def. 7). The root partition periodically receives the roots of the state trees of all transaction systems and creates uniqueness certificates for the ledgers of the transaction systems. By uniqueness certificate, we mean any proof of uniqueness of the ledger such as proof of work, proof of stake, or proof of authority. The definite specifications of the root partition and the uniqueness certificate depend on the consensus protocol of the root partition, the particular choice of which is out of the scope of this paper.
- *Atomicity partition.* The atomicity partition (or atomicity system) supports atomic multi-asset swap transactions. The entities of the atomicity partition are multi-asset swap transactions, where each transaction is described by the identifier ι, the terms, and the status (complete/incomplete) of the transaction. The atomicity partition is described in detail in Section VI-C.
- *Alphabill Money partition.* The Alphabill Money partition (or Alphabill Money system) provides the default medium of exchange of the platform. The implementation of the money partition is described in Section VI-B

The communication protocols in the Alphabill platform (Fig. 1) can be divided into five logical groups (*fabrics*):

- The *consistency fabric*, which establishes the rules of communication between client applications (wallets) and the transaction systems, as well as the reactions of the transaction system to the communication, i.e., how the transactions (enacted by wallets) change the state of the transaction system.
- The *certification fabric*, which establishes the rules of communication between the transaction systems and the root system, as well as how the root system creates uniqueness certificates for all the transaction systems.
- The *distribution fabric*, which describes the rules of wallet-to-wallet communication.
- The *coordination fabric*, which describes the protocols between wallets and the atomicity system as well as the reaction of the atomicity system to protocol messages.

[15]https://www.blackrock.com/us/individual/education/multi-asset-strategies

- The *governance fabric*, which describes the communication with the root system for registering, changing, and removing transaction systems from the framework.

In regard to the certification fabrics, any type of consensus protocols (proof of work, proof of stake, etc.) can be used for producing uniqueness certificates in the root partition of the Alphabill framework. This paper does not aim to contribute to the field of ledger consensus. Partitions send their root hash values and summary values to the root partition and receive uniqueness certificates from the root partition. The consensus protocol guarantees that only one hash and summary value pair from each partition is certified by the root partition per block and that partitions do not create the next block before the uniqueness certificate for the previous block has been received.

B. THE ALPHABILL DESIGN PRINCIPLES

The design goals of Alphabill are maximum security, maximum scalability and system viability [76]. Accordingly, the design principles of Alphabill are:

- *Secure-by-Design.* Alphabill is a partitioned blockchain system that operates according to abstract rules that cannot be broken unless the blockchain technology itself fails. Every transaction system in Alphabill has its specific rules. All communication between partitions is certified. Cross-partition messages have ledger certificates. A ledger certificate is a proof that the content of an inter-partition message is included in the ledger of the sender partition. The types of ledger certificates available in the Alphabill platform are described in Section VI-A3.
- *Scalable-by-Design.* The Alphabill framework provides tools for designing a transaction system in a way that the ledger of the system is dynamically decomposable into autonomous ledgers (*shards*). Each asset entity has its own independently verifiable sub-ledger.
- *Robust-by-Design.* The Alphabill framework provides tools for designing highly redundant transaction systems, following proven patterns of blockchain networks with many nodes.
- *Viable-by-Design.* The Alphabill framework is a self-adaptive system. It changes depending on the requirements of its business environment and the society [76]–[84]. This is only possible because Alphabill is a partitioned blockchain solution, where new transaction systems can be included, existing transaction systems can be modified, and transaction systems that are no longer needed can be removed from the system. This is not possible in monolithic permissionless blockchains such as Bitcoin, because in these blockchains all rules are fixed from the launch of the blockchain and henceforth cannot be changed.

IV. KSI CASH CENTRAL BANK DIGITAL CURRENCY

KSI Cash [20], [21], [85] is a central bank digital currency (CBDC). KSI Cash represents an option of how to implement a currency partition of the platform in case it has to be controlled by a central bank.

KSI Cash is implemented on the basis of a bill money scheme [21], [22], or just bill scheme for short. The key property of any bill scheme is that it allows for sharding without a need for atomicity control of cross-shard transactions. The bill scheme mimics traditional cash. Traditional cash scales, without any automation, and, in a certain sense, the bill scheme can be considered as a direct digitization of traditional cash. In this paper we present a proof of concept of KSI Cash that has been implemented to evaluate its performance. This concept study is based on a pure bill money scheme as described in Section IV-A.

A. KSI CASH DATA STRUCTURES

1) BILLS AND PAYMENT ORDERS

Figure 2 shows the data structures of KSI Cash consisting of the public blockchain data and the maintained system state. The digital asset of a bill scheme is called a *bill*. A bill has an owner and a value. During payments, it is possible to change the owners of bills, but it is not possible to change their values. This means that the value of a bill cannot be changed and the number and identities of all bills are fixed from the genesis of the ledger during all of its lifetime, i.e., bills cannot be destroyed, nor dynamically created. From that, it follows immediately, that there are no splits and joins of digital assets, as we have them, e.g., in Bitcoin UTXOs (unspent transaction outputs) [1], [44]. Later, in implementations of KSI Cash in the field, the rigid change policy of bills needs to be loosened to allowing for limited forms of splits and joins. The design solution is to allow in-shard splits and joins, but to avoid atomic cross-shard transactions.

Figure 2 shows a small example system with sixteen bills B_0 through B_{15}. Given a bill B_i, we say that i is the identifier (or number) of the bill. The bills are distributed over four shards $shard_1$ to $shard_4$, i.e., bills B_0 through B_3 are maintained by $shard_1$, bills B_4 through B_7 are maintained by $shard_2$ and so forth. In Fig. 2, time flows from left to right, divided into epochs (also called rounds), i.e., rounds of block creation, with n denoting the most recent epoch number. In a given epoch k, each payment order, denoted by B_i^k, that is successfully integrated into the block, changes the owner of the bill B_i (and, in the extended bill scheme, potentially also the value of that bill). A payment order B_i^k is a record

$$B_i^k = \langle i, owner(B_i^k), value(B_i^k)\rangle, \tag{1}$$

where $owner(B_i^k)$ is the public key address of the new owner and $value(B_i^k)$ the (the potentially new) value of the bill. Therefore, you can think of a payment order B_i^k, that is integrated into the ledger, simply as the status of the bill in the respective round. We therefore also talk about a payment order B_i^k simply as the bill B_i in round k.

Each payment order is signed by the most recent owner of the affected bill. We call a payment order together with a corresponding signature a *signed payment order*. Signed payment orders are the transactions that are recorded by

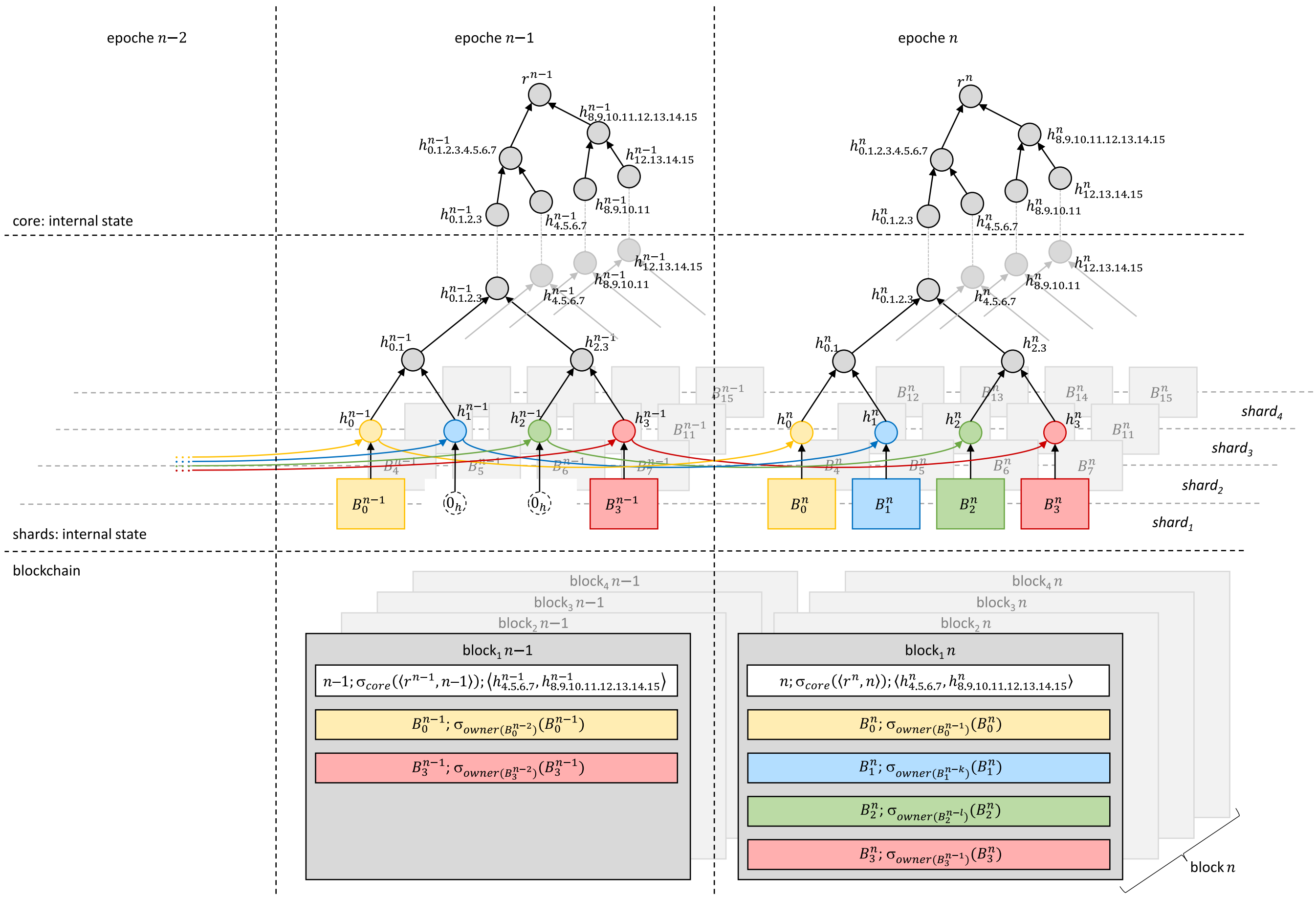


FIGURE 2. KSI cash data structures.

KSI Cash blocks. In Fig. 2, we use $\sigma_o(x)$ to denote the signature of data x created by the private key belonging to the public key address o. In a given round, only a small subset of all bills is affected by payment orders. Therefore, it is necessary to retrieve the current owner of a bill in the most recent block that contains a payment order with respect to this bill. For example, in Fig. 2, the most recent owners of bills B_0 and B_3 in round n, can be retrieved from the previous round n–1 as $owner(B_0^{n-1})$ and $owner(B_3^{n-1})$, whereas the owners of bills B_1 and B_2 need to be retrieved from the previous rounds $n-k$ and $n-l$ respectively, with both $k > 1$ and $l > 1$.

2) INTERNAL STATE

Each block needs to obtain further authentication data. We have designed a bill-based data structure that is optimized for performance, which will be analyzed in Section IV-C. For this, in each round, a Merkle tree is computed, which has one leaf for each bill, called the *hash of the bill* (in that round). If, in a given round, there exists a validated payment order for a bill, the corresponding leaf of the Merkle tree is computed from this payment order and the hash of the bill in the previous round, as depicted in Fig. 2 by various different colors (B_0: yellow; B_1: blue; B_2: green; B_3: red). If there exists no validated payment for a bill in a given round, we take the hash of the bill in the previous round again as hash of the bill in the current round (shown in Fig. 2 as computation from the zero hash, denoted by 0_h, and the hash of the bill in the previous round). This means that data from several rounds are related to each other ("chained together") at the level of hash tree leaves (and not, as, e.g., in Bitcoin, at root hash level).

In each round, the Merkle tree, which we also call the *hash tree of the round*, is computed by the shards and a further so called *core system*, or just *core* for short. Each shard validates payment orders and computes a Merkle tree for those bills that it is responsible for. The core then further integrates all of the shard trees into one single Merkle tree, see Fig. 2. In Fig. 2, the hash tree of a round k consists of nodes of the form h_s^k, where s is a sequence $i_1.i_2.\cdots.i_n$ of bill identifiers that uniquely determines the node's position in the hash tree.

We use h^k to denote the hash tree of round k and r^k to denote its root hash. We call the root hash of the hash tree, which is computed by $shard_i$ in round k, the *root hash of* $shard_i$ (in round k). We call the upper part of the hash tree of round k, that is maintained by the core system, the *core hash tree* of round k. Note that the root hash of a $shard_i$ belongs also, as a leaf, to the core hash tree of the respective round.

For example, in Fig. 2, we have for $h^k_{0.1.2.3}$ belongs both to the hash tree of $shard_i$ and the core hash tree of round k.

3) BLOCKS AND BILL PROOFS

The KSI Cash blocks are sharded. In a given round, each $shard_i$ builds a *shard block* denoted by $block_i\ k$. The block of round k, which is denoted as $block\ k$, consists of all shard blocks $block_i\ k$. In each round, the core system signs the root hash together with its round number $\langle r^k, k\rangle$ using the private key of the central authority, i.e., the respective KSI Cash system owner. We call such a signature a *root hash signature*, denoted by $\sigma_{core}(\langle r^k, k\rangle)$ in Fig. 2. As authentication data, each shard block $block_i\ k$ contains (i) the round number, (ii) the round's root hash signature and (iii) the authentication path [86]–[88] belonging to the shard's root hash in the core hash tree. For example, in Fig. 2, the shard root hash of $block_1\ n$ is $h^k_{0.1.2.3}$ and the authentication path of this shard root hash is the sequence $\langle h^k_{4.5.6.7}, h^k_{8.9.10.11.12.13.14.15}\rangle$.

After a payment order has been validated and successfully integrated into the block, the shard sends a respective proof to the respective wallet; either upon request of the wallet (pull) or automatically (push). We call proofs of successful payment orders also just *bill proofs*. With the given block design, each shard has the necessary authentication data to compile its bill proofs. A bill proof consists of (i) the bill number, (ii) the round number, (iii) the round's root signature, (iv) the hash of the bill in the previous round and (v) the complete authentication path of the bill in the current round's hash tree. For example, the bill proof of payment order B^n_0 in Fig. 2 is:

$$\left\langle 0, n, \sigma_{core}(\langle r^k, k\rangle), h^{n-1}_0, \langle h^n_1, h^n_{2.3}, h^n_{4.\cdots 7}, h^n_{8.\cdots 15}\rangle\right\rangle \quad (2)$$

The data contained in a KSI Cash bill proof is sufficient to authenticate a successful payment order, i.e., the wallet does not need to have all data of the full block or even the whole blockchain.

4) ON CORE SHARDING

The core system could also be sharded. In Fig. 2, the core hash tree is not sharded. However, a further tier of shards (or even tiers of shards) in the realm of the control system could be introduced, following the same pattern as exposed by the hash tree and block data structures in Fig. 2. However, core sharding would only become necessary, if the number of shards gets critically large. In balancing out concrete system designs, the number of bills per shard will be very large as compared to the number of shards. The number of bills per shard is determined by the expected performance of the system in terms of transaction throughput. Even for very large transaction loads, the number of shards can be expected to be relatively small (in regard to the need for core sharding), compare with Section IV-C and V.

B. THE KSI CASH EXCHANGE SERVICE

1) EXCHANGE SERVICE AND EXCHANGE STRATEGIES

The KSI Cash data structures described in Section IV-A are fully parallelizable. In a pure bill scheme, payment orders only change the owners of bills. Therefore, the set of all bills can be distributed over shards that can process payment orders in parallel without any communication and coordination. If a wallet wants to enact a payment, the wallet breaks down the payment to a set of payment orders. We need to assume that the wallet owns a reasonable mix of bills with different kinds of values, in particular, small enough values, so that it can map a concrete payment onto payment orders, possibly in different shards. In practice, it will happen that some payments cannot be mapped onto a mix of payments orders, given a certain mix of bills in the wallet.

In some of the KSI Cash runs, we simply ensure, via the system initialization, that all wallets have enough small-value bills to run all payments. This is possible, as we can steer, which payments are sent to wallets. Furthermore, we have implemented an exchange service in KSI Cash. This exchange service is implemented as a specialized wallet that is (i) owned by the central authority (central bank), and is (ii) trusted by all of the other wallets. We call the specialized wallet of the exchange service the *exchange wallet*, and we call all other wallets *user wallets*, or just wallets, if clear from the context.

The exchange service needs to be initialized with sufficiently many small-value bills in the beginning of a test run to enable all payments between user wallets. There are two wallet exchange strategies with respect to the usage of the exchange service, i.e.:

- *Re-Active Exchange.* Whenever the wallet cannot turn a payment into a set of payment orders, due to the lack of sufficiently small-value bills, it first uses the exchange service before the actual payment.
- *Pro-Active Exchange.* After each successful payment, the wallet analyzes, whether its combination of bills is able to potentially serve all possible payments (smaller than the total amount possessed by the wallet). If not, the wallet immediately triggers an exchange via the exchange service.

A usage of the exchange service is called a *money exchange*. A money exchange simply consists of two payments, i.e., a payment from the user wallet to the exchange wallet (containing some large-valued bills) and a simultaneous back payment from the exchange wallet to the user wallet (containing respectively many small valued bills).

2) TRANSACTION AND TRANSACTION ROUNDS

In the KSI Cash setting, we assume that the exchange wallet is trusted by the user wallets. Therefore, both payments of an exchange (i.e., the payment to the exchange wallet and back to the user wallet) can be enacted simultaneously, in one so-called *transaction round*. The transactions in an Alphabill transaction system might be organized hierarchically, depending on the respective domain and its technical implementation. In KSI Cash, we distinguish three levels of transactions, see Table 3. At the outermost level, KSI Cash transactions are simply called transactions. They are triggered by users in order to transfer a value from one user wallet

TABLE 3. KSI Cash transaction design and terminology.

KSI Cash Transaction Hierarchy	
transaction	user-triggered payment between wallets; consists of one or two transaction rounds
transaction round	payment between user wallets; *or* money exchange
payment order	changes the owner of a bill
Further KSI Cash Transaction Terminology	
money exchange	consists of two simultaneous payments: (i) from user to exchange wallet, (ii) trusted back payment from exchange to user wallet
transaction duration	started by a wallet sending payment orders; ends, when all bill proofs have been received; includes network travel times
tx/s	transactions per second (*transaction throughput*)

to another. At the next level, a transaction consists of either one or two so-called transaction rounds. A transaction round conducts a payment between two user wallets or otherwise a money exchange. At the lowest level, the transactions are the payment orders, which change the ownership of bills. In general, several payment orders are needed during a payment to fulfil its purpose.

A transaction must contain at least one transaction round, i.e., a payment between user wallets. Additionally, it might contain a further money exchange as second transaction round, according to the respective wallet exchange strategy (re-active or pro-active). Henceforth, we measure KSI Cash system throughput as transaction throughput (transaction rate), i.e., in terms of the number of transactions per second, which we denote as tx/s.

Transaction durations are defined via the start and end of transactions as follows: a transaction starts, when the respective wallet sends payment orders to the KSI Cash backend system; a transaction ends, when the receiving wallet has received all bill proofs of the transaction. This means, that transaction durations include network travel times. Network travel times depend on network performance; therefore, network travel time is an impact factor on KSI Cash performance measures (throughput) that is not KSI Cash specific, i.e., independent of KSI Cash technology. From a rigorous viewpoint, network travel times should be excluded from the duration times. It was decided to keep them in, in service of having realistic performance measures from a user perspective. In our test runs, network travel times had no large impact, e.g., a typical duration of creating a payment order by the wallet and sending it to the KSI Cash input router was in the range of 30-40ms only (as, e.g., compared to the standard round time of KSI Cash, which was set to 1,000ms, see Section V-A2).

3) BEYOND THE PURE BILL SCHEME

The exchange service described in Section IV-B is a solution that is consistent with the pure bill scheme, i.e., it works without introducing splits and joins in the money transactions. In practice, it needs to be ensured that always sufficiently many small-value bills are available in the exchange wallet. An approach to keep the number of small-value bills in the exchange wallet is to introduce strategies that trigger, in some appropriate way, also changes in the inverse direction, from the exchange wallet to user wallets and back.

A different way is introducing splits and joins. In an *extended bill scheme* splitting and joining bills is allowed; however, in such a way that cross-shard transactions are avoided entirely. This means, that the mechanisms of an extended bill scheme must be designed as in-shard only. The *dust collection* solution of Alphabill Money introduced in Section VI-B2 achieves this and, therefore, is an extended bill scheme.

If a way to split bills into smaller-valued bills is introduced, we also need to introduce mechanisms that join bills of smaller values into fewer bills of larger values. Without any joining mechanism, after some time, the system would end up in a state of too high granularity; theoretically, in a state in which there exits only bills of smallest value. However, if the bill granularity of the system gets too high, the system is flooded with too much payment orders and stops running efficiently.

C. ANALYTICAL PERFORMANCE ESTIMATIONS

In this section, we conduct some analytical time and cost estimations for the bill scheme of KSI Cash as introduced in Sections IV-A and IV-B. We deal with the costs of building the internal state and verifying ledger certificates. We use ρ_i to denote the number of payment orders processed by shard i in a given round. For the purpose of this analysis, we do not distinguish between payment orders that are requested by wallets and payment orders that are processed by the shards, i.e. we simply assume that all payment orders are validated correctly and eventually integrated into the block. We use β_i to denote the number of bills maintained by a shard and ς to denote the number of shards. See Table 4 for an overview of symbols used in this section and its results.

We use T_{hash} to denote the time needed to compute the hash of a single payment order and T_{sig} to denote the time needed to verify a payment order signature. A reasonable assumption is that the time needed to verify a signature is three orders of magnitude more than the time needed to compute a payment order hash, i.e.:

$$T_{sig} \approx 10^3 \times T_{hash} \tag{3}$$

With $T_{update,i}$, we denote the time needed by a shard i to update its internal state in a round (called *update time*), i.e., to compute the new hash tree for all the payment orders to be included into the block. The time $T_{update,i}$ corresponds directly to the size of the computed hash tree, i.e., its number of nodes. The extreme (and unlikely) case that there are as many payment orders as bills yields the following as an upper bound for the update time (according to the full number of nodes in the hash tree):

$$T_{update,i} < 2\,\beta_i \cdot T_{hash} \tag{4}$$

TABLE 4. KSI cash time and cost estimations: symbols and summary.

ρ_i	:	number of payment orders, shard i
β_i	:	number of bills, shard i
ς	:	number of shards
T_{hash}	:	hash compute time
T_{sig}	:	signature verification time
$T_{update,i}$	:	state update time, shard i
$T_{process,i}$	:	round processing time, shard i
$T_{process}$	:	total round processing time

$$T_{sig} \approx 10^3 \times T_{hash} \tag{3}$$

$$T_{update,i} \approx \rho_i \cdot log_2\beta_i \cdot T_{hash} \tag{7}$$

$$T_{process,i} \approx \rho_i \cdot \left(T_{sig} + log_2\beta_i \cdot T_{hash}\right) \tag{8}$$

$$T_{process} \leq 2\,\varsigma \cdot T_{hash} + \max_i T_{process,i} \tag{9}$$

If there are less payment orders than bills, accordingly less leave nodes and possibly also (many) inner nodes of the hash tree do not have to be computed. Remember from Section IV-A that the previous round's hash is taken for bills for which there is no payment order in the current round. Saving hash tree computations propagates through the tree, i.e., in updating the hash tree, the hash values from the previous round's internal state can be re-reused wherever possible (You start with computing the hashes for existing payment orders, flag the inner nodes that get affected at the next level, proceed with those nodes in the next round, and so on, until you reach the root.). Therefore, the extreme case that there is only a single payment order yields the following lower bound for the update time (according to the length of one path from leave to root in the hash tree):

$$T_{update,i} \geq log_2\beta_i \cdot T_{hash} \tag{5}$$

Now, given the number of payment orders ρ_i, we can give a better upper bound for the update time than (4) as follows:

$$T_{update,i} \leq \rho_i \cdot log_2\beta_i \cdot T_{hash} \tag{6}$$

We can assume that we have very few payment orders as compared to the number of bills in a shard. Similarly, we can assume that the payment orders are usually widely "distributed", i.e., that there is no significant overlap of the paths of successful payment order leaves in the hash tree. Altogether, we can see that, therefore, the upper bound (6) is actually a good estimation of the update time, in general, and therefore we state:

$$T_{update,i} \approx \rho_i \cdot log_2\beta_i \cdot T_{hash} \tag{7}$$

If we assume that there are much less payment orders than bills, i.e., $\rho_i << \beta_i$, we see that the number of payment orders ρ_i clearly dominates the update time (7).

As part of payment order validation, the shard has to verify the payment order signatures as provided by the most recent owners of the respective bills. For the moment, we neglect further payment order validation costs. With $T_{process,i}$ we denote the processing time of a shard i during a round, which encompasses the update time plus the signature verification times, i.e., we estimate:

$$T_{process,i} \approx \rho_i \cdot \left(T_{sig} + log_2\beta_i \cdot T_{hash}\right) \tag{8}$$

If we assume that T_{sig} is a thousand times more costly than T_{hash}, see (3), we have that the processing time in (8) is clearly dominated by the signature verification time T_{sig} for any realistic number of bills.

Finally, we can give an upper bound for the total processing time of a round as follows:

$$T_{process} \leq 2\,\varsigma \cdot T_{hash} + \max_i T_{process,i} \tag{9}$$

As shard computations can be parallelized, the total processing time in (9) needs to consider only the slowest shard (i.e., the one that needs the maximum processing time). In addition, the total processing time needs to incorporate the update time needed for computing the core hash tree. We can assume that all shard root hashes change during a round, i.e., that all shards receive at least one transaction. Therefore, the full size of the core hash tree is the adequate basis for estimating its update time. As discussed in Section IV-A4, we can consider the size of the core hash tree as relatively small, so that the shard processing time is the dominating component in (9).

V. KSI CASH PERFORMANCE EVALUATION

A. THE KSI CASH TEST IMPLEMENTATION

1) OVERVIEW

In order to evaluate the KSI Cash concepts and architecture, in the first place in regard to performance and scalability, but also in regard to availability, a first version (proof of concept) of KSI Cash and a respective test bench have been implemented. Particular effort and resources have been invested into simulation of a realistic scenario. This involves a realistic wallet simulation as well as the implementation of a backend system with professional input and output subsystems that are specified as if the system has to operate under real-world conditions. For example, TLS (Transport Layer Security) was imposed throughout the test runs, and session authentication was conducted for incoming payment orders.

We conducted a series of test runs under simulation of realistic usage as described in Section V-B. However, we also wanted to show that the system concept is, basically, capable of unlimited performance, by showing linear scalability of the involved core data structures and algorithms. Therefore, we also conducted a series of test runs with a cut-down version of the backend system and a simplified load test generation, simply to achieve ultra-high load with still reasonable testing costs. We describe these ultra-scalability test runs in Section V-C. We start with a description of the

TABLE 5. Implementation of KSI Cash and its wallet simulation in terms of lines of code, i.e., uncommented, non-blank lines L_{code}, comment lines $L_{comment}$, and blank lines L_{blank}.

KSI Cash Backend Code				
Language	**Files**	L_{blank}	$L_{comment}$	L_{code}
Go	659	14,724	14,138	87,704
CSS	1	2,452	2	8,619
YAML	88	384	406	3,571
JSON	8	0	0	2,398
Bourne Shell	35	227	152	957
HCL	17	95	23	589
Python	13	203	101	537
Protobuf	35	165	669	394
Markdown	9	113	0	322
Dockerfile	13	45	4	161
HTML	2	14	2	130
make	1	22	2	61
Bash	1	3	4	13
Sum	887	18,447	15,513	105,461
KSI Cash Wallet Simulation and Test Bench Code				
Language	**Files**	L_{blank}	$L_{comment}$	L_{code}
JavaScript	80	591	1,919	4,322
LESS	24	213	7	1,232
Java	21	202	1	848
YAML	7	47	5	454
Markdown	5	97	0	179
Bourne Shell	1	23	36	126
Gradle	5	15	1	110
JSON	4	0	0	79
DOS Batch	1	26	2	76
make	3	19	14	42
SVG	8	0	1	42
HTML	2	9	20	39
Dockerfile	2	8	1	22
Sum	163	1,250	2,007	7,571

backend system as used in the realistic usage scenario in Sect V-A2 and will explain the differences of the system as used in the ultra-scalability test runs in Section V-C.

Both the KSI Cash implementation, as well as the KSI Cash wallet simulation and load test generation have been deployed, as cloud-native applications, to the Amazon EC2 cloud.[16]

KSI Cash, i.e., the KSI Cash backend system, has been implemented mainly in Go.[17] The implementation encompasses approx. 105,000 (uncommented, non-blank) lines of code (LOCs), with 87,704 LOCs in Go plus additional code in diverse programming and DevOps-related languages, see Table 5. The KSI cash wallet simulation has been implemented mainly in JavaScript. Together with auxiliary test bench code, it encompasses approx. 7,500 LOCs, with 4,322 LOCs in JavaScript plus additional code in diverse languages, see again Table 5. The software development team consisted of ten Guardtime software developers (all senior full-stack plus DevOps software engineers).

The KSI Cash backend system has been deployed on a total of 1,382 virtual CPUs (vCPUS) with 8.27 TB RAM and 26.1 TB storage, compare with Table 6; and the wallet simulation has been deployed on 826 virtual CPUs with 1.8 TB RAM and 24.5 TB storage. Some smaller amount of extra resources has been needed to run auxiliary modules of the test bench, i.e., 24 virtual CPUs with 88 GB RAM and 1.8 TB storage, see Table 6.

[16]https://aws.amazon.com/ec2/

[17]https://go.dev/

2) THE KSI CASH BACKEND SYSTEM

Figure 3 shows the nodes of the KSI Cash backend system as deployed in Amazon EC2. The KSI Cash wallet depicted in Fig. 3 does not belong to the KSI Cash backend system – for the architecture of KSI Cash wallets simulation, see Fig. 4. The KSI Cash wallet sends payment order requests to and fetches bill proofs from KSI Cash through a high-availability proxy (HA proxy) consisting of 16 nodes (virtual machines). See Table 6 for technical specifications of deployment nodes, including the number of virtual CPUs (vCPUs), RAM size, storage size, processor type, Amazon EC2 instance type and the number of virtual machines (VMs). Henceforth, we use nodes and virtual machines as synonyms.

Handling payment orders in KSI Cash is sharded as described in Section IV. The test system consists of 14 shards. According to this, there are 14 so-called *gateway* nodes (Gateway 1-14) and, additionally, 14 corresponding *gateway output components* as illustrated in Fig. 3. The gateways and their output components can be considered the workers of the system, as they are responsible for building the shard hash trees, creating blocks and issuing bill proofs as described in due course. This explains the relatively large amount of compute resources dedicated to the gateways and their output components, with 896 virtual CPUs and 7.17 TB RAM, i.e., approx. 65% of vCPUs and 87% RAM of the total compute resources of the backend system (1,382 vCPUs, 8.27 TB RAM). We name the gateways and their output components together with the *core* module and the *controller* module the *key components* of the system, as these components are responsible for all of the tasks described in Section IV. The remaining components allow for the initialization and management of the system and ensure its client connectivity.

Payment orders are queued in data pipelines on the basis of Apache Kafka.[18] We use a configuration with three Kafka nodes, six Kafka proxy nodes and three Zookeeper nodes. Such configuration is standard; the three Kafka nodes are in service of availability, not because of performance optimization. There exists one queue for each gateway in the Kafka configuration. The *payment order router* analyzes incoming payment request and routes them to the respective gateway queues. The payment order router is a potential bottleneck for high loads of payment requests, however, more parallel payment router nodes can be arbitrarily added to deal with higher loads, as payment order routing is fully parallelizable. For the loads during our test runs, one node (with 8 vCPUs) was sufficient.

In the KSI Cash implementation, gateways are synchronized, i.e., they process payment orders and build blocks in rounds of pre-determined, fixed length called *round time*,

[18]https://kafka.apache.org/

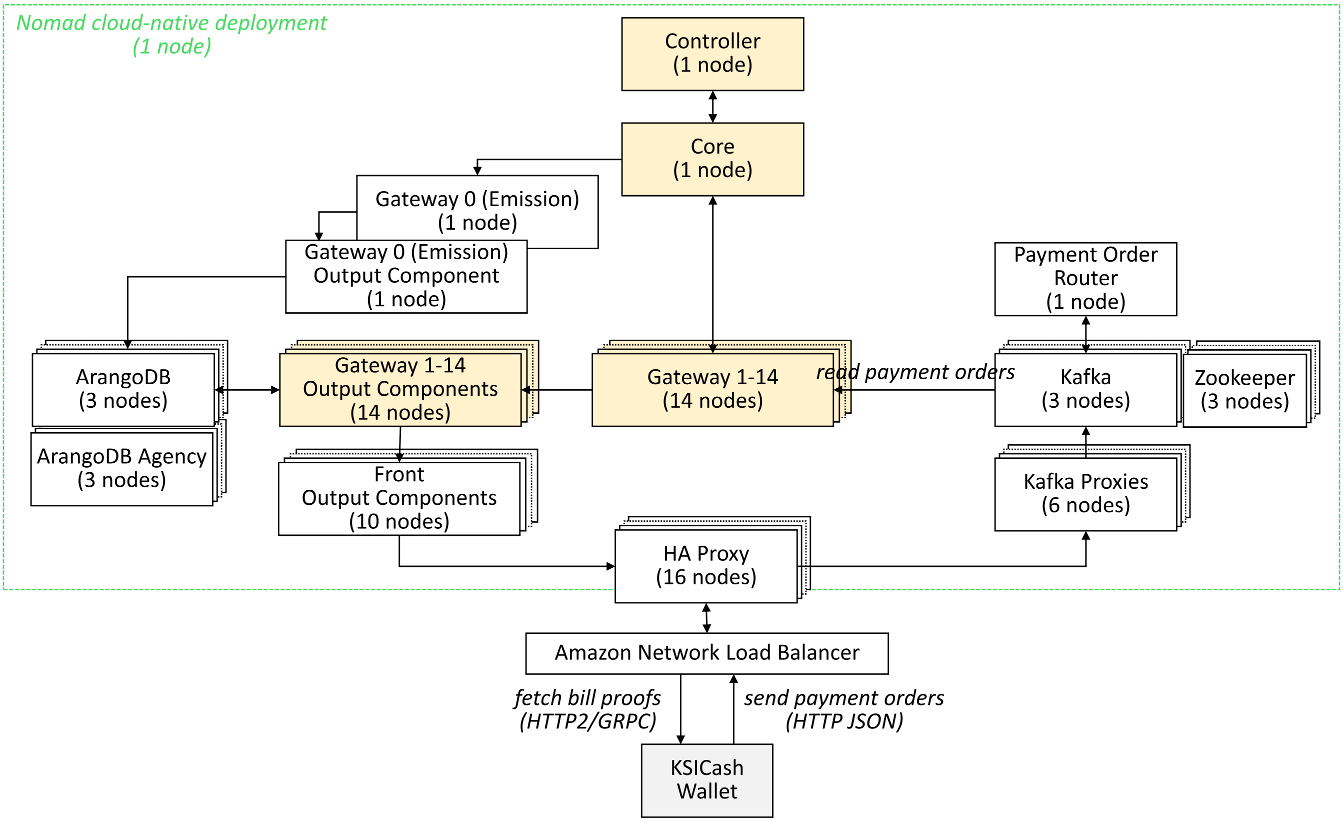


FIGURE 3. KSI cash backend system architecture.

where the round time is a tuning parameter of the system. In the test runs of KSI Cash that we present in this paper, the round time was set to a length of 1,000ms.

At the beginning of each round, each gateway pulls the payments orders which it is responsible for, from its gateway queue. It stops pulling requests after a fixed amount of time, called *payment order reading time*, set to 300ms in the test runs of this paper. Payment orders that could not be read in the pre-determined reading time, remain in the gateway queue for processing in further rounds. All gateways need to hand over calculation to the core node at least before a so called *core offset time*, which is set to 100ms in the test runs of the paper. The round time, the payment order reading time and core offset time are all tuning parameters of the system, which need to be configured in such way that all requested gateway calculations are finished during a round.

The core node is responsible for building the core hash tree. As soon as a gateway has received a certificate from the core node, it sends the block of the current round to its corresponding gateway output component. The gateway output stores the block in persistent storage. For storage of the blockchain we use the multi-model database ArangoDB[19] that we run as a database cluster consisting of six nodes, i.e., three ArangoDB nodes and further three ArangoDB Agency nodes. Furthermore, the gateway output component rebuilds the hash tree of the most recent round in its internal memory, which it needs to serve bill proof requests that are sent from KSI Cash wallets. The gateway output components serve bill proof requests via a scalable tier of ten *front output component* nodes. The front output components answer bill proof requests immediately, i.e., either with the bill proof or otherwise a message which indicates that the bill proof is not yet available. In case a bill proof is not yet available, the respective wallet has to retry the bill proof request after a reasonable time interval, which again is a tuning parameter.

An initialization system was built for setting up the testing environment from scratch, including deployment of components, configuring a new KSI Cash instance, emitting bills, initializing and pre-funding the wallets, in whatever configuration necessary. The Gateway-0 node, also called *emission gateway* and its output component are used to run this system initialization. The whole KSI Cash backend system has been developed as a cloud-native application. We have decided to use Nomad[20] as container orchestration system for KSI Cash. One node is dedicated as Nomad server for the KSI Cash backend.

[19]https://www.arangodb.com/

[20]https://www.nomadproject.io/

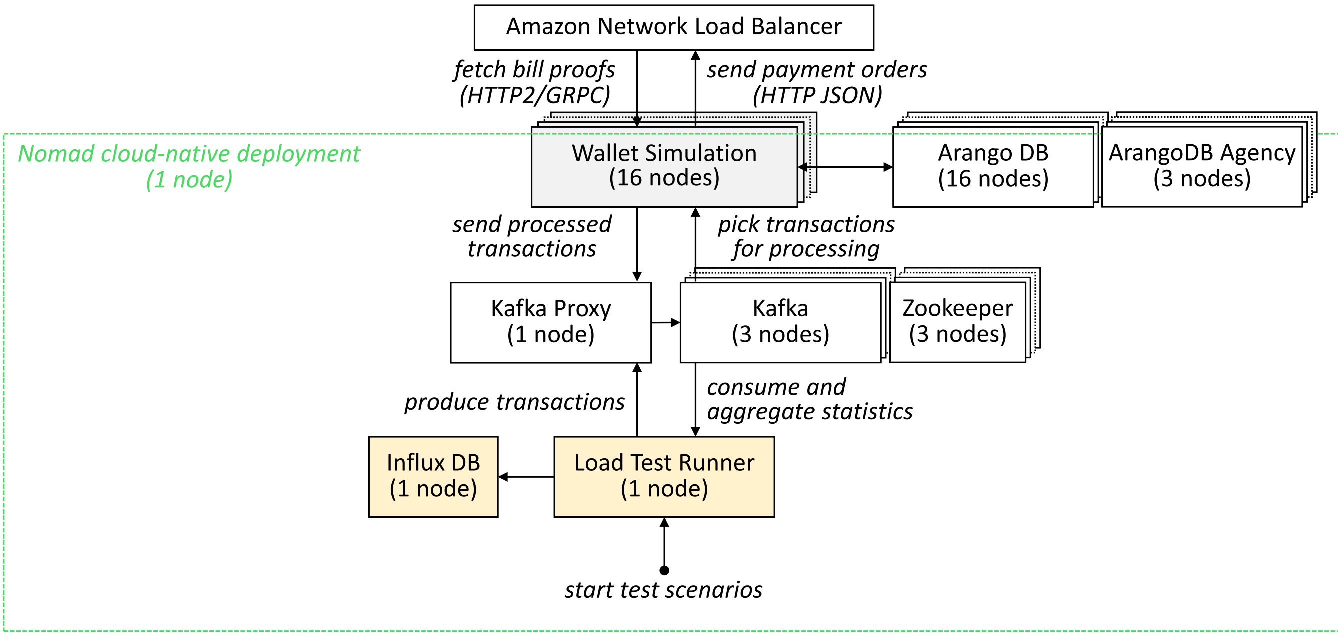


FIGURE 4. KSI cash wallet simulation and load test generation.

3) THE KSI CASH WALLET SIMULATION

Figure 4 shows the components of the KSI Cash wallet simulation. The test developer starts test scenarios via the *load test runner* component. For each test run, the load test runner generates a series of transactions among wallets. These transactions are buffered in a Kafka configuration consisting of three Kafka nodes plus one Kafka proxy node and three Zookeeper nodes. The *wallet simulation* is run by 16 nodes, connected with an ArangoDB database to hold the wallet data, run by 16 ArangoDB nodes plus 3 ArangoDB Agency nodes.

The wallet simulation has a series of tasks. First it picks transactions from the Kafka queuing system. In case of test runs including exchange services, it breaks down transactions to transaction rounds, and also steers the execution of these transaction rounds. It splits transaction rounds into payment orders, and sends these to the KSI Cash backend system. Furthermore, the wallet simulation requests bill proofs from the KSI Cash backend system. A simulated wallet waits a reasonable time after sending a payment order and before issuing the corresponding bill proof, which is at least the round time plus a certain *waiting time offset*. This time offset is a tuning parameter of the overall KSI Cash system and is set to 250ms in the current setting. If the KSI Cash backend system replies to a bill proof request that the bill proof is not yet available, the wallet will again wait the specified waiting time before issuing the bill request again.

Once a simulated wallet has received all bill proofs needed to prove a transaction, it sends it back to the load test runner as a successfully *processed transaction*. The load generator aggregated the results and submits them to the time-series database InfluxDB.[21] The test engineer can then use InfluxDB for various statistical analysis. Furthermore, InfluxDB was connected to the data analysis and visualization tool Grafana[22] for further analysis (not shown/listed in Fig. 4 and Table 6).

B. REALISTIC USAGE TEST RUNS

1) OVERVIEW

Several test runs have been conducted. The purpose of the *baseline test run* in Section V-B2 was to determine the transaction duration under very small transaction load, i.e., such that can be considered as minimally needed transaction duration. The purpose of the *production test run* in Section V-B3 was to set the system continuously under a high load, according to the key performance indicator of 10,000 tx/s, which was initially set for the proof of concept. The purpose of the *maximum throughput test run* in Section V-B4 was to push the test system to its limits, in regard to its current configuration and tuning parameter setting. The purpose of the *exchange service test runs* in Section V-B5 was to investigate differences of behaviors of KSI Cash's re-active versus pro-active wallet exchange strategies. We use the *peaks and lows test run* in Section V-B6 to investigate gateway calculation times. The purpose of the memory consumption test runs in Section V-B7 where to investigate the RAM usage of gateway nodes and gateway components under several different transaction loads. The *24 hours test run* in Section V-B8 was for the purpose of demonstrating the availability of the system.

[21] https://www.influxdata.com/
[22] https://grafana.com/

TABLE 6. The KSI Cash performance test server configuration (with node names referring to nodes in Figs. 3 and 4).

KSI Cash Server Nodes						
Node Name	**vCPUs**	**RAM**	**Storage**	**Processor**	**EC2 Instance Type**	**VMs**
Controller	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Core	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Gateway-0	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Gateway-0 Output Component	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Gateway 1–14	32	256GB	20GB	Intel Xeon Scalable (Cascade Lake)	r5n.8xlarge	14
Gateway 1–14 Output Component	32	256GB	20GB	Intel Xeon Scalable (Cascade Lake)	r5n.8xlarge	14
Front Output Components	16	32GB	20GB	AMD EPYC 7002	c5a.4xlarge	10
Payment Order Router	8	16GB	20GB	AMD EPYC 7002	c5a.2xlarge	1
Kafka	8	64GB	20GB+4TB	AMD EPYC 7000	r5a.2xlarge	3
Kafka Proxies	16	32GB	20GB	AMD EPYC 7002	c5a.4xlarge	6
Zookeeper	2	1GB	20GB	AMD EPYC 7000	t3a.micro	3
ArangoDB	16	32GB	20GB+4TB	AMD EPYC 7002	c5a.4xlarge	3
Arango Agency Node	2	4GB	20GB+20GB	AMD EPYC 7002	c5a.large	3
HAProxy	8	16GB	50GB	AMD EPYC 7002	c5a.2xlarge	16
Nomad Server	2	2GB	20GB	AMD EPYC 7000	t3a.small	3
total	1,382	8.27TB	26.1 TB	–	–	78
KSI Cash Client Nodes						
Node Name	**vCPUs**	**RAM**	**Storage**	**Processor**	**EC2 Instance Type**	**VMs**
WM Worker	32	64GB	20GB	AMD EPYC 7002	c5a.8xlarge	16
Kafka	8	64GB	20GB+2TB	AMD EPYC 7000	t3a.xlarge	3
Kafka Proxy	4	8GB	20GB	AMD EPYC 7002	c5a.xlarge	1
Zookeeper	2	1GB	20GB	AMD EPYC 7000	t3a.micro	3
Load Test Runner	8	16GB	20GB	AMD EPYC 7002	c5.2xlarge	1
ArangoDB	16	32GB	20GB+1TB	AMD EPYC 7002	c5a.4xlarge	16
Arango Agency Node	2	4GB	20GB+20GB	AMD EPYC 7002	c5a.large	3
Influx DB	8	16GB	20GB+1TB	AMD EPYC 7002	c5a.2xlarge	1
total	826	1.8TB	24.5TB	–	–	45
KSI Cash Auxiliary Nodes						
Node Name	**vCPUs**	**RAM**	**Storage**	**Processor**	**EC2 Instance Type**	**VMs**
ELK Stack	8	64GB	20GB+1.5TB	AMD EPYC 7002	r5a.2xlarge	1
HAProxy	2	2GB	20GB	AMD EPYC 7000	t3a.small	1
Nomad Backend Server	2	2GB	20GB	AMD EPYC 7000	t3a.small	1
Monitoring	2	8GB	50GB	AMD EPYC 7000	t3a.large	1
Consul	2	2GB	20GB	AMD EPYC 7000	t3a.small	3
Jump Host	2	4GB	20GB	Intel Xeon (not specified)	t2.medium	1
Docker Registry	2	2GB	100GB	AMD EPYC 7000	t3a.small	1
total	24	88GB	1.8TB	–	–	9

a: PERFORMANCE BENCHMARKS

It was decided to specify two own major key performance indicators (KPIs) as performance benchmarks as follows:

- Throughput: the proof of concept is able to process 10,000 transactions per second assuming 100M user wallets.
- Transaction durations:
  - 95% of all transactions are processed in less than 5 seconds,
  - 99% of all transactions are processed in less than 10 seconds.

The standard transaction processing benchmarks TPC-C [89] and TPC-E [90] of the Transaction Processing Council (TPC) had been considered. However, TPC-C and TPC-E were not particularly well suited for our purposes for a couple of reasons. The benchmarks have been originally designed as OLTP (online transaction processing) benchmarks in the field of database management systems, i.e., they are benchmarks for database products and not particularly designed for blockchain technology. Furthermore, since they are database product benchmarks, they are intrinsically oriented towards the account money scheme [22] when it comes to payments. Furthermore, both benchmarks are defined at application level, for example, TPC-C models an application "portraying the activity of a wholesale supplier" [89] encompassing operations not only for payments, but also operations for new orders, order status, delivery, and stock. TPC-E is exactly in the same vein, simulating "the OLTP workload of a brokerage firm" [89]. Therefore, it was decided to set own KPIs as performance benchmarks. Both of the above KPIs were fulfilled by the production test run in Section V-B3. Beyond the two major KPIs, a series of further KPIs have been specified in regard to the other test runs. They have all been met and we will not give them explicitly in this paper; instead, we directly present the results of the several test runs.

b: BILL AND WALLET INITIALIZATION

Most of the test runs in this paper (Section V-B2, V-B3, V-B6, and V-B8) have been initialized with the so-called *standard initialization* as follows:

- *Bills*: 2.8 billion bills of different values have been emitted, i.e., 200 million bills per gateway.

- *Wallets*: 100 million wallets; each wallet loaded with a combination of bills, so that the first payment is always possible without usage of the exchange service.

The maximum throughput test run in Section V-B4, the exchange service test runs in Section V-B5 and the memory consumption test runs in Section V-B7 have used different bill and wallet initializations.

c: TRANSACTION GENERATION

During transaction generation, the transaction sums were sampled randomly from log-normal distribution, with arithmetic average amount 25 EUR and with 1-cent precision. This pattern follows Eurozone statistics[23] of combined cash and card retail payments. On average, a generated transaction encompasses six payment orders. This means also, that six bill proof requests are needed, on average, to validate a transaction.

Payer wallets were chosen sequentially. Therefore, due to the test run durations, payer wallets have not been reused, i.e., a wallet has been used for payment at most once during a single test run. Due to this (and the fact that each wallet has been initialized with a respective combination of bills, see Section V-B1b), all transactions in the test runs of Section V-B2, V-B3, V-B5 and V-B8 have been single-rounded transactions, i.e., the KSI Cash exchange service has never been used in these test runs. On the other hand, the *exchange service test runs* provoked a series double-rounded transaction by re-using wallet configurations from a previous test run without re-initializing the system, as will be explained Section V-B5.

Receiver wallets were chosen randomly. Again, due to the test run durations, not all of the wallets have been used during single test runs.

We define the *duration of a test run* as the time from the start of the first round till the end of the last round, i.e., the test run durations do not include times for KSI Cash system initializations.

2) BASELINE TEST RUN

During the baseline test run, the system was loaded with 1 tx/s for a duration of 5 minutes, i.e., with a total of 600 transactions (with the standard initialization of 2.8B bills and 100M wallets). All of 600 transactions had finished successfully in less than 2 seconds. As the round time of the KSI Cash backend system was set to 1,000 ms, and the waiting time offset was set to 250ms, no transaction could have been faster than 1,250ms.

3) PRODUCTION TEST RUN

In the production test run, a steady load of 10,000 tx/s had been generated for a duration of 1 hour, see Fig. 5. The two spikes in Fig. 5 were caused by the wallet database and can be neglected. On average, approx. 60,000 bills were used in transactions every second. Wallets had been used, at most, once as sender for a transaction. This means, that approx. 36 million wallets (out of the standard initialization with 100M wallets) had started a transaction. This also means that the KSI Cash money exchange service had not been used and all transactions were single-rounded.

[23]https://www.ecb.europa.eu/press/pr/stats/

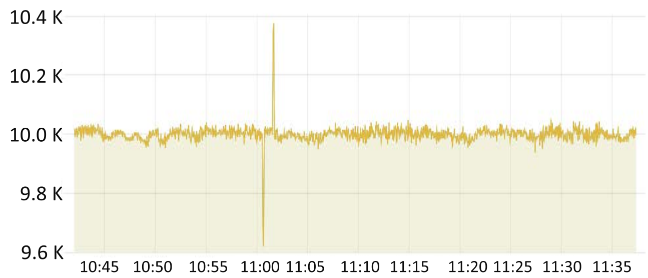


FIGURE 5. Production test run: total transactions per second.

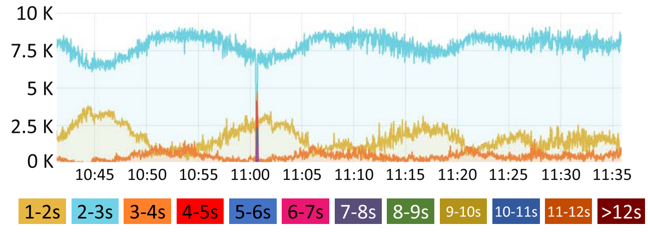


FIGURE 6. Production test run: transactions according to transaction durations.

All of the transactions of this run succeeded. Figure 6 shows the distribution of transaction durations of the run: 100% of the transactions finished in less than 7 seconds. Actually, despite during the first spike in the run (compare with Fig. 6), all transactions have been finished in less than 4 seconds. Overall, 16.51% of the transaction have been finished in less than 2s, 95.11% have been finished in less than 3s and 99.89% have been finished in less than 4s.

4) MAXIMUM THROUGHPUT TEST RUN

The purpose of this test run was to find the transaction rate limit of the current system setup, including the concrete setting of tuning parameters. The concept of this test run was to increase the transaction load during the test run in stages of 1,000 tx/s until the system starts failing. Here, we define *system failure* as the occurrence of a significant amount of failed transactions, i.e., on the basis of notions of *successful transaction* vs. *failed transaction*.

Definition 2 (Successful vs. Failed Transaction): A transaction is considered a *successful transaction*, if the bill proofs of all of its payment orders could have been successfully queried by a wallet during a specified *expiry time* ξ. Otherwise, it is considered as a *failed* transaction.

For the sake of this test run, we have set the expiry time for successful transactions to ξ =40sec (accounting to 40 rounds of block creation in our system setting). In real-world settings, in practice, this expiry time can be surely be relaxed further; the expiry time of ξ =40sec simply shows the high ambitions of the test run. We have not further pre-specified, exactly

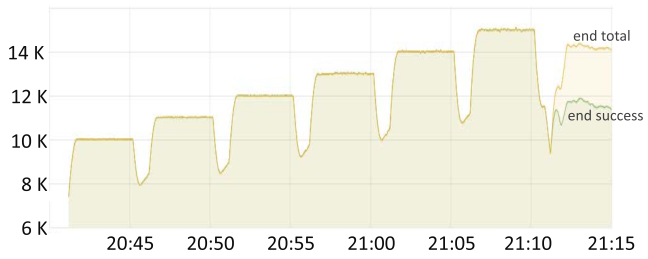


FIGURE 7. Maximum throughput test run: total successful transactions per second ("end total" = all transaction at the end of the test run; "end success" = successful transaction at the end of the test run).

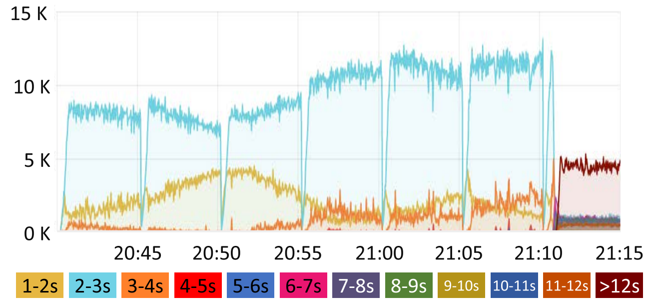


FIGURE 8. Maximum throughput test run: transactions according to transaction durations.

how many failed transactions are *significant*, i.e., need to fail before we consider the situation a system failure; actually, given the success of our test run, you can assume the most strict specification, i.e., we consider the system failing upon occurrence of the first failed transaction.

A transaction rate of 15,000 tx/s of successful transactions has been reached. No attempt was made to re-tune the system (in term of the KSI Cash tuning parameters) to reach even higher transaction rates with the available EC2 node deployment. It can be assumed that even higher transaction rates could have been reached with respective tuning. Still, the result gives a stable lower bound for the possible transaction load with the EC2 node deployment described in Section V-A2.

The test run was initialized with 111 million bills per gateway, and 55 million wallets. The transaction load started with 10,000 tx/s. Then, the load was increased by 1,000 tx/s every 5 minutes, see Fig. 7. Up to a load of 15,000 tx/s, the system operated with a rate of 100% successful transactions.

When the load was increased 16,000 tx/s, the system started failing, see Fig. 7, where we distinguish between all transactions (‘*end total*’) as opposed to successful transactions (‘*end success*’) during the last five minutes of the test run. First, we can see that a significant amount of transactions have failed. Second, we can see that the total number of transactions (including those failing) is less than the target load of 16,000 tx/s, and even less than the 15,000 tx/s of the previous, successful stage of the test run. This was so, because the wallet machines have been overwhelmed with requesting bill proofs at that stage, so that they failed generating the targeted number of transactions per second. Next, also the three most used gateway output components were also overwhelmed by the high number of bill proof requests at that stage. Consequentially, more and more payment orders were queued in their gateway input queues and a series of transactions failed.

We can summarize that, during system failure, the performance bottlenecks have been the wallet machines and the gateway output components. This opens opportunities for further system tuning (beyond optimizing the KSI Cash system tuning parameters) such as the introduction of queuing layer between the front output components and the gateway output components. Again, the target of the test run was not to push the performance to the limit on the basis of the available machine, but to find a reasonable lower bound for the performance of the system with the given compute resources.

Figure 8 shows the transaction durations of this test run. Transaction durations did not increase critically with more load until the system's limit was reached. Some more transactions with a duration of up to 4s appeared with increasing load, however, even at the peak load of 15,000 tx/s, still 99.9% of transactions were still executed in less than 5s.

5) EXCHANGE SERVICE TEST RUNS

The exchange services test runs have been conducted to test the KSI Cash money exchange mechanism and to understand its impact on the overall systems performance.

Three test runs have been conducted. The first one was a test run without usage of the exchange service, in order to have a baseline for comparison with the test runs that use the exchange service, see Section V-B5a. The other test runs provoked some transactions that needed to use the exchange service. In the second test run in Section V-B5b, the re-active exchange strategy was activated, compare with Section IV-B1. In the third test run in Section V-B5c, the pro-active exchange strategy have been activated.

Each of the three test runs was loaded with a steady transaction rate of 5,000 tx/s over a duration of 10 minutes each.

The system was initialized with 40 million bills in each gateway (a total of 560 million bills). A total number of 10 million user wallets has been created. The exchange wallet received 20 million bills. Therefore, on average, each user wallet received 54 bills. In each wallet, the combination of bills was composed in such way, that the wallet could make the first transaction without the need to use the exchange service.

a: NO EXCHANGE BASELINE TEST RUN

During the baseline run, each sending wallet was used only once, therefore it was possible to pay all sums without the help of the exchange service. Figure 9 shows the results of the first test run. The majority of 55.07% of the transactions have been finished in 1-2s, whereas 44.87% have been finished in 2-3s and all of the transactions in less than 4s.

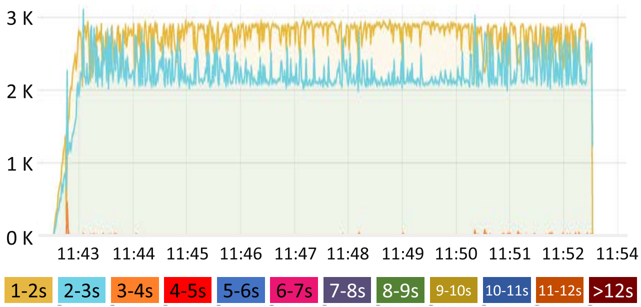


FIGURE 9. No exchange baseline test run: transactions according to transaction durations.

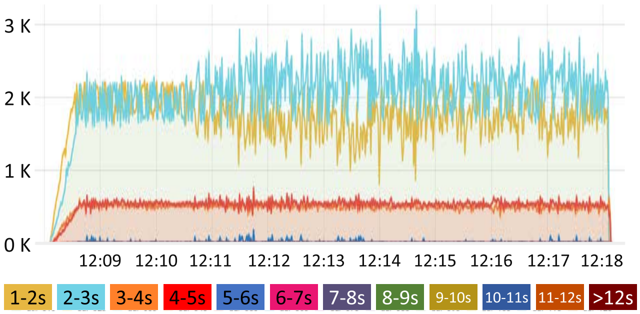


FIGURE 10. Re-active exchange strategy test run: transactions according to transaction durations.

TABLE 7. Exchange service test runs: transaction durations of three test runs with different money exchange strategies (no exchange, re-active, pro-active).

	transaction duration				
Test Run	**1-2s**	**2-3s**	**3-4s**	**4-5s**	**5-6s**
no exchange	55.07%	44.87%	0.06%	–	–
re-active	36.11%	42.15%	10.30%	10.94%	0.46%
pro-active	27.00%	68.91%	4.06%	0.03%	–

b: RE-ACTIVE EXCHANGE STRATEGY TEST RUN

The second test run was started after the first (baseline) test run from Section V-B5a without re-initialization, i.e., the wallets remained in the state as produced by the first test run. Therefore, some wallets missed some denominations for enacting a transaction at this point and had to use, re-actively, KSI Cash exchange wallet. This increased the transaction durations as can be seen in Fig. 10, when compared with Fig. 9. As opposed to the baseline test run, less transaction were finished in 1-2s (36.11%), more transaction needed 2-3s resp. 3-4s (42.15% and 10.3%) and transactions with longer duration than 4s also occurred (>11%), see also Table 7 for a summary.

c: PRO-ACTIVE EXCHANGE STRATEGY TEST RUN

For the third test run, the system was re-initialized with the same data as the first run (40M bills, 10M wallets, 20M bills in the exchange wallet etc.). In this run, the pro-active money exchange strategy was activated, i.e., an extra money exchange round was triggered after each transaction,

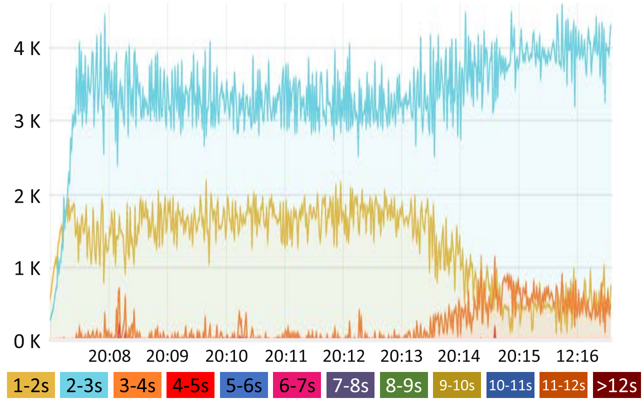


FIGURE 11. Pro-active exchange strategy test run: transactions according to transaction durations.

whenever it was in a state that could not serve directly all possible transaction requests any more, see Section IV-B1. Again, as compare to the baseline run in Section V-B5a, transaction durations increased, with less transactions of 1-2s (27%), more transactions of 2-3s (68.91%) and also more transactions longer than 3s, see Fig. 11 and again Table 7 for comparison.

When comparing the third test run (pro-active) with the second test run (re-active), there are much more transactions in the range of 2-3s in the pro-active run than in the re-active run. With 68.91%, it can be said that the bulk of transactions is 2-3s in the pro-active run, whereas even longer transaction more often occur in the re-active run (>21% as opposed <5% in the pro-active run). When comparing the overall averages of the two runs, the pro-active run is slightly better (2.2712s) than the re-active run (2.4735s). From these results, it is not possible to decide clearly, which of the money exchange strategies is better. More, longer simulations or mathematical modeling would be needed. At least, the results seem to indicate that the pro-active strategy should be preferred whenever the optimization goal is in minimizing very long transaction durations (>3s).

6) PEAKS AND LOWS TEST RUN

This test consisted of six phases of different transaction rates, each with a length of 18 minutes as follows (with the system standard initialization: 2.8B bills, 100M wallets):

1. 100 tx/s
2. 1,000 tx/s
3. 10,000 tx/s
4. 100 tx/s
5. 1,000 tx/s
6. 10,000 tx/s

The idea of the test run was to simulate peaks and lows in a real world environment, for the purpose of investigating gateway calculation time behavior under realistic conditions.

Figure 12 shows the result of the test run in terms of gateway *round calculation times* (for 14 gateways, one colored curve per gateway node in Fig. 12), i.e., the calculations made by a gateway during one round.

TABLE 8. Memory consumption of gateway nodes and gateway output components.

gateway node, 8,000 payment orders per second under full load					
Nr.	**Number of Bills per Gateway**	**RAM Usage Before Initialization**	**RAM Usage During Initialization**	**RAM Usage After Initialization**	**RAM Usage Under 10,000 tx/s Load**
#1	2,000,000	0.27GB	1.18	0.62GB	1.16GB
#2	20,000,000	2.89GB	12.31	6.41GB	12.42GB
#3	200,000,000	27.3GB	69.43	62.52GB	70.21GB
gateway output component, 9,500 bill proof request per second under full load					
Nr.	**Number of Bills per Gateway**	**RAM Usage Before Initialization**	**RAM Usage During Initialization**	**RAM Usage After Initialization**	**RAM Usage Under 10,000 tx/s Load**
[1mm] #1	2,000,000	0.27GB	2.05GB	1.21GB	2.18 GB
#2	20,000,000	2.89GB	17.24GB	11.94GB	23.97GB
#3	200,000,000	27.31GB	119.43GB	117.52GB	167.07GB

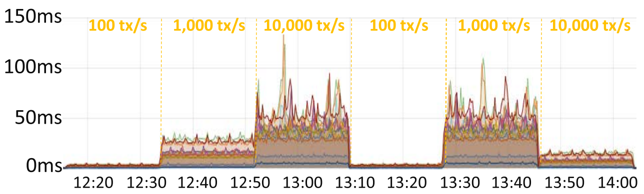


FIGURE 12. Gateway round calculation time of 14 KSI Cash backend gateway nodes.

Definition 3 (Gateway Round Calculation): A *gateway round calculation* includes:

- Validation of payment orders; here, processing is dominated by ECDSA (Elliptic Curve Digital Signature Algorithm) [91] signature checks. Go's cryptography library has been used for the purpose of signature checks.[24]
- Updating the state tree of the gateway machine. This works in batch mode, i.e., some leaves of the Merkle tree are changed when the represented bills change ownership, and then the Merkle tree is re-calculated to update its root, see Section IV-A2.

The gateway round calculation times increase with system load. The round calculation needs to fit into the standard time window of 600ms of the test run, i.e., round time (1,000ms) minus payment order reading time (300ms) minus core offset time (100ms), see Section V-A2. Even under a high load of 10,000 tx/s, the gateway calculation time never exceeded 150ms (with an average of approx. 75ms), see Fig. 12.

7) MEMORY CONSUMPTION TEST RUNS

The purpose of these test runs was to investigate the memory (RAM) consumption of the gateway nodes as well as the gateway output components in regard to different amounts of bills under production condition, i.e., 10,000 tx/s load. Three runs for have been conducted, one with 2 million bills, a second with 20 million bills and a third with 200 million bills per gateway, each with 100 million wallets. This means that the third run was run with the standard initialization (2.8B bills, 100M wallets). In each run we measured the memory usage (peak usage) of gateway nodes and gateway output components in four phases, i.e., before system initialization, during system initialization, after system initialization (when transactions have not yet been started) and during rounds with a load 10,000 tx/s. The durations of the test runs (in the narrow sense), i.e., after starting rounds have been 5 minutes each.

The results of the test runs are listed in Table 8. We have arbitrarily chosen one particular, representative gateway node and its output component for Table 8. The 10,000 tx/s transaction load had a footprint of approx. 8,000 payment orders per second for the particular gateway node, and a footprint of 9,500 bill requests per second for the particular gateway output component. Gateway memory consumption is in positive correlation to the number of bills. The memory consumption of gateway output components is higher than memory consumption of gateway nodes. Memory consumption is higher after initialization than before system initialization. Memory consumption is higher during transactions than before the starting transactions (after initialization). Memory consumption is higher during initialization than after initialization. This is due to the garbage collector of the Go runtime environment. However, it is only slightly higher in the case of many bills (test run #3). The peak memory usage of the gateway node was approx. 70GB and the peak memory usage of the gateway output component was approx. 167GB. This means that the available RAM (256GB) of the used virtual machines (compare with Table 6) has been under-utilized during these test runs.

8) 24 HOURS TEST RUN

The 24 hours test run was conducted in service of testing availability. The system was loaded with a moderate transaction load of 575 tx/s over a time of 24 hours (with standard initialization: 2.8B bills, 100M wallets), Fig. 13. All of the transactions were successful, i.e., system availability during the test run was 100%. All transactions were executed within 3 seconds, where the majority of the transactions were executed within 2 seconds, see Fig. 13.

[24] https://pkg.go.dev/crypto/ecdsa

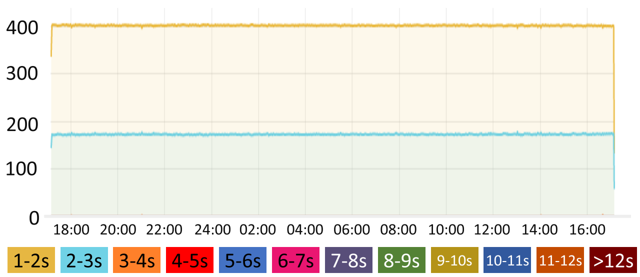


FIGURE 13. 24 hours test run: transactions according to transaction durations.

C. ULTRA-SCALABILITY TEST RUNS

We have conducted a series of test runs to show that KSI Cash shows unlimited performance through linear scalability in terms of needed compute resources. We call these test runs *ultra-scalability test runs*. The target was to scale out the system to a million payment orders and beyond. Conducting tests with such a high load on the full backend system and the full wallet simulation as provided by Section V-A2 and Section V-A3 would have been very costly without a clearly visible advantage of having the realistic usage simulation. We argue that ultra-scalability test runs provide enough evidence for unlimited performance although they are conducted under more simplified conditions, i.e., in what we call a *laboratory setting*; in particular, when combined with the evidence provided by the realistic test runs of Section V-C.

Figure 14 shows the KSI cash system configuration for the ultra-scalability test runs, please also compare with Fig. 3. For a specification of the nodes of the system, see Table 9, compare with Table 6. The major difference is in dropping the gateway output components. This means the ultra-scalability test runs do not actually construct and store the blockchain anymore. Also, the generating of bill proofs is not tested any longer. Consequentially, the ultra-scale test system also needs no front output components. Instead, the block creation is tested consisting of all necessary gateway calculations, including the validation of payment orders, updates to the state tree of the gateway machines (see Def. 3), and provision of block signatures by the core system's core component. We propose that such a test design provides enough evidence for the intended scalability result.

We name the gateways together with the core module and the controller module the *central components* of the system, as together they are responsible for the block creation.

As a minor detail, the stripped-down system no longer has a high-availability proxy. Instead, the system contains load generator nodes. These are added to the system and take over the role of the KSI Cash wallet simulation from Section V-A3. The load generation was conceptually simplified, i.e., payment orders were generated directly, and not on behalf of (as part of) simulated transactions. The load generators were single-threaded applications which were executed in parallel on 100 virtual machines. Each load generator was responsible to generate a load of 5,000 payment orders per second.

Some further details of the simplified conditions of the ultra-scalability test runs were as follows:

- No TLS, no session authentication for incoming payment orders; instead, the load generators send payment orders to appropriate gateways.
- No bill proof generation nor bill proof requests, instead relying on the gateways' internal logging to confirm that payment orders are verified and processed. Results are collected through metrics collection.
- No storage of blocks in the ledger database. Once blocks are created by a gateway together with an associated certificate provided by the core, the blocks are discarded.
- The gateways are relatively small, each of them handling 1 million bills.

We conducted four test runs with increasing compute resources. Table 10 shows the specification of the test runs together with the results. Tests were run with 25, 50, 100 and 200 gateways, deployed on the same number of virtual machines for each test run, with twice as many load generators for each test run. The load generators of the first three test runs were deployed to 25, 50 and 100 gateways respectively, where each single load generator produced a load of 5,000 payment orders per second. The system was configured so that each gateway processes 10,000 payment orders per second. In the fourth test run, we were short of 2 million payment orders per second. This was due to the fact that we could deploy the 400 load generators only to 100 virtual machines (as in the test run with 200 load generators). Therefore, the load generators' virtual machines' CPU usage reached 100% and could not produce enough payment orders. Therefore, the slightly smaller number of 1,960,000 payment orders per second was not due to any KSI Cash related limitations.

Assuming that a payment transaction consists on average of six payment orders (e.g., as noted in Section V-B1c, the results illustrate that KSI Cash operates effectively supplying more than 300,000 successful transactions per second.

It can be summarized that the ultra-scalability test runs demonstrate that KSI Cash is capable of processing approx. 2 million payment orders per second, scaling linearly in the available compute resources. The result is not surprising where the purpose of the test runs was to demonstrate that these transaction loads are possible, thus we conclude that even greater transaction loads are possible with additional compute resources.

D. KSI CASH CARBON FOOTPRINT

1) CARBON FOOTPRINT PER TRANSACTION

We provide an estimation of the carbon footprint of KSI Cash on the basis of the production test run in Section V-B3, i.e., on the basis of 10,000 tx/s and the compute resources needed for this test run, see Table 6.

For the estimation, we consider only components related to the implementation of KSI Cash, including orchestration

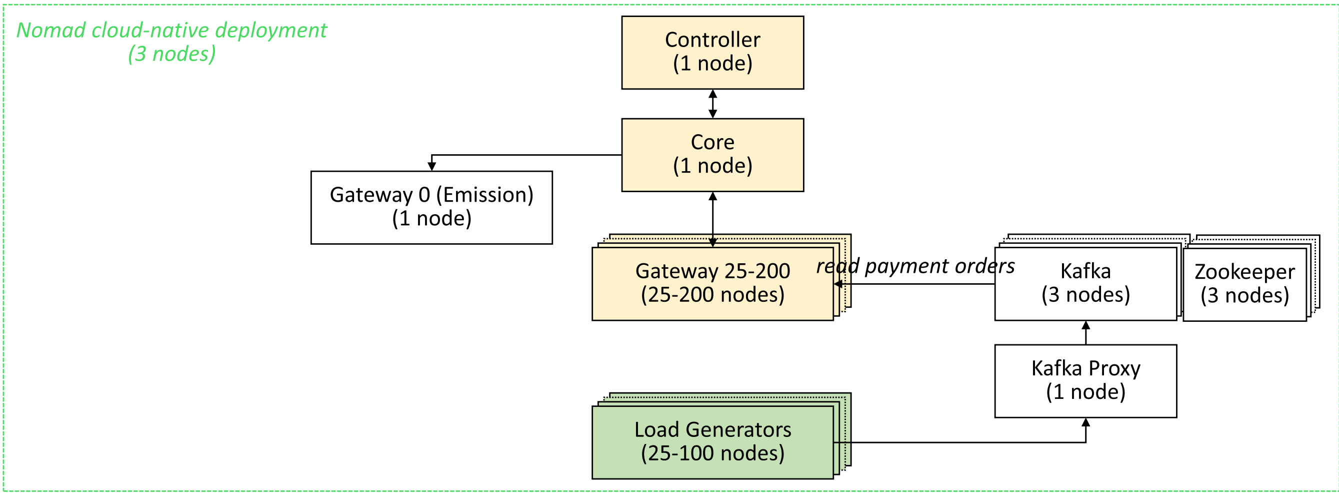


FIGURE 14. The KSI cash system configuration for the ultra-scalability test runs.

TABLE 9. Server configuration of the KSI Cash ultra scalability performance test run, with node names according to Fig. 14.

Node Name	vCPUs	RAM	Storage	Processor	EC2 Instance Type	VMs
Controller	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Core	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Gateway-0	2	4GB	20GB	AMD EPYC 7002	c5a.large	1
Gateway 1–200	8	16GB	20GB	AMD EPYC 7002	c5a.2xlarge	25-200
Kafka	8	32GB	20GB+4TB	AMD EPYC 7000	t3a.2xlarge	8
Kafka Proxy	4	16GB	20GB	AMD EPYC 7000	t3a.xlarge	1
Zookeeper	2	1GB	20GB	AMD EPYC 7000	t3a.micro	3
Nomad Server	2	2GB	20GB	AMD EPYC 7000	t3a.small	1
Load Generators	4	8GB	20GB	AMD EPYC 7002	c5a.xlarge	25-100

TABLE 10. Linear scalability of the transaction rate (in terms of payment orders) in dependence of the number of deployed gateways.

Number of Gateways	VMs (Gateways)	Number of Generators	VMs (Generators)	Payment orders per second
25	25	50	25	250,000
50	50	100	50	500,000
100	100	200	100	1,000,000
200	200	400	100	1,960,000

and edge networking, i.e. KSI Cash server nodes in Table 6. Load generation and analytics-related components are not included into the calculation. In total, 1,382 vCPUs (virtual CPUs) and 8.27 GB of RAM were provisioned. With respect to resource consumption, there was room for optimization, i.e., the weighted average CPU usage during the production test run was below 20% and the memory load was below 40%.

The experiments were run in the Amazon EC2 cloud environment. Amazon does not disclose its energy efficiency and environmental footprint in necessary detail; therefore, we use indirect calculations with the Dell EMC Enterprise Infrastructure Planning Tool.[25] For the estimation, we assemble a comparable hardware setup in the tool, choosing comparable high-end components and assuming virtualization.

[25]http://dell-ui-eipt.azurewebsites.net/

In order to substitute the needed vCPU count and amount of memory, we calculate with 6 instances of Dell PowerEdge R940 servers, each configured with 4 Intel Xeon 8180 processors, each providing 28 cores and 56 threads. The servers are configured with 44 RDIMM memory modules, each 32GB, which yields 1.4 TB RAM per server. The CPU maximal power requirement is 205 W, and memory approximately 200 W. In total, these 6 servers amount to 1,344 CPU threads and 8,448 GB of memory. We take these 1,344 CPU threads as an equivalent of 1,344 vCPUs in our production test run, as Amazon EC2 markets one CPU thread as one vCPU.

Dell's planning tool estimates the following power consumption for our configuration:

- in idle mode: 304 W
- under memory-intensive load: 1030 W
- under potential maximum load: 1704 W

In our estimation, we work with 6 kW under memory-intensive load for our six-server configuration. We add 1 kW for network equipment and assume an equal power consumption for cooling and other data-center losses, i.e., we work with a total of 14 kW for our scenario. Power consumption for cooling and auxiliary data center resources must not be neglected [92]. A factor of two is a usual estimate used in practice, and we therefore consider this factor as appropriate for the purpose of a rough estimation.

Next, we assume EU's 2019 average[26] of 275g CO2/kWh of carbon footprint. Based on that we can conclude the following estimations for the carbon footprint of KSI Cash:

- 34t CO2 per year (with permanent load of 10,000 tx/s)[27]
- 0.0001g CO2 per transaction[28]

Our estimation does not have the ambition to give an exact estimate of the carbon footprint of the KSI Cash production system. In real-world scenarios, additional resources might be required, e.g., to ensure a higher degree of availability, or due to a larger amount of bills covering a larger money supply. Still, we would suppose that, even under a pessimistic approach, this would not increase our estimation by more than 3–5 times. On the other hand, there is also the potential of energy savings, e.g., by further optimizing the KSI Cash system or using optimized hardware. To conclude, we suggest that our carbon footprint estimation is accurate enough to be used in orders-of-magnitudes comparisons, such as needed when comparing with Bitcoin and proof-of-work consensus, see Section V-D2.

2) BITCOIN CARBON FOOTPRINT

The energy consumption and carbon footprint of Bitcoin and proof-of-work consensus is alarming [93]. A study [94] from the Frankfurt School Blockchain Center from 2021 estimates the carbon footprint of Bitcoin as approx. 369.5 kg CO2 per transaction. This number is based on an estimation of the annual energy consumption of 90.86 TWh (in the period of 1 September 2020, to 31 August 2021), an estimation of the CO2 emission of 37.97 Mt CO2 and a total of 102,754,276 transactions in that period. The number of transactions is taken from the *statista.com* portal.[29] According to the *blockchain.com* portal, the number of confirmed Bitcoins transactions in that period[30] is 34,379,872, which would amount to a carbon footprint of 1,104.4 kg CO2 per transaction (based on the 37.97 Mt CO2 estimated annual emission from above); whereas the number of confirmed Bitcoins payments is 238,745,466,[31] which would amount to a carbon footprint of 159 kg CO2 per payment. In Bitcoin, a transaction can contain several payment to different recipients (called *batching*). The batched payments must not be confused with the payment orders of KSI Cash. In KSI Cash, a transaction consists of one or more payment orders; when comparing energy consumption of KSI Cash and Bitcoin, it is the Bitcoin batched payment that correspond to a KSI Cash transaction. Henceforth, we talk about a Bitcoin batched payment also as *payment transaction*.

A study [95] of the central bank of the Netherlands (De Nederlandsche Bank) from 2021 comes up with similar results, i.e., an annual energy consumption of 70 TWh for 2020 (45 Mt CO2; 402 kg CO2 per transaction) and 54 TWh for 2019 (36 Mt CO2; 300 kg CO2 per transaction).

Other estimates are:

- 21.5 Mt CO2 to 53.6 Mt CO2 in 2018 [96]
- 45.8 TWh and 22 Mt CO2 in 2018 [97]
- 1,216.51 kg CO2 per transaction[32]

Similar estimates are also reported in [98].

Given these data, it can be summarized that the carbon footprint of Bitcoin may be estimated as larger than 100 kg CO2 per transaction payment.

VI. ESSENTIAL ALPHABILL PLATFORM COMPONENTS

A. FUNDAMENTAL ALPHABILL DESIGN CONCEPTS AND ELEMENTS

1) ASSET REPRESENTATION

Each kind of asset is implemented by an individual transaction system having individual characteristics. Each partition maintains asset entities (units) and enables transactions that create new entities, delete identities, or change the attributes of entities.

Definition 4 (Entity Representation): Each entity is represented as a triple $\langle \iota, D, \varphi \rangle$ where:

- ι is a unique identifier of the entity.
- D is the data part that encompasses all relevant attributes of the entity.
- φ represents ownership – φ is a predicate (logical condition) that defines the rules and restrictions of the next transaction with the entity, i.e., so that in order to execute the next transaction T with the entity, the initiator has to provide an ownership proof s such that $\varphi(T, s)$ holds.

The system identifiers are totally ordered, i.e., for each pair of identifiers ι_1, ι_2, either $\iota_1 = \iota_2$, $\iota_1 < \iota_2$, or $\iota_2 < \iota_1$.

Ownership predicates are an essential concept. For example, in Bitcoin, such ownership predicates can be found in the form of so-called locking scripts [44]. The simplest example of an ownership predicate is public-key based signature verification, i.e.,

$$\varphi(T, s) \equiv \mathsf{Verify}(\mathsf{pk}; T, s),$$

which holds if s is a signature on T that verifies with the public key pk, where the public key pk is an additional parameter of φ.

2) PLATFORM ELEMENTS

All kinds of transaction systems can be integrated into the Alphabill infrastructure. It is assumed that all partitions follow the Alphabill design principles as described in Section III-B, in particular, achieving highest security and scalability. The Alphabill platform enables the integration of transaction systems and supports the adherence to its design principles by providing a series of elements and respective behavior in the Alphabill platform as follows:

[26] https://www.eea.europa.eu/data-and-maps/indicators/overview-of-the-electricity-production-3/assessment

[27] $14\text{kW} \times 24\text{h} \times 365.25\text{d} \times 275\text{g CO2/kWh} = 33{,}749\text{t CO2}$

[28] $14\text{kWh} \times 275\text{g CO2/kWh} / 3{,}600\text{s} / 10{,}000\text{tx} = 0.0001069\text{g CO2/tx}$

[29] https://www.statista.com/statistics/730806/daily-number-of-bitcoin-transactions/

[30] https://www.blockchain.com/charts/n-transactions

[31] https://www.blockchain.com/charts/n-payments

[32] https://digiconomist.net/bitcoin-energy-consumption

- The Alphabill platform defines a language for describing the functionality of transaction systems (syntax and semantics of state and transactions),
- provides libraries and toolkits for developing blockchained transaction systems in the Alphabill framework,
- registers and assigns identifiers α to transaction systems, based on the descriptions of the transaction systems,
- provides ledger certificates, through the blockchain mechanism, i.e., irrefutable proofs of the current (block n), the parameters D of entities and the transactions T with entities.

The Alphabill development libraries and toolkits unlock the technological know-how in sharding of KSI Cash and Alphabill Money to be used for implementing massively scalable transaction systems. All transaction systems that exploit these libraries and tools will show a design consisting of shards and a core system as described in detail in Section IV-A, see Fig. 1. The ledger certificates provided by the Alphabill platform are described in Section VI-A3.

In order to integrate a transaction system as a partition into the Alphabill platform, the steps described in Section VI-A8 have to be followed.

3) LEDGER CERTIFICATES

The Alphabill platform supports three kinds of ledger certificates:

- *Transaction Certificate*. A transaction certificate proves that a transaction T was included into the n-th block of a transaction system.
- *Data Certificate*. A data certificate proves the value of D (its attributes) of an entity ι in a block n.
- *Non-existence Certificate*. A non-existence proves that the entity with identifier ι does not exist in round n. Non-existence certificates play a crucial role in atomic multi-asset swap transactions.

A ledger certificate consists of:

- the *authentication path* of the state tree of the n-th block,
- the *uniqueness certificate* of the n-th block which depends on the root of the state tree and the consensus mechanism of the partition.

Alphabill partitions may be based on different kinds of state trees such as:

- *Pure Merkle tree* [86] – supports transaction certificates and data certificates, but does not support non-existence certificates.
- *Authenticated search tree* [99] – provides all three kinds of ledger certificates.
- *Count-certified hash tree* [100] – provides all three kinds of ledger certificates, and, in addition, also a proof of a summary value or an invariant of the transaction system (e.g., the total amount of money in case the transaction system represents a money scheme).

These three major kinds of sate trees can be used in different variations with additional security features such as chain length certification [101] and node type (leaf/non-leaf) certification [102].

4) SECURITY MEASURES

This paper does not aim to contribute to the field of protecting systems against the various forms of cyberattacks such as DDoS (distributed denial-of-service) attacks. All the best practices of protecting services and blockchains against cyberattacks are also applicable in Alphabill. We assume that the used consensus protocol guarantees that only one version of the uniqueness certificate is produced. On the other hand, it can be shown that whenever an adversary is able to produce two versions of Alphabill's ledger that both verify against the same uniqueness certificate, a hash collision can be extracted from these two versions and the uniqueness certificate. The proofs follow the reduction techniques used in provably secure time-stamping [99], [100], [103], [104] and are omitted in this paper.

5) TRANSACTION ORDERS

In the Alphabill platform, a unified message format is used for all transaction orders.

Definition 5 (Transaction Order Format): Each transaction order P in a transaction system is a tuple $\langle \alpha, \tau, \iota, A, T_0 \rangle$, consisting of:

- α – the system identifier
- τ – the message type identifier
- ι – the entity identifier
- A – a list of transaction attributes
- T_0 – a timeout specification

As usual, we use dot notation to denote the several items of a tuple, e.g., we use $P.\alpha$ to denote the system identifier of a transaction order P.

The concrete format of the list of attributes A depends on the message type τ. The timeout T_0 represents the expiry time of the message in terms of block numbers, i.e., P can be accepted in block n only if $n < T_0$. Timeout specifications allow for providing reliable evidence that a transaction has not and will not be accepted by the system.

A *signed transaction order* is a pair $\langle P, s \rangle$, where P is a transaction order and s is an ownership proof, such that $\varphi(P, s) = 1$, where $\langle \iota, D, \varphi \rangle$ is an entity.

6) SHARDING SCHEME

The Alphabill platform suggests a unified sharding strategy for all transaction systems. Each shard is assumed to manage a part of the partition's entities within a range of identifiers ι (see Def. 4) from an interval $[\iota_{\min}, \iota_{\max}]$.

Every shard has a shard identifier γ that is a finite bit string. Initially, the partition might start as consisting of only one single shard with the empty bit string as identifier γ. Upon a certain event trigger (e.g., a threshold number of transactions per shard), the shard with identifier γ is split into two shards with identifiers γ_0 and γ_1. The state tree of the shard is split by applying a tree-splitting algorithm. Hence, the set Γ of all shard identifiers is always a prefix-free code.

Definition 6 (Sharding Scheme): A sharding scheme is represented as a pair $\langle \Gamma, \rho \rangle$, where ρ is a function on Γ

that, given a shard number γ as input, outputs the interval $[\iota^{\gamma}_{\min}, \iota^{\gamma}_{\max}]$ of the shard γ.

7) TRANSACTION SYSTEM SPECIFICATION

Each partition in the Alphabill network is completely defined by a set of parameters, called *transaction system specification*.

Definition 7 (Transaction System Specification): A transaction system specification consists of

- α – the system identifier
- the type specification (meta data) of the data part D of the partition's entities
- a state tree type
- the initial sharding scheme and the shard-splitting trigger
- Σ_0 – the initial state of the system
- a set $\mathbb{T}$ of message types
- for each message type $\tau \in \mathbb{T}$:
  - A_τ – a list of attributes forming the message content
  - a specification of the message's effect onto the system state

Message types could also be called transaction message types or just transaction types for short, however, message type is common terminology in transaction system and data interchange technologies and standards such as in the field of EDI (electronic data interchange) [105]. Given a transaction of message type τ, we call this transaction also just a τ-transaction for short.

8) JOINING THE ALPHABILL PLATFORM

Joining a new transaction system into the Alphabill platform requires the following steps:

1. Describing the new system by defining the parameters of the transaction system, see Def. 7
2. Applying Alphabill toolkits to compile the system code
3. Preparing machines for each shard and deploying the code to machines
4. Registering the system and obtaining the system identifier α

The concrete registration procedure depends on design choices. In case of automatic registration, the registration is regulated by ledger rules, i.e., registration is a special blockchain transaction.

9) INTEGRATION WITH OTHER BLOCKCHAINS

One of the main design goals of the Alphabill platform is efficient certified communication between the partitions and shards of the partition inside Alphabill. The main resource of efficiency is that all shards and partitions share the same uniqueness certificate framework.

Efficient communication with other blockchains depends on how the certified communication between Alphabill and other blockchains can be organized. For efficient verification of certified messages from other blockchains, Alphabill's smart contract language has built-in cryptographic functionality of verifying digital signatures and hash chains. Whether another blockchain can verify Alphabill's certified messages depends on the flexibility and programmability of the blockchain. For example, blockchains with Turing-complete smart contract language such as Ethereum's Solidity [106] have sufficient flexibility for verifying Alphabill's certificates. For less flexible blockchains, interoperability solutions such as LayerZero [107] can be used.

B. ALPHABILL MONEY

Pure bill-type money schemes only use ownership transfer transactions, i.e., such transactions that change only the ownership conditions of bills. Pure bill schemes enable massively parallel decomposition of the money system [21], [22], but also have shortcomings. Similar to physical cash, it is not always possible for a party to pay exact amounts and therefore, some additional services, such as a money exchange service, are needed, compare with Section IV-B.

To implement a digital currency based on a pure bill-type money scheme, one has to start with a large number of bills existing already from day one. For a central bank money, such a solution is sufficient as a central bank (central authority) can always change the distribution of the monetary denominations by its authority. However, in the permissionless case, having too many bills might quickly become too costly for the validators of the blockchain. Here, it would be reasonable to start from a few high-value bills, enable the users to split the bills if needed and to provide an efficient mechanism to subsequently join bills. This idea is the design rationale behind our proposed dust collection solution in Section VI-B1 and VI-B2.

1) SPLIT PAYMENTS

An *extended bill money scheme* addresses the shortcomings of pure bill schemes by introducing *split payments* that make sure that exact payments are always possible. When applied to a bill, a split transaction reduces its value by an amount n and creates a new bill with that value n.

Split type transactions enable exact payments but introduce the critical problem that too many small-value bills (*dust bills*) emerge over time. Therefore, additional transactions and ledger mechanisms are needed to reduce the amount of dust bills by joining them to larger bills. We address the issue of dust bills in Section VI-B2 by introducing new types of transactions along with a specialized type of bill.

2) DUST COLLECTOR AND BILL SWAP PAYMENTS

In the extended bill scheme of Alphabill Money, we introduce a special type of ownership – the *dust collector* (DC). Bills owned by the DC are not in the usual money circulation but can be considered as frozen by the system. For the purpose of consolidating dust bills, further transactions with the DC-owned bills are conducted automatically by the ledger operation rules (block creation rules) and, hence, DC represents a built-in smart contract (a smart contract built in the system).

Furthermore, there is a special DC-owned bill with identifier ι_0 called *dust collector money supply*. The system is sharded and every shard must have its own DC money supply.

Users who want to get rid of their dust bills, can transfer them to DC via special ownership transfer transactions of message type transDC and get ledger certificates of them. These transactions are processed by the system in such a way that a new bill with value n is issued to the owner specified in the transaction orders, and simultaneously, the dust collector money supply (the value of the bill ι_0) is reduced by n.

By presenting those certificates to the system, users can then obtain new larger bills via bill swap transactions. Each bill swap transaction contains a list of transDC transactions with ledger certificates and an owner condition φ. Once a bill swap transaction is received by a shard of the Alphabill native currency partition, the ledger certificates of the listed transDC transactions are verified and a new bill is created with the sum value of the dust bills and with φ as owner. A bill swap transaction has a unique identifier that uniquely points to the shard in which it can be executed. This prevents that the same bill swap transaction is used in two different shards, i.e., double-spending the bill swap transaction.

Merely transfers to DC and corresponding swaps are not sufficient to reduce the number of small-value bills in the system. In order to achieve a consolidation of small-value bills, we need to introduce a mechanism for joining dust bills, as we will discuss in due course in Section VI-B2b.

a: BILL SWAP SCENARIO

To enact a DC swap, a user wallet selects a set of dust bills with identifiers $\iota_1, \ldots, \iota_m$ and values $v_1, \ldots, v_m$, respectively. The bill swap procedure consists of the following steps:

1. The user wallet computes
$$\iota \leftarrow h(\iota_1, \ldots, \iota_m),$$
where h is a cryptographic hash function.
2. For every dust bill ι_k, the wallet creates a signed transDC transaction order P_k that contains ι, an ownership condition a for a potential exchange bill, a timeout T_0, and sends P_k to the system.
3. It might be that some of the payment orders do not reach the system in time. Let $P_{k_1}, \ldots, P_{k_\ell}$ be the payment orders that were successfully received by the system. For these payment orders, the wallet obtains ledger certificates (transaction certificates) $\Pi_{k_1}, \ldots, \Pi_{k_\ell}$.
4. For the bills (from the list $\iota_1, \ldots, \iota_m$) that have not reached the system in time, the wallet obtains ledger certificates (data certificates) that the ownership in block $n \geq T_0$ has not changed. This ensures that these payments indeed failed.
5. The wallet sends a signed bill swap transaction $\langle P, s\rangle$ to the system, where
$$P = \langle \alpha, \mathsf{swap}, \iota, A, T_0\rangle,$$
$$A = \langle a, \iota_1, \ldots, \iota_m; P_{k_1}, \ldots, P_{k_\ell}, \Pi_{k_1}, \ldots, \Pi_{k_\ell}, v'\rangle,$$
and
   - a is the ownership of the potential new exchange bill,
   - $\iota_1, \ldots, \iota_m$ are the identifiers of the dust bills,
   - $P_{k_1}, \ldots, P_{k_\ell}$ are the successful payment orders of type transDC,
   - $\Pi_{k_1}, \ldots, \Pi_{k_\ell}$ are the ledger certificates of $P_{k_1}, \ldots, P_{k_\ell}$,
   - $v' = v_{k_1} + \ldots + v_{k_\ell}$ is the value of the exchange bill.
6. The system checks the following required conditions:
   - (i) $v' = v_{k_1} + \ldots + v_{k_\ell}$, where $v_{k_1}, \ldots, v_{k_\ell}$ are the values of the corresponding bills (v_{k_i} is included in P_{k_i}).
   - (ii) $v' \leq v_0$, where v_0 is the current value of the DC money supply, i.e., where there exists a sufficient DC money supply.
   - (iii) there exists no bill with identifier ι. If the hash function h is collision-resistant, this condition only fails in case of an attempt to use the same swap message twice, possibly, to maliciously double spend it.
   - (iv) $\{P_{k_1}.\iota, \ldots, P_{k_\ell}.\iota\} \subseteq \{\iota_1, \ldots, \iota_m\}$ – all bill identifiers in payment orders are elements of $\{\iota_1, \ldots, \iota_m\}$
   - (v) $\iota = h(\iota_1, \ldots, \iota_m)$ – the identifier ι of the new bill is properly computed.
   - (vi) $P_{k_1}.\tau = \ldots = P_{k_\ell}.\tau =$ transDC – bills were transferred to DC
   - (vii) $P_{k_1}, \ldots, P_{k_\ell}$ all contain ι.
   - (viii) the payment orders $P_{k_1}, \ldots, P_{k_\ell}$ all contain the proper ownership condition a (of the potential new exchange bill),
   - (ix) $a(P, s) = 1$, i.e., s is the signature of a on P,
   - (x) $\Pi_{k_1}, \ldots, \Pi_{k_\ell}$ are proper ledger certificates for $P_{k_1}, \ldots, P_{k_\ell}$.
7. If all of the required conditions have been verified, a new bill, represented by the triple $\langle \iota, v', a\rangle$, is created.

Now, let us analyze the security of the bill swap procedure.

It is not possible to obtain two different exchange bills (say, with identifiers $\iota \neq \iota'$) for the same set of dust bills, because the identifier ι is a deterministic function of $\iota_1, \ldots, \iota_m$.

The same swap cannot be used twice in a shard, because the swap transaction with bill identifier ι is not accepted whenever a bill with the identifier ι already exists.

The same bill swap transaction cannot be used in two different shards, because the sharding scheme uniquely defines the shard of ι.

The ownership a of the new bill cannot be modified by outsiders, because a is included in the payment orders $P_{k_1}, \ldots, P_{k_\ell}$ and this inclusion is verified.

It is not possible that swap dust bills are controlled by somebody else, because the signature s of the swap order has to contain a signature s that satisfies a.

If some (or even all) of the transDC payment orders are not received in time, their ownership does not change and they can be swapped next time. If just one transDC payment order goes through, the owner of this dust bill just receives an equivalent dust bill in a (possibly) different shard. Hence, there is no risk of losing money during a swap.

b: DUST COLLECTION PROCESS

In the extended bill scheme, dust collection is introduced as a necessary automatic functionality related to block creation, i.e., regularly (in a way well-defined by the ledger rules), each block creator has to delete a set

$$(\iota_1, v_1, DC), \ldots, (\iota_k, v_k, DC)$$

of dust bills and, simultaneously, raise the DC money supply (ι_0, v_0, DC) accordingly by increasing its value by

$$d = v_1 + \ldots + v_k.$$

All the activities related to dust collection preserve the total money of the system, including DC money.

The DC money is a purely technical, system-related measure and does not actively participate in the usual "business transactions", i.e., it is not meant to be directly accessed by system users.

3) THE STATE TREE OF ALPHABILL MONEY

Alphabill Money uses an AVL-type authenticated search tree, see Adelson-Velsky and Landis [108], where the indices are provided by the bill identifiers. In addition, it is a count-certified tree where the counters in nodes represent the total value of the corresponding sub tree. All nodes, not only the leaf nodes, contain data about bills.

The tree is represented as a pair $\langle \iota_r, N \rangle$, where ι_r is the bill identifier of the root node and N is an indexed array (dictionary) of nodes. We use $N[\iota]$ to denote the node that is associated with the bill with identifier ι. If there is no bill with identifier ι, we write $N[\iota] = \perp$.

Definition 8 (Alphabill Money State Tree Node): An Alphabill Money state tree node $N[\iota]$ is a tuple $\langle \varphi, v, x, V, h, \iota_L, \iota_R, d, b \rangle$ consisting of the following:

- φ – owner condition
- v – value of the bill
- x – bill hash computed as $H(x', T)$, where H is a hash function, T is the last transaction order with the bill ι and x' is the previous bill hash of the same bill.
- V – summary value computed as
$$V \leftarrow v + N[\iota_L].V + N[\iota_R].V,$$
where we assume that $N[\perp].V = 0$
- h – hash of the node, computed as
$$h \leftarrow H(\iota, H(\varphi, v, x), V; h_L, V_L; h_R, V_R),$$
where h_L, h_R are the hashes of nodes $N[\iota_L], N[\iota_R]$, and V_L, V_R are the summary values of $N[\iota_L], N[\iota_R]$ under the assumption that the hash of $\perp$ is the zero-hash 0_h
- ι_L – left node identifier (can be $\perp$ if there is no left node)
- ι_R – right node identifier (can be $\perp$ if there is no right node)
- d – depth of the sub-tree computed by
$$d \leftarrow \max\{N[\iota_L].d, N[\iota_R].d\} + 1,$$
assuming that $N[\perp].d = 0$.
- b – balance factor [108] computed by
$$b \leftarrow d_L - d_R,$$
where d_L and d_R are the depths of $N[\iota_L]$ and $N[\iota_R]$, respectively, assuming that $N[\perp].d = 0$.

In Def. 8, the value of $N[\iota_r].V$ represents the total amount of money in the system (or shard) and $N[\iota_r].h$ is the root hash of the system (or shard).

C. THE ALPHABILL ATOMICITY PARTITION

1) THE GOAL OF ATOMICITY CONTROL

Let $u_1, \ldots, u_m$ be entities with identifiers $\iota_1, \ldots, \iota_m$ and owner conditions $\varphi_1, \ldots, \varphi_m$, respectively. The entities $u_1, \ldots, u_m$ may belong to different transaction systems (partitions) with identifiers $\alpha_1, \ldots, \alpha_m$, respectively. It is assumed that in each of these partitions, there is a kind of transaction available for changing the ownership conditions of entities.

The goal is to transfer the entities to new owner conditions $\varphi'_1, \ldots, \varphi'_m$ atomically, i.e., in a way that either

- all of its component transfers take place – all entities $u_1, \ldots, u_m$ are transferred to the new owner conditions $\varphi'_1, \ldots, \varphi'_m$, or
- none of its component transfers take place – all entities will have owner conditions equivalent to the previous conditions $\varphi_1, \ldots, \varphi_m$.

All entities may potentially be controlled by different parties. We assume that these parties may communicate in order to agree on the atomic transfer, i.e., after communication, all parties know all of

$$\iota_1, \ldots, \iota_m, \varphi_1, \ldots, \varphi_m, \alpha_1, \ldots, \alpha_m.$$

The parties also agree on other transaction specific parameters.

If there is more than one party involved, such transfer is an *atomic swap*. If there is only single party involved, it is an *atomic multi-entity, single-asset transfer*.

2) IMPLEMENTATION RESTRICTIONS

To implement such a transaction, the parties send component transaction orders $T_1, \ldots, T_m$ with the new owner conditions $\varphi'_1, \ldots, \varphi'_m$ to the system. These new conditions have to be designed in a way that the atomicity condition is satisfied, i.e., the new owner of every entity ι_i can execute the next payment only if there is evidence that all other transactions $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_m$ have been accepted and included in the ledger. The previous owner of ι_i can execute the next transaction only if there is evidence that at least one other transaction has not been accepted.

In distributed databases technology, such orchestration of multi-shard transactions is usually achieved with two-phase commit protocols. In a partitioned blockchain technology, two-phase commits have to be implemented in a certified way – before executing a component transaction in one partition the status of other component transactions in other partitions have to be certified, i.e., there has to be evidence about the status of other component transactions.

Basically, there are two possible approaches. A usual approach used in known blockchain technologies is to define the new owner conditions $\varphi'_1, \ldots, \varphi'_m$ (locking scripts) in a way that every φ'_i includes direct verification of the status of all other transactions $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_m$. A shortcoming of this approach is that the conditions φ'_i are large in size and complex to verify.

Therefore, we use a different approach, where the status of the multi-component transaction is managed by a special stateful smart contract called *atomicity agent*. Instead of using direct verification of status of other component transactions, the condition φ'_i verifies the status of the multi-component transaction via checking the status of a particular atomicity agent.

The information about the status must be consistent, i.e., always the same for all verifying parties. Therefore, a single server implementation of atomicity agents is insufficient, because a malicious server may give inconsistent status information to the parties, resulting in a violation of the atomicity condition. Implementing the atomicity agent as a smart contract in a blockchain prevents inconsistent status information because of the uniqueness property of blockchains. Therefore, all atomicity agents are managed in a special partition of the system – the *atomicity partition*.

3) ATOMICITY PARTITION DESCRIPTION

There is a specific transaction system (partition) with identifier α_0 that provides necessary unique references for atomic multi-asset swap transactions. We call it the *atomicity partition*.

Entities of the atomicity partition are the atomic multi-asset swap transactions, i.e., each of these transactions has a unique pseudo-random identifier ι (referred to as *contract identifier*) in the atomicity partition. Furthermore, each entity has a data part D containing:

- a list $\langle\alpha_1, \iota_1\rangle, \ldots, \langle\alpha_m, \iota_m\rangle$ of system/entity identifier pairs,
- a timeout specification t_0 of the multi-asset contract expressed in terms of the block number of the atomicity partition,
- the status flag $\mathsf{status} \in \{0, 1\}$ of the contract (initially set to 0), where 1 means that the transaction is completed, and 0 means that it is not yet completed,
- the set $\mathsf{confirmed}$ of already confirmed transactions (initially $\emptyset$).

The transactions of the atomicity partition are:

- reg – registering a new atomic multi-asset swap transaction with contract identifier ι
- con – confirming an existing multi-asset swap transaction with contract identifier ι

To complete the implementation of swap transactions, an appropriate commit protocol has to be provided. We contribute a novel three-phase commit protocol for this purpose, as described in Section VI-C4.

4) THE 3-PHASE COMMIT PROTOCOL

a: PHASE 1: PREPARATION

The involved parties prepare transaction orders $P_1, \ldots, P_m$ (in their wallets) that transfer the ownerships of the entities $\iota_1, \ldots, \iota_m$ to special parameterized ownership predicates

$$\varphi_{\mathsf{ato}}(\alpha_0, \iota, t_0, \varphi_1, \varphi'_1; \_;\_,\_,\_)$$
$$,\ldots,$$
$$\varphi_{\mathsf{ato}}(\alpha_0, \iota, t_0, \varphi_m, \varphi'_m; \_;\_,\_,\_).$$

It is not necessary to specify how the parties agree on the terms of the multi-asset swap transaction and which communication channels they use for that. This depends on the domain logic of the involved partitions.

The contract identifier ι is computed as a deterministic pseudo-random function on the transaction orders $P_1, \ldots, P_m$ (without signatures, i.e., ownership proofs) and the ownership predicates $\varphi_{\mathsf{ato}}(\alpha_0, \iota, t_0, \varphi_i, \varphi'_i; \_;\_,\_,\_)$.

Definition 9 (Ownership Predicate φ_{ato}): The predicate $\varphi_{\mathsf{ato}}(\alpha_0, \iota, t_0, \varphi, \varphi'; P; \mathsf{status}, \Pi, s)$ (where the triple $\langle\mathsf{status}, \Pi, s\rangle$ represents the ownership proof), is true if either:

- $\varphi'(P, s) = 1$, $\mathsf{status} = 1$, and Π is a certificate in the partition α_0 in a round with number $t < t_0$ of the status of contract ι, or
- $\varphi(P, s) = 1$, $\mathsf{status} = 0$, and Π is a certificate in the partition α_0 in a round with number $t \geq t_0$ of the status of contract ι, or
- $\varphi(P, s) = 1$, and Π is a certificate in the partition α_0 in a round with number $t \geq t_0$ of the non-existence of contract ι

The first condition in Def. 9 means that, if all transactions $P_1, \ldots, P_m$ are confirmed in time, the new owners have control over the entities.

The second condition means that, if the status of the contract is still 0 at t_0, the previous owners have control over the entities.

The third condition means that, if the multi-asset swap transaction has not been registered in the atomicity partition at t_0, the previous owners have control over the entities.

b: PHASE 2: REGISTRATION

One of the parties creates the registration message

$$\langle\alpha_0, \mathsf{reg}, \iota, A, T_0\rangle,$$

consisting of:

- α_0 – the identifier of the atomicity partition
- reg – the message type
- ι – a contract identifier

- A – a list of attributes that contains the identifiers $\alpha_1, \ldots, \alpha_m, \iota_1, \ldots, \iota_m$ and the timeout t_0 of the multi-asset swap transaction
- T_0 – a timeout specification of the registration message expressed in terms of the block number of the atomicity partition.

The party sends the registration message to the atomicity partition. After it has received the message, the atomicity partition creates a new entity with identifier ι and with the data part

$$D = \langle \alpha_1, \ldots, \alpha_m, \iota_1, \ldots, \iota_m, t_0, \mathsf{status} = 0, \mathsf{confirmed} = \emptyset \rangle.$$

c: PHASE 3: CONFIRMATION

All parties sign their transaction orders $P_1, \ldots, P_m$ (by adding ownership proofs s_i to P_i) and send them to the corresponding partitions $\alpha_1, \ldots, \alpha_m$.

The parties wait until the next blocks in the partitions $\alpha_1, \ldots, \alpha_m$ have been formed and obtain the ledger certificates (of acceptance) $\Pi_1, \ldots, \Pi_m$ for the signed transactions $\langle P_1, s_1 \rangle, \ldots, \langle P_m, s_m \rangle$.

Next, each party k sends the confirmation message

$$\langle \alpha_0, \mathsf{con}, \iota, A_k, T_0 \rangle$$

to the atomicity partition, where

$$A_k = \langle P_k, s_k, \Pi_k \rangle.$$

Upon receiving a confirmation message

$$\big\langle \alpha_0, \mathsf{con}, \iota, \langle P_k, s_k, \Pi_k \rangle, T_0 \big\rangle,$$

the atomicity partition verifies $\langle P_k, s_k, \Pi_k \rangle$, and checks whether the identifiers α_k and ι_k are consistent with the data part

$$D = \langle \alpha_1, \ldots, \alpha_m, \iota_1, \ldots, \iota_m, t_0, \mathsf{status}, \mathsf{confirmed} \rangle \quad (10)$$

of the contract ι. After a successful verification, the triple $\langle P_i, s_i, \Pi_i \rangle$ is added to the set $\mathsf{confirmed}$.

If all transactions have been confirmed, i.e., if

$$\mathsf{confirmed} = \{(P_1, s_1, \Pi_1), \ldots, (P_m, s_m, \Pi_m)\},$$

the atomicity partition checks whether the contract identifier ι was correctly computed on the basis of $P_1, \ldots, P_m$, and finally sets the status flag $\mathsf{status} \leftarrow 1$.

5) STATE TREE OF THE ATOMICITY PARTITION

The state tree of the atomicity partition is similar to the state tree of Alphabill Money, except for the following: (i) instead of value v there is a state data record D (see (10)), (ii) there is no summary value V defined for the nodes, and (iii) the ownership φ of entities is not defined. Hence, the computation of the node hash is given by the formula

$$h \leftarrow H(\iota, H(D, x); h_L; h_R).$$

6) IMPLEMENTATIONS WITH PUBLIC BLOCKCHAIN TECHNOLOGY

Basically, the Alphabill platform can be implemented as a private (permissioned) as well as a public (permissionless) blockchain.

Due to the use of certified communication between the partitions and shards of each partition, there are some restrictions to the Alphabill blockchain consensus protocol. There are two categories of consensus protocols used in permissionless blockchains:

- *Persistent consensus* protocols, where a certified n-th block will, once created, be persistently stay in the blockchain. For example, proof-of-stake (PoS), multi-signature-based consensus mechanisms belong to this category.
- *Nakamoto consensus* protocols, where a certified n-th block will not necessarily stay in the blockchain, but instead may be replaced with an alternative block with stronger certificate – although the probability that the n-th block is replaced by a different block after the $(n+k)$-th block has been created becomes negligible when k is large. For example, proof-of-work (PoW) consensus mechanisms belong to this category.

If certified (with ledger certificates) information from the n-th block of one partition (or shard) is used as input to another partition (or shard), then this n-th block must be guaranteed to stay in the first partition (or shard). Otherwise, the certified information might be incorrect and should not be used in other partitions and shards.

Therefore, in case the partitions use independent consensus protocols, only persistent consensus protocols are suitable for the Alphabill platform.

VII. FUTURE DIRECTIONS

In this section, we discuss future perspectives of the proposed technology in regard to opportunities and challenges. We also discuss some ongoing and future work. We structure the discussion along the currently widely discussed concepts of DeFi and Web3. Furthermore, we briefly discuss legally binding smart contracts and, furthermore, opportunities for the development of Alphabill partitions.

A. POTENTIAL OF ALPHABILL IN GENERAL

In her speech at the Bank of England Conference in September 2017, Christine Lagarde said: "To be clear, this [virtual currencies] is not about digital payments in existing currencies – through Paypal and other »e-money« providers such as Alipay in China, or M-Pesa in Kenya. Virtual currencies are in a different category, because they provide their own unit of account and payment systems. These systems allow for peer-to-peer transactions without central clearinghouses, without central banks. For now, virtual currencies such as Bitcoin pose little or no challenge to the existing order of fiat currencies and central banks. Why? Because they are too volatile, too risky, too energy intensive, and because the underlying technologies are not yet scalable. Many are too opaque for regulators; and some have been hacked. But many of these are technological challenges that could be addressed over time. Not so long ago, some experts argued that personal computers would never be adopted, and that tablets would only be used as expensive coffee trays. So I think it may not be wise to dismiss virtual currencies.'' [109]

The quotation form Christine Lagarde's speech touches a main motive that was also a central driver for our research, i.e., that scalability is the prerequisite for cryptocurrencies becoming a utility beyond being merely financial instruments. But is it all just about speed performance and eco-friendliness? Lagarde mentions that current cryptocurrencies are too volatile, financially risky and ''too opaque for regulators'' [109]. All of these issues are essential, yet, they cannot be resolved by purely technical measures. On the other hand, it is also impossible to solve these issues merely by regulatory measures [110]. A technological innovation itself can never disrupt institutions (organizations, or societies as a whole); at best, it can enable disruptions. Furthermore, it would usually not lead to disruption, but rather to evolution of an institution – if the institutional analyses framework of Oliver Williamson [17], [111] is right, cultural changes need significantly more time than regulatory changes, and regulatory changes need again significantly more time than organizational changes. Still, there is a huge (seemingly endless) potential for improving the way we conduct business and the way we run our organizations today. Blockchain technology can be the key to unleash this potential [112]–[114]. The question is, how?

We have argued that it needs a platform for universal asset tokenization that allows for uncapped scalability and a maximum of both business-related and technological innovations; and we came up with a concrete design of such platform.

B. DECENTRALIZED FINANCE

1) THE CONCEPT OF DeFi

The least common denominator of DeFi is the utilization of blockchain or distributed ledger technology to implement financial infrastructure and services. Furthermore, it is essential for the concept of DeFi that it would enact innovations in regard to the institutional architecture of today's established financial economics; i.e., mere technological innovations in regard to underlying data structures and algorithms (typically in service of enhancing cross-cutting concerns such as security or utilization of emerging devices) would not be sufficient to classify as DeFi. The envisioned transformation connected with DeFi is disintermediation [127], which is not necessarily, but typically, associated with permissionless consensus [120] or even utilization of a concrete technological platform (such as Ethereum in [128]).

The notions of DeFi transactions and economic transactions [17], [111] are incommensurable. DeFi is about innovations to financial economics; whereas, *transaction cost economics* [17], [111] is a general theory of economics, which is applicable independent of whether established, transformed or entirely new financial services are utilized. DeFi transactions are financial transactions, as such, they are part of economic transactions (in the sense of transaction cost economics).

2) COMMERCIAL BANKING AND INVESTMENT BANKING

Today's financial services fall into two large categories, i.e., commercial banking and investment banking. Commercial banks have an essential role in today's monetary system of fractional reserve banking.

a: COMMERCIAL BANKING

The business model of commercial banks is commonly described as ''banks borrow short and lend long''; however, although this is not completely wrong, it does not describe adequately what commercial banking is about in today's tiered monetary system. Commercial banks provide the money supply in granting collateralized credits; whenever they grant a credit, by way the largest fraction of the money is freshly generated. Therefore, it is not the central bank that ''prints money out of thin air'' (as we sometimes hear in cryptocurrency tech talks), instead, the central bank steers the money supply via a set of complex regulatory measures. In [16], we have described the money supply as a continuous transformation of collateralized anchors. In this transformation process, commercial banks are surveilled (audited) by dedicated financial supervisory authorities.

Any financial service innovation should be aware of the exact mechanisms in the established monetary system if it aims to add value. In the vast amount of cryptocurrencies that have been launched in the past decade, we have seen many that nurtured a narrative of an over-centralized, over-controlled banking system that needs to be overcome, or at least, have borrowed from such narrative. The Alphabill vision does not have to rely on such narrative and, therefore, can be neutral in this regard. Many of today's stakeholders in the banking system seek for innovative solutions and are open for improvements of the institutional framework; and the Alphabill platform is open for them to join with their services, as it is open for entirely newly designed financial services.

b: INVESTMENT BANKING

Investment banking (including wealth management towards affluent individuals) shows a huge potential for service innovation. Today's investment banking services are highly asymmetric, in that they are designed for (and exclusively offered to) a limited group of clients, including businesses and wealthy individuals. The current narrative of DeFi is very much centered around disintermediation; however, the asymmetries in today's investment banking can and should be addressed more in the future. The Alphabill platform can host any kind of transaction systems and is therefore also suited for financial service innovations in the investment banking sector.

TABLE 11. Some typical characteristics of the Web3 vision found in current discussions, compared to the current state of Web 2.0.

	Web 2.0	Web3
Payments	Online bank transfers between accounts hosted by commercial banks; "digital payments in existing currencies – through Paypal and other »e-money« providers such as Alipay in China, or M-Pesa in Kenya" [110]; M1-money	Cryptocurrencies; direct payments between web users, without intermediaries; currency neither owned by central bank nor collateralized (being neither M0-, nor M1-money); (central bank digital money is usually not considered part of Web3)
Financial services	financial services are not considered as part of Web 2.0 (although they might be made accessible through web-based e-commerce services)	Built-in DeFi (see Sect. VII-B); financial services are considered integral part of Web3
Identity concepts	public key infrastructure (including established routines of personal identity proofing [116]); also: cloud-based identity	self-sovereign identity [117]
Data ownership	data owned and utilized by companies	data owned and utilized by users
Trust anchors	authorities, companies	peer-to-peer [118], consensus protocols
Protocol characteristics	stateless (protocols connect siloed applications, protocols regulate the "transmission of data, not how data is stored" [119])	stateful ("collectively maintained universal state for decentralized computing" [119])
Business models	Silicon Valley tech giants (Alphabet, Amazon, Metaverse); super-scaling e-commerce; social media/networks (commercialization of customer data)	decentralized autonomous organization (DAO) [120]; also: genuine DeFi business models (decentralized payment services, decentralized fundraising, decentralized contracting) [121]
Use cases	(i) Usual narrative: content consumption (Web 1.0); content production (Web 2.0): social media/networks, collective intelligence systems [122]. (ii) Practically: business-to-customer (B2C) e-commerce (dotcom [123] and post-dotcom era); (iii) despite SOA (Web services) [124], business-to-business (B2B) is rather not considered a Web 2.0 use case	all Web 2.0 use cases, however, disintermediated; disintermediated B2B is considered an integral part of Web3; non-fungible tokens (NFTs) [125], [126] ("can represent real-world items like artwork and real estate" [127], "can also ... represent individuals' identities, property rights, and more" [127])

3) DeFi VERSUS FinTech

The concepts of DeFi and FinTech (financial technology) are incommensurable, as DeFi is a concrete technological vision (albeit not yet settled, see Section VII-B1), whereas FinTech [129], [130] is a subject area (also considered an industry). FinTech is always about the utilization of the respective latest technology for the provision of financial services. As such, it currently encompasses blockchain technology as well as artificial intelligence, cloud computing and big data technology. DeFi is about the utilization of blockchain for financial services, with a strong stance in favor of disintermediation (at least in its current dominant narrative).

C. Web3

In [131], Web3 has been characterized as "a decentralized, blockchain-based internet ecosystem owned and operated by its users". Web3 (not to be confused with the Semantic Web, Web 3.0[33]) takes disintermediation to a next level by making it ubiquitous, encompassing not only payments and financial services but also digital identities, data and business models. Although Web3 is still in its infancy, it has gained massive attention by major technology analysts such as Gartner [132], Forrester [133] and Forbes Technology Council [134] as well as the Harvard Business Review [131], [135], [136], and the expectations are high towards Web3 being "our chance to make a better internet" [131]. Yet, it is too early to give an exact definition of Web3. Instead, in Table 11, we have compiled a series of characteristics that we found most typical for Web3 in the current discussion and compare them to corresponding characteristics of Web 2.0.

[33]https://www.w3.org/standards/semanticweb/

Alphabill aims at taking a pragmatic instead of dogmatic approach to Web3:

- We do not want to assess Web3 through *completeness of vision* [137] as we would do with innovative products.
- The Web3 is not a product that is engineered, instead, it is a complex adaptive system [81] that emerges.
- The Alphabill platform positions itself as the framework that enables the emergence of Web3.

With Alphabill, the Web3 can emerge – evolutionary – through the appearance and disappearance of more or less viable systems [76], [79], [80], as described in Section III-B in regard to Alphabill's viable-by-design principle. A plethora of research opportunities exist in regard to innovative domain-specific transaction systems, as will be discussed in due course in Section VII-E.

Our own research interest is in reconsidering Internet and WWW protocols, from scratch, in regard to Web3. Currently, we conduct a feasibility study of integrating blockchain technology into the application layer protocols (DNS, TLS, SMTP, IMAP, PGP, FTP, ...) so that it would become possible to explicitly encode the exchange of digital rights. As another concrete step, we are currently designing a Web3 browser as a kind of Alphabill *super wallet* which tightly integrates Web 2.0 browsing and token payments from scratch.

Furthermore, given the "chance to make a better internet" [131], we are currently also interested in more fundamental research questions:

- What can we hope from a "better internet"?
- What needs to be required from a "better internet"?
- How to design a "better internet"?
- What are the obstacles to creating a "better internet" and how to overcome such obstacles?

It was as early as 1960, when Ted Nelson founded the original hypertext [138] project Xanadu [139]. Today, more than 50 years later, the requirements[34] that have been formulated for Xanadu read like a wish list for the "better internet" including a document type system, transclusion [140], secure user identification, access rights management, data replication and many others. Analysing today's enterprise application landscape [141] leads to similar requirements in regard to crosscutting concerns. The fact that today's enterprise applications are implemented as web-based applications gives us an idea of another huge opportunity for Web3 that has been overlooked so far: the systematic convergence of intranet and internet (where we think of the intranet as a potential enterprise application backbone [141]). Unfortunately, a severe challenge for such a design is in existing "path dependence in technologies" [142].

D. LEGALLY BINDING SMART CONTRACTS

With ISO/DTS 23259,[35] the ISO started a standards development on "Blockchain and distributed ledger technologies – Legally binding smart contracts", which has been deleted (currently having the status "deleted"). The notion of legally binding smart contracts reveals a fundamental problem: major technological changes usually need to team together with institutional changes [143], [144]. A legally-binding smart contract language cannot be contributed by merely technological measures. A smart contract language can only be legally binding in so far this is reflected in respective regulations. Therefore, successfully designing a legally-binding smart contract language needs to be, to a large extent, a juridical research endeavor; although, there exist also some interesting technological research questions in regard to legally-binding smart contract languages such as programming language semantics [145] and programming language pragmatics [145], [146].

E. PROSPECTIVE ALPHABILL PARTITIONS

1) POTENTIAL

A great deal of future work is available in regard to prospective Alphabill partitions at various levels, i.e., in regard to business sectors, business domains and even single enterprises, as well as on cross-cutting aspects such a business-to-business communication [147], [148]. Various research studies have been conducted in regard to several business sectors such as e-health [149], energy [150], manufacturing [8], [10], [151], insurances [152], air traffic management [153], as well as public sector domains such as taxation, [154], e-procurement [155], [156] and land registration [157], just to name a few.

We forecast that research and development activities in regard to domain-specific blockchain solutions will even increase more in the future. The Alphabill platform can ease and streamline these research and development efforts by enabling performant implementations and the integration with other domains.

2) A GENERIC CI PARTITION

Collective intelligence systems [158] form an extremely important class of web-based applications with Wikipedia and Reddit being just two examples. Recently, enterprises have discovered the potential of CI for their endeavors [159], where Blackrock's Aladdin[36] system and Genpact's Cora system[37] are particular good examples. As a next concrete research step, we will design and implement a generic collective intelligence (CI) platform as an Alphabill partition. The platform will be designed on the basis of the generic CI framework provided by [121]. With the envisioned generic CI platform, it will be possible to launch tailored CI systems for specialized purposes.

3) B2C PARTITIONS

The current business-to-customer (B2C) sector offers many opportunities for innovations. Today's e-commerce platforms fulfill a mix of functions. Large platforms are sometimes brokers, logistics providers and insurances at the same time. In particular with respect to the brokerage function, existing asymmetries need to be identified and understood better. For example, if 10% provision are taken merely for brokerage of a service (although only the service provider invests, provides the service and bears the business risk), it is worth to investigate the potential for disintermediation at platform level. The research question is then, how to design peer-to-peer e-commerce marketplaces [160], [161] and which business models they would follow.

4) B2B PARTITIONS

Further research is needed in regard to business-to-business (B2B) communication. Value-added B2B services are natural candidates to join the Alphabill platform as transaction systems. The Alphabill platform comes with a solution for multi-asset swap transactions, i.e., the atomicity partition described in Section VI-C. Still, there is potential for value-added services on top of this fundamental mechanism. In the EDI era [105], valued-added networks were quite common. Since then, the value chain has become more and more fragmented [162] and today's B2B is highly asymmetric, with large manufacturers dominating smaller suppliers. The last systematic attempt to implement a more symmetric B2B was the millennium B2B vision with its UDDI (Universal Description, Discovery, and Integration) registry [123]; however, it never took off and is more than twenty years ago now. B2B always was a highly relevant research area, and it can be easily predicted that it will receive even more relevance through the emerging real-time economics [77], [78].

[34] https://xanadu.com.au/general/faq.html#2
[35] https://www.iso.org/standard/75095.html
[36] https://www.blackrock.com/aladdin
[37] https://www.genpact.com/cora

VIII. CONCLUSION

We have argued that uncapped scalability and universal tokenization are the sine qua non game changers for blockchain technology to become as impactful as claimed by the plethora of diverse blockchain technology visions stated so far over the last decade, ranging from decentralized (micro-) payment systems over DeFi to now Web3 – to name a few (not yet to speak about the many domain-specific blockchain-based technology visions).

In this paper, we contributed:

- A new form of electronic money scheme, the bill scheme, which unlocks, through its decomposability results, unlimited scalability in both permissioned and permissionless scenarios.
- KSI Cash, a central bank digital currency that implements the bill scheme (as outcome of a research cooperation between the Estonian Central Bank and Guardtime) as technological feasibility study of a digital euro. In particular, we have contributed:
  - The design of a bill-based data structure that is optimized for performance. We have provided analytical performance estimations for this data structure.
  - The design of an exchange service, based on a pro-active and on a re-active exchange strategy.
- Exhaustive performance evaluations of KSI Cash, conducted with the European Central Bank (together with a group of eight central banks from the Eurosystem) encompassing:
  - *A Realistic Backend System Implementation.* Particular effort has been invested to enable realistic usage testing, resulting into a backend system implementation consisting of approx. 100,000 LOCs.
  - *Realistic Usage Tests.* We have conducted a series of test runs under simulated real-world conditions. Most importantly, we conducted a maximum throughput test run, showing the system operating with 15,000 transactions per second. (further test runs have been conducted to analyze round calculation times, memory consumption and the behavior of exchange services).
  - *Ultra-Scalability Tests.* We have conducted a series of ultra-scalability tests under limited (laboratory) conditions, showing the system managing a load of about 2 million payment orders per second (meaning an equivalent of more than 300,000 transactions per second), demonstrating the linear scalability of KSI Cash (in terms of compute resources)
  - *Carbon Footprint Estimation.* We are able to estimate the carbon footprint of KSI Cash as 0.0001g CO_2 per transaction, again under the realistic usage scenario (Bitcoin = 100 kg and more).
- We have described the architecture of the Alphabill platform, which enables universal asset tokenization. We have defined the platform's elements, asset presentations, ledger certificates, transaction orders, sharding schemes and transaction system specifications.
- We have introduced Alphabill Money, which is the genuine money partition of the Alphabill platform, as an extended bill scheme.
- We have specified the dust collection mechanism of Alphabill Money by elaborating the bill swap scenario and defining the dust collection process.
- Based on the specification of the dedicated Alphabill swap partition, we have defined a 3-phase-commit protocol for atomic heterogeneous (multi-asset) predicate-based swap transactions.
- Finally, we have discusses opportunities and challenges of the proposed platform in terms of the currently widely discussed concepts of DeFi and Web3. Furthermore, we have outlined opportunities for the development of Alphabill partitions.

In the design of the Alphabill platform, we have followed the design principles of security-by-design, scalability-by-design, robustness-by-design and viability-by-design. As the consequence of this, with the Alphabill platform, we are able to provide a universal tokenization platform that allows for universal asset tokenization, transfer and exchange as a global medium of exchange.

REFERENCES

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Apr. 20, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] N. Szabo, "Formalizing and securing relationships on public networks," *1st Monday*, vol. 2, no. 9, Sep. 1997.
[3] L. Grassi, D. Lanfranchi, A. Faes, and F. M. Renga, "Do we still need financial intermediation? The case of decentralized finance–DeFi," *Qualitative Res. Accounting Manage.*, vol. 2022, pp. 1–22, Feb. 2022.
[4] Visa USA. *Visa Acceptance for Retailers*. Accessed: Mar. 11, 2022. [Online]. Available: https://usa.visa.com/run-your-business/small-business-tools/retail.html
[5] F. Meyer, L. Kuhlmann, and S. Müller, (2019). *Card Schemes—Europe Could Irreversibly Lose its Position in the Payment Market. CORE SE*. Accessed: Mar. 11, 2022. [Online]. Available: https://core.se/techmonitor/card-schemes
[6] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2020.
[7] Z. Yang, R. Yang, F. R. Yu, M. Li, Y. Zhang, and Y. Teng, "Sharded blockchain for collaborative computing in the Internet of Things: Combined of dynamic clustering and deep reinforcement learning approach," *IEEE Internet Things J.*, early access, Feb. 17, 2022, doi: 10.1109/JIOT.2022.3152188.
[8] S. Hameed, S. A. Shah, Q. S. Saeed, S. Siddiqui, I. Ali, A. Vedeshin, and D. Draheim, "A scalable key and trust management solution for IoT sensors using SDN and blockchain technology," *IEEE Sensors J.*, vol. 21, no. 6, pp. 8716–8733, Jan. 2021.
[9] M. A. Bouras, Q. Lu, S. Dhelim, and H. Ning, "A lightweight blockchain-based IoT identity management approach," *Future Internet*, vol. 13, no. 24, pp. 1–14, 2021.
[10] A. Vedeshin, J. M. U. Dogru, I. Liiv, S. Ben Yahia, and D. Draheim, "A secure data infrastructure for personal manufacturing based on a novel key-less, byte-less encryption method," *IEEE Access*, vol. 8, pp. 40039–40056, 2020.
[11] A. Vedeshin, J. M. U. Dogru, I. Liiv, D. Draheim, and S. B. Yahia, "A digital ecosystem for personal manufacturing: An architecture for a cloud-based distributed manufacturing operating system," in *Proc. 11th Int. Conf. Manage. Digit. EcoSyst. (MEDES)*, 2019, pp. 224–228.

[12] G. Sagirlar, B. Carminati, E. Ferrari, J. D. Sheehan, and E. Ragnoli, "Hybrid-IoT: Hybrid blockchain architecture for Internet of Things—PoW sub-blockchains," in *Proc. 11th IEEE Int. Conf. Internet Things (iThings)*, Jul. 2018, pp. 1007–1016.
[13] *Informatics Study on Ultra-Scalable Blockchain Technology*. Project 19H01103. Nara Institute of Science and Technology. Accessed: Jun. 21, 2022. [Online]. Available: https://kaken.nii.ac.jp/en/grant/kakenhi-project-19h01103/
[14] M. Kasahara, M. Sasabe, J. Kawahara, and Z. Mototama. (2020). *Informatics Study on Ultra-Scalable Blockchain Technology*. 2020 Fiscal Year Annual Research Report. Nara Institute of Science and Technology. Accessed: Jun. 1, 2022. [Online]. Available: https://kaken.nii.ac.jp/en/report/kakenhi-project-19h01103/19h011032020jisseki/
[15] S. Kasahara, "Performance modeling of bitcoin blockchain: Mining mechanism and transaction-confirmation process," *IEICE Trans. Commun.*, vol. 104, no. 12, pp. 1455–1464, Dec. 2021.
[16] A. Buldas, D. Draheim, T. Nagumo, and A. Vedeshin, "Blockchain technology: Intrinsic technological and socio-economic barriers," in *Proc. 7th Int. Conf. Future Data Secur. Eng. (FDSE)*, in Lecture Notes in Computer Science, vol. 12466. Cham, Switzerland: Springer, 2020, pp. 3–27.
[17] O. E. Williamson, "Transaction cost economics: How it works; where it is headed," *De Economist*, vol. 146, no. 1, pp. 23–58, 1998.
[18] O. Rikken, M. Janssen, and Z. Kwee, "Governance challenges of blockchain and decentralized autonomous organizations," *Inf. Polity*, vol. 24, no. 4, pp. 397–417, Dec. 2019.
[19] (Jul. 2021). European Central Bank, Eesti Pank, Bank of Greece, Deutsche Bundesbank, Central Bank of Ireland, Banco de España, Latvijas Banka, Banca d'Italia, and De Nederlandsche Bank. *Work Stream 3: A New Solution—Blockchain & eID*. Accessed: Mar. 28, 2022. [Online]. Available: https://www.ecb.europa.eu/paym/digital_euro/investigation/profuse/shared/files/deexp/ecb.deexp211011_3.en.pdf
[20] R. Olt, T. Meidla, L. Ilves, and J. Steiner. (Dec. 2021). *Summary Report: Results Eesti Pank—Guardtime CBDC Research*. Eesti Pank, Guardtime. Accessed: Mar. 11, 2022. [Online]. Available: https://haldus.eestipank.ee/sites/default/files/2021-12/EP-Guardtime_CBDC_Research_2021_eng.pdf
[21] A. Buldas, M. Saarepera, J. Steiner, L. Ilves, R. Olt, and T. Meidla, *Formal Model of Money Schemes and their Implications for Central Bank Digital Currency*, Eesti Pank, Guardtime, 2021. Accessed: Mar. 11, 2022. [Online]. Available: https://haldus.eestipank.ee/sites/default/files/2021-12/EP-A_Formal_Model_of_Money_2021_eng.pdf
[22] A. Buldas, M. Saarepera, J. Steiner, and D. Draheim, "A unifying theory of electronic money and payment systems," *TechRxiv*, pp. 1–45, May 2022, doi: 10.36227/techrxiv.14994558.v1.
[23] J. Herrera-Joancomartí and C. Pérez-Solà, "Privacy in bitcoin transactions: New challenges from blockchain scalability solutions," in *Proc. 13th Int. Conf. Modeling Decisions Artif. (MDAI)*, in Lecture Notes in Artificial Intelligence, vol. 9880. Cham, Switzerland: Springer, 2016, pp. 26–44.
[24] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Proc. 17th Int. Symp. Stabilization, Saf., Secur. Distrib. Syst. (SSS)*, in Lecture Notes in Computer Science, vol. 9212. Cham, Switzerland: Springer, 2015, pp. 3–18.
[25] J. Poon and T. Dryja. (Jan. 14, 2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Draft Version 0.5.9.2*. Accessed: Apr. 18, 2022. [Online]. Available: https://lightning.network/lightning-network-paper.pdf
[26] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE Netw.*, vol. 33, no. 5, pp. 166–173, Sep. 2019.
[27] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 17–30.
[28] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," Cryptol. ePrint Arch., Paper 2017/406, May 2017, pp. 1–16. [Online]. Available: https://eprint.iacr.org/2017/406
[29] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 583–598.
[30] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14155–14181, 2020.
[31] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125244–125262, 2020.
[32] *Blockchain and Distributed Ledger Technologies—Vocabulary*, Standard ISO 22739:2020(E), International Organization for Standardization, 2020.
[33] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NISTIR 8202, Oct. 2018.
[34] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[35] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," 2017, *arXiv:1708.03778*.
[36] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–16.
[37] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1–18.
[38] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 95–112.
[39] J. Drake. (May 17, 2018). *Ethereum Sharding*. Accessed: Mar. 26, 2022. [Online]. Available: https://youtu.be/J4rylD6w2S4
[40] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, Jun. 2019, pp. 123–140, doi: 10.1145/3299869.3319889.
[41] H. Chen and Y. Wang, "SSChain: A full sharding protocol for public blockchain without data migration overhead," *Pervas. Mobile Comput.*, vol. 59, Oct. 2019, Art. no. 101055.
[42] A. Manuskin, M. Mirkin, and I. Eyal, "Ostraka: Secure blockchain scaling by node sharding," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Sep. 2020, pp. 397–406.
[43] M. Du, Q. Chen, and X. Ma, "MBFT: A new consensus algorithm for consortium blockchain," *IEEE Access*, vol. 8, pp. 87665–87675, 2020.
[44] A. M. Antonopoulos, *Mastering Bitcoin: Programming the Open Blockchain*. Sebastopol, CA, USA: O'Reilly, 2017.
[45] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*, 2016, pp. 279–296.
[46] R. Pass and E. Shi. (2016). *Hybrid Consensus: Efficient Consensus in the Permissionless Model*. Cryptology ePrint Archive, Report 2016/917. [Online]. Available: https://ia.cr/2016/917
[47] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *Proc. 31st Int. Symp. Distrib. Comput. (DISC)*, vol. 91, Oct. 2017, pp. 1–39.
[48] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[49] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 10401. Cham, Switzerland: Springer, 2017, pp. 357–388.
[50] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Oper. Syst. Principles (SOSP)*, Oct. 2017, pp. 51–68.
[51] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable bias-resistant distributed randomness," in *Proc. 38th IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 444–460.
[52] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.
[53] J. Sousa and A. N. Bessani, "From Byzantine consensus to BFT state machine replication: A latency-optimal transformation," in *Proc. 9th Eur. Dependable Comput. Conf. (EDCC)*, May 2012, pp. 37–48.
[54] V. Buterin, "A next generation smart contract and decentralized application platform," Ethereum, White Paper, 2015. [Online]. Available: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[55] N. Okanami, R. Nakamura, and T. Nishide, "Load balancing with in-protocol/wallet-level account assignment in sharded blockchains," *IEICE Trans. Inf. Syst.*, vol. E105.D, no. 2, pp. 205–214, 2022.
[56] L. Ren, K. Nayak, I. Abraham, and S. Devadas, "Practical synchronous Byzantine consensus," 2017, *arXiv:1704.02397*.
[57] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, and J. Stern, "Addendum to 'scalable secure storage when half the system is faulty,'" *Inf. Comput.*, vol. 205, no. 7, pp. 1114–1116, Jul. 2007.
[58] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. 1st Int. Workshop Peer-to-Peer Syst.*, in Lecture Notes in Computer Science, vol. 2429. Berlin, Germany: Springer, 2002, pp. 53–65.
[59] N. Sohrabi and Z. Tari, "ZyConChain: A scalable blockchain for general applications," *IEEE Access*, vol. 8, pp. 158893–158910, 2020.
[60] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative Byzantine fault tolerance," *ACM Trans. Comput. Syst.*, vol. 27, no. 4, pp. 1–39, 2010.
[61] *Vitalik Buterin explains Ethereum*. Accessed: Jun. 13, 2022. [Online]. Available: https://www.youtube.com/watch?v=TDGq4aeevgY
[62] A. Hafid, A. S. Hafid, and M. Samih, "A novel methodology-based joint hypergeometric distribution to analyze the security of sharded blockchains," *IEEE Access*, vol. 8, pp. 179389–179399, 2020.
[63] D. Jia, J. Xin, Z. Wang, and G. Wang, "Optimized data storage method for sharding-based blockchain," *IEEE Access*, vol. 9, pp. 67890–67900, 2021.
[64] H. Huang, Z. Yue, X. Peng, L. He, W. Chen, H.-N. Dai, Z. Zheng, and S. Guo, "Elastic resource allocation against imbalanced transaction assignments in sharding-based permissioned blockchains," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2372–2385, Oct. 2022.
[65] M. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. London, U.K.: Morgan & Claypool, 2010.
[66] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.
[67] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–14.
[68] (Feb. 3, 2022). Federal Reserve Bank of Boston and Massachusetts Institute of Technology Digital Currency Initiative. *Project Hamilton Phase 1—A High Performance Payment Processing System Designed for Central Bank Digital Currencies*. Federal Reserve Bank of Boston. Accessed: Mar. 26, 2022. [Online]. Available: https://www.bostonfed.org/-/media/Documents/Project-Hamilton/Project-Hamilton-Phase-1-Whitepaper.pdf
[69] L. Zhong, Q. Wu, J. Xie, Z. Guan, and B. Qin, "A secure large-scale instant payment system based on blockchain," *Comput. Secur.*, vol. 84, pp. 349–364, Jul. 2019.
[70] W. K. Chan, J.-J. Chin, and V. T. Goh, "Simple and scalable blockchain with privacy," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102700.
[71] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review," *J. Netw. Comput. Appl.*, vol. 1491, Jan. 2020, Art. no. 102471.
[72] S. Meiklejohn and C. Orlandi, "Privacy-enhancing overlays in bitcoin," in *Proc. 19th Int. Conf. Financial Cryptography Data Secur.*, in Lecture Notes in Computer Science, vol. 8976. Berlin, Germany: Springer, 2015, pp. 127–141.
[73] G. Maxwell. *Coinjoin: Bitcoin Privacy for the Real World*. Accessed: Jun. 1, 2022. [Online]. Available: https://bitcointalk.org/index.php?topic=279249
[74] G. Wood. (2016). *Polkadot: Vision for a Heterogneous multi-Chain Framework, Draft 1*. Accessed: Feb. 3, 2022. [Online]. Available: https://polkadot.network/PolkaDotPaper.pdf
[75] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," 2019, *arXiv:1906.11078*.
[76] S. Beer, "The viable system model: Its provenance, development, methodology and pathology," *J. Oper. Res. Soc.*, vol. 35, no. 1, pp. 7–25, Jan. 1984.
[77] "The real-time revolution," *The Economist*, pp. 22–24, Oct. 2021.
[78] "Instant economocis," *The Economist*, p. 13, Oct. 2021.
[79] S. Beer, *Brain of the Firm*. Baltimore, MD, USA: Penguin, 1972.
[80] S. Beer, *The Heart Enterprise*. Hoboken, NJ, USA: Wiley, 1979.
[81] J. H. Holland, "Studying complex adaptive systems," *J. Syst. Sci. Complex.*, vol. 19, no. 1, pp. 1–8, Mar. 2006.
[82] J. H. Miller and S. E. Page, *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*. Princeton, NJ, USA: Princeton Univ. Press, 2007.
[83] B. Latour, *Reassembling the Social: An Introduction to Actor-Network Theory*. Oxford, U.K.: Oxford Univ. Press, 2005.
[84] K. McBride and D. Draheim, "On complex adaptive systems and electronic government: A proposed theoretical approach for electronic government studies," *Electron. J. e-Government*, vol. 18, no. 1, pp. 43–53, Apr. 2020.
[85] Eesti Pank. (Dec. 13, 2021). *Eesti Pank Ran an Experiment to Investigate the Technological Possibilities of a Central Bank Digital Currency Based on Blockchain, Eesti Pank*. Accessed: Mar. 11, 2022. [Online]. Available: https://www.eestipank.ee/en/press/eesti-pank-ran-experiment-investigate-technological-possibilities-central-bank-digital-currency-13122021
[86] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Secur. Privacy*, Apr. 1980, pp. 122–134, doi: 10.1109/SP.1980.10006.
[87] R. C. Merkle, "Method of providing digital signatures," U.S. Patent US4 309 569A, Jan. 5, 1982.
[88] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. 7th Conf. Theory Appl. Cryptograph. Techn.*, in Lecture Notes in Computer Science, vol. 293. Berlin, Germany: Springer, 1987, pp. 369–378.
[89] *TPC Benchmark C—Standard Specification, Revision 5.11.*, Trans. Process. Council, Feb. 2010. [Online]. Available: https://www.tpc.org/information/who/whoweare5.asp
[90] *TPC Benchmark E—Standard Specification, Revision 1.14.0.*, Trans. Process. Council, Apr. 2015. [Online]. Available: https://www.tpc.org/information/who/whoweare5.asp
[91] *Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)*, Standard X9.62-2005, Accredited Standards Committee X9, Nov. 2005.
[92] N. Rasmussen, "Calculating total cooling requirements for data centers," Schneider Electr.'s Data Center Sci. Center, Vienna, Austria, APC White Paper, Tech. Rep. 25, Revision 3, 2017.
[93] S. Foteinis, "Bitcoin's alarming carbon footprint," *Nature*, vol. 554, p. 169, Feb. 2018.
[94] P. Sandner, C. Lichti, C. Heidt, R. Richter, and B. Schaub, *The Carbon Emissions Bitcoin From Investor Perspective*. Frankfurt, Germany: Frankfurt School Blockchain Center, 2021.
[95] J. P. Trespalacios and J. Dijk. (2021). *The Carbon Footprint Bitcoin*. De Nederlandsche Bank. Accessed: Mar. 29, 2022. [Online]. Available: https://www.dnb.nl/media/1ftd2xjl/the-carbon-footprint-of-bitcoin.pdf
[96] C. Stoll, L. Klaaßen, and U. Gallersdörfer, "The carbon footprint of bitcoin," MIT Center Energy Environ. Policy Res., Cambridge, MA, USA, Tech. Rep. CEEPR WP 2018-018, Dec. 2018.
[97] C. Stoll, L. Klaaßen, and U. Gallersdörfer, "The carbon footprint of bitcoin," *Joule*, vol. 3, no. 7, pp. 1647–1661, 2019.
[98] L. Badea and M. C. Mungiu-Pupazan, "The economic and environmental impact of bitcoin," *IEEE Access*, vol. 9, pp. 48091–48104, 2021.
[99] A. Buldas, P. Laud, and H. Lipmaa, "Eliminating counterevidence with applications to accountable certificate management," *J. Comput. Secur.*, vol. 10, no. 3, pp. 273–296, Sep. 2002.
[100] A. Buldas and S. Laur, "Knowledge-binding commitments with applications in time-stamping," in *Proc. 10th Int. Conf. Pract. Theory Public-Key Cryptography*, in Lecture Notes in Computer Science, vol. 4450, 2007, pp. 150–165.
[101] A. Buldas, A. Kroonmaa, and M. Saarepera, "System and method for generating keyless digital multi-signatures. U.S. Patent U.S. 8 874 921 B2, Oct. 28, 2014.
[102] A. Buldas, P. Laud, M. Saarepera, and J. Willemson, "Universally composable time-stamping schemes with audit," in *Proc. 8th Int. Conf. Inf. Secur.*, in Lecture Notes in Computer Science, vol. 3650. Berlin, Germany: Springer, 2005, pp. 359–373.
[103] A. Buldas and M. Saarepera, "On provably secure time-stamping schemes," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3329. Berlin, Germany: Springer, 2004, pp. 500–514.
[104] A. Buldas and M. Niitsoo, "Optimally tight security proofs for hash-then-publish time-stamping," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 6168. Berlin, Germany: Springer, 2010, pp. 318–335.
[105] M. A. Emmelhainz, *EDI: A Total Management Guide*. New York, NY, USA: Van Nostrand Reinhold, 1993.
[106] K. Solorio, R. Kanna, and D. H. Hoover, *Hands-on Smart Contract Development With Solidity and Ethereum—From Fundamentals to Deployment*. Sebastopol, CA, USA: O'Reilly, 2019.
[107] R. Zarick, B. Pellegrino, and C. Banister, "Layerzero: Trustless omnichain interoperability protocol," 2021, *arXiv:2110.13871*.

[108] G. Adelson-Velsky and E. Landis, "An algorithm for the organization of information (in Russian)," *Doklady Akademii Nauk Proc. Russian Acad. Sci.*, vol. 146, no. 2, pp. 263–266, 1962.
[109] C. Lagarde, "Central banking and FinTech—A brave new world?" Int. Monetary Fund, Washington, DC, USA, 2017.
[110] D. Draheim, "Blockchains from an e-governance perspective: Potential and challenges—EGOSE'2020 Keynote," in *Proc. 7th Int. Conf. Electron. Governance Open Society: Challenges Eurasia*, in Communications in Computer and Information Science, vol. 1349. Cham, Switzerland: Springer, 2021, pp. 11–12.
[111] O. E. Williamson, "Transaction cost economics: The governance of contractual relations," *J. Law Econ.*, vol. 22, no. 2, pp. 233–261, 1979.
[112] D. Furlonger and C. Uzureau, *The Real Business of Blockchain—How Leaders Can Create Value in a New Digital Age*. Boston, MA, USA: Harvard Bus. Review Press, 2019.
[113] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, and S. Debois, "Blockchains for business process management–challenges and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 1, pp. 1–16, 2018.
[114] M. Janssen, V. Weerakkody, E. Ismagilova, U. Sivarajah, and Z. Irani, "A framework for analysing blockchain technology adoption: Integrating institutional, market and technical factors," *Int. J. Inf. Manage.*, vol. 50, pp. 302–309, Feb. 2020.
[115] *Information Technology—Security Techniques—Identity Proofing*. Standard ISO 29003:2018, International Organization for Standardization, 2018.
[116] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Comput. Sci. Rev.*, vol. 30, pp. 80–86, Nov. 2018.
[117] A. Oram, *Peer to Peer: Harnessing the Power of Disruptive Technologies*. Sebastopol, CA, USA: O'Reilly, 2001.
[118] S. Voshmgir, *Token Economy—How the Web3 reinvents the Internet*, 2nd ed. Berlin, Germany: BlochainHub Berlin, 2020.
[119] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized autonomous organizations: Concept, model, and applications," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 5, pp. 870–878, Oct. 2019.
[120] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *J. Bus. Venturing Insights*, vol. 13, Jun. 2020, Art. no. e00151.
[121] S. Suran, V. Pattanaik, and D. Draheim, "Frameworks for collective intelligence: A systematic literature review," *ACM Comput. Surveys*, vol. 53, no. 1, pp. 1–36, Jan. 2020.
[122] J. E. Stiglitz, *The Roaring Nineties: A New History of the World's Most Prosperous Decade*. New York, NY, USA: W. W. Norton, 2004.
[123] D. Draheim, "The service-oriented metaphor deciphered," *J. Comput. Sci. Eng.*, vol. 4, no. 4, pp. 253–275, Dec. 2010.
[124] M. Niforos. (Jun. 20, 2022). The Promising Future of NFTs Remains in a State of Flux. Financial Times. Accessed: Jul. 21, 2022. [Online]. Available: https://www.ft.com/content/a449b288-3f49-43d1-bb2a-5a5b30c20176
[125] M. Dowling, "Fertile LAND: Pricing non-fungible tokens," *Finance Res. Lett.*, vol. 44, Jan. 2022, Art. no. 102096.
[126] R. Sharma. (Jun. 22, 2022). What is a Non-Fungible Token (NFT)?" Investopedia. Accessed: Jul. 21, 2022. [Online]. Available: https://www.invetopedia.com/non-fungible-tokens-nft-5115211
[127] D. A. Zetzsche, D. W. Arner, and R. P. Buckley, "Decentralized finance," *J. Financial Regulation*, vol. 6, no. 2, pp. 172–203, 2020.
[128] F. Schär, "Decentralized finance: On blockchain- and smart contract-based financial markets," *Federal Reserve Bank St. Louis Rev.*, vol. 103, no. 2, pp. 74–153, 2021.
[129] "The FinTech revolution," *The Economist*, May 2015. [Online]. Available: https://www.economist.com/leaders/2015/05/09/the-fintech-revolution
[130] P. Gomber, R. J. Kauffman, C. Parker, and B. W. Weber, "On the fintech revolution: Interpreting the forces of innovation, disruption, and transformation in financial services," *J. Manage. Inf. Syst.*, vol. 35, no. 1, pp. 220–265, Jan. 2018.
[131] L. Jin and K. Parrott. "Web3 is our chance to make a better internet," *Harvard Bus. Rev.*, vol. 10, pp. 1–12, May 2022. [Online]. Available: https://hbr.org/2022/05/web3-is-our-chance-to-make-a-better-internet
[132] J. Wiles. (Feb. 15, 2022). *What is Web3? Gartner*. [Online]. Available: https://www.gartner.com/en/articles/what-is-web3
[133] M. Bennett. (May 10, 2022). *Web3 Isn't Going to Fix the Shortcomings of Today's Web. Forrester*. [Online]. Available: https://www.forrester.com/blogs/web3-isnt-going-to-fix-the-shortcomings-of-todays-web/
[134] B. Platz. (Jun. 1, 2022). *Why Web3 is so Confusing. Forbes Technology Council*. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2022/06/01/why-web3-is-so-confusing/
[135] T. Stackpole, "What is Web3?" *Harvard Bus. Rev.*, vol. 10, pp. 1–9, May 2022. [Online]. Available: https://hbr.org/2022/05/what-is-web3
[136] J. Esber and S. D. Kominers, "Why build in Web3," *Harvard Bus. Rev.*, vol. 16, pp. 1–38, May 2022. [Online]. Available: https://hbr.org/2022/05/why-build-in-web3
[137] N. Pollock and R. Williams, "The sociology of a market analysis tool: How industry analysts sort vendors and organize markets," *Inf. Org.*, vol. 19, no. 2, pp. 129–151, Apr. 2009.
[138] T. H. Nelson, "A file structure for the complex, the changing and the indeterminate," in *Proc. 20th ACM Nat. Conf.*, 1965, pp. 84–100.
[139] K. Knowlton, "Ted Nelson's Xanadu," in *Intertwingled—The Work and Influence of Ted Nelson* (History of Computing), D. R. Dechow and D. C. Struppa, Eds. Cham, Switzerland: Springer, 2015, pp. 25–28.
[140] T. H. Nelson, "The heart of connection: Hypermedia unified by transclusion," *Commun. ACM*, vol. 38, no. 8, pp. 31–33, Aug. 1995.
[141] D. Draheim, "On the radical de- and re-construction of today's enterprise applications—CENTERIS'2019 Keynote," in *Proc. 10th Int. Conf. Enterprise Inf. Syst.*, in Procedia Computer Science, vol. 164, 2019, pp. 120–122.
[142] C. Castaldi, G. Dosi, and E. Paraskevopoulou, "Path dependence in technologies and organizations," in *The Palgrave Encyclopedia of Strategic Management*, M. Augier and D. Teece, Eds. London, U.K.: Palgrave Macmillan, 2018, pp. 1–4.
[143] J. Koppenjan and J. Groenewegen, "Institutional design for complex technological systems," *Int. J. Technol., Policy Manage.*, vol. 5, no. 3, pp. 240–257, 2005.
[144] D. Draheim, R. Krimmer, and T. Tammet, "On state-level architecture of digital government ecosystems: From ICT-driven to data-centric," in *Special Issue in Memory of Roland Wagner. Transactions on Large-Scale Data- and Knowledge-Centered Systems*, vol. 48. Berlin, Germany: Springer, 2021, pp. 165–195.
[145] V. Dwivedi, V. Pattanaik, V. Deval, A. Dixit, A. Norta, and D. Draheim, "Legally enforceable smart-contract languages: A systematic literature review," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–34, Jun. 2021.
[146] A. Dixit, V. Deval, V. Dwivedi, A. Norta, and D. Draheim, "Towards user-centred and legally relevant smart-contract development: A systematic literature review," *J. Ind. Inf. Integr.*, vol. 26, Mar. 2022, Art. no. 100314.
[147] A. Norta and D. Draheim, "First workshop on blockchains for inter-organizational collaboration (BIOC)," in *Proc. Int. Workshops*, in Lecture Notes in Business Information Processing, vol. 316. Cham, Switzerland: Springer, 2018, pp. 100–102.
[148] A. Norta, B. Leiding, D. Draheim, D. Karastoyanova, L. Pufahl, and S. Schöning, "Joint workshop on blockchains for inter-organizational collaboration and flexible advanced information systems BIOC & FAiSE 2019," in *Proc. CAiSE Int. Workshops*, in Lecture Notes in Business Information Processing, vol. 349. Cham, Switzerland: Springer, 2019, pp. 149–153.
[149] A. Zhang and X. Lin, "Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain," *J. Med. Syst.*, vol. 42, no. 140, pp. 1–18, 2018.
[150] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renew. Sustain. Energy Rev.*, vol. 100, pp. 143–174, Feb. 2019.
[151] Z. Li, W. M. Wang, G. Liu, L. Liu, J. He, and G. Q. Huang, "Toward open manufacturing: A cross-enterprises knowledge and services exchange framework based on blockchain and edge computing," *Ind. Manage. Data Syst.*, vol. 118, no. 9, pp. 303–320, Feb. 2018.
[152] M. Raikwar, S. Mazumdar, S. Ruj, S. S. Gupta, A. Chattopadhyay, and K.-Y. Lam, "A blockchain framework for insurance processes," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–4.
[153] M. D. Clementi, N. Larrieu, E. Lochin, M. A. Kaafar, and H. Asghar, "When air traffic management meets blockchain technology: A blockchain-based concept for securing the sharing of flight data," in *Proc. IEEE/AIAA 38th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2019, pp. 1–10.
[154] D. A. Wijaya, J. K. Liu, D. A. Suwarsono, and P. Zhang, "A new blockchain-based value-added tax system," in *Proc. 11th Int. Conf. Provable Secur.*, in Lecture Notes in Computer Science, vol. 10592. Cham, Switzerland: Springer, 2017, pp. 471–486.

[155] E. Abodei, A. Norta, D. Azogu, C. Udokwu, and D. Draheim, "Blockchain technology for enabling transparent and traceable government collaboration in public project processes of developing economies," in *Proc. 18th IFIP Conf. e-Business, e-Services e-Soc.*, in Lecture Notes in Computer Science, vol. 11701. Cham, Switzerland: Springer, 2019, pp. 464–475.

[156] T. I. Akaba, A. Norta, C. Udokwu, and D. Draheim, "A framework for the adoption of blockchain-based e-procurement systems in the public sector," in *Proc. 19th IFIP Conf. e-Business, e-Services e-Soc.*, in Lecture Notes in Computer Science, vol. 12066. Cham, Switzerland: Springer, 2020, pp. 3–14.

[157] N. Lazuashvili, A. Norta, and D. Draheim, "Integration of blockchain technology into a land registration system for immutable traceability: A case study of Georgia," in *Proc. 17th Int. Conf. Bus. Process Manage.*, in Lecture Notes in Business Information Processing, vol. 361. Cham, Switzerland: Springer, 2019, pp. 219–233.

[158] T. W. Malone and M. S. Bernstein, *Handbook of Collective Intelligence*. Cambridge, MA, USA: MIT Press, 2015.

[159] D. Draheim, "Collective intelligence systems from an organizational perspective," in *Proc. 21st Int. Conf. Inf. Integr. Web-based Appl. Services (iiWAS)*, 2019, pp. 3–4.

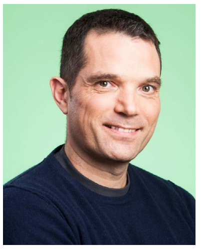
[160] F. Celata, C. Y. Hendrickson, and V. S. Sanna, "The sharing economy as community marketplace? Trust, reciprocity and belonging in peer-to-peer accommodation platforms," *Cambridge J. Regions, Economy Soc.*, vol. 10, no. 2, pp. 349–363, Jul. 2017.

[161] S. Brauckmann, "City tourism and the sharing economy—Potential effects of online peer-to-peer marketplaces on urban property markets," *J. Tourism Futures*, vol. 3, no. 2, pp. 114–126, 2017.

[162] C. W. Stern, "The deconstruction of value chains," in *The Boston Consulting Group on Strategy—Classic Concepts and New Perspectives*, C. W. Stern and M. S. Deimler, Eds., 2nd ed. Hoboken, NJ, USA: Wiley, 2006, pp. 198–201.

AHTO BULDAS (Member, IEEE) received the degree in computer science from the Tallinn University of Technology, in 1991, and the M.Sc. degree in simulation techniques for Boolean circuits and the Ph.D. degree in computational algebraic graph theory, in 1992 and 1999, respectively. He is a Professor of cryptography with the Tallinn University of Technology. His time-stamping related research started in 1997 and he has published papers in the conferences Crypto, Asiacrypt, and PKC. He participated in the development of the Estonian Digital Signature Act and the Estonian eID card, from 1996 to 2002. He is the Co-Founder of Guardtime and also of Cybernetica AS. His research interest includes applied cryptography. His current research interests also include risk analysis methods, including attack-tree semantics and game-theoretical approaches to risk analysis.

DIRK DRAHEIM (Member, IEEE) received the Ph.D. degree from Freie Universität Berlin and the Habilitation degree from Universität Mannheim, Germany. Currently, he is a Full Professor of information society technology with the Tallinn University of Technology, Estonia, where he is the Head of the Information Systems Group. The Information Systems Group conducts research in large and ultra-large-scale IT systems. He is also an Initiator and the Leader of numerous digital transformation initiatives. He is an author of the Springer books *Business Process Technology*, *Semantics of the Probabilistic Typed Lambda Calculus*, and *Generalized Jeffrey Conditionalization*; and a coauthor of the Springer book *Form-Oriented Analysis*.



MIKE GAULT received the Ph.D. degree from the University of Wales, in 1994. He is the CEO and the Founder of Guardtime, a company he has led for the last ten years. He started his career on a postdoctoral fellowship conducting research with the Tokyo Institute of Technology on the numerical analysis of quantum devices. He then spent ten years as a Quant and Derivatives Trader with Credit Suisse Financial Products and Barclays Capital before starting Guardtime.

RISTO LAANOJA is currently pursuing the Ph.D. degree with the Tallinn University of Technology, working on provable security of hashing-based constructs and the development of "strong pre-image awareness" and "bounded pre-image awareness" concepts. He is a member of the Research and Development Team, Guardtime.

TAKEHIKO NAGUMO received the M.Sc. degree in development finance from the University of London and the M.B.A. degree in strategic management from Georgetown University. He is a Senior Managing Executive Officer of Mitsubishi UFJ Research and Consulting. He is an Adjunct Professor in strategic management with the Kyoto University Graduate School of Management. Numerous times, his research on balanced scorecard has been published by the Harvard Business School. He is one of the thought leaders in Japan in the field of digital economy and society: he is a member of the Japanese Government's Regulatory Reform Promotion Committee; a Research Fellow of the National Institute of Advanced Industrial Science and Technology; the Co-Founder and the Executive Director of the Smart City Institute Japan; and a fellow of the World Economic Forum, Centre for Fourth Industrial Revolution Japan.

MÄRT SAAREPERA received the doctoral degree in real-time systems verification from the Tokyo Institute of Technology. He has extraordinary experience in developing blockchain systems. Also, he is an Experienced Business Developer, having shown his leadership skills by growing small startups to strong enterprises. He was the Founder of Guardtime, in 2006, which has now grown to one of the leading industrial blockchain developing companies providing reliable data integrity assurance solutions for industry and the public sector. He has been a coauthor of several research papers on cryptography, integrity protection, and privacy.

SYED ATTIQUE SHAH (Member, IEEE) received the Ph.D. degree from the Institute of Informatics, Istanbul Technical University, İstanbul, Turkey. During his Ph.D. degree, he studied as a Visiting Scholar with The University of Tokyo, Japan; the National Chiao Tung University, Taiwan; and the Tallinn University of Technology, Estonia, where he completed the major content of his thesis. He has worked as an Associate Professor and the Chairperson with the Department of Computer Science, BUITEMS, Quetta, Pakistan. He was also engaged as a Lecturer with the Data Systems Group, Institute of Computer Science, University of Tartu, Estonia. Currently, he is working as a Lecturer in smart computer systems with the School of Computing and Digital Technology, Birmingham City University, U.K. His research interests include big data analytics, the Internet of Things, networks security, and information management.

JOOSEP SIMM received the master's degree in computer science from the Tallinn University of Technology, in 2010. He is a Software Engineer with Guardtime. One of his specializations is performance testing and tuning of computer systems. In Guardtime, he has been integrating Guardtime technology to third party systems. He was the Lead Engineer in the KSI Cash Project. He has been active in software development, since 2005.

JAMIE STEINER received the bachelor's degree in aeronautical engineering from the United States Air Force Academy and the master's degree in business administration from the NYU Stern School of Business. He has 25 years of experience in intelligence, finance, and technology. While in the U.S. Air Force, he collected and analyzed signals and imagery intelligence, and crafted the Air Force's premier course on precise geographical positioning for targeting. His experience in finance includes making markets in inflation linked government bonds for JPMorgan Chase, overhauling the fixed income trading systems used by the JPMorgan Private Bank, and managing regulatory and compliance issues. As the Head of electronic trading with Phoenix Partners LLC, he designed, tested, and managed one of the top electronic liquidity venues for credit default swaps.

TANEL TAMMET is a Full Professor of applied artificial intelligence with the Tallinn University of Technology. He has also worked in the area of cybersecurity and has been involved in numerous large commercial and public sector IT projects. He was one of the initiators of the Estonian X-ROAD interoperability framework and has helped to develop several other core IT infrastructure systems in Estonia. His main research interests include crowd-sourced knowledge bases, automated reasoning, and commonsense reasoning.

AHTO TRUU received the M.Sc. degree in computer science from Tartu University and the Ph.D. degree from the Tallinn University of Technology, where his thesis was on hash-based server-assisted digital signature solutions. He is a Software Architect and a member of the Research and Development Team, Guardtime, where he has worked on developer tools for integrating the company's KSI technology into customer solutions, trained engineers both internally and externally, and was one of the inventors of the BLT digital signature scheme. Before joining Guardtime, he worked for Aprote, leading Estonia's first integration of EMV payment card support into a point-of-sale system among other projects. He has also been teaching programming-related courses with Tartu University and the Tallinn University of Technology and coached Estonian teams for international programming competitions.