TRX+
Blockchain Technology +AI

Combining Trust and Intelligence with TRX+

v1.0

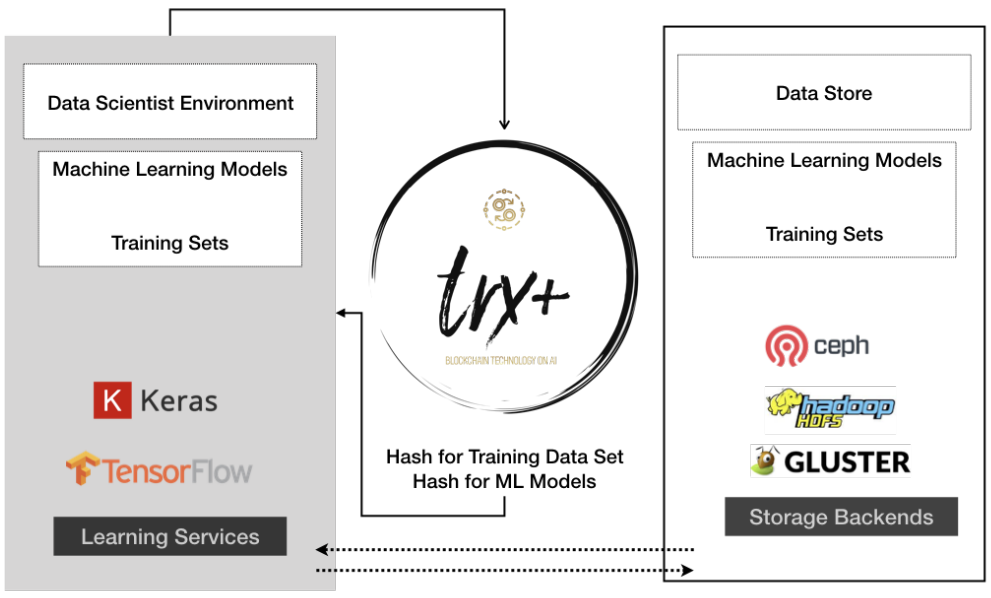
TRX+ R&D Department
research@TRXplus.com

Abstract

In the 21st century, technology and IT has helped the growth of Artificial Intelligence in different aspects of our life. Even our daily life is affected by these new technologies. Machine Learning has shaped new systems, AI has helped for a better life. On the other hand, blockchain has been prospering in the field of privacy and security. In the meanwhile, blockchain technology has shaped decentralized systems and contracts that are open to public, transparent, and secure.

TRX+ stands where AI and Blockchain converge ...

The main purpose of TRX+ technology is distributed transaction processing while establishing trust between the participants operating the peer nodes in the peer-to-peer network representing a blockchain. TRX+ is characterized by: immutability, data integrity, resiliency to attacks, being deterministic, and decentralized.

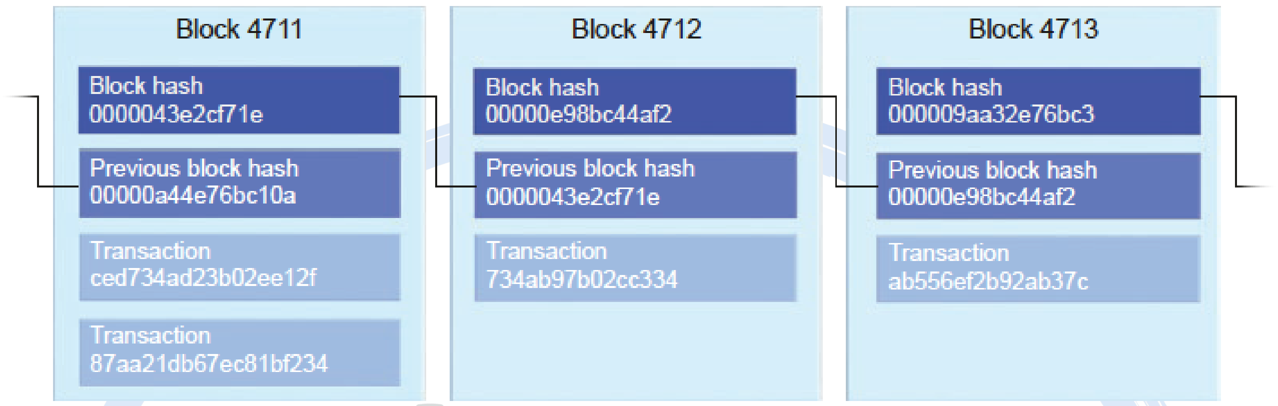

Data Scientist Environment
Machine Learning Models
Training Sets
Keras
TensorFlow
Learning Services
trx+
BLOCKCHAIN TECHNOLOGY ON AI
Hash for Training Data Set
Hash for ML Models
Data Store
Machine Learning Models
Training Sets
ceph
hadoop HDFS
GLUSTER
Storage Backends

Contents

I. Introduction

Blockchain is at its core an immutable, shared ledger application which can be used to record transactions of assets. Assets can be tangible assets of the real world like containers, cars, houses, and many more or digital assets like currencies. Any blockchain technology today employs a couple of key concepts. First, there is the smart contract which is a piece of code that encapsulates the business terms executed within transactions which are recorded by the blockchain. The second key concept is the shared ledger which is conceptually a distributed database holding the records of the recorded transactions.

The records are created by multiple participants in a peer-to-peer network. The peer-to-peer networks can be public or private. Each record goes into a block. The block is linked to its predecessor and successor block in a linked list structure – hence the name blockchain as illustrated in Fig. 1.



Block 4711
Block hash 0000043e2cf71e
Previous block hash 00000a44e76bc10a
Transaction ced734ad23b02ee12f
Transaction 87aa21db67ec81bf234
Block 4712
Block hash 00000e98bc44af2
Previous block hash 0000043e2cf71e
Transaction 734ab97b02cc334
Block 4713
Block hash 000009aa32e76bc3
Previous block hash 00000e98bc44af2
Transaction ab556ef2b92ab37c

Fig. 1. How blockchain works.

Each block is timestamped and cryptographically secured. The blocks are immutable and cannot be changed once created by a participant in the blockchain peer network.

A key difference between blockchain technologies and the well-known database technologies is the following: A blockchain supports only two operations. Create a new transaction and the read operation of a recorded transaction. As a result, think of blockchain technologies as an append-only persistency. Database technologies have four operations: creation of a record, update of a record, deletion of a record, and a reading of a record (collectively referred to as CRUD operations). The mechanism to prevent that a single participant can change something is done through consensus mechanisms. There are different techniques used to implement consensus such as Practical Byzantine Fault Tolerance (PBFT) or multi-signature. The chain is stored at every participant's node. A real-time synchronization keeps the individual peers synchronized by the blockchain fabric at all times. Strong cryptographic algorithms are used to enforce that all transactions are secure, authenticated, and verifiable. In permissioned blockchains, they are also used to ensure that participants can only access those parts of the ledger they are entitled to see.

Transactions created through permissioned blockchains which are used more broadly in the enterprise space are typically of more interest to enterprises from an analytical standpoint where AI comes into play.

II. Blockchain

The promise of a blockchain-based solution for enterprises is to establish a shared, trusted ledger in a much more effective and efficient way with appropriate security controls in place.

Hyperledger is an open source blockchain technology supported by many companies. Enterprises adopt Hyperledger because it provides a shared, replicated, and permissioned ledger with consensus, provenance, immutability, and finality features as shown in Fig. 2. Each participant is running a peer node of the Hyperledger peer network. The peers are kept in sync through a replication mechanism so that all ledgers of all six participants have the same data at all times. Through the Hyperledger channel and private data collection features, you can control access to the transactions. You can use these features so that, for example, three participants collaborate on one channel, whereas a different group participates on another channel and only see subsets of data they have permissions for. Whether or not that is needed depends on the business processes and participants using the Hyperledger blockchain.

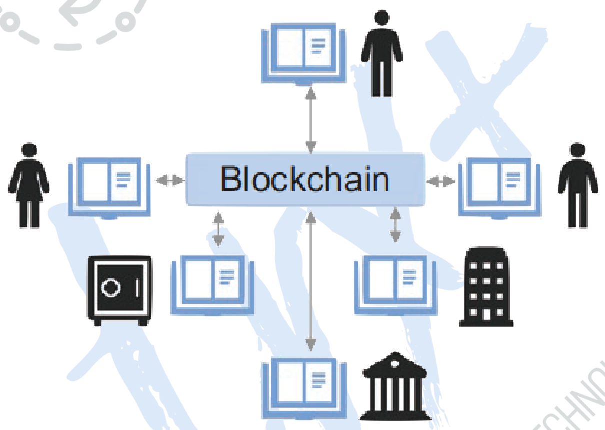


Fig. 2. How TRX+ works.

Tradelens

Tradelens has been created by Maersk and IBM and has dozens of other companies from the logistics ecosystem as partners already. Several governments also participate in the solution since clearing freight at a border is a government-controlled process. The goal of the Tradelens solution is to streamline the data flow between all participants of the logistics supply chain such as logistics companies, government border control authorities, companies operating large ports or airport terminals, and so on. While the data exchange is streamlined, it still needs to be secure so that you have proof when, for example, a container switches from one custodian in the supply chain to another.

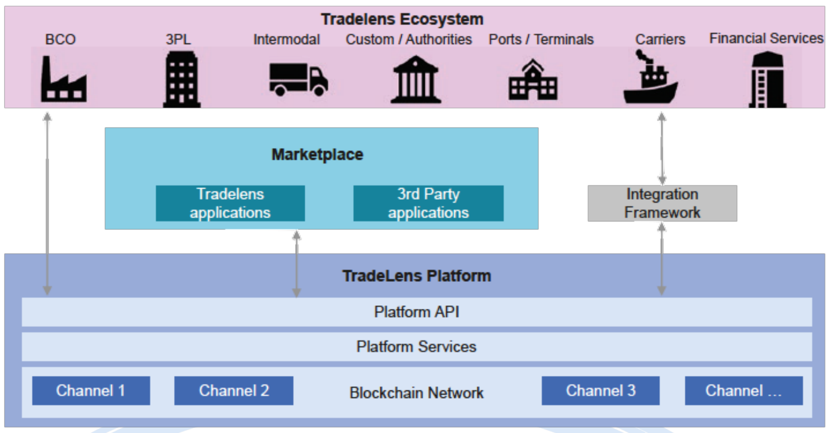


Fig. 3. Tradelens architecture.

Another goal is to remove as much as possible of the paper-based, manual document processing. As a result, much better visibility of goods in the supply chain is achieved and freight in ports or terminals can be cleared faster with a higher degree of automation. Authorities benefit from complete insights into the transport chain with more accurate information and can focus their attention more on risk analysis and others. The foundation of the Tradelens platform is Hyperledger blockchain where multiple channels can be created for different participants. On top of Hyperledger blockchain are the Tradelens platform and API services. Through the APIs, participants in the ecosystem of the Tradelens solution like custom authorities, beneficial cargo owners (BCO), ocean carrier companies, and so on can post transactions onto the blockchain when they received, for example, a container or cleared customs for freight. The solution architecture9 is shown in Fig. 3. With the new API-based access, a lot of the pain of the EDI framework which was previously used as the integration framework (gray box in Fig. 3) can be increasingly avoided as more and more participants switch to the new, modern APIs. Key components of the solution are running on the IBM Cloud. At the time of this writing, over 1 billion event transactions, more than 8 million documents, and over 21 million container transactions have been processed by this blockchain platform.

III. Artificial Intelligence (AI)

Artificial intelligence (AI) has been a vision of humans for a long time. Works of fiction have explored the topic of AI from many angles. For instance, Neuromancer, 2001: A Space Odyssey, Terminator, A.I., Star Trek, Alien, Mother, and so forth feature AI in many different manifestations: some human-like and some very different, some serving, some working with, and some even fighting against humans. While artificial general intelligence (AGI) as featured in science fiction and movies remains more than elusive, there has been a lot of progress in several practical fields of AI which have already moved from fiction to reality. Especially, the AI areas of machine learning (ML) and deep learning (DL) have advanced from research to practice and are meanwhile applied by a large number of companies and organizations to a stunning breadth of use cases around the world. We are now at a point where leveraging ML and DL is state of the art for modern enterprises, yet larger-scale adoption still lies ahead. Early adopters will go deeper and broader in their ML and DL applications, and those who did not yet start in earnest will need to follow soon.

AI for the enterprise

Enterprise AI entails leveraging not only advanced ML and DL but also natural language processing (NLP) and decision optimization (DO) to come to automated actions, robotics, and other areas to optimize existing business processes and to implement new use cases. AI in the enterprise1 aims to discover organizational knowledge and deliver and infuse analytical insight into decision processes in a way that is as aligned with how a human person would go about these tasks, but accelerates these processes by orders of magnitude.

A discussion of AI in the enterprise requires an intensive treatment of an AI information architecture and the challenging operationalization aspects of AI, including DevOps in the context of AI. No enterprise can sustain today's business dynamics and required business agility without a robust information architecture (IA), including aspects like data storage and management, governance and change management, and master data management (MDM).

IV. TRX+AI

Going one step further, usually the input data for an ML-based prediction is relatively small. You can use the blockchain to create a transaction each time an AI model is used for a decision. Such a transaction on the blockchain would just need to include two things:

- The hash value of the used ML model, because all details about this model can be found by looking at the transaction where the hash of this model was originally recorded on the blockchain and optionally going from there to the external data store.
- The input values if they are small enough or a hash value of the input values. If the input values are too large, they could go to a data store outside of the blockchain where all the full values are recorded.

This approach, for example, would be useful for the robotic swarm ecosystem (decentralized system like the blockchain itself) or the AI-infused business process examples mentioned earlier. Blockchains can be characterized by the following traits: immutability, data integrity, resiliency to attacks, being deterministic, and decentralized. If you compare this to AI which is based on confidence in a prediction with a certain probability, volatility, continuous change (model needs to be retrained to avoid model thrift), and central decisions if you have the right model based on deep analysis, you can see that these two technologies are very different, but complementary to each other: if you bring them together though for AI governance, you achieve

- Enhanced data security
- Decentralized intelligence
- High efficiency
- Improved trust in AI-infused business process decisions or robotic swarm decisions

In a nutshell, using TRX+ blockchain, you can improve your AI governance capabilities by creating a ledger of all AI-related artifacts and business decisions where AI was involved.

V. Apply AI on TRX+ blockchain data

With advancements in IoT and the emergence of Blockchain, AI is on brink of a second revolution, one that involves creating complete product offerings with intelligent software and custom hardware.

The next generation of AI technologies will take software solutions several steps ahead. AI 2.0 is something bigger than AI alone. It's no longer about just creating intelligent software. This has been made possible by recent advancements in cognitive technologies (AI), the Internet of Things (IoT), and Blockchain. IoT and Blockchain are relatively recent developments but have co-existed with AI for some time now.

Improvements in AI technologies have made it increasingly easier to develop "complete" product offerings using AI, IoT, and Blockchain. To reiterate, it's not about just software anymore but it's about software and hardware on top of a highly secure and flexible network.

The main purpose of the blockchain technology is distributed transaction processing while establishing trust between the participants operating the peer nodes in the peer-to-peer network representing a blockchain. In the Hyperledger blockchain technology, the persistency is based on LevelDB or CouchDB, and the data is stored in binary or JSON format. A lot of analytics tools such as IBM Cognos, IBM SPSS, Tableau, and so on and data science tools like IBM Watson Studio or AWS SageMaker, which have been built over the last decades, have been optimized for data stores like relational databases supporting SQL. Many of these data stores have been optimized supporting advanced analytics like columnar databases. Another major approach for analytics is based on data lakes using the Hadoop platform with horizontal scale-out across a large number of compute nodes.

Many advanced AI capabilities which require significant computational resources can't be applied without significant performance and scalability impact to the transaction processing on the blockchain. On the other hand, the blockchain data is still in binary or JSON format and not transformed into a data format where analytics can be done with much better performance and scale. Also, many AI scenarios use data from a variety of sources, where blockchain transactional data is just another source. Therefore, if your AI solution requires data from multiple sources, on-chain analytics is architecturally not an appropriate solution because it is impractical to move the data from other sources onto the blockchain – a system designed for transaction processing and not for analytics. The only effective way to consume blockchain transaction data in a scenario where you want to keep the data on-chain and still be able to combine it with data from other sources is through federation. IBM Db2 federation supports federated SQL queries across multiple different data sources, and in the most recent release, the support has been extended to peer nodes of Hyperledger blockchain.

Off-chain analytics in TRX+ blockchain

Let us assume the following scenario from the insurance industry. Insurance companies and the insurance regulator collaborate using a blockchain network. The insurance companies place transactions on that blockchain network. Each transaction represents one claim which has been settled relative to natural disasters where the type of disaster and the amount paid are recorded. The natural disasters could be wildfires, floods, and so on. The insurance regulator can see all these transactions on the blockchain network and would be interested to predict under which circumstances of natural disasters one or more insurance companies need to file for bankruptcy because they can't settle the related claims anymore. For developing such prediction models, the insurance regulator needs to extract the transactions from the blockchain into an analytics environment where these transactions can be combined, for example, with live weather data from other sources, to determine if there are risks of large wildfires, floods, and so on. This is of course just one out of many examples where the blockchain transactions could be a relevant data source for AI. Generally speaking, such scenarios require that the data is moved from a peer node into an analytics environment where the AI techniques are applied as shown in Fig. 4.

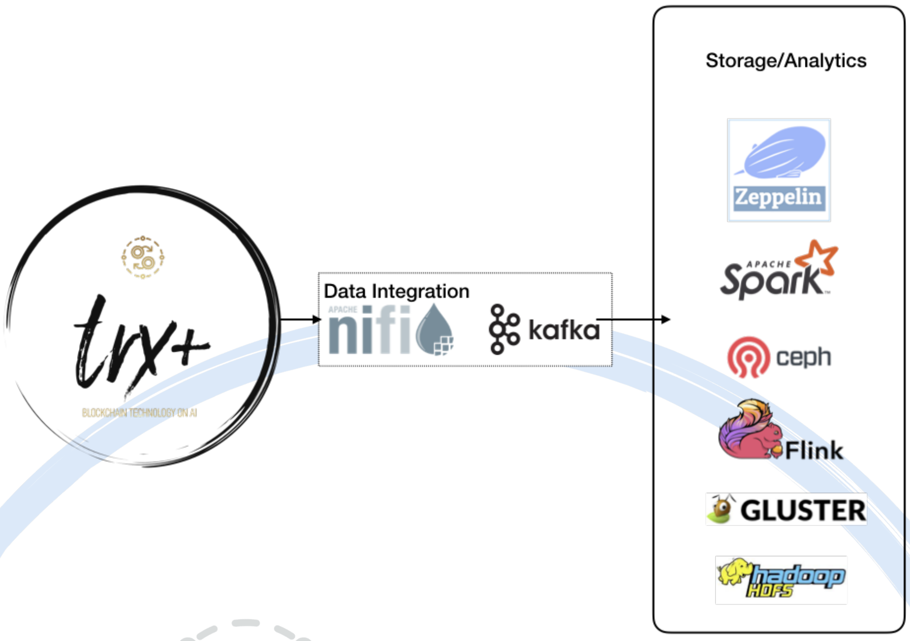


Fig. 4. Off-chain analytics.

Depending on the scenario, a participant running a peer in a blockchain network would like to move a portion or all of the data the participant is entitled to see to an analytics environment. As shown in Fig. 4, there are many different possible targets such as data warehouses, data lakes, data marts, and so on. They can be located in private cloud or public cloud environments. The data transfer can be done as batch transfer using traditional data integration tools from vendors like Informatica or IBM or open source tools like Nifi or Airflow12. If you need the blockchain transactions as they are posted on the blockchain near real time replicated to your analytical systems, then you need messaging tools like the open source Kafka messaging system or commercial counterparts. TRX+ has an Event Hub mechanism where you can basically create subscriptions for transactions events which can be published onto the messaging bus and routed from there to your analytics system. Regardless if you do batch or ongoing data transfer or an initial batch with all further changes being continuously replicated, once the batch transfer finishes, the technology you deploy must support integration with the security architecture of your blockchain system. For Hyperledger, this requires authentication and digital certificates integration. In off-chain analytics scenarios, you have also an additional advantage that you can transform the binary or JSON data from blockchain into formats much more suitable for AI. For example, you can use Kafka to seamlessly replicate from Hyperledger to IBM Cloud Object Storage on the public cloud as is to land the data next to other data assets you might have there. You can then use the SQL query13 service to reformat the JSON blockchain data into a more analytics-friendly format and then run either large-scale SQL-based analytics or any other AI algorithms on it.

The off-chain analytics approach supports a much broader set of use cases, has better scale and performance characteristics, and allows to execute substantially more AI techniques on the blockchain data.

VI. TRX+ BLOCKCHAIN TO GOVERN AI

AI governance is a growing concern. The European Union released an AI regulation17 in February 2020 to address the following key questions:

- How can we ensure that data sets are not biased and sufficiently representative?
- How can we provide mandatory documentation of training, testing, data selection, and AI algorithms being used?
- How can we provide information to consumers about AI capabilities applied and their limitations in an automatic fashion?
- How can we ensure that AI decisions are robust, accurate, and reproducible?
- How can we integrate human oversight through approvals and monitoring into our AI-infused business processes?

To better understand why these questions are critical, think about the following: if AI-based prediction models are used by banks to determine if and under which conditions you might or might not get a mortgage or if insurances use AI-based models to determine whether or not your claim is accepted and how much money will be paid to you, you might want to know how reliable these models are, who trained them, and so on. Individual agents in robotic swarm ecosystems work toward a swarm goal. However, their decision making is decentralized and the decisions can be based on AI algorithms. The robots in such a use case make decisions and outcomes are based on majority rules. The vote based on an AI algorithm is basically the transaction. Again, how do you know how and why a conclusion was reached in a robotic swarm ecosystem using AI-based decision making by the robots?

Tools like IBM Watson OpenScale allow you to address some of the preceding questions like whether or not the training set was biased or if the model drifted over time and so on. IBM Watson OpenScale and similar tools do not address the need to provide a tamperproof audit trail with full transparency, who trained the machine learning model, which data sets was the model trained with, which version was used at a particular point in time, and so on. This is where AI and blockchain technologies nicely complement each other as shown in Fig. 5.

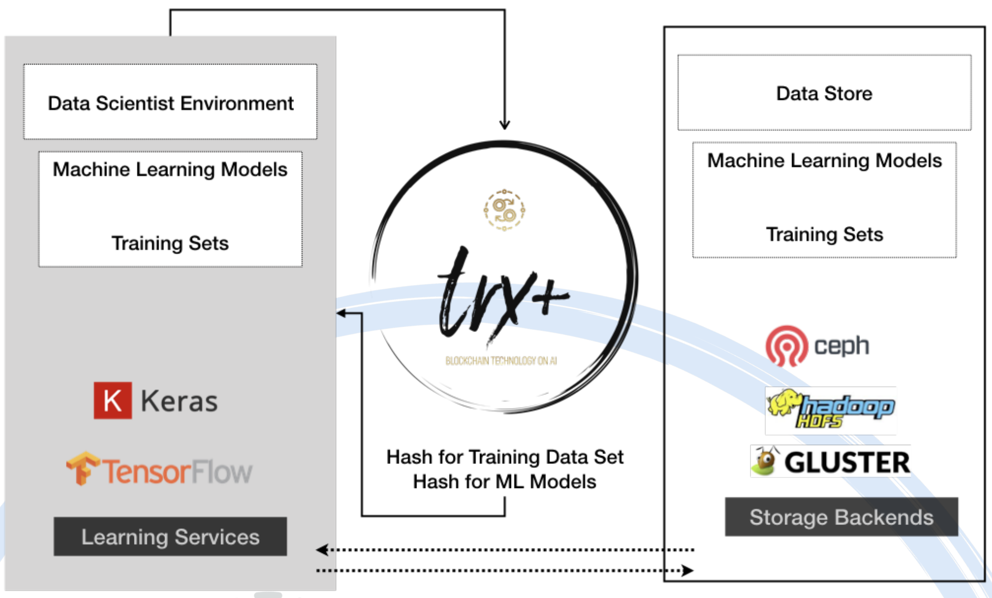


Fig. 5. TRX+ blockchain governs AI.

The key idea is basically to establish proof of existence on the blockchain, which ML model has been trained by whom and when on which data set, which version was deployed, and so on. Pushing large data volumes onto the blockchain which might have been used in your AI projects to train the models might not be possible. This can be either due to the volume or the size of individual items in structured or unstructured data sets. Hence, a more practical approach is to store the AI models and the training data sets in a data store outside of TRX+ blockchain as shown in Fig. 5. However, you push cryptographically secure hash values – conceptually a unique fingerprint of the model and the training data set alongside critical other attributes like who trained the model, version number of the model, and so on onto the TRX+ blockchain itself. If you want to verify later that it was indeed a particular ML model and the training data set you have in the data store which was used, you could easily detect if someone tampered that because the hash value would not be the same as the hash value you find on the TRX+ blockchain. So the immutability of the TRX+ blockchain guarantees that you have a tamperproof way to show which AI model was running at the time a critical business decision based on AI-infused business processes was taken.

VII. TRX+AI Applications

A. *TRX+AI in supply chains*

TRX+ blockchain technology can be used in manufacturing and supply chains to record any exchange, agreements/contracts, tracking, and payment. Because every transaction is recorded on a block and across multiple copies of the ledger that are distributed among users, this recording method is highly transparent. It's also highly secure. It will be extremely efficient and scalable. You will be able to see the whole record of a product or a component, including where it comes from and where it has been. This is called traceability. TRX+ blockchain technology can increase the efficiency and transparency of supply chains. For example, multinational retailer Walmart has partnered with IBM to track food staples from supplier to shelf using its Hyperledger Fabric blockchain. The technology has also been used to track art works, antiques, jewelry, and other valuables for authentication and proof of ownership and to combat counterfeiting.

Supply chain management is crucial to the success of many industries and growing economies around the world; some consider supply chains to be the nervous system of trade. Managing a supply chain is an operation-oriented practice that requires efficiency and effectiveness, from the early phases of planning, procurement, and warehousing, to the logistics of shipping goods from producers to consumers. The following sample diagram identifies the stakeholders involved in the supply chain management of a laptop Fig. 6:

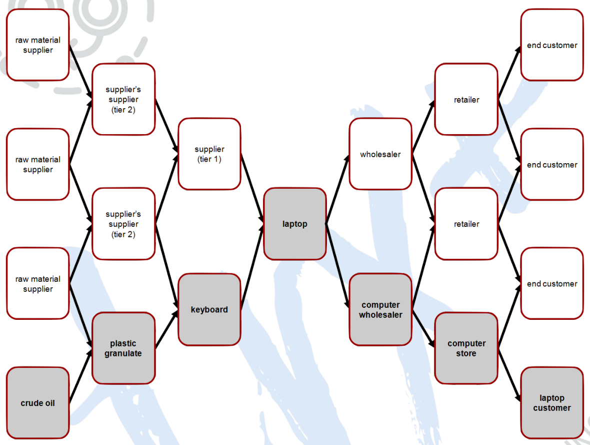


Fig. 6. TRX+AI in supply chain

The following are three of the top issues faced by the supply chain industry:

- Operational costs and risks: A high proportion of the supply chain is still in paperwork mode. This means that data across the system is highly vulnerable to being tampered with. The data in supply chains is crucial for identifying the value of the products, including its basic attributes such as validity and shelf life. If the paperwork of the shipment has been tampered with, there could be a large gap across the value chain. Manual paperwork-based supply chains can also attract financial glitches due to potential scenarios such as double billing, thereby creating more audit problems and confusion among the stakeholders. Although relatively newer supply chain solutions have been effective in digitally transforming the paperwork nature of a few supply chain transactions, there are fewer systems technically capable of handling potential data loss or forgery in the process. Blockchain can alleviate these problems with the help of smart contracts and can automate several critical processes.

- Security and authenticity: Although the end-to-end processes of the supply chain are digitally connected using traditional enterprise software, they are loosely defined and coupled among themselves. This means that data integrity is not treated as a first-class feature. This can be resolved with blockchain technologies based on the immutability concepts and ensure that no records can be stolen, withdrawn, or replaced with inaccurate data. Since transactional data on the blockchain is open, relevant stakeholders can access information and verify the single source of truth.
- Real-time visibility: Transactions on native digital platforms that are closely coupled to traditional finance can also invite other business challenges, such as delays in settlements. Businesses in the supply chain heavily rely on working capital, and the major source for capital is either through loans or the revenue for their services. Enabling real-time payments across the value chain can enable faster payment processing at relatively lower costs compared to traditional finance. This is a game-changer for the industry.

Let's go through some of the TRX+ solution combining blockchain and AI to solve the problems in the supply chain industry. TRX+ blockchain network provides real-time intelligence and actionable recommendations. This suite offers a wide range of features across supplier management, inventory management, and order management. It is also notable that the suite offers an open platform and a developer hub for building tailored solutions in the supply chain using blockchain and AI. With the aim of making supply chains more connected, collaborative, intelligent, and secure, TRX+ is working on an autonomous and intelligent supply chain that can be used to apply AI, IoT, and blockchain.

Although AI and blockchain are being used to track orders and manage inventory more effectively, there are many business gaps that need to be filled. It is important to lower the bar of entry to bring in a diverse set of stakeholders across the globe so that they can participate in a globally verifiable supply chain network with a vision to increase quality and efficiency and promote transparency. This can happen by exposing APIs so that not all vendors have to be running on the same blockchain or network; instead, they could simply exchange trustworthy data and record them as cross-transactions across the networks for better compatibility. Effort must be made to make these products interoperable so that it becomes easier for the global supply chain economy to thrive on the diversity offered by the blockchain.

B. *TRX+AI in Banking and Financial Services Industry (BFSI)*

The BFSI is the backbone of the economic operations of the whole world as we know it. With trillions of dollars of assets under management, managing money effectively at a digital scale has become a lucrative opportunity and hence needs to be revisited due to some inefficiencies in the current systems that generally rely on traditional methods. The following diagram depicts the relationship between stakeholders in the BFSI industry Fig. 7:

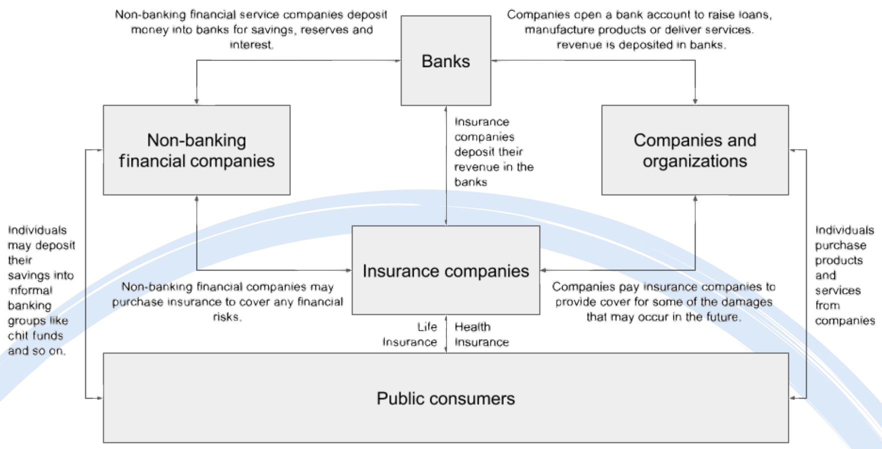

Non-banking financial service companies deposit money into banks for savings, reserves and interest.
Banks
Companies open a bank account to raise loans, manufacture products or deliver services. revenue is deposited in banks.
Non-banking financial companies
Insurance companies deposit their revenue in the banks
Companies and organizations
Individuals may deposit their savings into informal banking groups like chit funds and so on.
Insurance companies
Individuals purchase products and services from companies
Non-banking financial companies may purchase insurance to cover any financial risks.
Companies pay insurance companies to provide cover for some of the damages that may occur in the future.
Life Insurance
Health Insurance
Public consumers

Fig. 7. Principle of Banking and Financial Services Industry (BFSI)

As shown in the preceding diagram, public consumers like you and me access capital and services through banks, insurance companies, and Non-Banking Financial Companies (NBFC). Also, note that companies and organizations that provide services to us also rely on these financial institutions.

The following are the top three issues faced by the BFSI industry:

- Access to capital: The unbanked and the underbanked are severely affected by the problem of eligibility criteria, thereby creating a gaping hole in the market, thus leading to informal economies. If accessing the capital is made easier, we can streamline the informal economy and maximize its potential.
- Fraud and delinquency: It's estimated that financial fraud and delinquency are affecting the global economy through a loss of 600 billion USD. This is a serious problem plaguing the BFSI industry, especially in India due to the recent revelations about bad loans. At the time of writing, India ranks second in bad loans, piling up to 160 billion US dollars according to a Bloomberg article.
- Lack of proper process automation: Although numerous attempts have been made to secure and automate the BFSI sector, we need a better model that can bring about a striking balance between transparency in the industry but also preserves order among institutions. Hence, using blockchain to bring about transparency and using AI techniques to enable optimized workflows for automation could be the key to revolutionizing this industry.

TRX+ provides a peer-to-peer network of lenders and borrowers that enable crypto-backed loans. Lenders can provide their cryptocurrencies for an interest rate, while borrowers can raise loans against collateral at a discounted price of the pledged asset, followed by a payment of a premium at the end of the loan's term. At the time of writing, the Nuo network has been used by many users to raise loans of up to 30 million USD. Also, it is important to note that TRX+ Network is non-custodial, meaning that the wallets with funds in them are not in direct or indirect control of the company. This is enabled by using the TRX+ blockchain, where smart contracts are used to trade TRX+ and TRX+ tokens.

C. TRX+AI and real estate

The following are two of the top issues faced by the real estate industry:

- Inconsistency in land records: A sale or transfer of rights pertaining to property involves several departments. Unfortunately, these departments work as silos, thereby making way for discrepancies if the same data is not available to all the authorities. This can lead to complicated legal positions for landowners.
- Lack of instant traceability: If a buyer wishes to buy a specific property, there must be a method to trace the transfer of ownership through the title deeds or any other complying documents within seconds. Today, the services offered by local authorities usually take some time and may come at a considerable cost.

TRX+ provides a solution for enterprises and governments to benefit from immediate consensus, real-time information sharing, and smart contracts. TRX+ network enables issuing digital signature certificates. This solution proposes recording sale deeds so that they're stored on a blockchain, thus allowing various government bodies to access it, such as registration and stamp revenue departments, survey and settlement departments, revenue departments, and courts, along with business entities such as banks.

VIII. TRX+ BLOCKCHAIN IN DEPTH

We believe that the lack of scalability, scripting safety and cheap access to real-world data of current "smart contract platforms" come down to three core issues. *First*, the currently prevailing stateful design makes smart contracts written for the platform hard to analyze[1], and statefulness combined with sequential transaction ordering complicates scalability [need cit.]. *Second*, the high cost of bringing real-world data into the system in a decentralized, trustless and reliable way complicates or outright prevents the realization of many promising applications [need cit.]. *Third*, the platforms are limited in their abilities to update themselves, in order to adapt to new technological or economical knowledge. We believe that each of these three problems have clear solution paths that should be explored.

First, recent research into state channel technology suggests that for many use cases, keeping state on-chain is not necessary [need cit.]. It is very often entirely possible to store all information in state channels, and only use the blockchain to settle any economic

[1]The difficulty of analyzing stateful contracts was very clearly demonstrated by the re-entrance bug that brought down "The DAO". This happened despite the code having been audited by several of Ethereum's creators as well as the general community [need cit.].

results of the information exchange, and as a fallback in the case of dispute. This suggests an alternative approach to blockchain architecture in which Turing-complete smart contracts exist in state channels but not on-chain. This increases scalability since all transactions become independent and can thus be processed in parallel. Additionally, this means that contracts never write to shared state, greatly simplifying their testing and verification. We believe that this design emphasizes that blockchains are about financial logic rather than data storage; there exist decentralized storage solutions that complement blockchains perfectly.

Second, applications such as Augur have attempted to bring real-world data onto the blockchain in a decentralized way—in the process essentially building a consensus mechanism inside smart contracts **augur**, instead of utilizing the consensus mechanism of the underlying blockchain. This leads to inefficiencies but doesn't increase security. The natural conclusion from this is to generalize the blockchain's consensus mechanism so that it can provide information not only on the next internal state, but also on the state of the external world. It could thus be argued that the blockchain's consensus mechanism determines the result of running what complexity theory dubs an *oracle machine*: a theoretical machine that is more powerful than a Turing machine because it has answers to some questions that can't necessarily be computed, such as "Who won football game X?" [*need cit.*].

Third, it seems natural that the consensus mechanism could also be used to determine the parameters of the system. This allows it to adapt to changing external conditions, as well as adopting new research and recent developments in the field.

The rest of this section introduces the TRX+ blockchain in greater detail, starting with a brief overview of accounts, tokens, names and the structure of blocks. This is followed by an explanation of our approach to state channels and smart contracts, and then a discussion on how the blockchain's consensus mechanism can be used both to create an efficient oracle mechanism and to govern the system. Finally, we discuss scalability from several different angles.

A. Tokens, accounts and blocks

Despite being "stateless" from the contract developer's point of view, the TRX+ blockchain keeps track of several predefined state components. We will now explain these, as well as the content of each block. For simplicity, this section assumes that every node keeps track of the whole blockchain. Possible optimizations are described in section VIII-E.

A.1) Access token, TRX+: To use the blockchain is not free, but requires that the user spends a token called *aeon*. TRX+ are used as payment for any resources one consumes on the platform, as well as the basis for financial applications implemented on the platform.

The distribution of aeon in the genesis block will be determined by a smart contract hosted on Ethereum. Further aeon will be created via mining.

All system fees get paid with aeon, all smart contracts settle in aeon.

*A.2) **Accounts:*** Each account has an address and a balance of aeon and also a nonce which increases with every transaction and the height of its last update. Each account also has to pay a small fee for the amount of time it is open. The costs of creating and keeping accounts prevents spam and disincentivizes state-bloat. The reward for deleting accounts incentivizes the reclaiming of space.

*A.3) **Name system:*** Many blockchain systems suffer from unreadable addresses for their users. In the vein of Aaron Swartz' work and Namecoin, TRX+ features a name system that is both decentralized and secure, while still supporting human-friendly names **aaron**. The blockchain's state includes a mapping from unique human-friendly strings to fixed-size byte arrays. These names can be used to point to things such as account addresses on TRX+, or hashes e.g. of Merkle trees.

*A.4) **Block contents:*** Each block contains the following components:

- The hash of the previous block.
- A Merkle tree of transactions.
- A Merkle tree of accounts.
- A Merkle tree of names.
- A Merkle tree of open channels.
- A Merkle tree of oracles which haven't answered their respective questions.
- A Merkle tree of oracle answers.
- A Merkle tree of Merkle proofs.
- The current entropy in the random number generator.

The hash of the previous block is required to maintain an ordering of the blockchain. The transaction tree contains all transactions that are included in the current block. With the exception of the consensus vote tree, all the trees are fully under consensus: if a tree is changed from one block to the next, this change has to be due to a transaction in the new block's transaction tree, and a Merkle proof of the update has to be included in the block's proof tree. The purpose of the three remaining trees will hopefully become clear in the following sections.

B. State channels

One of the most interesting developments in the blockchain space lately is that of state channels. They operate on the basic principle that in most cases, only the people affected by a transaction need to know about it. In essence, the transacting parties instantiate some state on a blockchain, e.g. an Ethereum contract or a Bitcoin multisig. They then simply send signed updates to this state between each other. The key point is that either one of them could use these to update the state on the blockchain, but in most cases, they don't. This allows for transactions to be conducted as fast as information can be transmitted

```
1 macro Gold f870e8f615b386aad5b953fe089 ;
2
3 Gold oracle
4 if 0 1000 else 0 0 end
5 0
```

Fig. 8. A simple contract encoding a bet on the price of gold. The language used is the Forth-like *Chalang*, which will be presented in section X-A.

and processed by the parties, instead of them having to wait until the transaction has been validated—and potentially finalized—by the blockchain's consensus mechanism.

On TRX+, the *only* state update that can be settled on the blockchain is a transfer of aeon, and the only aeon that can be transferred are the ones that the transacting parties already deposited into the channel. This makes all channels independent from each other, which has the immediate benefit that any transactions related to channels can be processed in parallel, greatly improving transaction throughput.

The blockchain is only used to settle the final outcome or to resolve conflicts that arise, roughly analogous to the judicial system. However, because the blockchain's behavior will be predictable, there is no gain in disputing the intended result of a state channel; malicious actors are incentivized to behave correctly and only settle the final state on the blockchain. All taken together, this increases transaction speed and volume by several orders of magnitude, as well as privacy.

B.1) Smart contracts: Despite that the only state that can be settled on-chain is a transfer of aeon, TRX+ still features a Turing-complete virtual machine that can run "smart contracts". Contracts on TRX+ are strictly agreements that distribute funds according to some rules, which stands in stark contrast to the entity-like contracts of e.g. Ethereum. Two of the more notable practical differences is that by default, only the involved parties know about a given contract, and only parties that have an open state channel can create a valid contract. If the parties agree to a contract, they sign it and keep copies for future reference. It is only submitted to the blockchain if its outcome is disputed, in which case the code is only ever stored as part of the submitted transaction, never in any other state. If this happens, the blockchain distributes the tokens according to the contract and closes the channel.

As an example, fig. 8 shows a very simple contract that encodes a bet on the price of gold at a certain time. On line 1, the macro `Gold` saves the identifier of the oracle in question, which will return `true` if the price of gold is below $38/g on December 1st, 2016. The body of the contract is displayed on lines 2-4: we first push the gold oracle's identifier to the stack and call it using `oracle`, which will leave the oracle's answer on the top of the stack. We use this to do a conditional branching: if the oracle returns `true`, we push 0 and 1000 to the stack, indicating that 0 aeon should be burned and 1000 aeon should go to the first participant in the channel. Otherwise, we push 0 and 0,

```
1 : hashlock
2 swap
3 hash
4 == ;
```

Fig. 9. A simple hashlock.

```
1 macro Commitment a9d7e8023f80ac8928334 ;
2
3 Commitment hashlock call
4 if 0 100 else 0 50 end
5 1
```

Fig. 10. Using the hashlock to trustlessly send tokens through a middleman.

with the second 0 indicating that the other participant receives all aeon in the channel. Finally we push 0, which is taken to be the nonce of this channel state. In actual usage, the nonce would be generated at deployment.

One important thing to note is that contracts on TRX+ don't maintain any state of their own. Any state is maintained by the transacting parties and submitted as input at execution. Every contract is essentially a *pure function* that takes some input and gives a new channel state as output[2]. The benefits of using pure functions in software development in general, and in the development of financial applications in particular, has been extensively documented in academia and industry for decades **formal˙contracts**[need cit.].

a) Contract interaction and multi-step contracts: Even though all contracts are stateless and execute independently of each other, contract interaction and statefulness can still be achieved through *hashlocking* [need cit.]. A simple hashlock is shown in fig. 9. On line 1, we define a function called `hashlock` that expects the stack to contain a hash h and a secret s. It swaps them on line 2, in order to hash the secret on line 3, before calling the equality operator on $hash(v)$ and h on line 4. This returns true if the secret is a preimage of the hash. This function can be used to predicate the execution of code branches in different contracts on the existence of the same secret value.

As a simple example usage, hashlocks make it possible for users that don't share a state channel to trustlessly send each other aeon, as long as there is a path of channels between them. For example, if Alice and Bob have a channel and Bob and Carol have a channel, then Alice and Carol can transact through Bob. They do this by creating two copies of the contract shown in fig. 10, one for each channel. The `Commitment` on line 1 is the hash of a secret that Alice chooses. On line 3 we push it to the stack and call the `hashlock` function. Which branch of the `if` that gets executed depends on the return value of `hashlock`. Once these contracts have been signed by all parties, Alice reveals

[2]It should be noted that since contracts can read answers from oracles and some environment parameters, they aren't completely pure functions. However, oracle answers never change once they've been provided and can be argued to be due to the computational richness of the oracle machine, rather than being an impurity. Environment parameters are deemed a "necessary evil" and will ideally be compartmentalized appropriately by high-level languages.

```
macro Commitment a9d7e8023f80ac8928334 ;

Commitment hashlock call
if State33 else State32 end
call
```

Fig. 11. A simplified example of using the hashlock to play a multi-player game in channels.

the secret, allowing Bob and Carol to use it to claim their aeon.

Hashlocking can also be used to e.g. play multi-player games in the channels, as shown in fig. 11. Everyone makes a channel with the game manager, which publishes the same contract to every channel. Say we are in game state 32, defined by the function `State32`, and we want to trustlessly simultaneously update all the channels to state 33. When the game manager reveals the secret, it causes all the channels to update at the same time.

*b) **Metered execution:*** Contract execution is metered in a way similar to Ethereum's "gas", but TRX+ uses two different resources for its metering, one for time and one for space. Both of these are paid for using aeon by the party that requests the execution.

This could be seen as undesirable, because it is probably another party that is causing the need for the blockchain to resolve the dispute in the first place. However, as long as all money in the channel is not used for betting, this can be effectively nullified in the contract code, since it has the ability to redistribute funds from one party to the other. It is in fact generally good practice to avoid using all funds in a channel to transact, because it disincentivizes the losing party to cooperate when closing the channel.

*B.2) **Example:*** Let's bring all of these ideas down to earth. In practice, if Alice and Bob want to transact using a state channel on TRX+, they go through the following procedure:

1) Alice and Bob sign a transaction that specifies how much money each of them is depositing into the channel, and publish it to the blockchain.
2) Once the blockchain has opened the channel, they can both create new channel states, send them between each other and sign them. Channel states can be either a new distribution of the funds in the channel or a contract that determines a new distribution. Each of these channel states has an increasing nonce and are signed by both parties, so if a dispute arises, the latest valid state can be submitted to the blockchain, which enforces it.
3) The channel can be closed in two different ways:
    a) If Alice and Bob decide that they have finished transacting and agree on their final balances, they both sign a transaction indicating this and submit it to the blockchain, which will close the channel and redistribute the money in the channel accordingly.
    b) If Alice refuses to sign a closing transaction for any reason, Bob can submit the last state that both of them signed and request to have the channel closed using

this state. This starts a countdown. If Alice believes that Bob is being dishonest, she has the opportunity to publish a state with a higher nonce that both of them have signed before the countdown finishes. If she does so, the channel closes immediately. Otherwise it closes when the countdown has finished.

C. Consensus mechanism

TRX+ uses a hybrid Proof-of-Work and Proof-of-Stake consensus mechanism. The block-order will be determined via Proof-of-Work. Certain system variables will be determined via on-chain prediction market system, which allows the users to participate and bring in their knowledge. For the PoW algorithm we currently favor a variant of Tromp's Cuckoo Cycle, one which is memory bound, and also is an "indirectly useful Proof-of-Work", as it requires less electricity to run, but instead has another limiting factor, the one of memory latency availability. This also makes it feasible to mine with a smart phone.

Tromp writes about his work:

> "[Cuckoo Cycle is] an instantly verifiable memory bound PoW that is unique in being dominated by latency rather than computation. In that sense, mining Cuckoo Cycle is a form of ASIC mining where DRAM chips serve the application of randomly reading and writing billions of bits.
> When even phones charging overnight can mine without orders of magnitude loss in efficiency, not with a mindset of profitability but of playing the lottery, the mining hardware landscape will see vast expansion, benefiting adoption as well as decentralization."

Preview: The consensus mechanism has a somewhat non-standard role in TRX+. In addition to agreeing on new blocks for the blockchain, it also agrees on both answers to oracle questions and the values of the system's parameters. In particular, the consensus mechanism can change itself. However, it should be noted that this is not entirely unproblematic. For example, if a simple proof-of-work mechanism was used, it would be rather cheap to bribe the miners to corrupt the oracle. Therefore TRX+ is going to use a novel hybrid Proof-of-Stake Proof-of-Work algorithm, leveraging the benefits of both. Independently from this, PoW is going to be used to issue new aeon tokens.

*C.1) **Oracles:*** A crucial feature for most contracts, whether encoded as text or as code, is the ability to refer to values from the environment, such as the prices of different goods or whether a certain event occurred or not. A smart contract system without this ability is essentially a closed system and arguably not very useful. This is a generally accepted fact and there are already several projects that attempt to bring external data into the blockchain in decentralized way **augur**. However, to decide whether a supplied fact is true or not, this essentially requires the implementation of a new consensus mechanism along side the block-ordering consensus mechanism.

Here is how the oracle will work. Anyone can launch an oracle by paying enough money to run the oracle. They ask a yes/no question, and they give a timeframe for when we will know the answer to the question. The oracle is a market that measures the coorelation between the difficulty of the blockchain, and how the question will get answered. We expect that the fork of the blockchain that has higher difficulty is also the one who's oracle is honest.

D. Governance

Governance of blockchain-based systems has been a big problem in the past. Whenever a system upgrade needs to be done, this requires a hard fork, which usually leads to big discussions among all value holders. Even simple things, like correcting an arbitrarily set variable in the source code, as we have seen with the block size debate in Bitcoin, seem to be very hard in a system where the users' incentives are not aligned with the decision makers, and where there is no clear upgrade path. We have also seen more complicated governance decisions, like fixing a single smart contract bug in "The DAO", which required quick intervention by system developers.

The primary problem of these systems is easily identifiable—the decision-making process for a protocol upgrade or change is not well defined and lacks transparency. TRX+'s governance system is part of the consensus. It uses prediction markets to function as efficiently and transparently as possible.

Moreover, the consensus mechanism is defined by a number of variables that determine how the system functions and that are being slightly updated by each new block. From how much it costs to make transactions or ask an oracle, to modifications of fundamental parameter values like the block time.

By having prediction markets about the variables that define the protocol, the users can learn how to efficiently improve the protocol. By having predictions markets about potential hard forks, we can help the community come to consensus about which version of the code to use. Each user chooses for itself which metric it seeks to optimize, but a simple default strategy would be to maximize the value of its holdings.

E. Scalability

*E.1) **Sharding trees:*** The architecture that has been presented thus far is highly scalable. It is possible to run the blockchain even when each user only keeps track of the part of the blockchain state that they care about and ignores everyone else's data. At least one copy of the state is needed for new users to be certain about the substate that they care about, but we can shard this data across arbitrarily many nodes so that each node's load is arbitrarily small. Merkle trees are used to prove that a substate is part of the state **github-merkle**. It is easy to imagine a scenario where certain nodes specialize on keeping track of the trees and get paid for inserts and look-ups.

*E.2) **Light clients:*** Light clients don't download the entire blocks. First the user gives their client a hash in the history of the fork they prefer, a technique also known as *weak subjectivity* **weak-subjectivity**. Then the client knows only to download forks that include a block with that hash. The client only downloads the headers of the blocks. The headers are much smaller than full blocks; very few transactions are processed. For simplicity, we made no mention of the block headers when discussing the block structure in section VIII-A.4, but they contain the following:

- The hash of the previous block.
- The root hash of all of the state trees.

*E.3) **State channels and parallelism:*** State channels have immense throughput and most transactions inside them are never executed or even recorded on the blockchain. Additionally, the channels don't write to any shared state on-chain, so all transactions that actually *do* get recorded on the blockchain can be processed in parallel. Given that most consumer hardware sold today has at least four processing cores, this has the immediate effect that transaction throughput is multiplied by roughly a factor of 4.

Furthermore, the fact that there will never be any complex concurrent interaction suggests that sharding this blockchain architecture should be relatively easy. Since blockchain sharding is still fairly experimental, we have deliberately chosen not to pursue any sharding techniques in the initial design of TRX+. However, if this changes in the future, TRX+ should be one of the easiest blockchains to shard.

*E.4) **Transactions per second at a given memory requirement:*** The variables that define the protocol are all constantly being updated by the consensus. From their initial default values, we can calculate the initial default rate of transactions per second.

```
Note that this is a draft and will likely
change.

We define the following variables for the following calculations:

B = block\_size in bytes
F = blocks\_till\_finality
R = time\_till\_finality in seconds
T = transaction size in bytes

transactions per second = B * F / (T * R)

B = 1000000 bytes = 1 megabyte per block
F =  24*60*2 blocks per day
R / F = 30 seconds per block
R = 24*3600 seconds per day
T = 1000 bytes per transaction

1000000 * 24*60*2 / 1000 / 24*3600
= 1000000 / 1000 / 30
= ca. 32 transactions per second (fast enough to sign up every human within 8 years)
```

To operate a node, we need to keep a copy of all the blocks since finality, and we need to be able to record 100 times more information, in case there is an attack. Estimating that finality is 2 days, then there would be 5760 blocks till finality. So the memory requirement is 5760 * one megabyte * 100 = 576000 megabytes = 576 gigabytes. When there isn't an attack happening, one would only need about 5.76 gigabytes to store the blocks.

IX. Applications

The stateless nature of the TRX+ smart contracts makes it easy to build the following applications on TRX+'s blockchain. It is especially suitable for high-volume use-cases.

A. Blockchain essentials

Blockchain essentials are necessary primitives like aeon, wallets, names and related concepts. They modularize reusable components which can be used as application foundations and can be improved on.

*A.1) **Identities:*** Each account will have an associated unique ID number. Users can register unique names, and link names to the Merkle-root of a data structure. The data structure can contain one's unique ID as well as other information about one's account. We aim to use Schema.org's JSON format to represent things like persons or companies **schemaorg**.

*A.2) **Wallet:*** A wallet is a piece of software that is used to interact with TRXplus. A wallet manages private keys for the aeon and creates and signs transactions. One can use the wallet to send channel transactions, and use apps in the channel network.

*A.3) **Proof of existence:*** One transaction type allows for the publishing of the hash of any data. System participants can use the headers to prove that the data existed at that point in time.

B. State channel applications

Smart contracts in state channels are perfect for micro-services on the web that require a high transaction throughput.

*B.1) **Toll API:*** Most APIs existing today are publicly available for anyone to call, or else they are secured by a username-password–scheme or unique access tokens. Payment channels allow for a new kind of API, where one pays for every call to the API, possibly every HTTP-request. Paying to access an API solves DDoS problems, and it makes it easier to build high-quality APIs that are always available. API responses that require a payment are fundamental for the creation of as of yet impossible types of businesses and can play an important role in the emergence of the decentralized economy. They create incentives for information owners to make otherwise private data publicly available.

*B.2) **Insured crowdfunding:*** We can implement insured crowdfunding using dominant assurance contracts [need cit.]. These are smart contracts that are used to raise money for a public good, like a new bridge, a school or a market.

Dominant assurance contracts differ from traditional assurance contracts like Kickstarter, in that they make it a dominant strategy to participate. If the good is not funded, all participants get their aeon back plus interest, so they are insured against reducing their liquidity without receiving the good. Using an oracle, we can ensure that the provider of the good or service only gets paid if the good or service is actually provided.

*B.3) **Cross-chain atomic swaps:*** Cross chain atomic swaps allow for trustless exchange of aeon for bitcoins **atomic**, **interledger**. These can be implemented using a hashlock, that locks the transactions on both blockchains under the same value.

*B.4) **Stable value assets and portfolio replication:*** We can use smart contracts to program synthetic assets that stay nearly the same price as a real world asset. For example, we could make an asset that stays the same price as gold. Synthetic derivatives are created in equal and opposite pairs. For one user to have an asset that moves with gold, a different user will have to have an asset that move inversely to gold. For example, Alice could make a contract with Bob so that Alice owns 1 gram of gold. Out of the money in the contract, one gram of gold worth of aeon will go to Alice, and the leftover money goes to Bob. The contract has an expiration date when the price of gold will be measured, and the funds distributed to Alice and Bob accordingly.

*B.5) **Event contracts:*** Event contracts pay when an event happens and don't pay when an event does not happen, as per the oracle's telling. Apart from being interesting in themselves, these can be used by several different applications:

*a) **Insurances:*** We can use event contracts to implement insurances. For example, expensive music event tickets can become worthless if the weather goes bad. However, if the concert-goer receives money if the oracle decides that it rained on the day of the event, the investment can be protected so that one can afford to find an emotionally-adequate alternative. Slightly more seriously, farmers are often interested in the total number of inches of rain in a season. We can insure them against their crops wilting from dryness.

*b) **Whistleblowing:*** Event contracts can also be used to incentivize revealing sensitive information. For example, we could bet on the event "Information indicating that Company A has used illegal pesticides was released on or before January 24th, 2017". Any person with access to such information would be incentivized to first bet that the event will happen and then release it.

*B.6) **Prediction markets:*** A prediction market works by letting users bet on whether a future event will happen. From the price of the bets we can predict the future likelihood **hivemind**, **augur**, **promisepredmarket**. They are the most accurate way to measure the future at a given price [need cit.]. Once the event has happened, the market is settled using the oracle.

As noted in section VIII-D, we can for example use prediction markets to predict which updates to the software will be beneficial, and which will be harmful. We can also use them to estimate how much candidates in an election will actually be able to accomplish, so lies and baseless promises can be detected more easily.

*a) **Multidimensional prediction markets:*** Multidimentional prediction markets allow us to predict the correlation between possible future events. So for example, one could

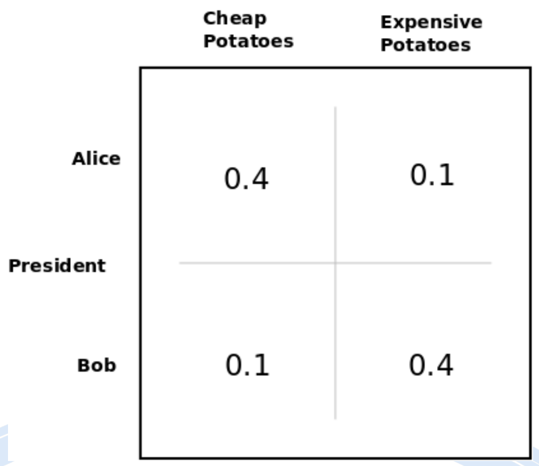


Fig. 12. Multidimensional prediction market.

predict that if Alice is elected leader, the price of potatoes will go down, and that if Bob wins, the price will go up. One could learn that if Google uses plan A for the next 3 months, that it will probably earn more money, and that if it uses plan B, it will probably earn less. Or, as in fig. 12, we can see that if Alice would be elected president, there is a high likelihood of the price of potatoes being rather low.

B.7) Market with batch trading at a single price: There are two approaches available to attackers that want to rob aeon from a market. They can take advantage of the market being split in time, or they can take advantage of it being split in space.

- If the market is split in space, then the attacker does arbitrage. He simultaneously makes trades in both markets at once so that his risk cancels out and he earns a profit.
- If the market is split in time, then the attacker front-runs the market. He reads the transactions coming into the market and creates buy and sell orders immediately before and after.

To combine markets in space, everyone should use the same market maker. To combine markets in time, we need to have trading done in batches, at single price. The market maker needs to commit to each person what price he decided, and if anyone can find contradictory commitments from the market maker, then all of his customers should be able to drain all of his channels. If the market maker commits to a fair price, then he will match the same volume of buyers and sellers together, as fig. 13 shows. Otherwise, he will end up in a situation similar to fig. 14, thus taking a large risk.

X. Implementation

Most key concepts already have proof-of-concept implementations in Erlang. This includes the blockchain itself, the contract language and VM, the oracle and governance mechanisms, as well as an old version of the consensus mechanism. We have used Erlang/OTP because it makes it easy to write code that can respond to many requests in

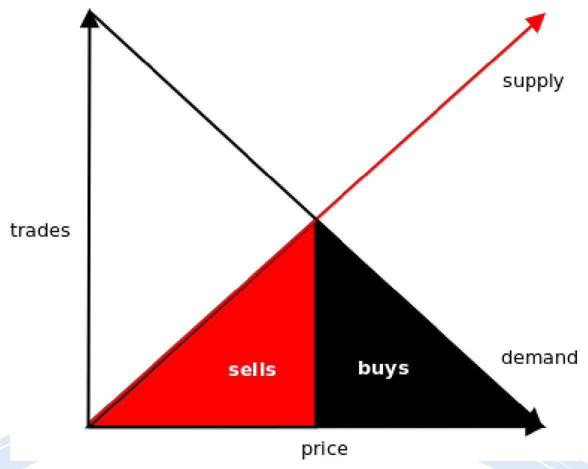


Fig. 13. The black line is the demand curve, the red line is the supply curve. The sells in red are the same size as the buys in red. The vertical line is the price the market maker selected. Everyone willing to buy at a higher price traded at that price, everyone willing to sell at a lower price traded at that price.

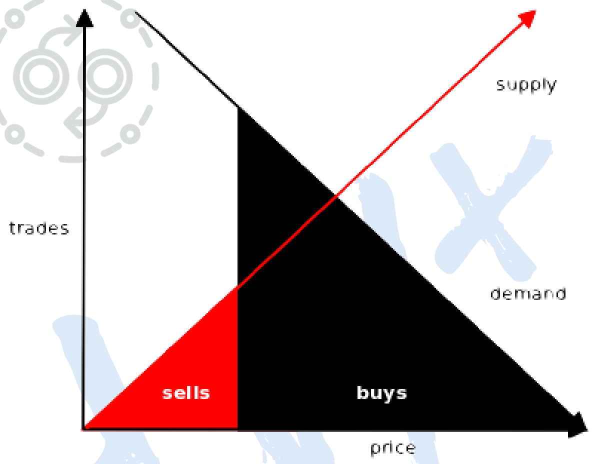


Fig. 14. The black is much bigger than the red. The market maker is selling many more shares than it is buying, thus taking on a lot of risk.

parallel and does not crash. The servers with the highest up-time in the world are based on Erlang. It has been used for industrial applications for 30 years, proving itself to be a reliable and stable product.

A. Virtual machine and contract language

The virtual machine is stack-based and similar to Forth and Bitcoin' scripting language, although in comparison to the latter, it is rather rich. The VM supports functions instead of gotos, making its semantics relatively simple to analyze. A list of the VM's opcodes can be found on our Github[3].

[3]https://github.com/TRXplus/chalang/blob/master/opcodes.md

Additionally, there exists a higher-level Forth-like language called *Chalang*, which compiles to bytecode for the VM. It supports macros and variable names, but keeps the stack-based execution model **chalang**. Examples of Chalang code can also be found on our Github[4].

B. Adoption via web-integration

The web is the most popular application platform. We will provide easy-to-use web-development tools, such as JS-libraries and JSON-APIs for the core features of the TRX+ blockchain.

C. Open source modules

In order to be easily re-used for private blockchain consortium and other use-cases, the software will be written in MIT-licensed modules, such as a consensus module, that can be adapted to specific needs.

D. Usability and UX design

Frictionless human interaction will be a big focus of our development efforts. More specifically, we will make sure that who controls the identity, keys and transactions is clearly established. Also, offering easy access via web-gateways will be a central focus of future development. Users participating in prediction markets via a Tinder-like (swipe left/right) mobile interface, and simple web-wallets that can be easily integrated in a website through an iframe will be the new norm.

XI. Discussion

AI has surfaced and resurfaced in several waves, but it's only recently that it has become commonplace. The widespread affordability and use of AI in software development has been seen as a revolution. The first AI revolution, the one we are currently witnessing, is about AI-as-a-Service. The book Artificial Intelligence for .NET: Speech, Language, and Search (Apress, 2017) gives an in-depth of creating AI-enabled software applications. With advancements in IoT and the emergence of Blockchain, AI is on brink of a second revolution, one that involves creating complete product offerings with intelligent software and custom hardware. "In the revolutions we have talked about, the real hero is not Cloud, nor it is AI or IoT or Blockchain. These are only the enablers. What do they enable? That's easy—data, which will always be the real hero in all important revolutions. Think about it. Data is what's collected (IoT). Data is what's analyzed (AI). Data (results) is what's stored and propagated (Blockchain).

Now we have TRX+ Blockchain, a mechanism to openly and securely store large amounts of data that is available to each member on the network. TRX+ use is with

[4]`https://github.com/TRXplus/chalang/tree/master/examples`

transactional data, and it is often used in conjunction with NoSQL and relational databases.

We have provided an explanation of how to architect a fundamentally more efficient value transfer system. The described system is in fact a global oracle machine that can be used to provide decision making services at global scale. In particular, all the applications proposed in section IX can be built easily and efficiently on top of TRX+.

However, our approach has both fundamental limitations and avenues for improvement. These are discussed here.

A. Limitations and tradeoffs

While we do believe that the tradeoffs made in our architecture are reasonable given the resulting performance increase in other areas, TRX+ is not a catch-all solution for decentralized applications. It should rather be viewed as a synergistic complement to existing technologies. There are several caveats that one need to be aware of.

A.1) On-chain state: Despite having many advantages, TRX+'s lack of programmable state makes it unfit for applications that require a custom state to be under consensus. For example, this includes DAOs as they are usually conceived, custom name systems and subcurrencies which are not tied to the value of an underlying asset.

A.2) Free option problem: If Alice and Bob have a channel and Alice signs a contract, she essentially gives Bob a free option when she sends it to him: Bob can choose to sign and return (i.e. activate) the contract at any time in the future. Often this is not what is intended. To avoid this problem, channel contracts aren't immediately activated with the full amount. They are divided up in time or space. Both participants would sign up for the contract in small intervals so that neither user ever offers a large free option to the other.

For example, if the parties want to bet 100 aeon, then they might sign up to it in 1000 steps that each increase the bet by 0.1 aeon. This would require about 1000 messages to pass, 500 in each direction, which is cheap enough since the contract is never submitted to the blockchain. As another example, if one wanted to make a financial asset that would last for 100 days, one might sign up in 2400 steps of one hour each. This would require about 2400 messages to pass, 1200 in each direction.

A.3) Liquidity loss and state channel topologies: When composing channels using hashlocks as demonstrated in section VIII-B.1, any middlemen have to lock up at least twice as many aeon as will be transmitted through them. For example, if Alice and Carol want to transact through Bob, Bob will act as Carol when interacting with Alice, and vice-versa.

Since this is expensive for Bob, he would most likely earn a fee as compensation. If Alice and Carol expect to conduct many trades between each other, they can avoid this by creating a new channel and trustlessly moving the active contracts to the new channel using a hashlock.

Still, since keeping an extra channel open impacts one's liquidity negatively, going through middlemen is expected to be desirable in many cases, especially in cases where the parties don't expect to trade a lot in the future. Thus, a channel topology where certain rich users make money from trustlessly transmitting transactions between other users is expected to emerge.

It should be noted that this does not constitute a single point of failure, since we do not trust these transaction transmitters with anything. If a transmitter goes offline before the secret to a hashlock has been revealed, the transaction doesn't go through. If it goes offline afterwards, the only possible "negative" effect is that the transmitter is not able to claim its aeon.

B. Future work

There are several possible ways to improve on the current architecture.

B.1) Functional contract language: A reasonable future direction would be to experiment with high-level languages that adhere more closely to the functional paradigm. Keeping track of an implicit stack is generally error-prone and arguably not suitable for a high-level, developer-facing language. This should be rather easy given that programs are already pure functions (modulo some environment variables), and would greatly simplify both development and formal verification of contracts. If this is done, it could also make sense to revise the VM to be tightly coupled with the new language, to make the compilation less error-prone and less dependent on trust in the developers. Ideally, the translation from surface language to VM code would simply be a direct transcription of peer-reviewed research, though pragmatic concessions will likely have to be made.

B.2) Multi-party channels: Currently, all channels on TRX+ are limited to two parties. While multi-party channels can *de facto* be achieved through hashlocking, this can be expensive. Hence, we plan to investigate the possibility of adding support for n-party channels, with a m-of-n settlement mechanism.

ACKNOWLEDGMENTS

Thanks to Dinis Guarda for insightful discussions.

REFERENCES

[1] Satoshi Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf (accessed April 2020).
[2] Stuart Haber, W. Scott Scornetta: How to Time-stamp a Digital Document. www.anf.es/pdf/Haber_Stornetta.pdf (accessed April 2020).
[3] Bitcoin: https://bitcoin.org/en/ (accessed April 2020).
[4] Ethereum: https://ethereum.org/en/ (accessed April 2020).
[5] Ripple: https://ripple.com (accessed April 2020).
[6] Aleksander Berentsen, Fabian Schär: A Short Introduction to the World of

Cryptocurrencies. https://cdn.crowdfundinsider.com/ wp-content/uploads/2018/01/St.-Louis-Federal-Reserve-ashort- introduction-to-the-world-of-cryptocurrencies.pdf (accessed April 2020).
[7] Cambridge Bitcoin Electricity Consumption Index: www.cbeci. org (accessed April 2020).
[8] Hyperledger: www.hyperledger.org/ (accessed April 2020).
[9] Ala Al-Fuqaha, Nishara Nizamuddin, M. Habib Ur Rehman, Khaled Salah: Blockchain for AI: Review and Open Research Challenges. IEEE, 1.1.2019, p. 10127–10149, Electronic ISSN: 2169- 3536, DOI: https://doi.org/10.1109/ACCESS.2018.2890507.
[10] Tradelens: www.tradelens.com (accessed April 2020).
[11] Tradelens solution architecture: https://docs.tradelens.com/ learn/solution_architecture/ (accessed April 2020).
[12] Hyperledger Explorer: www.hyperledger.org/projects/explorer (accessed April 2020).
[13] Vinayak Agrawal, Sanjeev Ghimire: Perform analytics on blockchain transactions. https://developer.ibm.com/patterns/ use-db2-and-sql-to-perform-analytics-on-blockchaintransactions/ (accessed April 2020).
[14] Nifi: https://nifi.apache.org (accessed April 2020).
[15] Airflow: https://airflow.apache.org (accessed April 2020).
[16] IBM SQL Query: https://cloud.ibm.com/catalog/services/ sql-query (accessed April 2020)
[17] Boahua Yang: 10 Practical Issues of Blockchain Implementations. www.hyperledger.org/blog/2020/03/31/title-10-practicalissues- for-blockchain-implementations (accessed April 2020).
[18] Lukasz Golab, Christian Gorenflo, S. Keshav, Stephen Lee: FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second. https://arxiv.org/abs/1901.00910 (accessed April 2020).
[19] Mark Rakhmilevich: Blockchain Tables in Oracle Database. https://blogs.oracle.com/blockchain/blockchain-tablesin- oracle-database:-technology-convergence (accessed April 2020).
[20] BigChainDB: www.bigchaindb.com (accessed April 2020).
[21] Fluree: https://flur.ee (accessed April 2020).
[22] ProvenDB: https://provendb.com/homepage/ (accessed April 2020).
[23] Amazon Quantum Ledger Database: https://aws.amazon.com/ de/qldb/ (accessed April 2020).
[24] European Commission: On Artificial Intelligence – A European approach to excellence and trust. https://ec.europa.eu/info/ sites/info/files/commission-white-paper-artificialintelligence- feb2020_en.pdf (accessed April 2020).
[25]https://www.bloomberg.com/news/articles/2019-09-27/asia-s-640-billion-bad-loan-pile-lures-investors-deloitte-says.

[26] Nishith Pathak. “IoT, AI, and Blockchain for .NET.” iBooks.