БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ
ИНСТИТУТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

Jivko Jeliazkov

Smart Contract Platforms

DISSERTATION

for awarding the educational and scientific degree "Doctor"

in the professional field 4.6. "Informatics and Computer Sciences"

scientific specialty "Informatics"

Scientific advisor: Assoc. Prof. Hristo Kostadinov

Sofia, 2025 г.

Table of Content

Introduction

The objective of this dissertation is to develop approaches that address several pervasive challenges, including the incentivization of scientific research, the deployment and ongoing maintenance of business software systems in real-world operational environments, and the generation of provably random numbers.

The solutions are implemented using a distributed blockchain [1-10] based architecture that meets the requirements of modern software for the delivery and implementation of business software systems. A major part of the dissertation (Chapter 3) covers the stages of the systems development life cycle (SDLC) [11-14], with special attention paid to the phase related to the implementation of the software by end customers, as well as subsequent updates of the software versions. The importance of this process requires that it be carried out in a reliable manner, with minimal disruption to business processes and full traceability of the responsibilities of each participant. The architecture of the blockchain-based systems

envisages the automation of steps related to specific responsibilities and key performance indicators (KPIs) through smart contracts, so that the affected parties are notified in a timely manner. An important place is also occupied by the analysis of the security and reliability of the processes, guaranteed by blockchain protocols based on cryptographic algorithms.

Blockchain technology is a relatively new technology, and its beginning is often associated with the Bitcoin [15-18] cryptocurrency. However, unlike cryptocurrency, what blockchain software developers are interested in are the advantages of the technology itself with its characteristics such as distributivity, security, reliability and qualitatively new architectures of communication between participants in business processes. These characteristics make it very suitable as the basis for the developed blockchain system. Due to its modern nature and relevance, there are currently dozens of variants of blockchain platforms with seemingly very similar, but at the same time quite different characteristics. This requires a detailed examination of the popular available platforms and the selection of the most suitable one to meet the requirements of each process.

An important part of the dissertation is the development of a prototype that demonstrates how the main modules, the connections between them and the communication protocols solve the set goals for the given architecture. The prototype in chapter 4 is built on the EOSIO/Antelope/Vaulta blockchain platform [19-21], which is open source and is distinguished by processing transactions in a fast, scalable and secure manner. For large-scale data storage, the InterPlanetary File System (IPFS) [22-24] is used - a new generation of decentralized distributed storage network called the Interplanetary File System. IPFS uses content addressing to uniquely identify each file in a global namespace. Files uploaded to IPFS are distributed across multiple computers and assigned a hash value that allows users to locate them.

Relevance of the problems

1. Creating distributed systems for incentivization of scientific research is a topic that is not widely discussed. It allows different parties to contribute to driving scientific research. Some by programmatically define a scientific research topic, others may add to the define price for finding the solution and most importantly

the ones that try and solve the defined problem. The whole set of interactions or parts of it may strictly be modelled by blockchain interactions is a distributed application.

2. The most important steps in the development life cycle of any software product are planning, implementation, testing, deployment and maintenance of the software. The stages of software deployment and maintenance are examined in greater depth, because in most cases they are critical for the business of companies that manage their processes through IT products. That is why the SDLC topic continues to be relevant and even becomes more critical for large business companies that have completely digitized their business and every area of their production, sales and deliveries depends on working and reliable software. Software that is updated on time and is resistant to hacker attacks. Software in which all problems are fixed on time, without interrupting production. Software that has the necessary speed and is compatible with new operating systems and new hardware servers and modules. For participants in the SDLC processes that ensure the implementation and maintenance of software in large business companies, there are challenges.
3. Provably random number generation is well-known topic with application in many IT and non-IT areas. Blockchains by design are naturally transparent and immutable medium that may be programmed to follow strict processes required to guarantee fairness of the selection process to all participants and external observers. A prototype is demonstrated in this dissertation that prove that even for high stake scenarios of selection and random number generation the results are clear and undeniable by any involved party.

Concurrently, blockchain technology is profoundly transforming established industries. Owing to its intrinsic properties—such as strong security guarantees, resilience derived from its distributed architecture, reliable transaction processing, and immutable preservation of the complete operational history without the possibility of modification or deletion—blockchain is regarded as a promising solution in domains that have traditionally struggled to meet these requirements.

Since the mining of the first Bitcoin in 2009 and the introduction of Ethereum [25–27] in 2015 as a blockchain platform supporting smart contracts, numerous industries have gained confidence in the reliability and advantages of this technology. This has triggered a surge of

novel software architectures that fundamentally reshape conventional business models and sectors. Prominent examples include insurance, healthcare, banking, finance, and others, all characterized by the participation of multiple actors in each process, where transactional information must be stored permanently and remain tamper-proof. In these contexts, dependable and traceable communication among participants is essential, and in many cases, the presence of an arbitrating entity is required to resolve disputes related to liability and compensation.

It is precisely the relevance and core characteristics of blockchain technology—its disruptive impact on traditional industries and its capacity to address complex, high-stakes scenarios—that enable the design of qualitatively new architectures grounded in blockchain platforms.

Dissertation structure

The dissertation is divided into an introduction and five chapters. The dissertation contains 135 pages, 48 figures, 3 tables, 57 cited literature sources and 2 appendices. 2 publications have been published on the dissertation, all of which are reports from international conferences.

The first chapter provides an overview of DLT technologies, their key characteristics, types and classifications. Concepts that are used further in the dissertation are explained. The main elements of the functioning of DLT are described and analyzed, as well as how communication security is ensured and how consensus is achieved. Main conclusions are made based on studied literature sources and directions and functionalities that are of importance for the design of the proposed new blockchain systems are formulated. The choice of a blockchain platform and technological means for implementing the system is justified. Characteristics of the various existing platforms are described, and an analysis of how their specificities could address the challenges in different areas is analyzed.

Chapter 2 describes a distributed system for incentivization of scientific research. It is based on EOSIO technology and allows different parties to set specific tasks for other parties to solve and provide incentives in the form of blockchain tokens.

Chapter 3 presents the traditional SDLC area with its key scenarios, process participants, as well as challenges in implementing and updating the software. The challenges are also described in terms of requirements for the new SDLC system, which should innovatively overcome the difficulties associated with typical software update steps. The most essential part, presented in Chapter 3, is the definition of the innovative SDLC blockchain design and an analysis of how it overcomes the existing classic SDLC challenges.

Chapter 4 describes the architecture of a practical system for generation of provably random numbers and results from building a simple but robust prototype based on the innovative architecture, describing in detail the environment in which it operates: interfaces, smart contracts, modules and agents, communication protocols, etc.

Chapter 5 describes the contributions of the dissertation, future directions of work, as well as scientific publications on the dissertation are described.

The final chapter 6 summarizes the results of the development of the three scenarios described in chapters 2, 3, and 4 and analyzes the different scenarios successfully addressed by the innovative architectures. The advantages are demonstrated, especially in the areas of traceability, security and reliability of the results.

Chapter 1. Overview and Analysis of Distributed Ledger Technology (DLT)

Blockchain is the most common type of distributed ledger technology (DLT), which consists of growing lists of records called blocks that are securely linked together using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. The timestamp proves that the transaction data existed when the block was created. Because each block contains information about the previous block, they effectively form a chain, with each block linking to those before it. Transactions in a blockchain are irreversible because, once recorded, the data in each block cannot be changed retroactively without changing all subsequent blocks, as shown in Figure 1.

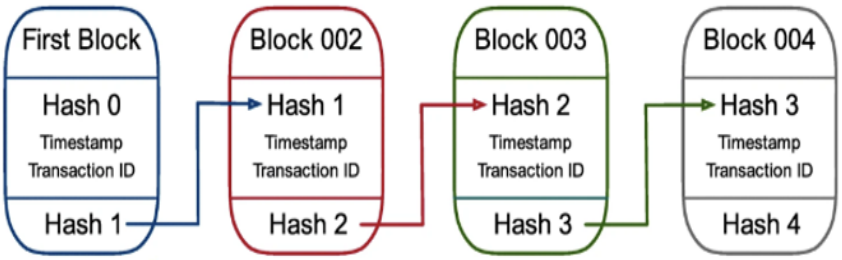

Figure 1 Blockchain structure

The blockchain is stored in a distributed network, where each participant in the network has a copy of the chain on their computer. In this way, there is no single, specific master copy, and there is no risk of failure, loss, or manipulation of information. The participants in the network are equal (peer to peer) and follow a specific protocol for validating new blocks. Once validated and recorded, no block can be changed without the approval (consensus) of the other participants in the blockchain chain, as shown on Figure 2.

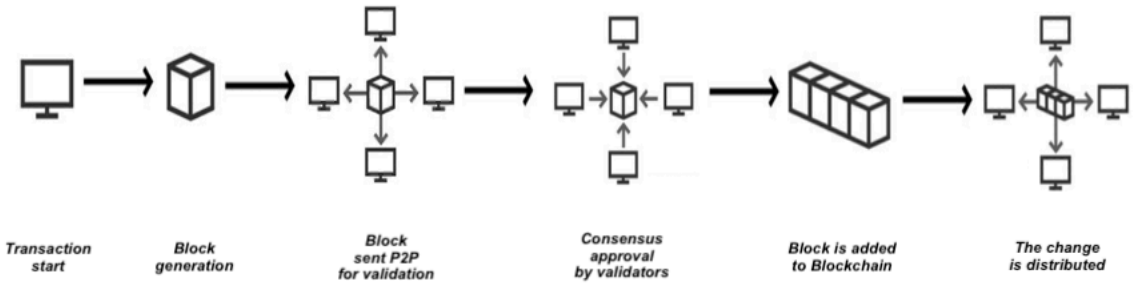

Figure 2 Adding a block to a blockchain

1.1. Historical information

The idea of a blockchain first appeared in 1982, when cryptographer David Chaum presented a dissertation titled "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups" [28], which described a design for a distributed computer system that could be created, maintained, and trusted by participants who did not know each other. A year later, he published the paper "Blind Signatures for Untraceable Payments" [29], which detailed a new form of cryptography that he said could enable an automated payment system in which third parties could not see the payment information. The idea behind blockchain technology was further developed in 1991, when researchers Stuart Haber and W. Scott Stornetta introduced a computationally practical solution for marking digital documents [30] so that they could not be backdated or forged. The system uses a cryptographically secure chain of blocks to store time-stamped documents, and in 1992 Merkle trees [31] were incorporated into the design, making it more efficient and allowing multiple documents to be combined into a single block. However, this technology remained unused, and the patent expired in 2004, four years before Bitcoin was created.

The concept of Proof-of-Work, which is key to the functioning of blockchain, dates to 1992, when Moni Naor and Cynthia Dwork were trying to deal with the problem of email spam. The concept was intended to discourage spam by requiring mail senders to perform some computational exercises before sending an email. A detailed description is published in the article "Pricing via Processing, Or, Combatting Junk Mail, Advances in Cryptology" [32].

In 1997, Adam Back developed a protocol called Hashcash [33], a more advanced concept for preventing email spam. Hashcash itself now includes a solution for double-spending, incorporating the so-called “double-spend protection” concept. Interestingly, these initial concepts were not themselves called Proof-of-Work.

In 1998, Wei Dai with his idea for b-money and Nick Szabo with his concept for Bit Gold made significant contributions to the future emergence of Bitcoin.

In 1999, two years after Adam Back’s concept of Hashcash, an academic essay by Ari Juels and Markus Jakobsson entitled “Proofs of Work and Bread Pudding Protocols” [34] was published. It was then that the term Proof-of-Work (PoW) first appeared. The essay describes the concept of proof-of-work as “a protocol in which a prover demonstrates to a verifier that he or she has expended a certain level of computational effort over a specified time interval.” This version is like the Proof-of-Work concept used in Bitcoin. The main difference is that this version is not based on an economic incentive.

In 2004, Hal Finney developed the concept of “reusable Proof-of-Work (RPOW),” which built on Adam Back’s concept of Hashcash. As such, this concept is still considered an important part of the development of digital money and a major precursor to Bitcoin.

In late 2008, an architectural document introducing a decentralized peer-to-peer electronic cash system called Bitcoin was published on a cryptographic forum by a person or group using the pseudonym Satoshi Nakamoto. Satoshi Nakamoto’s “Bitcoin: A Peer-to-Peer Electronic Cash System” contains the first detailed description of Bitcoin/blockchain. The nine-page document describes a concept for the reliable execution of financial transactions in a distributed network based on cryptographic algorithms.

Based on the Hashcash proof-of-work algorithm, but instead of using a hardware-trusted computing function like RPoW, Bitcoin's double-spend protection is provided by a decentralized peer-to-peer protocol for tracking and verifying transactions. In short, Bitcoins are "mined" for a reward using the proof-of-work mechanism by individual Bitcoin miners and then verified by decentralized nodes on the network.

On January 3, 2009, the first Bitcoin block was mined by Satoshi Nakamoto, who received a reward of 50 Bitcoins. The first recipient of Bitcoin was Hal Finney, who received 10 Bitcoins from Satoshi Nakamoto in the world's first Bitcoin transaction, which took place on January 12, 2009.

In 2013, Vitalik Buterin, a programmer and co-founder of Bitcoin Magazine, stated that Bitcoin needed a scripting language to build decentralized applications. Unable to reach consensus within the community, Vitalik began developing a new blockchain-based distributed computing platform, Ethereum, which includes scripting functionality called smart contracts. Smart contracts are programs or scripts that are implemented and executed on the Ethereum blockchain. They can be used to perform a transaction if certain conditions are met. Smart contracts are written in specific programming languages and compiled into bytecode that a decentralized virtual machine, called the Ethereum Virtual Machine (EVM), can read and execute.

Gradually, blockchain is being upgraded, and so from a technology initially created only as a carrier of electronic currency, over time, it has developed into a standalone and independent platform on which traditional business scenarios can be based, implemented in a new, much more secure, intelligent and automated way.

1.2. Relationship between blockchain technology and DLT

Very often, DLT and blockchain are used as the same definition for distributed digital databases and the difference between them remains misunderstood. The basis is the concept of a distributed database between many servers, without the need for a central server, thus ensuring reliability and secure access to the database, as well as, of course, other important features such as authenticity, security, etc. And while the definition of DLT is exactly as described above, blockchain is a specific and most popular implementation of this definition. In simple terms, the relationship in this case is like a fruit - an apple. Sometimes it is easier to describe an apple and then generalize what a fruit is. It can be said that blockchain made DLT a popular term after the entry of Bitcoin into the market in 2009. Since then, due to different scenarios and business needs, different subtypes of DLT have been defined, which differ in the way they are implemented.

DLT is a definition of an encrypted and distributed database that serves as a ledger in which records of transactions are stored. At the heart of DLT is an innovative approach to a database, in which cryptography is used for each transaction update and verification becomes possible in the specific distributed network. Depending on how this distributivity is implemented, depending on how the information is structured and validated, we observe

several types/subsets of DLT. Blockchain is the most recognizable type of DLT. In it, transactions are added in blocks that are sequentially linked by a cryptographic hash. Another very important property of blockchain is that all "nodes" in the network are equal, have access to the entire chain and have the right to participate in the validation of a new block. Some of the disadvantages (for specific business scenarios) of blockchain are the expensive mechanism for validating each block, as well as the ability of each "node" to see all transactions. In real life, there are business scenarios in which a company wants only a subset of its customers, partners, or subcontractors to have access to certain information. This requires that there are "nodes" with special privileges, and that the distribution of information is controlled (public/private). The same applies to the validation of information, and the process does not always have to be slow and energy-consuming. Sometimes there may be trusted nodes that have the privilege of approving transactions. To respond to these scenarios, subtypes of DLT appear, some of which do not store information in blocks, and not all nodes are equal. Figure 3 shows DLT subtypes according to the way they are accessed and the way transactions are stored. Figure 4 shows a classification of blockchains according to the way information is validated.

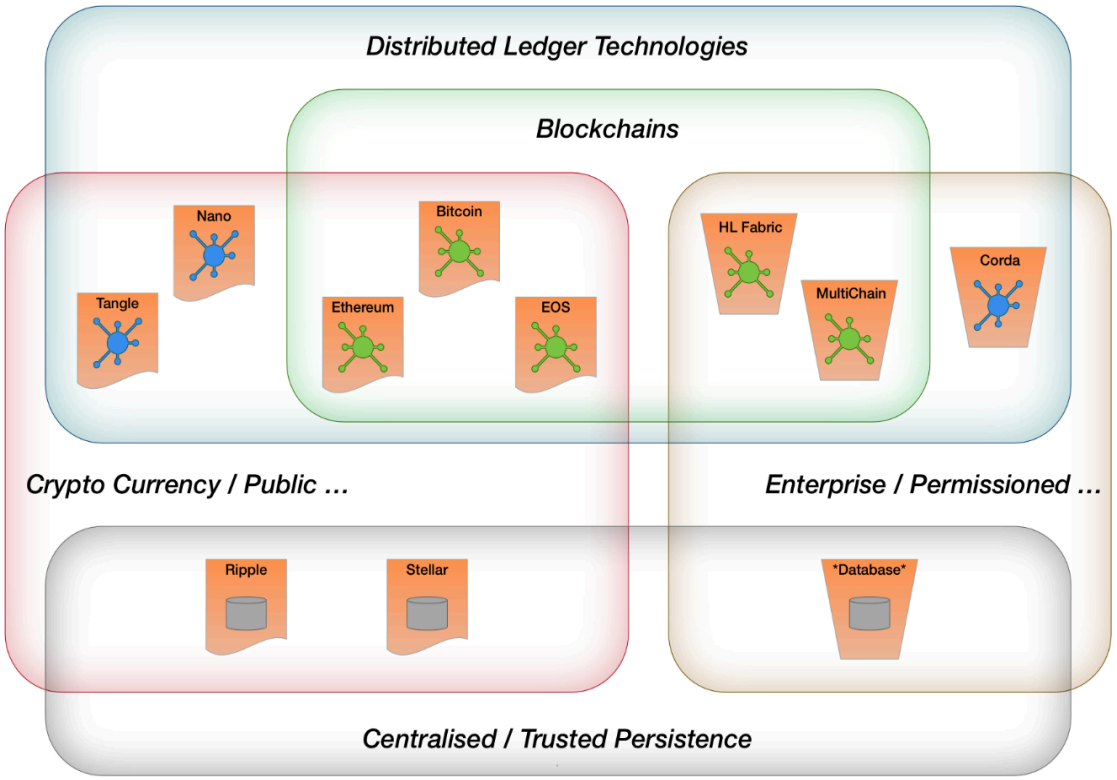

Figure 3 Types of DLT

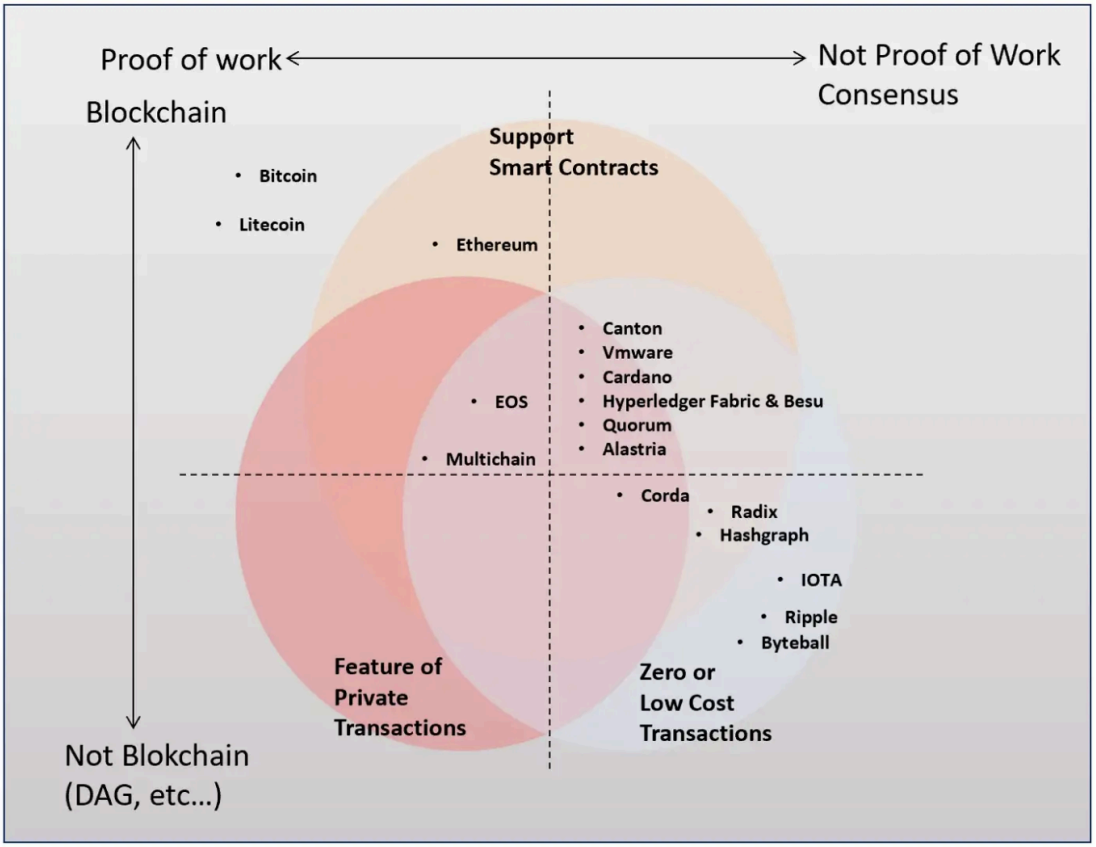

Figure 4 Consensus mechanism in different types of DLT

Before describing the advantages and disadvantages of each DLT subtype, it is necessary to consider in more detail the main concepts and stages of DLT operation. In the next chapter, along with the similarities of the main stages of validation, distribution and storage of information, some of the significant differences between the different subtypes will be discussed. This will allow us to assess at a later stage which subtype is suitable for solving a specific business scenario.

1.3. How it works

At the beginning, the main stages of the blockchain operation will be considered - from the initialization of a transaction, its validation, the creation of a block and the distribution of the entire blockchain in a distributed p2p network. It will also be analyzed who the participants in this process are and what rights and capabilities they have. In the course of the

description, the most significant differences between blockchain and other subtypes of DLT will be outlined.

The familiar graph is used again and the first step of creating a block containing information (transactions) is considered and how this block is validated by the participants in the network is tracked.

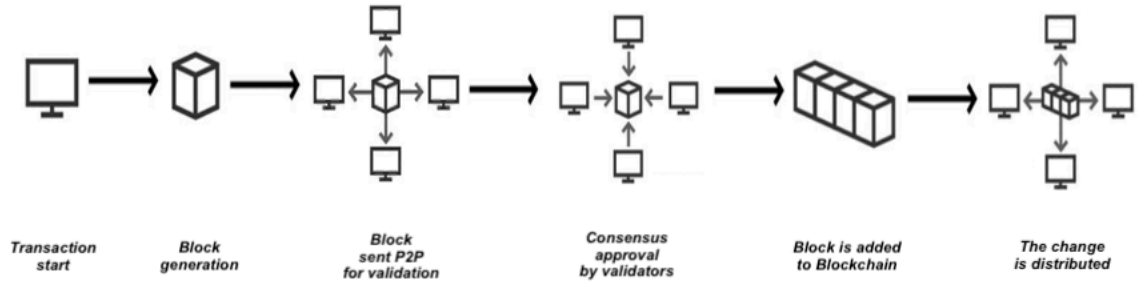

Figure 5 Executing a transaction on a blockchain

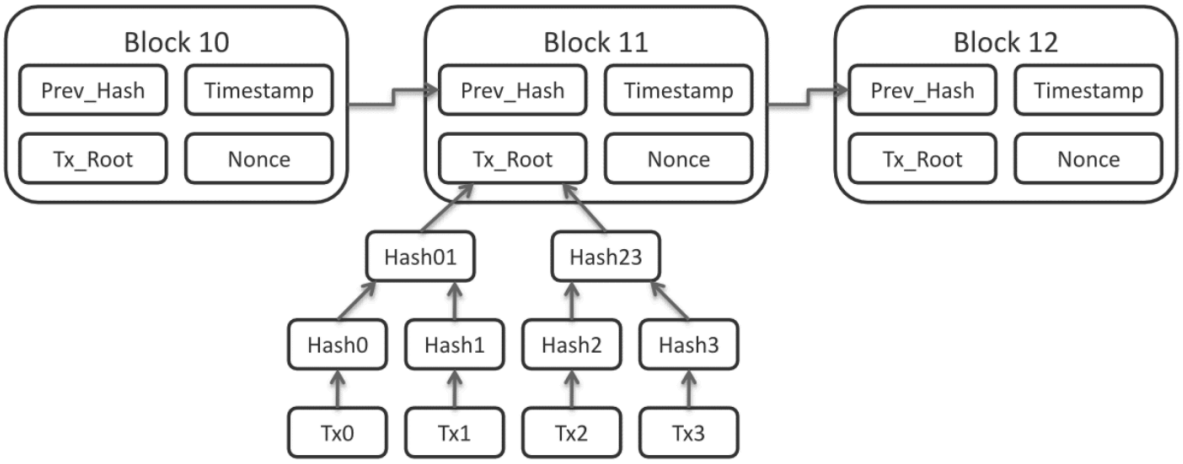

Figure 6 Typical block on the blockchain

Each block contains the hash code of the previous block, thus forming the connection in the chain. It also contains the timestamp of its creation. The hash root of the Merkle tree is built by recursively hashing pairs of transactions and thus encompasses the hash values of all transactions in the block. It also contains a nonce - a 32-bit integer. The nonce value is randomly generated when the block is created, after which a hash is generated for the block

header, with the target difficulty determining the minimum number of zeros at the beginning of the hash of the newly created block.

To track the recording of a new transaction in the distributed ledger, the process is divided into several steps:

- Status (at the time of writing) in the Bitcoin blockchain there are 923686 blocks, these details can be seen at https://www.blockchain.com/explorer/blocks/btc

- Newly arrived transactions are recorded in a new block, which must be added to the chain of the current 923686 blocks. The new block will contain the hash of the previous one, in this case 000000000000000001a0203dde0f822e6276f15148e6bb1fad020a52a70be4. In this way, the chain cannot be manipulated in practice, because if someone modifies, for example, the hundredth block, they will automatically change its hash value, and the next 101 blocks will no longer be connected to the chain. And all other participants in the process take the longest chain as the correct one.

-The next step is related to who will sign the new block and what the hash value of the new block will be. The Bitcoin blockchain uses the SHA-256 hashing algorithm, which, when applied to the input (block), will produce a 64-character string that is unique. Naturally, if the input is changed, the output string will also change. And this is where the nonce field comes into play. This field is added additionally and has no relation to the financial transactions in the block. Its purpose is to solve a cryptographic puzzle, in most cases requiring the hash code of the new block to start with 20 zeros. The process of repeatedly changing the nonce and hashing the block data to find a suitable signature is called mining and is what miners do. They have powerful computers and a lot of computing power, randomly changing the composition of the block (nonce) and hashing it until they find a suitable signature (output). The first one to guess the nonce receives a reward of several bitcoins.

-The last step is for the network participants to validate that the new nonce solves the crypto puzzle and to add it to the chain. After that, everything starts from the beginning. Considering that the maximum number of attempts to guess the nonce is at least 4 billion, and signing a block takes about 10 minutes, the more computing power a participant has, the greater the probability that he will guess the puzzle first. The difficulty of the puzzle itself is

configurable and depending on the number of participants and their computing power, the difficulty can increase.

The described process for validating transactions is called a consensus algorithm. The algorithm used in the blockchain is called “proof of work” (PoW), in which a cryptographic puzzle must be solved. In different DLTs, consensus algorithms are different and, in most cases, faster and do not require as much computing power. Each consensus algorithm has its own advantages and disadvantages, as described in chapter 1.7.

1.4. Cryptography and security in blockchain

It is often mentioned that DLT platforms are extremely secure and can withstand any hacker attacks. This is indeed the case thanks to the use of proven cryptographic algorithms. This chapter presents some of the concepts used. The blockchain platform is examined as a platform that has proven itself over time as impenetrable. With the clarification that if some platforms have been successfully attacked, this was in most cases due to incorrect implementation and not due to the architecture itself or the cryptographic functions used. Also, the frequent news about stolen cryptocurrency is in most cases due to a stolen private key or negligence on the part of the end customer.

After these clarifications, the concepts of wallet, address, transaction and signing of a transaction (money transfer) in blockchain are examined below:

In a blockchain for money transfer, an IP address is not used as an identifier of a sender or a recipient. Instead, an address is used that is generated using mathematical and cryptographic functions in a way that is secure and maintains the anonymity of its owner. The address uses the Pay To Public Key Hash (P2PKH) format, with a standard P2PKH having 26 to 34 alphanumeric characters and starting with 1. The concept is that when you pay for Bitcoin, you pay to a hash of some public key. Figure 7 describes how a P2PKH address is created in detail.

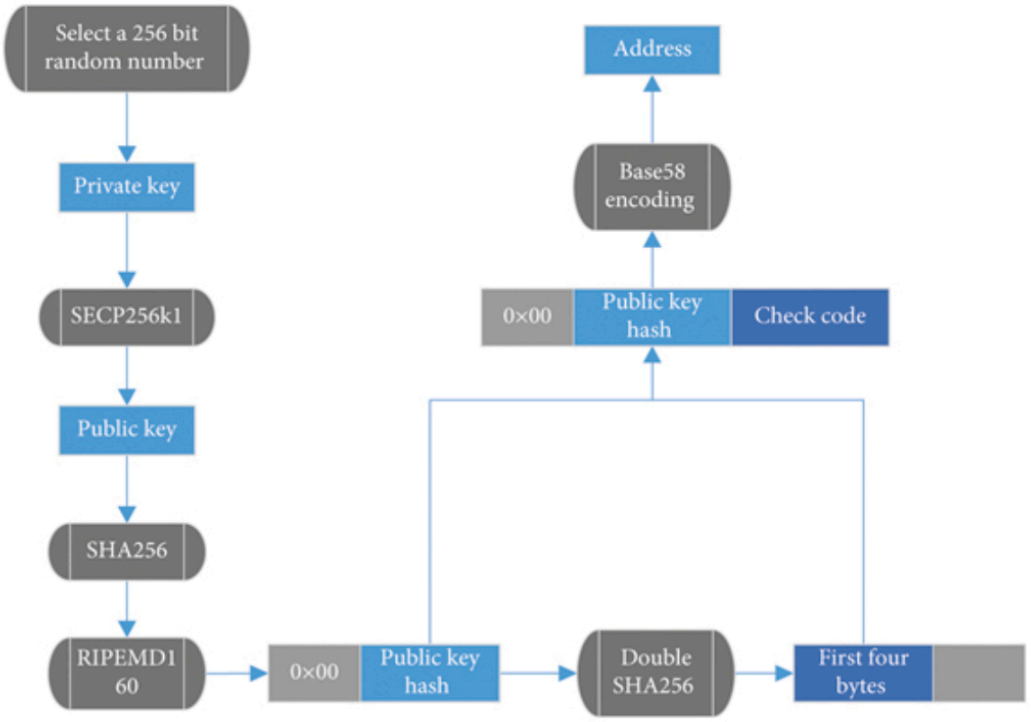

Figure 7 Pay To Public Kay Hash

To generate a Bitcoin address in P2PKH format, first a random 256-bit binary number is generated and used as the private key. The private key is then used to generate the public key using the SECP256k1 algorithm, and then a hash of the public key is generated using the SHA256 algorithm and the RIPEMD160 algorithm using the public key. 0×00 is the version number of the P2PKH address. The SHA256 algorithm is run twice on the public key hash, and the first four bytes are extracted as a checksum. Finally, the combined version number, public key hash, and checksum are Base58 encoded to generate a new Bitcoin address. The result is an address that looks like the number 17VZNX1SN5NtKa8UQFxwQbFeFc3iqRYhem.

The address is necessary in the case of financial transactions to ensure that payments reach the correct destination. It is a unique and secure identifier that facilitates payments. For greater security, as well as to prevent it from being traced, different addresses are usually generated. Most crypto wallets allow multiple private/public key pairs and, accordingly, multiple addresses to be generated from the "master private key". Something that even

modern banks have only recently started to offer as an option – a credit card for a one-time payment.

Using the described cryptographic scheme, the sender can sign the transaction with the recipient's public address or key, and only the recipient can read the transaction using his or her private key.

When determining the specific values of the attributes, two hash functions are used - one for calculating the hash of the block header, and the other for calculating the Merkle-root hash of all included transactions. The figure shows a detailed block structure showing all attributes and hash functions, where: HBH(Block) is a hash function H() for calculating the hash of the block header, and HT(TXN) is a hash function H() for calculating the hash of a transaction. The cryptographic algorithm SHA256 is used and is applied twice, which is known as "double SHA-256".

$$H_{BH}(Block) = SHA256\big(SHA256(Block)\big)$$

$$H_{T}(T_{XN}) = SHA256\big(SHA256(T_{XN})\big)$$

Considering the cryptographic task of calculating the nonce, which can reach over 4 billion attempts to guess it, we come to the conclusion about the precise design and security of DLT platforms, and particularly the blockchain.

Science of recent decades, especially mathematics, cryptography and the theory of distributed networks, has led to the creation of a qualitatively new platform (DLT), which can find application and fundamentally change typical business processes. Applications of DLT in business are discussed in more detail in the section "Applicability of DLT in business" at the end of this chapter.

1.5. Types of blockchain

Blockchain platforms can be built in different ways, with different rights for nodes in the network, different access to the network, and different consensus mechanisms. Depending on this, different types [35] of blockchains can be defined, as described in Figure 8.

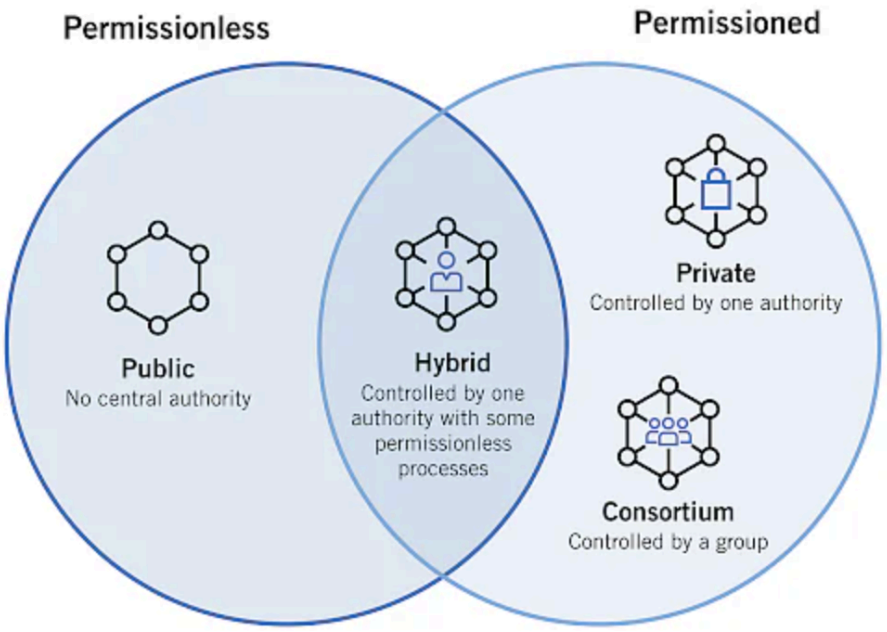

Figure 8 Types of DLT

Public Blockchain

A public blockchain is a decentralized network that anyone can join and participate in. Public blockchains allow all nodes to have equal access to the blockchain, allowing for the addition of new blocks of data and the validation of existing blocks of data. Popular public blockchains are Bitcoin, Ethereum, and Litecoin. They are most used for cryptocurrency exchange and mining.

Advantages: Trust, security, transparency.

Disadvantages: Slower transactions, high energy consumption to achieve consensus.

Private Blockchain

Private blockchains are centralized and managed by a person or organization that decides who can access the blockchain and be added as a node. Transactions on a private blockchain are not publicly visible but are verified through a consensus process among network members. Although its functionalities are like a public blockchain network, when it comes to peer-to-peer connectivity and decentralization, a private blockchain has a significantly narrower scope. For data privacy reasons, as well as for enterprise-level network sharing, a

private blockchain is the preferred type. Another example of its use is a B2B (business to business) virtual currency exchange based on Hyperledger.

Advantages: Speed, privacy.

Disadvantages: Building trust is more difficult, a network issue can limit nodes' access to key functionality.

Hybrid blockchain

Hybrid blockchains combine the characteristics of both public and private blockchain networks. This type of blockchain is often used in business applications where multiple organizations need to share data securely. A hybrid blockchain can share certain information publicly while keeping other information private. This allows for greater security and transparency while still maintaining a degree of confidentiality. An example is supply chain management, where subcontractors can easily join a private/corporate blockchain with multiple public blockchains.

Advantages: Building an ecosystem, security, privacy.

Disadvantages: Complex network to build and maintain.

Consortium Blockchain

In this type, a group of companies or organizations manage the process of granting permission to access the blockchain. They are more decentralized than a private blockchain, which provides greater security. Pre-defined nodes manage the consensus procedures in a consortium blockchain. This type of blockchain has a validator node, whose main function is to initiate a transaction, receive it, and validate it. Participant nodes can send or receive transactions. An example in this category is the Corda platform used in the field of finance and banking.

Advantages: Speed, building an ecosystem, confidentiality.

Disadvantages: Complex to build and maintain a network.

Based on the described classification, taking into account the advantages and disadvantages of each type of blockchain to solve a specific business scenario, it is assessed which type is most suitable to use.

1.6. Popular blockchain platforms

Figure 9 Popular blockchain platforms

Ethereum is one of the most popular and widely used blockchain platforms. It is also the first platform to allow developers to write new applications that can be configured and run on it. Created in 2015, it also introduces one of the unique concepts of blockchain, namely smart contracts. Smart contracts (also commonly known as intelligent contracts) are a type of program that contains commands and state/context. Each smart contract runs on a specific address on the platform. This allows them to send transactions and have a balance. All these operations are performed on the Ethereum Virtual Machine (EVM).

Developers develop and run decentralized applications (dapps) on the EVM. These are digital applications or programs powered by smart contracts that run on blockchains, rather than on centralized servers. They look and feel like regular mobile and web applications. However, the difference lies in the way they work and their key features:

▪ dApps do not need to be managed by a central server. This makes them resistant to censorship and fraud;

▪ dApps are open source;

▪ All data and records generated by dApps are stored on an immutable, public blockchain;

▪ dApps provide rewards to validators.

Ethereum smart contracts are written in the Solidity language. Ethereum is permissionless and open to the public. Its consensus mechanism was originally proof-of-work but later it was switched to proof-of-stake.

Ethereum has a cryptocurrency called Ether. Ether is used for payment, to create and initiate transactions on the Ethereum blockchain.

Hyperledger Fabric is a permissioned blockchain developed by Hyperledger Hub (Linux Foundation) and is aimed at enterprises that want to use, integrate or build blockchain-based solutions and applications. Hyperledger Fabric is like Ethereum not only in terms of the type of DLT, but also in its modular architecture. This modularity gives Fabric a kind of configurable interface where one can choose the preferred services, such as consensus algorithm, smart contract types, etc. Fabric smart contracts can be written in Go, Java and JavaScript.

EOSIO is a high-performance open source blockchain platform launched in 2018 by Block.one. EOSIO offers a fast, reliable, and highly secure platform for building blockchain applications.

EOSIO supports smart contracts written in the C++ programming language.

Developers choose the EOSIO platform for their blockchain projects because it is: fast and efficient, highly configurable, secure, compatible, and developer focused.

Corda is an open source blockchain platform created by the R3 Consortium in 2015. Corda was originally designed for financial institutions but has since expanded to serve additional areas such as healthcare, insurance, digital assets, and finance.

- Corda is a permissioned DLT and supports smart contract functionality. Corda smart contracts can be written in Java or Kotlin.

- The platform does not have a mining feature, so some transactions are never seen by most nodes. In other words, Corda transactions are not open to all nodes. There are no cryptocurrencies or tokens in Corda. Corda has configurable consensus, which means there are many consensus algorithms to choose from. An example of this is

- Validity consensus checks whether the transaction has been accepted by the contracts of each state and input and whether the transaction has all the required signatures

- Unique consensus agrees on a value if the input data for a transaction is unique and has not been used in other transactions.

Quorum is an open source blockchain platform based on Ethereum. Founded around 2016, it was designed to serve the financial industry and allow enterprises to "use Ethereum for their high-value blockchain applications."

Quorum was recently acquired by JP Morgan's ConsenSys. Many companies have implemented it in their businesses, including Microsoft, JP Morgan, Covantis, the South African Reserve Bank, SiaChain, Komgo, and others.

Quorum is permissioned, but it also allows for customization based on customer needs. In addition, Quorum supports both public and private networks, as well as smart contracts. Just like in Ethereum, smart contracts in Quorum are written in Solidity, which makes it very easy to switch from Ethereum to Quorum. Later in this chapter is given a detailed comparisons between several smart contract platforms to choose the platform that is most suitable for solving specific application development challenges and which will be the basis for the new smart contract platform applications.

1.7. Consensus mechanism

In the procedure for changing or adding data to the ledger, various approval methods are used. The way in which a decision on authenticity is made is called consensus among participants. This is a process of approval, confirmation of data by all or a majority of participants and ends with the signing of transactions. For example, the Bitcoin network uses the Proof-of-Work (PoW) protocol.

Depending on how the nodes in the network communicate with each other, as well as whether the network is part of a corporation or is public, there are different consensus mechanisms[36-38]. Some of the consensuses used are shown in Figure 10:

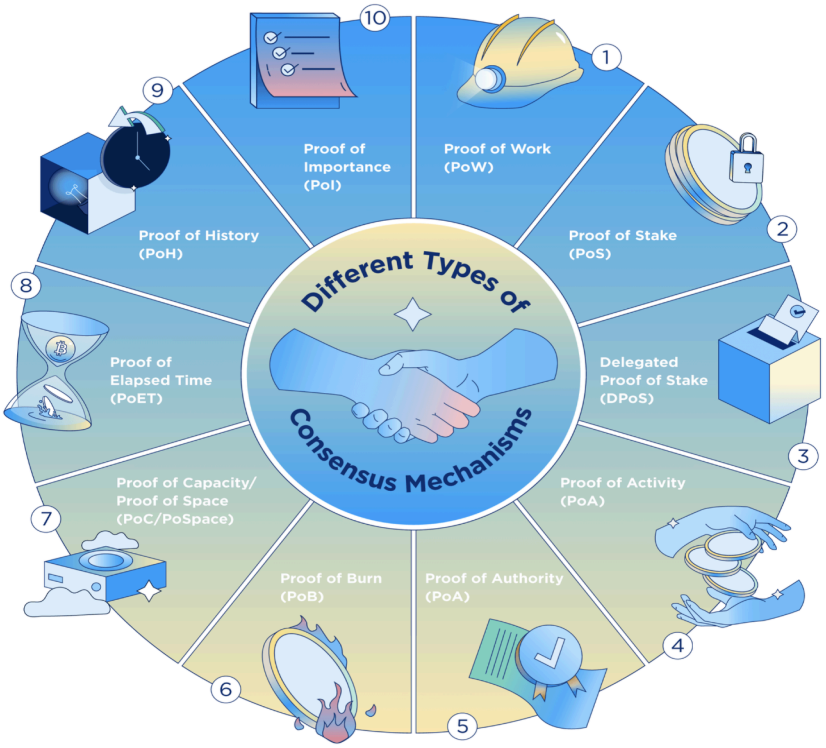

Figure 10 Types of Consensus Mechanisms

Proof of Work (PoW)

Scope: Bitcoin, Ethereum, Litecoin, etc.

The Proof-of-Work algorithm is the most popular consensus mechanism used in blockchain. The essence of the algorithm is that the network node performs intensive calculations to prove the transaction and add it to the chain, and the result can be easily compared with other

results of network calculations. The first person to complete the necessary calculations receives a reward for the work done, i.e. it also has a competitive nature.

Advantages of PoW:

- Confirmation cannot be forged;
- Proof cannot be obtained in advance, since the appearance of each new block requires the start of a new calculation cycle;
- Fairness in the distribution of rewards.

Disadvantages of PoW:

- Huge energy costs.
- Calculation takes a long time. This leads to a much slower transaction process, and systems with such a consensus mechanism are not suitable for micropayments or fast money transfers.

Proof of Stake (PoS)

Scope: Cardano, Ethereum

Proof of Stake (PoS) uses a randomized process to determine who has a chance to produce the next block. Blockchain users can lock up their tokens for a set amount of time to become validators. Once they become validators, users can create blocks. Typically, the user who owns the largest stake or has held coins for the longest period has a better chance of creating a new block.

Validators are typically rewarded for their work with a portion of the transaction fees for all transactions made in the block they create. With this approach, Proof of Stake offers incentives to validators to maintain the blockchain network. To remain fair and stable, the system requires mechanisms that would allow even smaller holders of currency to compete for the right to sign a block. Proof of Stake is more energy efficient compared to Proof of Work.

Pros of PoS:

- Reduced energy consumption compared to PoW. The verification process is fully virtualized and does not require large calculations.
- Faster decision-making. Since key players are given additional votes, it takes less time to reach consensus, making transactions faster.

Disadvantages of PoS:

- Lack of motivation for decentralization, as well as not many validators because nodes with large stakes win more often, which can potentially lead to security issues.
- Less proven security compared to PoW. While PoS has many benefits, it is less proven in terms of security compared to the long-established PoW mechanisms.

Delegated Proof of Stake (DPoS)

Scope: EOSIO/Antelope/Vaulta

In the delegated proof of stake process, users can stake their coins and vote for a certain number of delegates. The weight of a user's vote is based on their stake. The delegate who receives the highest number of votes gets a chance to produce new blocks. Delegates are rewarded with transaction fees or a certain amount of coins, just like in Proof of Stake.

The delegated proof of stake (DPoS) mechanism is one of the fastest consensus mechanisms in blockchain. It can handle a larger number of transactions compared to the proof of work mechanism. Due to its stake-weighted voting system, DPoS is often considered a digital democracy.

Pros of DPoS:

- Network members care about the fairness of transactions.
- Transaction speed is faster compared to PoS. Delegates can collaborate and not compete. Partial centralization allows for faster consensus.
- No high computational performance required. No need to list the entire blockchain.
- High network stability

Byzantine Fault Tolerance (BFT) (Byzantine Generals' Task)

Scope: Hyperledger

Byzantine Fault Tolerance is the ability of a computer system to continue operating even if some of its nodes fail or act maliciously.

In a BFT consensus algorithm, all nodes participate in the voting process to add the next block to the existing chain. When 2/3 of the participants vote to add the block, this means that the validation has passed successfully. Consensus works under the assumption that the number of malicious nodes cannot be equal to or more than one third of the total number of nodes in the network.

Advantages of BFT:

- Low commissions.
- Network scalability.

Disadvantages of BFT:

- Some degree of centralization through delegation of decision-making authority.

Consensus protocols can be classified by whether they are proof-based or voting-based, as well as by the type of blockchain they are used in. Part of the classification is shown in Figure 11.

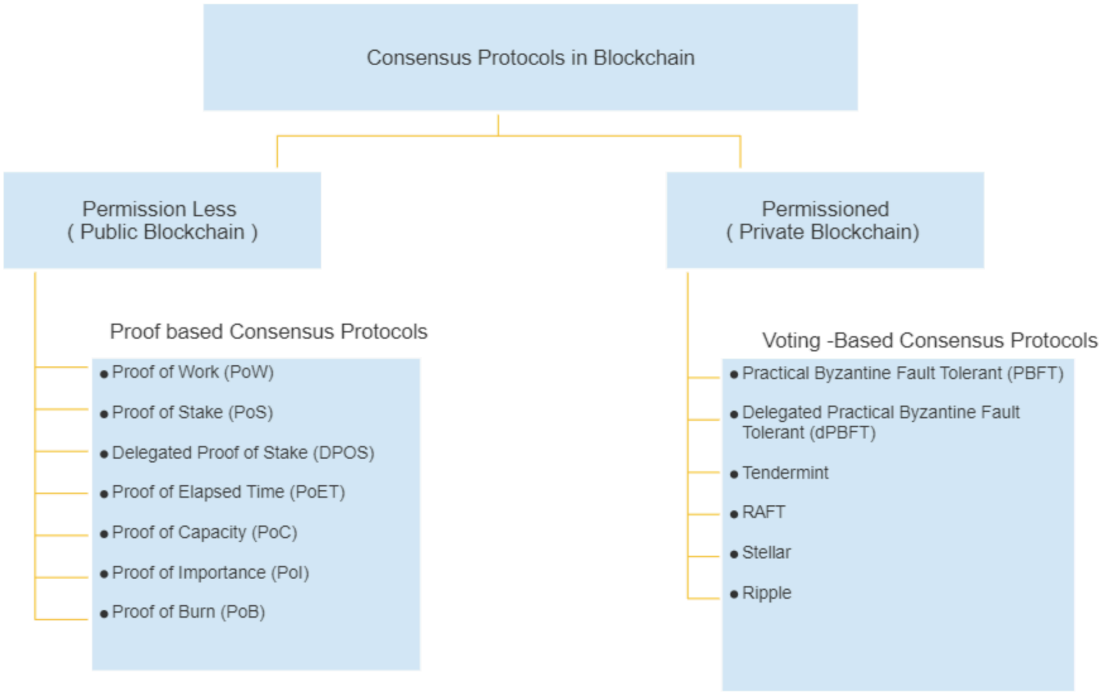

Figure 11 Classification of consensus blockchain protocols

1.8. Advantages and key features

Thanks to the use of p2p networks, cryptographic functions and consensus algorithms, DLT provides the following advantages:

▪ Decentralization - information is always available, because every computer in the network has a copy of the register.

▪ Transparency - information about transactions is stored in the public domain. However, this data cannot be changed.

▪ Unlimited - theoretically, the register can be supplemented with entries to infinity.

▪ Accessibility - everyone has access and can write, of course this is valid for certain types of DLT.

▪ Reliability - verification of information through consensus, as well as the impossibility of changing it at any time after recording in the register.

▪ Security - reliable cryptographic mechanisms are used to validate and protect information.

▪ Information can be anonymous or named - again due to the cryptographic functions used (public-private key).

▪ Automation – thanks to smart contracts, execution can be automated, for example, in the event of a flight delay, cash compensation can be automatically transferred to passengers.

▪ Efficiency – maintaining the register and translating transactions does not require high investments.

▪ Speed – transactions occur instantly depending on the choice of DLT.

All these qualities of DLT can drastically change current business processes, making them more efficient, accessible and reliable. The change in some business scenarios can be of enormous scale and lead to a qualitatively new experience for users. An example of this would be the process of registering a car or entering a property in the register. The next chapter examines the applications of these features in some of the most widely used businesses and services.

1.9. Applicability of DLT in business

Due to their key features, blockchain applications have great potential and are already being used in many industrial sectors [39-49]. They provide better product quality, greater consumer satisfaction, reduce costs, increase transparency and fairness, and improve market efficiency. Blockchain allows for the efficient storage of transaction, customer, and supplier data in a transparent, immutable online ledger.

Blockchain and DLTs have been successfully applied in practice in the many areas such as: Supply Chains and Logistics, Healthcare, Retail and e-commerce, Finance, Real Estate, Media, NFT Markets, Heavy Industry and Manufacturing, Music, Cross-border Payments, Internet of Things, Gaming, Privacy, Government and Voting, Advertising, Original Content Creation, Automotive, Smart Contracts.

An overview of blockchain usability in business is shown in Figure 12.

Figure 12 Blockchain Applications

Here are a few examples of applications in specific business areas.

Supply Chain: Every B2B is based on relationships with other companies, whether they are international partners or companies in the same region. Well-organized logistics contribute to business efficiency, reduce downtime, save money and optimize the overall workflow. Blockchain information can be shared between stakeholders, organizations, teams and countries. It allows all parties to track products and their corresponding information as they are delivered, shipped, purchased, used or consumed. The digitization of manual

processes, such as customs documentation, which relies on physical data entry and paper documentation, will improve both visibility and transparency across the industry. The same applies to the execution of contracts with a supplier without the need for an intermediate auditor – everything can be traced – transfer of ownership, origin. This creates a trustworthy relationship between every participant.

Government: Government blockchain applications can improve local political engagement, improve bureaucratic efficiency and accountability, and reduce enormous financial burdens.

Government blockchain applications have the potential to reduce millions of hours of bureaucracy each year by holding government officials accountable through smart contracts and digital ledgers that provide absolute transparency and create public records.

Blockchain voting applications can also revolutionize electoral processes. Blockchain-based voting can improve civic engagement and reduce voter apathy by providing a level of security and integrity that allows voting to be conducted on mobile devices.

Food: Using blockchain to store food supply chain data offers improved traceability of product origin, batches, processing, expiration dates, storage temperatures, and delivery.

Retail: Secure P2P marketplaces can track P2P retail transactions, with product, shipment, and waybill information entered into the blockchain, as well as payments made via Bitcoin.

Healthcare: Electronic medical records stored on blockchains enable the democratization of patient data and facilitate the transfer of records between providers. Blockchain networks are used in the healthcare sector to store and share patient data across hospitals, diagnostic labs, pharmacies, doctors, and nurses.

Insurance: The term smart insurance is increasingly being used, where when an insured event occurs, claims are automatically paid out through smart contracts on the blockchain. Premium costs are reduced as the need for auditing and data verification is eliminated.

Education: Educational institutions could use blockchain to store data to verify grades, degrees, and transcripts.

NFT blockchain applications: Unique (counterfeit-resistant) tokens used to prove ownership of digital, physical, or intellectual property.

Blockchain real estate applications: Exclusive real estate investments will become available as an investment opportunity for everyone. For the first time, real estate-based security tokens allow for proportional ownership of a plot of land or building, increasing market participation for everyone, and opening new financing opportunities.

Another factor that real estate blockchain applications can achieve is the highly efficient valuation of real estate investments based on anonymous and comparable data.

In this way, parties involved in a transaction can save on the financially expensive independent real estate valuations, since all parties have access to all available data on the blockchain. The entire history of the property is both transparent and traceable.

When purchasing a property and taking out a mortgage, property buyers find the current process risky and labor-intensive. The mortgage process is highly vulnerable to human error and can incur additional costs, as it involves several parties - notaries, real estate agents and financial institutions. With the introduction of blockchain applications for property, the need to rely on paper and individual communication is reduced, costs and human errors in the property acquisition process are reduced. A more streamlined lending process and the elimination of intermediaries will benefit borrowers and financial institutions, who can offer more competitive prices and lower labor costs.

Blockchain's use in the financial sector: This is where blockchain's application is most widespread, and it's no surprise, since the platform itself was originally designed to serve scenarios for creating, acquiring, and transferring cryptocurrencies. Here are some of the many uses of blockchain in this industry:

- Money transfers.
- Added security to transactions.
- Automation through smart contracts.
- Customer data storage.

Money transfers: Since its inception, blockchain technology has been designed to move funds from point A to point B without a central governing body. One example in this area is Ripple, a company that uses blockchain technology for RippleNet, a global payments network. Transactions on RippleNet are processed within five seconds and cost only a fraction of a cent.

Financial institutions that use blockchain technology could offer more efficient money transfers. These international money transfers, which sometimes take hours or days, can now be completed in seconds and without expensive fees.

Added transaction security: Financial companies are always subject to fraud. Digital payments carry the risk of information theft during the transaction process as they pass through payment processors and banks.

Blockchains use cryptographic algorithms to process and record transaction blocks. This cryptography is a way for financial companies to reduce the level of risk when processing transactions.

Automation through smart contracts: Ethereum introduces smart contracts. These are contracts that execute themselves when certain conditions are met. Contracts are a huge part of the financial services industry, and companies spend a significant amount of time on them. A self-executing contract can make this process much more efficient.

Customer data storage: The global financial industry stores trillions of records - from personal customer accounts to accounting records of stock market transactions. A large part of these transactions could be recorded using digital ledgers on a blockchain, which would be immutable and thus prevent fraud.

In addition, the decentralized nature of transactions will give banks better security over the records. Since other parties involved in the transaction also receive a record, disputes over things like missing or incorrect transactions would be a thing of the past.

Here are just a few of the applications of blockchain in business. The data suggests that this is the beginning of a technological revolution that will fundamentally change the way business is done. Gartner predicts that the business value generated by blockchain will grow rapidly, reaching $176 billion by 2025 and $3.1 trillion by 2030.

That is why any research in this area is important and significant, so that the transition to this secure technology can be paved by scientific organizations, mathematicians and specialists in the information industry. They will have to investigate the security and applicability of different blockchain platforms, draw new scenarios and show their effectiveness. Something that is also embedded in this scientific work.

1.10 Choosing a DLT Platform

The first chapter provides an overview of some of the popular blockchain platforms. To select a suitable platform that solves SDM challenges, a detailed comparison of two public blockchains: Ethereum and EOS and two enterprise blockchains: Hyperledger Fabric and R3 Corda will be made.

One of the main criteria when choosing what type of DLT to use is public or enterprise. Having a solution based on a public DLT imposes a set of rules that cannot be changed by the developer. For example, a system based on the public Ethereum blockchain will require the use of immutable smart contracts and different block production times. The choice of DLT platform has a significant impact on the further life cycle of the overall solution. The costs associated with using public DLT and distributed storage also vary significantly. Enterprise DLTs are more flexible to develop – their smart contracts are usually variable – with the ability to upgrade and expand. However, when using enterprise DLT, one must consider the complexity of maintaining the DLT infrastructure itself and choose either a more centralized approach or one where each participant must maintain their own piece of the infrastructure to participate in the distributed process.

Ethereum: Ethereum is the first blockchain platform to support smart contracts. It currently uses a proof-of-stake (PoS) consensus mechanism. The average block creation time is 12 seconds, and the overall throughput is around 15 transactions per second worldwide. Ethereum contains an abstraction layer with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications, where they can create their own arbitrary rules, transaction formats, and state transition functions. The smart contract language is Solidity, a Java script clone designed specifically to run Ethereum smart contracts. A distinctive feature of Ethereum is that its smart contracts are immutable by design. This approach is designed to ensure that all data processing is fixed

price from the start and cannot be changed to anyone's advantage later. There are methods like upgradable proxy pattern that allow logic to be updated.

Both Bitcoin and Ethereum have their own currency, for Ethereum it is ether (ETH). In Ethereum, users have their own accounts, with the number of ethers held by each account recorded on the blockchain. The term gas identifies the fee or pricing value required to successfully conduct a transaction or execute an agreement on the Ethereum blockchain system.

Creating a user account on Ethereum is free, but each transfer of assets and the deployment and execution of a smart contract are charged in ether and gas, respectively. DApp users are charged a gas fee for each call they make. When comparing platforms, the cost of computing resources should be considered.

EOSIO/Antelope/Vaulta: EOSIO is a third-generation blockchain platform based on Delegated Proof of Stake (DPoS). In DPoS, each EOS holder can vote for trusted block producers. A total of 21 producers will be elected, and they will be responsible for processing transactions by hashing them into blocks. Blocks will be produced exactly every 0.5 seconds, and exactly one producer is authorized to produce a block at any given time. The DPoS protocol allows the network to process transactions within seconds and without fees. Fewer nodes are required to verify transactions, so blocks can be produced much more frequently without requiring much energy to operate the network. Transaction processing does, however, consume a small amount of network resources. Instead of paying, transaction initiators must rent the resources (specifically, CPU & NET) they want to use by staking EOS tokens. Resource usage will be cleared every day, and resources that are no longer needed can be returned at any time to get EOS back. This resource leasing model eliminates transaction fees, so it will be free for users to transfer tokens and use all kinds of dApps on EOS.

DPoS is also very stable on a trustless network. The voting process ensures that stakeholders ultimately control block production, because they have the most to lose when the network is not running smoothly. Anyone can participate in block production if they gather enough votes, and block producers will be re-elected every 63 seconds (126 blocks have been produced, which is 6 blocks for each of the 21 producers). A block producer

caught cheating will be quickly removed. Byzantine Fault Tolerance is also being added to the EOS network, requiring 15 producers to sign blocks to make them irreversible.

EOS provides parallel execution of smart contracts through horizontal scalability, which allows the EOS network to process up to 100,000 transactions per second. Horizontal scalability combines multiple hardware and/or software objects into a single object that acts as a single transaction processing unit. Traditionally, most networks have used vertical scalability, which involves adding more computing power to a single object (i.e., adding more CPU or RAM to a single computer) that processes transactions. EOS smart contracts are typically written in C++ compiled to Web Assembly. By default, EOS smart contracts are mutable. Corrections and extensions are relatively easy to make. If a developer decides to make an existing smart contract immutable, there is an option to remove the public keys of their account. This makes the contract virtually immutable, as a private key cannot be used to implement changes. One important feature of EOS is its support for Ricardian contracts. Using this feature of the EOS smart contract can clearly define the purpose and limits of the contract and legally bind its use, both for the provider and the consumer. The pricing model for EOS is very different from Ethereum and is based on the allocation and use of the three main EOS resources - CPU, NET and RAM. Invoking smart contracts and transferring funds on EOS networks is free. Its internal costs in CPU and NET are automatically recovered for each account of its users. Creating accounts requires a minimum amount of resources that must be allocated to ensure access to the EOSIO network resources.

Along with the price advantage, EOS also provides a very good environment for developing decentralized applications.

Hyperledger Fabric: Hyperledger Fabric [50-52] is part of the Hyperledger open-source community. It is provided by IBM and under the auspices of the Linux Foundation, it is designed specifically for enterprise use. It does not have its own cryptocurrency. Fabric smart contracts are called chaincode and are typically developed in Golang. Fabric chaincode is the only way to access or modify data in Hyperledger Fabric. There are concepts of Ledger, State DB and Side DB. For example, accessing the Hyperledger Fabric “ledger” gives access to all transactions that have occurred in the past, regardless of the current state of the included objects (such as “deleted”). The state database is rebuilt for each node based on the Ledger information and contains only the current state of

the objects and does not include objects that have been set as “deleted”. Side DB is a repository for off-chain data that can still be transmitted and validated by Hyperledger Fabric but is not part of its blockchain structures. Technically, each Fabric channel is a separate blockchain with its own chaincode and users. Parties that have access to the channel see all the information that is part of that channel. Additional channels can be used to limit the visibility of certain information. Hyperledger Fabric consensus protocols can be customized using the approval, ordering, and validation phases of adding data to the Fabric. Hyperledger Fabric’s flexibility can be an advantage for complex SDM scenarios. Its ability to add entire organizations as participants and its support for LDAP (Lightweight Directory Access Protocol) allows for easy integration across companies. Since Hyperledger Fabric is a blockchain, each node/user has a copy of and access to all the information that is relevant to it. This makes it suitable for ensuring the consistency and security of the system. It provides a good environment for sharing information between participating parties, as well as a common repository of information that can be used to collect historical data for past SDM procedures. Often, the performance of the system can be significantly improved by properly configuring its smart contracts and policies. Its main advantage is its flexibility and adaptability to different and changing processes. In addition, it is designed to integrate easily with already established corporate systems.

R3 Corda: Corda [53] is a non-blockchain DLT created by the R3 Consortium primarily for use in financial institutions. It has improved scalability compared to any classic blockchain due to its innovative architecture. Unlike Hyperledger Fabric, consensus on Corda is achieved at the transaction level (no blocks) and always involves interaction with one or more nodes of the so-called notary cluster to ensure the uniqueness of the transaction. Corda transactions are by default visible only to the participating parties, and there is the possibility to trace the full history of the funds upon request. Corda smart contracts are typically written in the Kotlin language. By design, Corda smart contracts (as well as Hyperledger Fabric smart contracts) are not guaranteed to be deterministic. Therefore, it is the responsibility of the smart contract developer to avoid writing probabilistic algorithms to avoid complex attacks on the system. For improving SDM processes, Corda can offer the best scalability under load and configurable consensus based on a notary cluster. Its security model regarding data visibility is the most complex, so it is much more difficult to test and analyze. Considering the performance aspect, when properly configured, Corda can be the best choice

for enterprise DLT. Corda can support a complex system with many participants for SDM processes, operating under high load.

Comparison of the DLTs

End users decide to use a given system based on the benefit it brings to them, its usability, initial costs and maintenance costs. Other leading characteristics for choosing a platform are functionality, performance, reliability, security, scalability. Finally, the size of the ecosystem that uses the platform is also important, whether it is constantly improved, whether many new functionalities are written, and whether the environment for writing new applications is easy and intuitive to work with.

The comparison between the four DLTs highlights the differences that already exist between the different technologies. The use of corporate DLTs (HL Fabric and R3 Corda) requires significant infrastructure costs to be created and maintained by the participating parties. This includes the allocation of resources for both hardware and software, as well as personnel responsible for system updates and configuration changes. Certain service qualities such as high availability and disaster recovery, following best security practices, etc. must be planned and implemented by all participants. For an organizational distributed system, these costs can be significant if they are implemented in a way that does not rely on trust in other participants in the process. As an additional factor, it should be considered that if there is already a good level of trust between all parties, the overall architecture can be significantly simplified by using classic databases instead of DLT.

The use of public DLTs (Ethereum and EOSIO) on the other hand guarantees trust between all participants because it is based on a public p2p network. The infrastructure of public DLTs is already created and maintained by other parties, which gives a cost advantage and easier maintenance. DApps running on public DLTs can be consumed directly even from mobile devices. These properties of public DLTs give them a decisive advantage over corporate DLTs when it comes to servicing a distributed SDM system.

The overall costs of managing SDM dApps on Ethereum and EOS are based on quite different models. If all the costs for each participant in the procedure are analyzed, it is evident that using Ethereum necessarily adds daily costs, which is why users continue to pay for using the dApp, since each step of the implementation and especially of using the dApp

costs a certain amount of gas. On the other hand, the daily costs of using EOSIO dApps are mostly recoverable. The CPU and NET used to make calls to EOS smart contracts are recovered over time and even the committed resources can be reduced if they are no longer fully used. The cost of RAM is paid for the initial implementation of the dApp and is used to store all the data collected by the dApp and its user. The EOS pricing model is more flexible compared to Ethereum and allows for different cost models and optimizations. As an additional benefit, the time to produce a block is significantly shorter, allowing for faster responsive applications. An additional advantage for starting prototypes is the ease of making any modifications to already implemented smart contracts, as well as the high speed at which blocks are generated.

The listed similarities and differences, as well as key features, are summarized in Table 1.

	Ethereum	EOS (Mainnet)	HL Fabric	R3 Corda
Type of DLT	Public	Public	Permisioned	Permisioned
Consensus Type	Proof of Work	Delegated Proof of Stake	Configurable per channel	Notary service + UTXO
Smart Contract Type	Immutable	Mutable/Immutable/BP	Mutable	Mutable
Smart Contract Programming Language	Solidity	C++	Golang	Kotlin
Legal Binding Agreement	- - -	Ricardian Contracts	- - -	Ricardian Contracts
Deployment Cost for dApp Publisher/Operator	Gas price for deployment	Cost: RAM (code + data)	(Distributed) Infrastructure	(Distributed) Infrastructure
Maintenance Cost for dApp Publisher/Operator	- - -	Cost: RAM delta (+/-)	Maintain (Distributed) Infrastructure	Maintain (Distributed) Infrastructure
Motivation to run DLT	Miner's reward per block (inflation)	Block producers' reward (inflation)	Infrastructure maintained by involved parties	Infrastructure maintained by involved parties
End user joining	Free	~ Free minimum RAM, CPU, NET	Restricted	Restricted
End user costs	Gas cost per dApp call	Practically free / Replenishable resources	- - -	- - -
On-chain data visibility	All persistent dApp data is visible globally	All persistent dApp data is visible globally	Full visibility per Fabric channel	Only participants see transactions. Transaction proof may be shared.
Scalability	Limited due to PoW used	Improved due to DPoS and limited number of BPs	Each channel is separate blockchain	Excelent, Native sharding support
Consensuis	Proof of Work	Delegated Proof of Stake	Configurable per channel	Verify: Required signers Unique: Notary service
Security	Top 3 Mining pools control 64% of hashrate	Top 21 Block Producers votable each 63 sec.	Customizable per scenario	Notary service
Transactions per Second	About 15 transactions per second globally	Limited to 4000 TPS for Mainnet	Configurable per channel	N/A
Block producing time	~ 15 seconds	0.5 seconds	Configurable per channel	N/A

Table 1 Comparison between selected DLTs

Ethereum has now officially switched to a Proof of Stake (PoS) consensus mechanism in 2022 as secure and energy-efficient way to validate transactions and add new blocks to the blockchain. Based on the analysis, considering all the important features, EOSIO is a more suitable platform choice for the development of distributed application prototypes. Along with the speed, the trust coming from its public type, as well as the popular language (C++) used to write smart contracts, there is another leading factor for choosing EOSIO/Antelope/Vaulta, namely the convenient environment for writing dApp applications.

1.11 Key features of EOSIO/Antelope/Vaulta

The EOSIO blockchain platform is a next-generation open-source platform with industry-leading transaction speed and flexibility of use. As a blockchain platform, EOSIO is designed for typical enterprise use cases and is built for both public and private blockchain deployments. EOSIO can be customized to meet a wide range of business needs across industries, such as role-based access provisioning and secure transaction processing by applications.

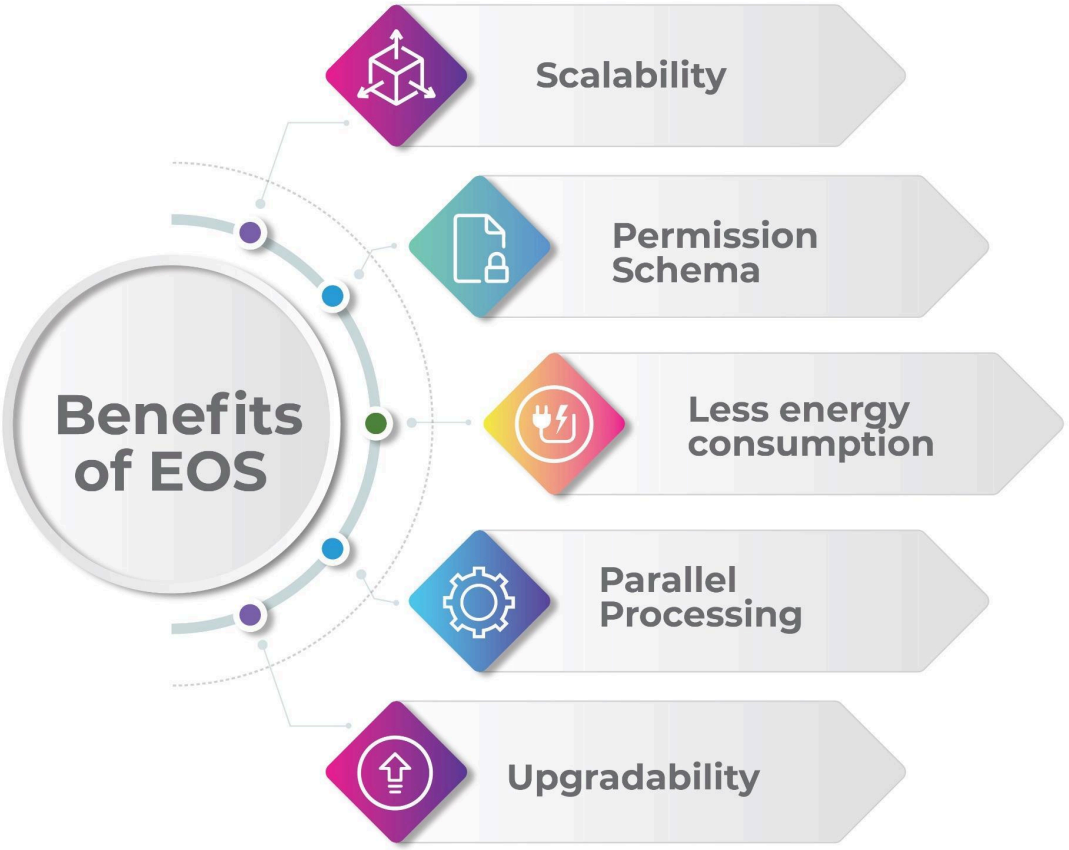

Figure 13 Advantages of the EOSIO platform

Building distributed applications on EOSIO follows familiar development models and programming languages that are also used outside of blockchain environments and the EOSIO ecosystem. For application developers, familiarity with the development environment and technologies leads to a seamless and rapid start of work on new software on the platform, whether it is smart contracts, entire applications, or new EOSIO extensions. The following points discuss the main development tools for components and communication protocols provided by EOSIO.

1.11.1 Platform and tools

The EOSIO platform provides several components and libraries that are used to work with blockchain nodes, collect blockchain data, interact between nodes in the network, and build smart contracts. The main component is **nodeos**. This is the main server process

running on each node in an EOSIO network. It can be configured to start together with the operating system. This component takes care of producing new blocks, providing interfaces for communication with the blockchain (APIs), controlling the writing of data to the blocks, etc. Cleos, on the other hand, is a command-line tool that interacts with **nodeos** locally on the same computer or remotely. **Cleos** provides the full set of functionalities used to manage the blockchain, allowing to send commands, read data, generate keys, install new smart contracts, and configure them. **Cleos** also interacts with **keosd**, a local component that stores EOSIO keys securely.

The C++ language is used to create smart contracts. However, this would be a very difficult task without the help of libraries that can be reused by programmers. For this purpose, EOSIO.CDT was created, which contains a set of tools and libraries that allow programmers to focus on the functionality being developed and abstract away from the specifics of the platform. The code is also compiled with a special compiler, part of EOSIO.CDT, which generates bytecode in **wasm** files. These files are in fact smart contracts that can be installed directly on the blockchain.

The basic relationship between these components is illustrated in the following diagram:

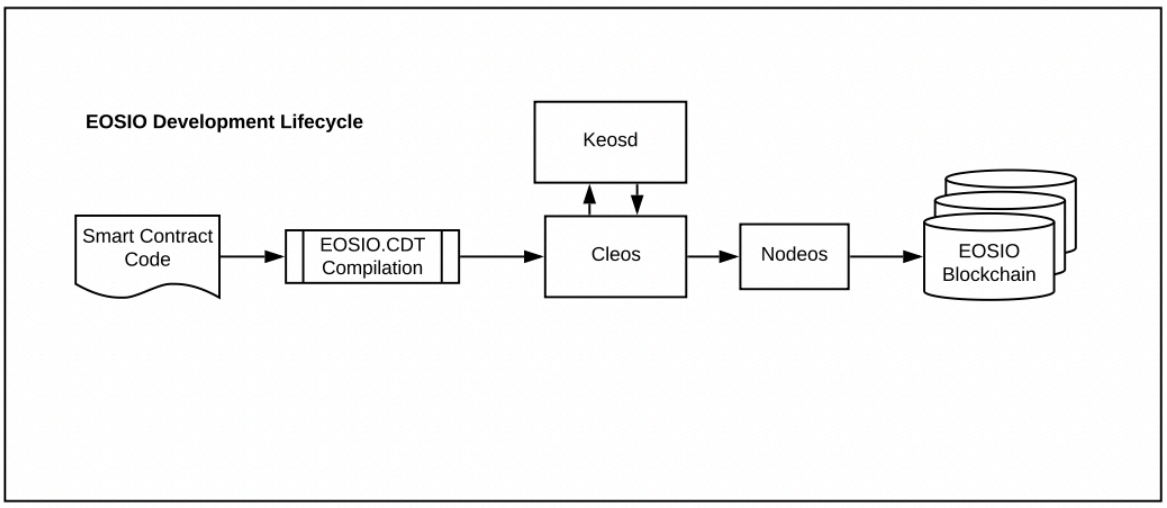

Figure 14 EOSIO components and tools

Nodeos

Nodeos is the main operating system process in each EOSIO node. **Nodeos** takes care of storing blockchain data, connecting all nodes in the network, taking care of data replication, and producing and validating new blocks. For the purposes of building development environments, **nodeos** allows a single node blockchain network to be set up. **Nodeos** offers a wide range of features through plugins that can be enabled or disabled at the initial startup time via command line parameters or configuration files.

Cleos

Cleos is a command line tool that interacts with the REST APIs provided by **nodeos**. **Cleos** can be used to deploy and test smart contracts on EOSIO. This is sufficient for communication with the EOSIO blockchain, both from the user and developer side. Any **cleos** command line tool is also often used for process automation through various scripts.

Keosd

Keosd is a tool for managing and storing private keys and signing digital messages. **Keosd** provides a secure medium for storing keys that will be encrypted in the associated wallet file. **Keosd** also defines a secure enclave for signing a transaction created by **cleos** or an application integrated into the platform.

EOSIO.CDT

EOSIO.CDT is a set of tools to facilitate writing contracts for the EOSIO platform. In addition to a set of general-purpose WebAssembly tools, EOSIO-specific optimizations are also available to aid in building EOSIO smart contracts.

1.11.2 Basic concepts

To better understand how EOSIO works, it is necessary to understand some of the basic concepts of its architecture. The main concepts are accounts, wallets, authentication and authorizations, smart contracts, and delegated proof of stake (DPoS) as a consensus mechanism.

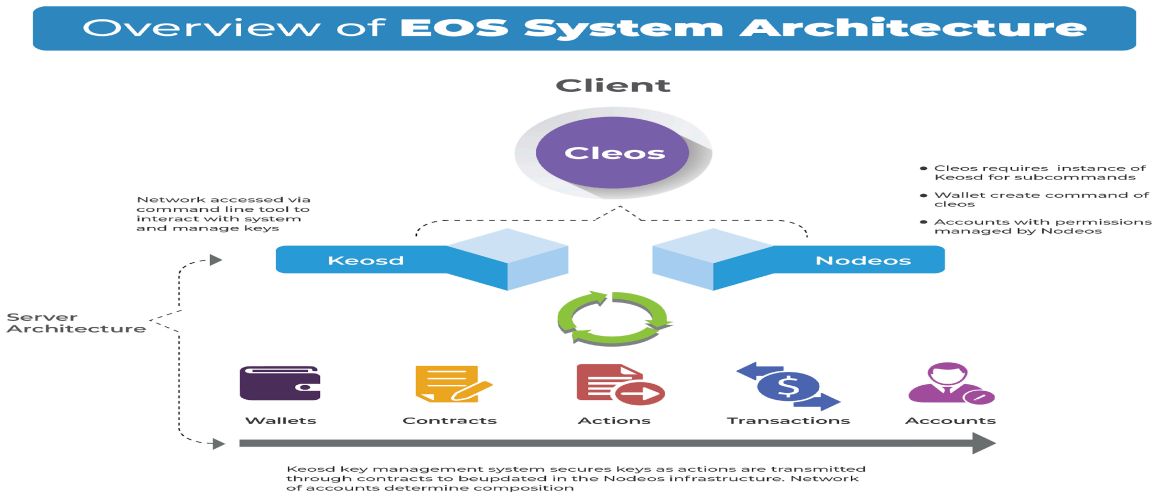

Figure 15 EOSIO Architecture

Accounts

The account has a name that is stored in the blockchain. It can be owned by an individual or a group of individuals through authorization, depending on the access configuration. An account is required to transfer or send a valid transaction to the blockchain. Also, a basic EOSIO concept is that each smart contract must be installed in a separate account. The reason for this is to control the resources used in the blockchain.

Wallets

Wallets are client-side software that stores keys. Keys can be associated with access rules for one or more accounts. Ideally, the wallet has a locked (encrypted) and unlocked (decrypted) state, which is protected by a high-entropy password. The combination of the previously described **cleos** and **keosd** implements this model. It is also possible to use other wallets, because they take care of the storage of a public and private key pair. The platform itself needs to provide this key pair and does not depend on where they are stored.

Authentication and authorization

The platform provides a comprehensive system for providing access rights, which allows for different configurations to suit the respective scenarios. For example, access rights can be

created and configured for a specific functionality of a specific smart contract. These access rights can be distributed and divided between different accounts with different access levels. The corresponding access levels and roles can be integrated into the end applications.

Smart Contracts

A smart contract is a piece of code that can be executed on a blockchain and preserve the state of the contract's execution as part of the immutable history of that blockchain. Developers can therefore rely on that blockchain as a trusted computing environment where the inputs, execution, and outputs of a smart contract are independent and free from external influence. The underlying WebAssembly C++ programming language provides a powerful environment for developing and authoring smart contracts and effectively interacting with the EOSIO ecosystem.

Oracles

The purposeful isolation of the blockchain from external factors and the lack of mechanisms to react to external influences leads to the need for modules external to the system that take care of providing real-world data to the smart contracts in EOSIO. As an example, an oracle can provide up-to-date information about the current price of airline tickets to other smart contracts. In this way, this information can be used to purchase tickets from different participants in the blockchain.

Delegated Proof of Stake, DPoS

The EOSIO platform implements a proven decentralized consensus algorithm capable of meeting the performance requirements of blockchain applications, called Delegated Proof of Stake (DPOS). The most popular alternatives to DPoS are the Proof of Work (PoW) mechanism used in Bitcoin and known for its energy-intensive operation, as well as the classic Proof of Stake (PoS) mechanism used by several blockchains such as Ethereum. Under the DPoS algorithm, users holding tokens on an EOSIO-based blockchain can elect block producers through a continuous approval voting system. Anyone can choose to participate in block production and will be given the opportunity to produce blocks, if they can convince token holders to vote for them.

1.11.3 System resources

The EOSIO blockchain defines three types of resources to compensate for network usage. To perform the operations associated with an account, it must have sufficient resources of its own or provided by another account:

Memory (RAM) – sets the limit for data storage

Processor (CPU) – limits the computing time

Network (NET) – limits the network usage

Memory (RAM)

RAM, in an EOSIO-based blockchain, is one of the important system resources consumed by blockchain accounts and smart contracts. RAM acts as persistent storage and is used to store account names, access definitions, token balances, and information for quick access to on-chain data. RAM must be purchased and is not based on staking, as it is a finite, persistent resource.

Processor (CPU)

CPU represents the processing time of an action and is measured in microseconds (μs). CPU is referred to as CPU bandwidth in the output of the **cleos** get account command and indicates the amount of processing time that the account has at its disposal when directing actions to a contract. CPU is a transient system resource and falls under the EOSIO staking mechanism.

Network (NET)

Besides CPU and RAM, NET is also a very important resource in EOSIO-based blockchains. NET is the network bandwidth measured in transaction bytes and is referred to

as net bandwidth in the **cleos** get account command. NET is also a transient system resource and falls under the EOSIO staking mechanism.

1.11.4 Technical characteristics

WebAssembly C++ compilation

EOSIO primarily uses C++ as the programming language for smart contracts. C++ developers do not need to learn a new programming language to understand how to create smart contracts and how EOSIO supports their development through C++ classes and structures. Any C++ developer with current knowledge can get involved in EOSIO development quickly and start programming smart contracts using EOSIO specific canonical C++ code constructs and smart contract APIs.

The EOS VM virtual machine runs smart contract code and is designed for the high demands of blockchain applications. The choice of compiling to **Wasm** allows EOSIO to use an optimized and tested toolkit that is maintained and improved by a broad community. In addition, adopting the **Wasm** standard also makes it easier for compiler developers to port other programming languages to the EOSIO platform.

High Throughput, Faster Confirmations, and Lower Latency

The Delegated Proof of Stake (DPoS) consensus mechanism achieves high transaction throughput because DPoS does not have to wait for all nodes to complete the transaction to achieve finality. This behavior results in faster confirmations and lower latency.

The EOS VM mentioned earlier, and all its features and enhancements add significant contributions to the overall performance of EOSIO-based systems.

Programmable Economics and Management

The resource allocation and management mechanism of all EOSIO-based blockchains is programmed through smart contracts. The system's smart contracts can be modified to customize the resource allocation model and governance rules of the EOSIO blockchain.

Staking Mechanism

In EOSIO-based blockchains, access to system resources is governed by a process called the staking mechanism. The system resources that fall under the scope of the staking mechanism are CPU and NET. One can directly interact with the blockchain via **cleos**, RPC API, or an application to access CPU and NET by staking system tokens.

When staking CPU and NET tokens, one gains access to system resources proportional to the total amount of tokens staked by all other users for the same resource at the same time. This means that transactions can be made for free, but within the limits of the staked tokens.

One can also allocate system resources to EOSIO accounts programmatically by customizing the resource allocation model of a smart contract in the system.

Business model flexibility

Applications built on EOSIO can adopt a freemium model, where application users do not have to pay for the cost of resources required to execute transactions.

Freemium transactions are facilitated by applications co-signing transactions with the user. Alternatively, applications can also stake enough system tokens to guarantee the necessary resources.

Upgrade Options

Applications implemented on EOSIO-based blockchains are upgradeable. This means that code can be modified, features can be added, and application logic can be changed if the necessary permissions are granted. It is also possible to implement smart contracts that cannot be modified on EOSIO-based blockchains. These decisions are at the discretion of the developers, not limited by the protocol.

Efficient energy consumption

With DPoS as its consensus mechanism, EOSIO consumes much less energy to validate transactions compared to other popular consensus algorithms.

Chapter 2. Research Incentivization based on Smart Contract Platform

2.1. Introduction

An explosion of new ideas and technologies followed the success of Bitcoin network as decentralized financial systems. It proved that a stable system can be build and operated there was how distributed systems could enhance and replace existing technologies. Ethereum showed that smart contracts add to blockchains abilities that are make it flexible and programmable which led to a bunch of new decentralized applications (dApps) based on a set of related smart contracts running on the public blockchain. It took some more years and the third generation of public blockchains got traction adding sophisticated new methods for reaching consensus, reducing the transaction costs and improving network accessibility. The expected explosion of dApps did not happen as fast as expected. Public blockchains got swarmed by various games, gambling and decentralized exchange apps. Being a new technology public blockchains took major share of the technological research efforts. This brought many technologies in the spotlight such as zero knowledge proofs, advanced security protocols, analysis of consensus algorithms, and various distributed ledger technologies based on directed acyclic graph instead of blockchain.

Apart for the indirect boost to various science branches and technologies public blockchains have the unique ability to offer medium that is visible, immutable and decentralized. All these properties are desirable by most science research activities. Can we build a system that could help different scientific entities to better define the problems that they put their efforts? A system that is visible so even an external or anonymous participant could contribute to solution of a specific tasks by automatically incentivizing achievements of researchers.

In this chapter we will describe a decentralized system based on public blockchain for incentivizing measurable and provable achievements in various research fields.

Let's think about one practical example that is in the area of medical research: The protein folding is a physical process by which a protein chain acquires its native 3-dimentional structure. The primary structure of a protein as a linear amino-acid sequence fully describes the protein's content. The secondary protein structure is the first step from the folding process for the protein to finally fold into its native structure. Secondary structure consists of sets of alpha helixes, beta sheets and connections between them. Their final single-protein structure is reached after tertiary structure get the secondary structures fixed in the 3-dimensional

space. The quaternary structure describes formations of already-folded proteins assembled into bigger structures. Getting from the deterministic primary structure to the final 3-dimational protein structure takes from microseconds and minimizes the structures Gibb's free energy. Since the full information for the protein is a string of amino acids finding the native protein structure could be defined as energy-optimization problem (Fig.16). When we have a stable low-energy solution that do not fit what we expect and do not match the structure observed in practice we could assume that the native protein structure is different.

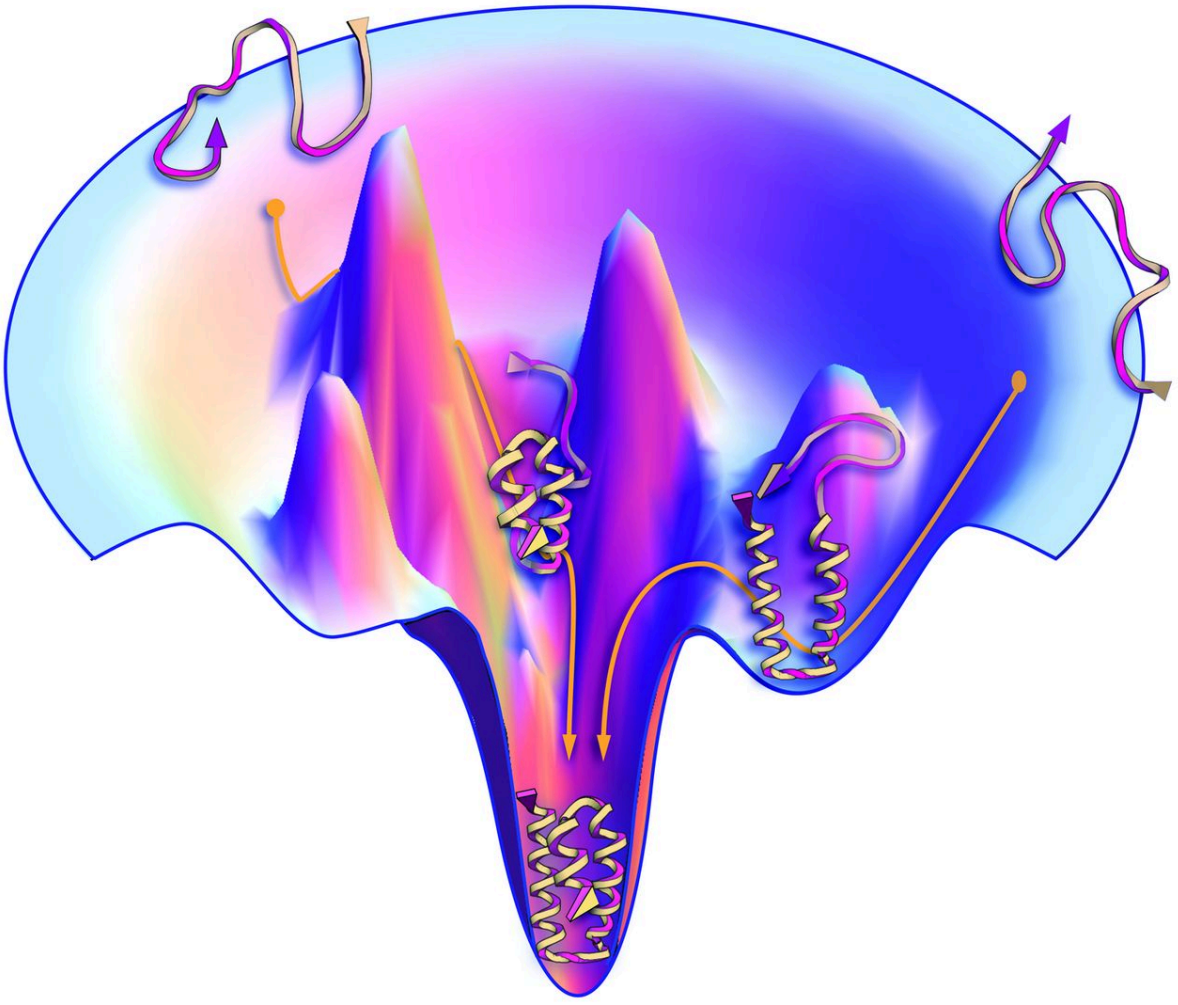

Figure 16 Optimizing Gibbs energy during protein folding

A task to search for a matching structure that minimize protein's Gibbs energy could be an important step in the research of various protopathic diseases such as Alzheimer's disease, Parkinson's disease, amyloidosis and many others. Although this computational challenge is easy to state, it is hard to solve, and the effectiveness of proposed solutions are easy to compare based on their calculated Gibb's energy values.

2.2. Main Parties Involved in the Incentivization Process

Understanding the roles of the participants and their interactions with the system is the first step for analyzing such system. Due to the specifics of the systems not all participants need to actively communicate to the system. It is expected that for some participants will threat the system as read-only source of information and other may interact only indirectly. Based on their functions several groups of people are involved in incentivization for research topics.

Framework Providers are typically organizations or individual developers that create and maintain the distributed coding for the supported frameworks in the smart contract platform. Their responsibility is also to maintain them by providing improvements and fixes when needed. For example, a group could offer a module to the system for calculating Gibb's energy for a specific 3-dimentional structure of a protein. Using this module, the system could automatically compare two or more different folding solution for a single protein and calculate that some of the proposed solutions are not the natural one due to their higher energy state. The provided coding makes sure no one could get the reward except the one that supply the best solution in the given timeframe. Thinking about architecture - there could be many different frameworks supported by the system. Different frameworks do not compete with each other but enhance their scopes and, in some cases, they could even share reward pools. And, since they all share the same smart contract platform based on public blockchain, they indirectly improve each-others security and immutability.

Reward Providers are the next important participants. Unlike *Framework providers*, the *Reward providers* does not have to be experts in the software or blockchain development. They need only to have scientific domain knowledge for the researched topic. For example, researchers investigate specific protein that is involved in specific prion disease. Knowing the protein's 3-dimensional shape is essential for determining its function and properties. Setting a reward for providing the best energy-efficient folded structure matching the protein's sequence of amino acids may lead to finding vital information for researching methods for treatment of the illness. Some initial monetary (in public blockchain sense) reward will be set for the entity that provides proper solution to the defined problem. Having the computational problem set and initial reward structure supplied to the system is all the *Reward provider* needs to do to have the problem defined and open for accepting possible solutions.

Reward Contributors are optional participants in the incentivization, but they may have a key role to focus attention to specific research problems. They could be either organizations

or individuals that want specific problem(s) to find a solution. *Reward contributors* are not required to have domain or technical knowledge to contribute to the process. In our example *Reward contributors* could be organizations investing in finding cures for the specific prion decease. Their role is to contribute financially to the rewards for finding solution for one or more problems they choose.

Researchers are again organizations or individuals that work for finding solutions to the defined and modelled in the system computational problems. Once they have a fitting solution, they submit their proposed solution to the system. After validation, depending on the configurations, the reward could be either transferred immediately or at a previously defined moment in time to allow further (and possibly better fitting) submissions. The reward in this case is the accumulated currency rewards up to the moment the price is transferred, and the problem's state is considered solved according to the configured boundaries set by the *Reward provider*.

Read-only / Off-chain Participants are essential part of the proposed system. Sharing the information for the frameworks, research problems, and available reward in the media is important part of the overall process. The category of *Off-chain Participants* also includes researchers that compete for the reward or collaborate off-chain. The system is used as a medium for sharing problems and incentivizing research. In many cases this may happen, and it should happen with no on-chain traces.

In the perfect case all participants need to participate accordingly for the system to contribute for the successful scientific research. At the end of each full process and behind every reward given there could be significant research breakthrough.

2.3. Classification of Incentivization Tasks

Some of the task types are given in Table 2. The tasks that are set for the system may vary by their defined duration – they may either be set in the system for indefinite interval or could have a predefined duration. The indefinite time tasks assign their full reward the first time a solution matching defined parameters is supplied to the system. The limited time tasks are more flexible. They may assign the reward to the first fitting solution, or they may wait for all submissions to accumulate and reward only the best one after the end of the period of competition.

ID	Task Duration	Private results	Allow Contributors	Allow Cancelation	Typical tasks	Examples
1	Infinite	No	Yes	No*	Theoretical	Counterexample for a hypothesis
2	Infinite	No	Yes	Yes	Research	Protein folding (improvement)
3	Fixed	No	Yes	Yes	Research	Protein folding (initial), Space travel (public)
4	Fixed	No	Yes	No	Practical	Divisor of a big number (public)
5	Fixed	Yes	No	No*	Practical	Divisor of a big number (secret), Space travel

Table 2 Responsibilities and SLA

In some cases, sharing the results for the winning solution may not be desirable due to security or financial reasons. For example, if a strength of a security key is tested by rewarding possible successful attacks it may not be desired. Also, a complex task such as a space travel plan may lead to negative financial consequences if specific parameters (launch dates, supplies) become public too early. For this kind of tasks solution submission should be possible in such form that only the *Reward Provider* could read the solution. This approach will require additional handshake between supplier of the solution and the Reward Provider. One effect of this complication is that these scenarios should not allow additional reward contributions as any *Reward Contributor* will not have access to the winning solution due to its private nature.

One very important aspect of the task lifecycle is the situation when no reward is won. This happens when a task is either cancelled or it reaches the end of its active period without valid solution to be submitted. In this case the accumulated reward must be returned to the *Reward Provider* and all *Reward Contributors*.

2.4. Architecture of the system

There are several basic elements of the proposed system (Fig. 17) that will make it run. For the prototype we chose EOSIO as a public blockchain technology. Some of the reasons for choosing this third generation blockchain are its flexibility for development, efficient programming language C++, support for Ricardian contracts, use of replenishable/free

resources such as CPU and NET for operations and very short block producing time - 0.5 seconds and block confirmation times - 99.9% in 1.5 seconds, 100% confirmation in 3 minutes. The choice of public blockchain such as Ethereum and EOSIO technology [3] brings additional benefit that participants does not have to invest heavy in own infrastructure as block producers are paid by block producing rewards or blockchain inflation. For comparison a blockchain network based on Hyperledger Fabric or Corda would require most or all key participants to maintain their own hardware resources raising the costs for operation.

User Management is a key module that keeps track of blockchain accounts that represent on-chain participants and their relations with supported *Task frameworks*, *Tasks*, *Rewards* and *Solution submissions*. In many cases there are option an account that is not yet known to the system to execute operations. For example, this could be an anonymous *Task Contributor* or *Researcher* which first interaction with the system is claiming a reward for finding solution of one of the defined problems. In these cases, no user registration is required for the system to be used but the blockchain account will be added to the *User Management* data after the interaction, being it successful or not. On-chain collaboration is also an option for several individuals or organizations to set clear reward boundaries by defining how rewards will be split on success. On-chain collaboration is an entirely optional feature which intention is mostly for sharing the visibility of participants for the achievement. If only monetary reward is concerned this could be achieved entirely off-chain as long as the partners trust each other for doing so.

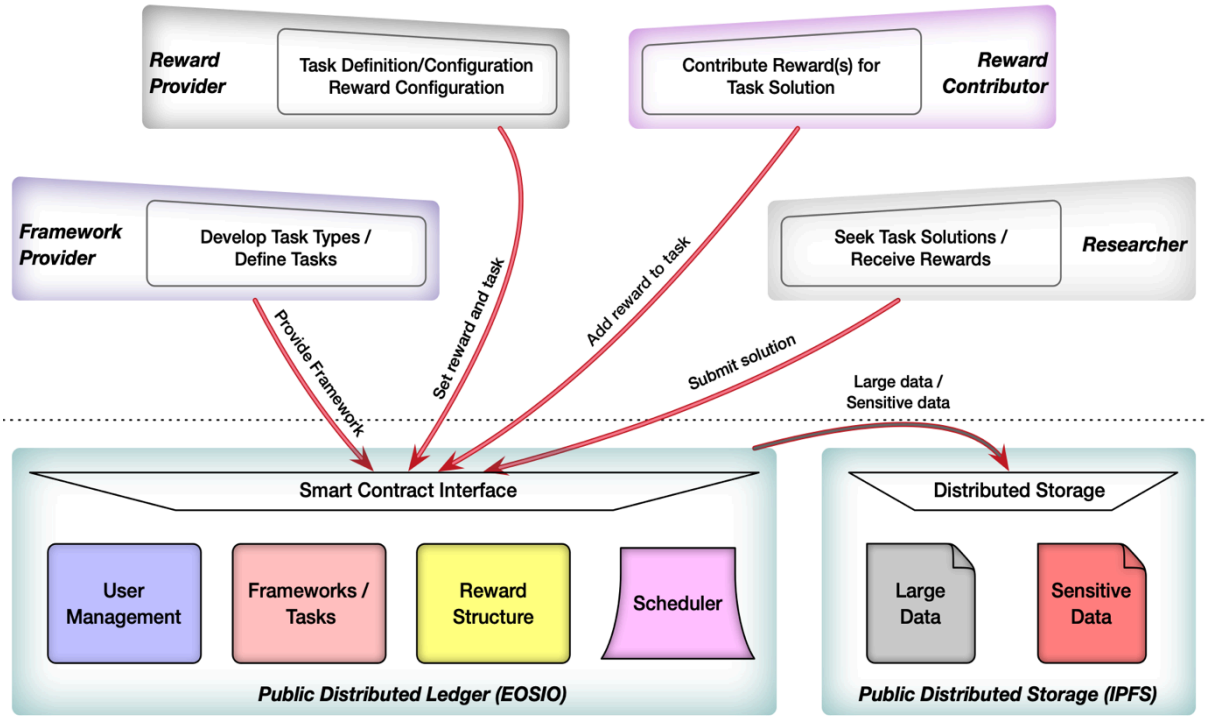

Figure 17 Main actors and interactions for the system

Task Frameworks are a set public blockchain components that are open for contribution by developers.

The main parts for contribution are the *Validation components* and *Task interfaces*.

Validation components are supplied and maintained by *Framework Providers*. It is important that they are stable and error-free contracts as they may be used for long time periods. Any changes to their interface must be backward compatible to a large degree. It is strongly advisable that supported task frameworks are open source and could be validated by anyone involved.

Task interfaces describe the technical parameters set by reward providers, as well as the format that the solutions are received. The period for which the reward for finding a solution is offered is a key parameter for such task. Another parameter is the type of competition. In some cases, the first validated solution may immediately receive the reward. In other cases, the problem may have many solutions, and the first submitted solution may later be improved by other submissions. In this case the rewards are given to the best solution submitted in the configured period, if any.

Reward structure is a set of data that keeps track of the rewards offered for solutions of specific calculation problems and additional configuration data. A single reward

may be as simple as one-to-one relationship between the token amount offered for problem's solution. They could also have complicated structure where one token amount may be assigned for a breakthrough in a variety of problems. Also, for each problems a variety of contributors may assign differently configured rewards. The native limitation is that the period for active secondary rewards (set by *Task contributors* that are not the original *Task provider*) should not extent the period of original task reward.

Reward type is a key parameter for the system that need to be considered.

- Each major public blockchain offers tokens as their **default cryptocurrency**. It is natural that rewards are offered and collected in blockchain's native currency. The main advantage for this approach is the simplicity for implementation and overall security.
- The flexibility of modern public blockchains allows each dApp to offer and operate its **own token**. A solution with own currency attached may be used in addition to finance building and operating of such system. The drawback of such approach is the increased complexity of using the system by its users and additional fluctuations in its value.
- Another approach is to set reward in a specific **stablecoin** is on the other side of the scale. Its rewards are independent of the fluctuations of the blockchain currency. Rewards set in stablecoin may be preferred due to the reduced risks of currency volatility.
- The most complex approach is not to fix a single reward token and allow **several tokens** to be valid rewards. As this approach is most flexible and customizable it opens the system to new types of threads if no control is enforced on the allowed tokens for rewards such as custom malicious coding executed as part of normal token operations.

Reward target could be defined as flexible as the system allows. Each reward amount could be set for solving anyone of a predefined set of tasks. In their most flexible implementation, a single reward can be set for providing solution for problems defined in different task frameworks.

Scheduler component makes sure that rewards can be received after a predefined period. Its responsibility is to calculate and compare the submissions at the end of the configured period. In the specific case that there are no valid solutions submitted the rewards need to be returned to the *Task provider* and the *Task contributors*.

These are the key components for functioning of the system that are addressed in the EOSIO-based prototype of the system.

2.5. Additional factor to be considered

The proposed incentivization system differs from most "standard" dApps running on a public blockchain as the reward may be claimed with no restrictions. The monetary reward is "protected" by finding a solution to a puzzle that is clearly and openly defined. This difference makes normal public blockchain interactions vulnerable to specific attacks.

Denial-of-Service (DoS) type attacks could limit system availability and potentially prevent researchers from receiving their properly earned rewards. The natural solution for protection from DoS attacks is making sure all key resources for using the distributed system are provided by the participant that use the system functionalities. For example, these are the CPU, NET and RAM used for running an EOSIO-based system. Special case here is when the system tracks rewards in several tokens. Without a maintained whitelist for allowed tokens in reward structure the system may be flooded with rewards in custom tokens that could be deprecated at any time. Setting pre-configured period for "receiving" the token rewards (claiming its RAM price) will protect the system's resources from being overwhelmed by unclaimed rewards.

Security/Protection of data supplied. Interaction with public blockchains is typically done by sending signed message to a chosen blockchain endpoint. If one call is a solution to a problem, the submission theoretically could be intercepted at the endpoint, data from submission to be extracted and a new submission to be sent to blockchain, claiming the reward. Such man-in-the-middle attacks could be avoided by establishing a two-phase submission process. Initial call contains encrypted solution that set the time for submission. Once the system is set in "solution expected mode" the encryption key may be sent in separate calls potentially to several endpoints during the locked period for reward transfers. Timeout and costs for "locking" need to be chosen carefully to avoid specific DoS attacks.

Private solutions. Sometimes the solution for which rewards are offered may not be public by nature. To address this case special handshake procedures, must be established, based on zero-knowledge-proof, to make sure the key solution details are accessible only to the party/account that provides the rewards. Naturally, when solution's privacy is a requirement, the only *Reward Contributor* is the original *Reward Provider* as they are the only ones that may benefit from the solution.

Reward Cancellation/Expiration. Under some circumstances a reward(s) for specific problem can be canceled or its period could expire with no valid submission. Keeping information about who provided the rewards is essential in case the reward currency must be returned to *Reward Provider* and *Reward Contributors*. A notice period for reward redraw should be in place to make the researchers know in advance that part or all the rewards are canceled. The cancelation policy should be part of the initial task configuration to limit the possibilities for abusing the system. To avoid system abuse, certain small fees could be charged for removing parts or all the rewards for solving a problem before the deadline is reached. This should not be possible for the original *Reward Provider* as various *Researchers* may have already invested significant efforts in finding a solution and possibly some *Reward Contributors* have already invested resources to incentivize the problem solution.

2.6. Benefits and risks for using DLT-based Incentivization System

In this chapter we described and analyzed a distributed system based on public blockchain that can be used to incentivize research breakthroughs and collaboration. The proposed distributed system would allow many computational problems to be publicly shared. The rewards are set for every potentially anonymous participant that succeeds in providing their proper solutions. Some of the advantages of using the distributed application are its flexibility and transparency, by-design immutability, no-admin operations, as well as its full traceability of both problem definition, reward structure and solution proposed. The developed EOSIO technology prototype covers the key roles and interactions for the proposed systems. In the paper was shown that even the simplest incentivization scenarios must be built carefully to avoid many potential attacks specific for the third generation public blockchains.

Chapter 3. DLT-based system for managing SDM processes

Chapter 1 described how blockchain, with its features such as security, transparency and reliability, can transform traditional business and offer a qualitatively new architecture and simplified processes for the end user. And while many developers are currently focused on transforming their current business, there is one IT area that could also benefit from blockchain, namely software development life cycle management. In SDLC, most of the challenges occur in the phase of implementation, configuration, updating and maintenance of the software at the end customer (System Deployment and Maintenance - SDM). It is in this phase that the use of blockchain would solve the typical problems accompanying the initial installation and configuration of the software in a productive environment.

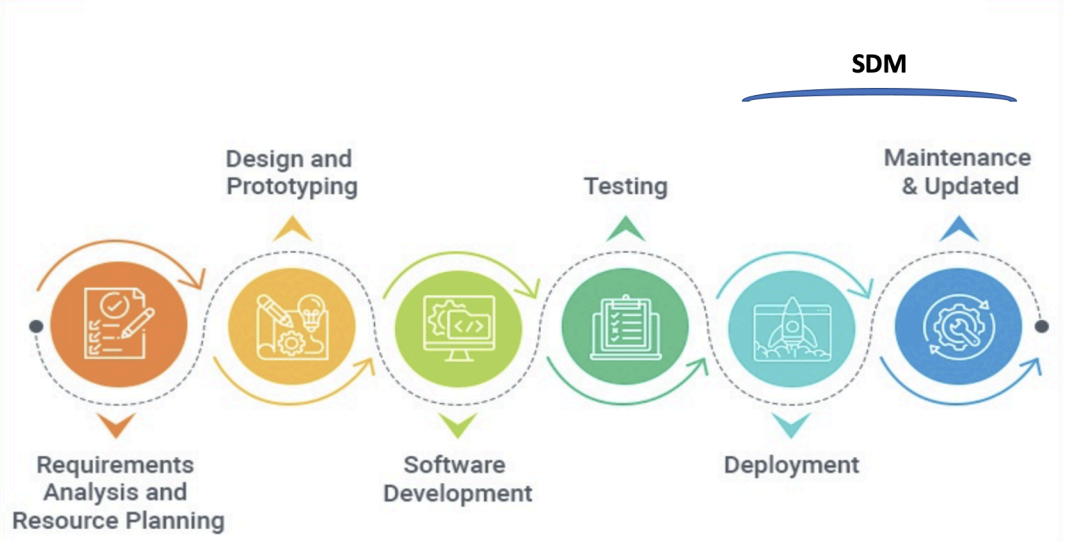

Figure 18 Definition of SDM

3.1 Introduction to SDM

SDM covers the stages of software development related to installing, configuring, updating and maintaining complex software systems. Large productive IT systems very often consist of multiple servers, computing power in the cloud environment and edge sensors and devices (Internet of Things - IoT). Software deployed on such a heterogeneous environment has complex dependencies, with each product with a given version assumed to function reliably and be able to communicate with another product with a specific version of the same

communication environment. Quality requirements, such as high service availability and disaster recovery, are mandatory, this adds more servers and devices to the productive environment, thus increasing the complexity of the overall solution. Accordingly, the procedures used to install, configure and maintain products in this heterogeneous environment become many times more complex.

Another factor that contributes to the complexity of SDM procedures are the many participants who are involved in the implementation of a productive system, and each participant has different rights and obligations. Relevant SDM information about the system properties is often limited to only some of the participants, sometimes even missing (intentionally or not), and in many cases there is wrong information because of manipulations by different parties. The interconnection between the participants, the responsibility of each of them, the level of confidentiality that should be ensured in communications is in some cases difficult to define and track.

The presence of records/history of SDM procedures applied to the production environment in each sequence could help for each subsequent production installation. Unfortunately, such a history is very often missing. Moreover, after installation or configuration, a large part of the logs is deleted. In this way, it becomes quite difficult to make an accurate prediction about the time it will take to implement a given product, as well as how long a given production system will be unavailable to end customers.

Finally, there is the issue of reliable software support and specific responsibilities - which participant is responsible for what type of support and for what period this commitment is undertaken. Due to the ever-changing nature of software systems, it is very common for a system to be updated/changed several times a year. Some systems have a lifespan of over 20 years. The participants who have maintained the system have changed many times during this period. The hardware that was used at the beginning has also most likely been replaced several times. The sequence of which participant should do what, by when and after whom is also sometimes difficult to establish. And although the interest of each participant in the SDM process is for the project to be successful, sometimes serious delays are observed. That is why clear responsibilities, sharing plans, data and commitments is an important part of the project's success.

3.2 Typical SDM scenarios

This section examines an IT scenario that is close to a real business system. It consists of various software products running on local servers, in a cloud environment, and integrated IoT devices. The typical actors responsible for implementing such a complex system are:

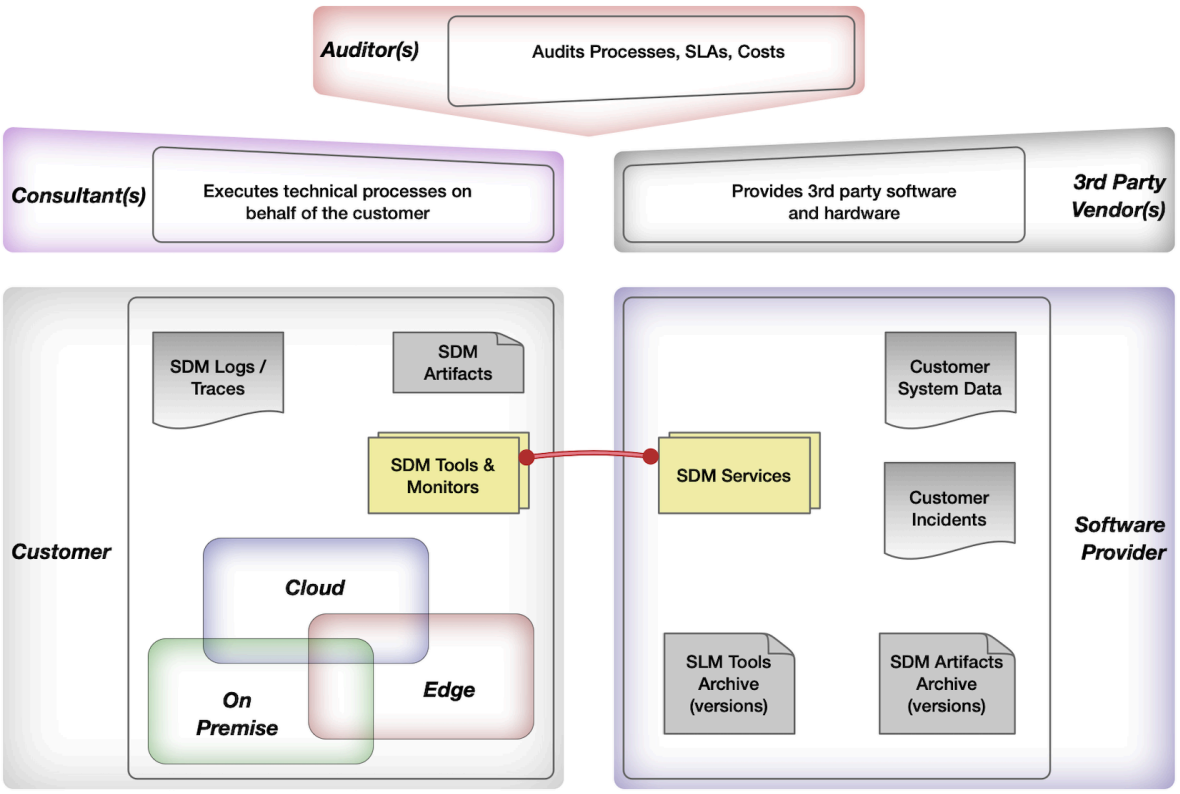

Figure 19 SDM environment and participants

The **Software Provider** is the IT company that has developed the software and offers ongoing support to its customers. The provider delivers software artifacts for the SDM procedures, which can be installation files, system upgrades, updates, patches (bug fixes), configuration tools, and more. The responsibilities of the software provider include delivering and supporting their products according to defined strict time and quality indicators (KPIs) to address various potential customer issues.

Externally or publicly funded projects usually have a designated **Auditor** assigned to the project. Although auditors do not directly contribute to the technical implementation, they are

an important part of the overall project and need reliable information about status and resources, as well as upcoming and met deadlines. Auditors need to assess whether the project is being implemented at the agreed speed and within the given budget. They also assess whether all modules that were described in the initial task have been reliably integrated into the overall scenario.

Technical consultants play a key role in the implementation of projects, especially in the implementation phase. They perform technical steps on behalf of the client. Their scope of work usually includes the client's on-premises environment, cloud-based components and services, and end-to-end IoT devices. Consultants also communicate at a technical level with all other parties involved. Very often, technical consultants are certified by the software vendor and are well-prepared for the SDM procedures used.

In most cases, a single software vendor cannot provide absolutely all the components involved in building the end system. Very often, the hardware, as well as the operating systems, are manufactured by other companies, which are collectively referred to as **third-party vendors**. Thus, the different manufacturers of the operating system, databases, servers, and IoT hardware add to the complexity of the SDN procedures. Compatibility testing of hardware and software supplied by third parties is an essential part of any SDM procedure.

All this is done for the benefit of the **end customer**, who pays for and uses the provided software. Customers are the participants who productively use the delivered software and participate in defining KPIs for the SDM procedures of the supported system. The maximum period of unavailability, availability requirements, requirements for protection (copies) of information in the event of a disaster, access to company resources and other KPIs are agreed with the customer.

In the first stage, all participants are engaged in the initial installation of the software. The customer pays for the software. The technical consultant downloads the installation files and starts the installation itself. During the installation, all important steps are recorded in the log files. After a successful installation, a configuration process follows, including settings and connection to databases, configuration of module addresses, the connection between them and other settings.

Another scenario that comes at a certain time after the initial installation is a **software version update.** In this case, the system usually stops for a certain interval of time, each module is replaced with its newer version and in some cases the tables in the databases are updated. During the update, all important steps are again recorded in log files.

A type of upgrade is the so-called **migration**. In this case, the differences between the launch and the new version are so great that the only option for transition is to install the new version cleanly. After that, only the data from the old system is migrated to the new one, and in many cases transformation rules are applied.

These typical scenarios are accompanied by many challenges, which are presented in the next chapter.

3.3 Challenges with traditional SDM

The first challenge that arises in traditional SDM is during the implementation of a new product. When deadlines are not met, participants often start asking themselves the following questions:

- Why was the hardware not delivered on time, were the suppliers informed about the deadlines (by phone, email) and did they confirm? And is the person who confirmed not on leave? Are there strict contracts with well-described penalties in case of delivery delays?

- Did the technical consultants take the latest version of the product before installing it?

- If the software does not work, is there enough information describing in what sequence the installation steps were performed, which the software supplier can check and assess more precisely where the problem is.

- To date, the answers to these questions are difficult to find. Often, communication and agreements are oral, by email or in long contracts with many clauses that may require long time to find and interpret properly the specific information.

Another important stage, and perhaps the most frequently performed, is when a software needs to be updated. The update may be planned and due to an improved version with more functionalities, but very often an urgent update (patch) is also required. In the case of a patch, the characteristic is that an error has been discovered in the current version of the software or

a security vulnerability, then a version must be urgently delivered in which the given malfunction has been removed. In the update scenario, in addition to the above questions, the questions also arise - which customers should be notified, as well as how they should be informed in a secure and confidential manner.

More details on the relationships between the participants in the SDM process, as well as the difficulties that are observed are presented in the next section.

System integrity and security

After an initial installation or update, it often happens that some SDM artifacts - executable files, configuration scripts or log files - are manipulated in the customer's network. Sometimes deleting such files is even part of the established procedures of the customer environment. In some cases, the person who has access to the scripts may consciously choose not to apply (accidentally or intentionally) a particular fix or fixes to the system. In this way, the system may be left in a broken state, or its security may be compromised. If the person has access to the registry files and logs, they can cover up all traces of their actions and hide such manipulations from the client and other parties.

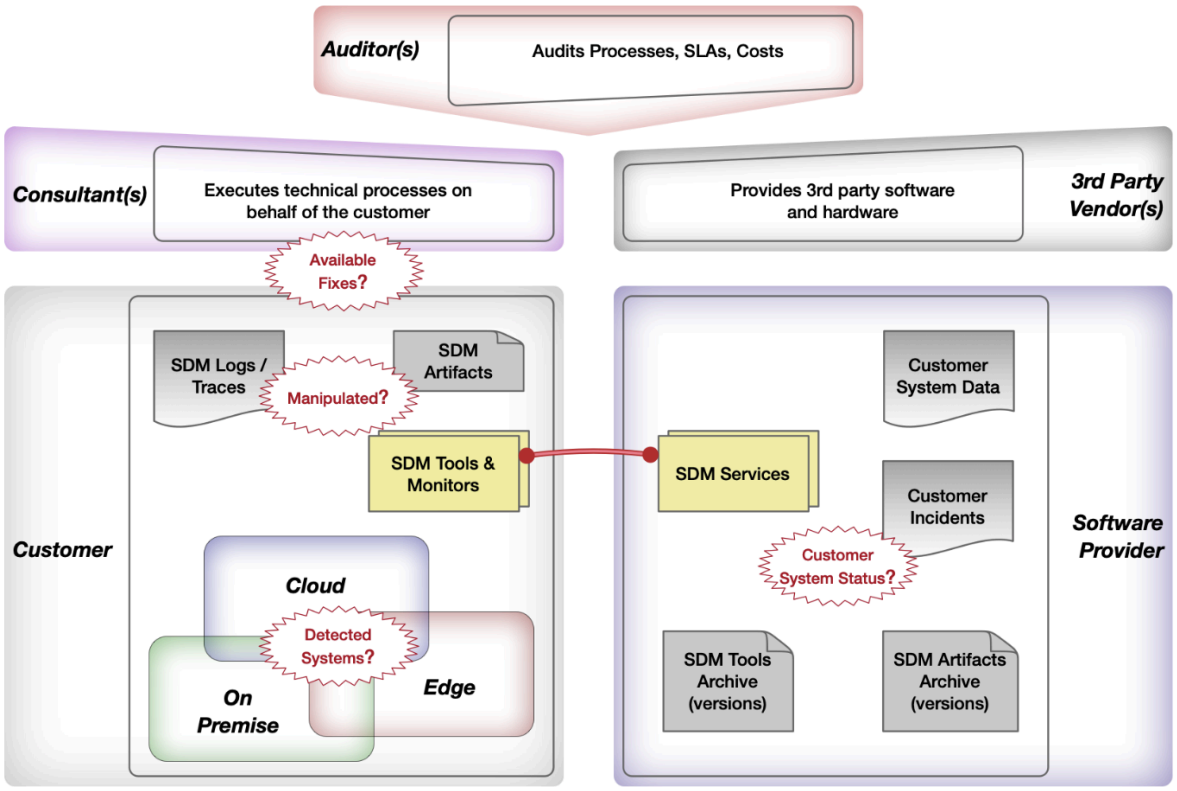

Figure 20 System integrity and security

Later, in the event of a dispute, the software provider can be held liable for all damages caused using the compromised system, without them being aware of what happened. In these cases, it is also very difficult for the audit firm to assess who is responsible for the given state of the system and who owes penalties. But along with liability, the bigger problem is that the integrity of the system has been violated for a long period of time, and it is difficult to find a way to restore it. A real security problem is also the situation in which the customer relies on that they have switched to a more secure version, but the update did not happen. The misleading thing here is that someone has only changed the version of the system in the registries without applying an update.

Data Visibility

Data visibility in the customer environment is an important aspect of supporting any system. A software vendor needs to know what applications are being used by their customers, what version they are on, and whether they are properly updated and configured. It is also important to know if there are early indications of problems that could cause the customer's system to crash while it is being used. On the other hand, the customer and their consultants need to know about all new versions of the software provided. All security patches should be applied as soon as possible, preferably before they become public knowledge and before successful attacks can be carried out.

It is also important where the bytecode of the installation files is obtained, is it signed by the vendor, and does the hash code match the original. All these visibility topics are extremely important for reliable support of productive systems.

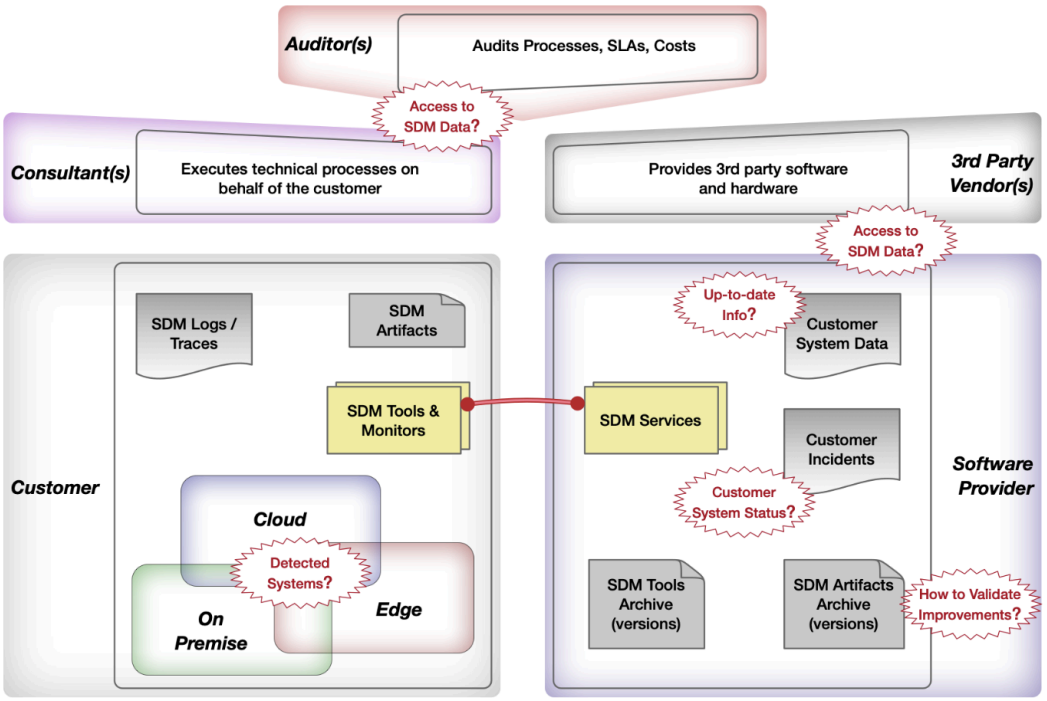

Figure 21 Data Visibility

System downtime and risk management

For many SDM procedures, it is important to predict the expected duration of the procedure steps, as well as the resource load, as shown on Figure 22. One of the most important stages for prediction is the interval of interruption of the systems in which they will be unavailable. In addition, some of the procedures may contain critical steps that could potentially cause additional problems depending on the type of databases used, operating system and other specifics of the environment. Therefore, it is good to historically store (anonymously) information from the procedures that have been executed on customers and based on this information to make a risk assessment. Having this information, the software provider can assess the risk of applying a procedure to a new customer or adapt some configuration parameters. For example, if the databases are from a given manufacturer and the operating system from a different manufacturer, specific optimal parameters for such an environment could be applied. Another example is, if it is known that for a given database size, the software update procedure takes 2 hours, and for a larger size it takes 8 hours, then the client can very precisely assess when to apply the update so that the system downtime affects the smallest possible number of its end users.

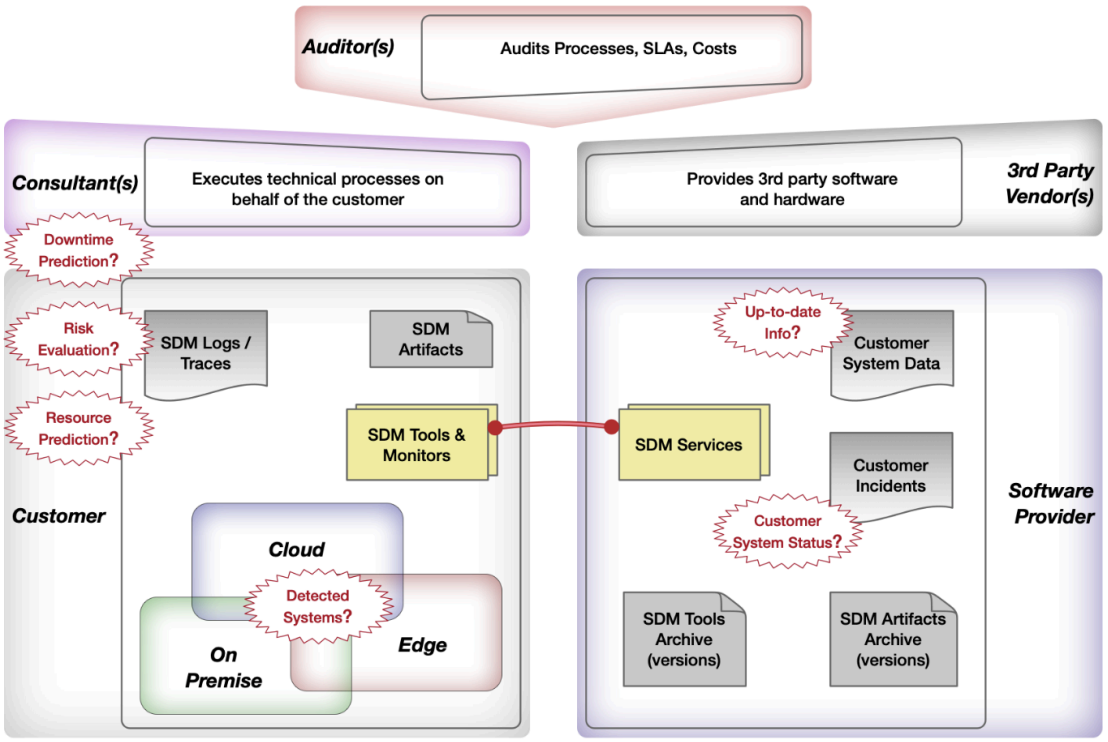

Figure 22 System downtime and risk management

Predicting system behavior

Sometimes, slow response from databases or warning messages appearing in the system log files are an early indication that the system is at risk and likely to crash. Very often, the customer and technical consultants do not pay attention to these early indications or cannot correctly interpret them.

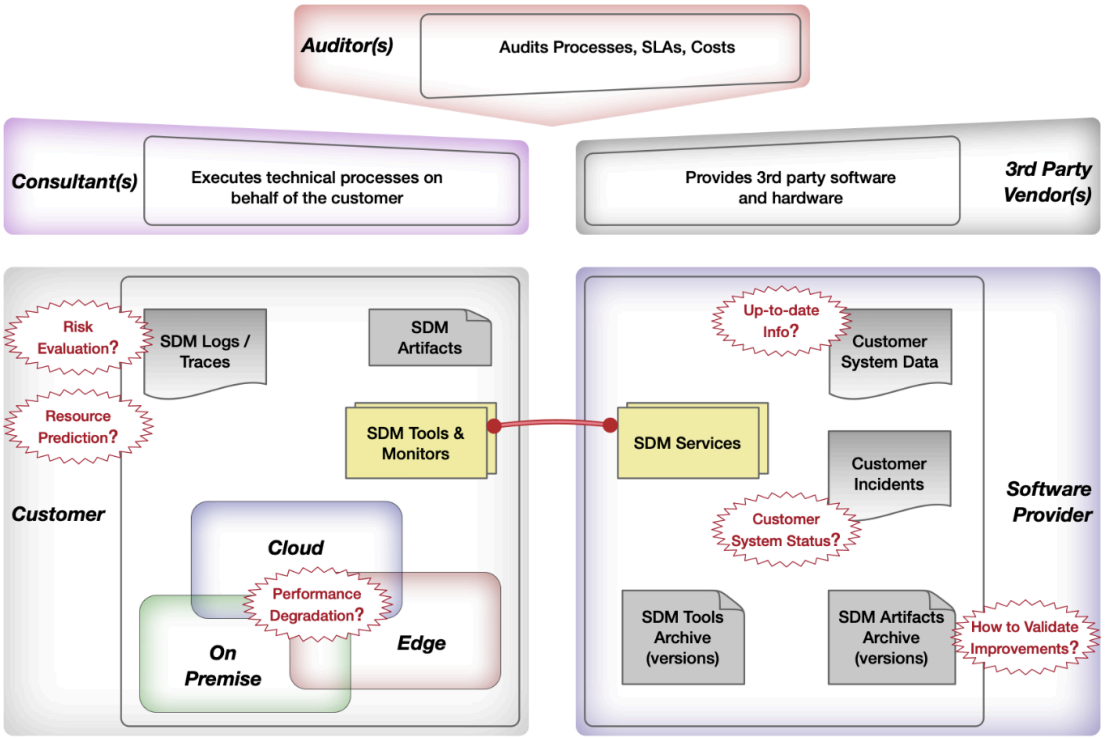

Figure 23 Predicting system behavior

That is why it would be beneficial to the software vendor to have access to this information and be able to prevent a crash for a specific client, but even more importantly, to analyze, assess and provide an improved version to all its clients who use a similar environment. Cases of critical system behavior caused by excessive, unplanned load that does not correspond to the initial declared (hardware and computing power do not correspond to the initially declared) are often observed. But even in this "easy" case, the end customer often has difficulty getting to the root cause. That is why key parameters such as processor load, database size, execution speed should be confidentially shared with the software manufacturer.

Responsibilities and SLAs

The way of negotiating delivery dates for components for a productive system, be it hardware, product software, operating system or databases, is still sometimes done verbally, by phone or email. Also, the sequence of which participant should do what, after whom and by what date is very difficult to track.

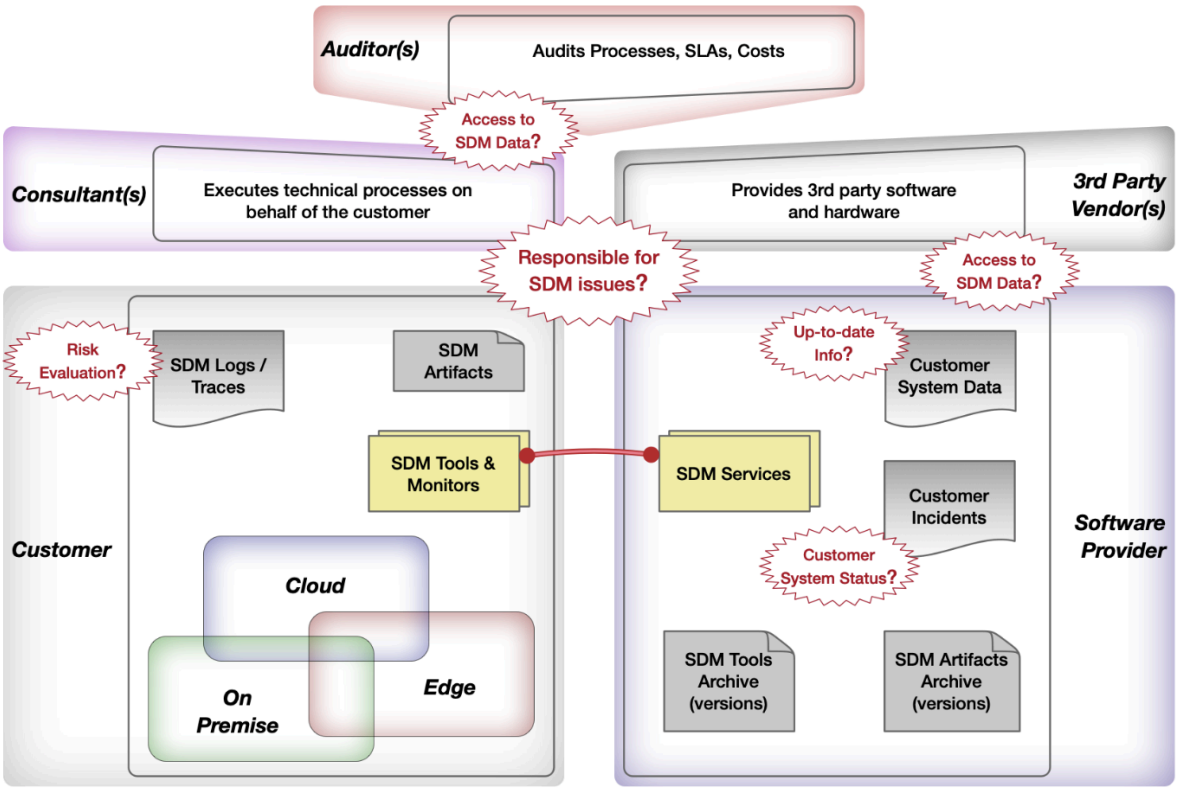

Figure 24 Responsibilities and SLA

Traceability in most cases is related to reviewing official meeting minutes and tracking which meeting was the last with the latest agreements. This blurring of responsibilities is a reason for project delays, as well as for audit firms to have maximum difficulty in determining who is responsible for a specific delay. This accordingly makes it difficult or sometimes even impossible to seek compensation for delayed projects.

To summarize, in the hybrid environment with many participants described above, there are five types of challenges:

System integrity and security: There is always a risk that some of the artifacts such as important executables, system configurations, log files and traces can be manipulated. Correctly determining the version of the component executables is also an important aspect of the overall security of the system. One possible attack on SDM data could be to manipulate the indication that a certain critical patch has already been applied to the customer's production systems, when the system has not yet been patched and is exposed to malicious attacks. It is important to have a simple and reliable method to verify the integrity of the system to avoid such attacks.

Data visibility: Each participating party has access to only a part of the overall data, most often only information related to its contribution. Often the overall picture is not completely transparent to all decision makers. For example, a software vendor does not always know which components and product versions are used by a particular customer. SDM and software usage data are visible to the customer and possibly to technical consultants but cannot be used by the software vendor to accurately predict the important parameters of an upcoming update procedure.

Downtime and risk management: Combined and anonymized data is used to make a precise forecast of the resources and time for each future procedure. The presence of a large set of data from previous procedures allows the forecasting to be performed by various algorithms, including artificial intelligence. Thus, each customer could predict the time of unavailability of their system for end users.

Predicting system behavior: Obtaining information from running systems such as execution speed, processor load, database size, error messages or warning messages could allow the software vendor to advise the customer or the technical company that serves the customer in a timely manner of possible problems and risks. This information could be used for possible improvements.

Responsibilities and SLAs: It should be possible for each stage and implementation of given agreements to be easily tracked and in case of violation, penalties can be automatically triggered as compensation to the affected participant.

Before examining the next chapter on a completely new architecture based on DLT that solves the listed challenges, let's compare the problems again with DLT characteristics.

Using DLT as a medium for data sharing, easy traceability and the impossibility of deleting and manipulating information would solve some of the challenges.

Smart contracts can automate various processes, as well as document responsibilities and even trigger money transfers as part of compensation for missed deadlines.

Secure and confidential informing of customers who have security problems will be realized by DLT. Through its cryptographic functions, public and private key DLT can provide the necessary messages in the most reliable and confidential way possible.

DLT and IPFS can provide reliable bytecode storage, with hash functions showing that a client is using the correct version. All procedure logs could be recorded in DLT, leaving a permanent record of the changes made and the participant who made them.

A valid question would be, if SDM is based on a DLT system, is there a need for an auditor as a participant in the implementation of a project or are all steps so visible and all penalties well defined and programmed that this role becomes redundant.

The rest of this chapter describes an innovative blockchain-based architecture that aims to solve the problems that exist during the implementation and maintenance of software systems in a complex production environment. The first two subchapters focus on the selection of a blockchain platform, comparing the most used platforms, as well as reviewing the main characteristics of the platform chosen for the purposes of the prototype presented in the next chapter.

3.4 Architecture of the SDM system based on DLT

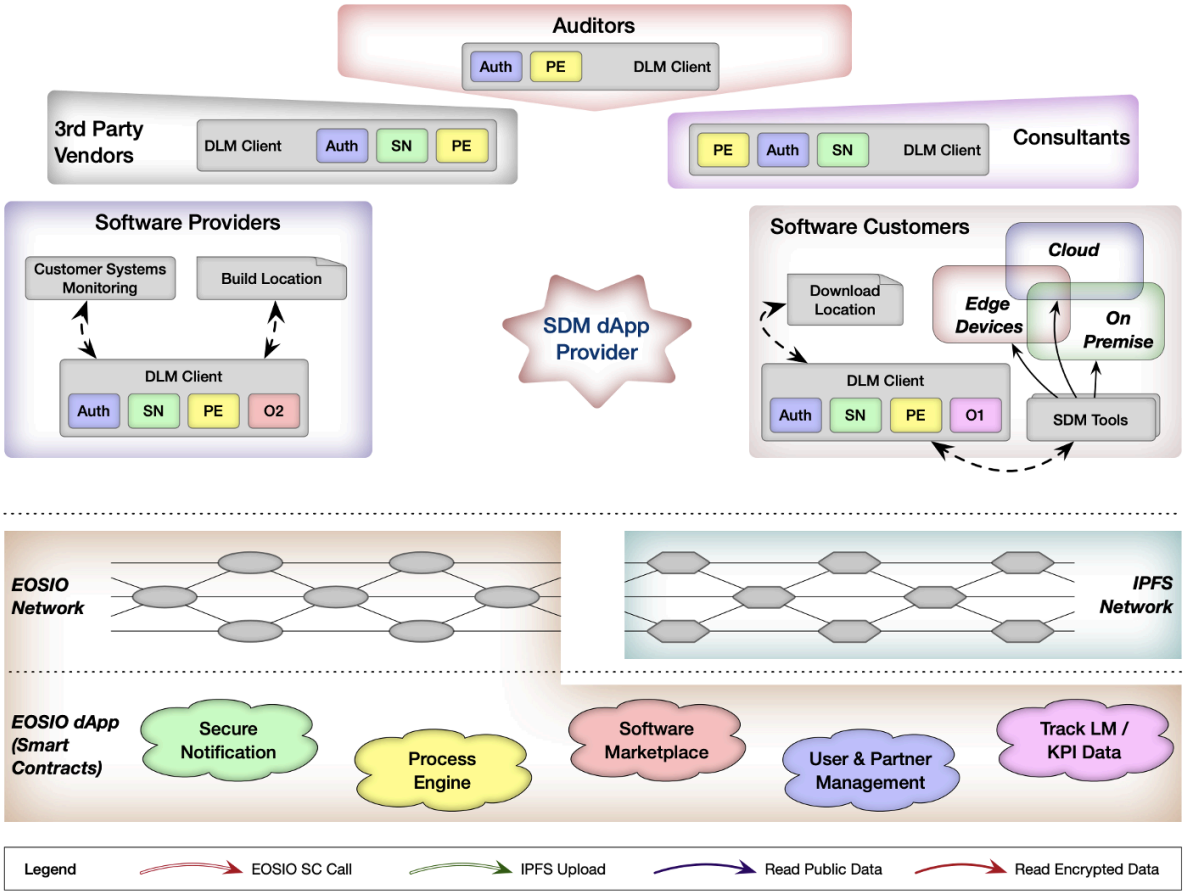

Figure 25 SDM system architecture

Up to this point, a review of blockchain technology has been made. Details and a comparison of the most suitable types of blockchain were shown, namely Ethereum, EOS, Hyperledger Fabric and R3 Corda. Based on the analysis performed, EOS was selected as a platform for implementing SDM architecture as the most applicable for the SDM case study. It was described how EOS works - development environment, smart contracts and other important modules and key features. The previous chapter showed what the main challenges facing the SDM topic are. Based on this, we can move on to defining and describing a blockchain-based architecture that addresses the problems presented in chapter 2.

The main elements of the system are a public EOSIO blockchain, a distributed file storage system IPFS, a single dApp that is published on EOSIO and uses pointers to files stored in IPFS. Each participant in the distributed SDM process uses one or more locally installed Distributed Lifecycle Management (DLM) clients/agents to communicate with the

blockchain/dApp, and this communication will in most cases also reach the other participants in the SDM process and/or the database (IPFS).

The architecture is based on five smart contracts [56,57], which can be defined as modules: a user management module (User and Partner Management), a process management module (Process Engine), a module providing secure communication (Secure Notification), a software marketplace module (Software Marketplace) and a module containing data from the client for tracking LM/KPI data (Track LM/KPI Data). These modules are part of the dApp.

The role of the local distributed agents (DLM) is to provide the ability for clients to authenticate and use the smart contracts of the SDM dApp, to attach resources in the form of files to IPFS. The data that DLM writes to EOSIO or IPFS is not encrypted, but, if necessary, in order to maintain data security, it will be encrypted using the public keys of the parties to the process who have the right to read the data.

Specific to EOSIO is the automatic authentication of each participant who changes the state of the process by adding, changing or “deleting” data from the current state of EOSIO. In fact, deleting and changing data from the state does not actually delete it from the system, since it remains forever in the blockchain history, but only marks it as deleted. The authentication of the participant is performed by encrypting the data with a specific client private key and checking the participant by the public EOSIO blockchain against the list of public keys associated with this EOSIO account. The use of public keys in EOSIO can be used not only to authenticate the account, but also to encrypt and decrypt data to which the client needs to have access. Such public and private keys are called transport keys because their primary role is to provide encrypted communication to the account. Technically, public keys for account authentication and transport keys for EOSIO are the same, and any generated and associated public/private key can be used for both purposes. From a security perspective, it is good to have separately generated transport keys to ensure independence of key re-generation cycles and subsequent invalidation of the old key.

The role of IPFS is to provide a database for large files and is important for the SDM process. Another role of IPFS is to enable the system to delete personal and confidential data that is only stored temporarily in SDM processes. Government regulations such as GDPR, for example, require that personal data be removed from storage after a period in which this data is needed to perform the process. Since the data in each blockchain is immutable, the only way to use confidential data in SDM processes, but also to allow their deletion after the

completion of the processes, is to store the data in an external system, with only a pointer to the specific file being stored in the blockchain. As is known, the pointers to files in IPFS are their hashes, which are stored in the EOSIO dApp. In this way, the system guarantees the immutability of the IPFS data used by each SDM process. Very often, the data stored in IPFS should be accessible only to a certain group of participants in the SDM processes. Examples of such files are logs from the execution of the systems at the clients or binary and other files related to critical fixes related to improving the security of certain components that are not yet publicly available. The protection of such data in IPFS is carried out by encrypting the files with a symmetric key generated individually for the file, and the key itself is encrypted using the public key of the specific participants in the process and is recorded in the EOSIO dApp. In this way, only the participant whose public key was used can decrypt the symmetric key and use it to decrypt the contents of the specific file in IPFS.

Another role of IPFS is purely financial. Data stored in EOSIO is relatively expensive to store compared to IPFS. When there are large amounts of data, it is much more efficient to store them in IPFS than to use the EOSIO CPU and EOSIO Net resources while uploading the data to EOSIO and blocking EOSIO RAM while it is stored. The difference in the cost of storing data in EOSIO and IPFS is several orders of magnitude.

The role of the client in the SDM application (client_DLM) is key. Its main role is to enable each participant to fully participate in the SDM processes, as the DLM decrypts and provides participants with information that is not public but is shared with them by other participants in the process. Another role of the DLM is to automatically perform complex steps of the SDM process, such as those that require the application of data during the process execution. In particular, when there is a file, such as a log file, that is relevant to the performance of specifically defined KPIs by the customer and the software provider, what DLM does is encrypt the file with a symmetric key, upload it to IPFS, add its hash/pointer to it in EOSIO, and write there the symmetric key encrypted with all the transport public keys of the participants who have the right to read that file. In this way, all participants automatically receive the information added by DLM about the participant who added that particular file to the SDM process, and those participants who have the right to use this data can read it using their own private transport keys to read and use the symmetric key.

One important role of DLM is to be able to automatically execute various steps of the SDM processes. This is important for some types of blockchain oracles that regularly publish

data to the SDM dApp. This is implemented through a simple or more complex scheduler that executes steps of the process at fixed or flexibly defined intervals. For example, an oracle that feeds the SDM process with logs or status of specific client systems that are relevant to defined KPIs can be triggered once a day, or when a specific pre-fixed amount of logs is present, or when specific exceptional events occur. When this step of the process is triggered, one or more of the files monitored by the oracle are encrypted, uploaded to IPFS, and a notification is sent to EOSIO via DLM. In this way, all interested participants automatically gain access to important data that can affect the SDM processes.

In some scenarios, the DLM client can be called directly from the systems running at the client and used to send data about the SDM process to the smart contracts. Thus, any SDM procedure that has a reference to the blockchain in its code will look for the DLM and send the information to it. Usually, the software provider itself has set the procedures to be executed only if there is a properly configured DLM and it correctly makes records in the blockchain. In this way, all authorized participants have real-time access to important data about the execution of the specific SDM process. This type of data about the SDM processes, once collected from different clients, can be used to automatically predict future executions of the SDM process at other clients. An important part of this type of data collection is the anonymization of the input data. Data about the execution of the SDM processes at a specific client should not be accessible or obtainable from the system, for example, by their competitors. This can be achieved by allowing certain smart contracts with highly restricted access to collect and anonymize the collected data, which can be used to predict the performance of future instances of the same or similar SDM processes. Oracles [58] provide external data to smart contracts running on blockchain technology. They are essentially a form of communication between the outside world and the blockchain world. An example is the oracle O1, which reports various system KPIs to the EOSIO and IPFS smart contract platforms via the DLM client. Another example is O2, which is responsible for all necessary interactions with the blockchain regarding the software product catalog.

3.4.1 Addressing the requirements for a modern SDM system

In this section, we will present how the requirements described in Chapter 2 for a modern SDM system, namely "System Integrity and Security", "Data Visibility", "System

Downtime and Risk Management", "System Behavior Prediction", "Responsibilities and SLA" are addressed in the proposed architecture.

System integrity and security

An important aspect of the operation of modern software systems is the guarantee that the code that is executed on the client systems is the same code that the software vendor has provided to the client. In many cases, malicious actors are motivated to replace one of the supplied installation files with a version that has previously implemented weaknesses that could lead to suboptimal system operation or could give unauthorized persons access to critical system resources. In some specific cases, the different version of the code is simply an older version in which vulnerabilities have been discovered that have been addressed in the latest versions. In this type of attack, the malicious intervention can be something as simple as replacing the version of a critical module in the declaration without replacing the files and solving the vulnerability problem. In this way, system monitors would provide information that the problem was solved based on the declared version of the code, but the vulnerability is available and can be exploited by malicious parties.

Let’s look at how the blockchain-based SDM architecture solves these two types of problems that lead to incomplete or vulnerable code on the client. In the more common scenario, the client or their technical partner wants to install a product and downloads the installation files accordingly. In the case of the blockchain architecture, they will check via the DLM client, via the Market Basket module, which is the latest version of the given product, as well as what is its hash in IPFS. After that, they can download the necessary files locally, comparing their hashes again. Moreover, if the installation does not go successfully, and accordingly the auditor, who also has access to this information, determines that there is a delay, then the log files produced by the installer can be viewed. In this case, the log files of the procedure are automatically recorded on the blockchain via the DLM client and, accordingly, cannot be changed or deleted at a later stage. Based on this information, it can be assessed whether the procedures were performed in the established order, whether the hardware met the requirements, and whether the correct operating system with the required version was used.

When moving to a higher version, a similar procedure is again performed, here the peculiarities are that the version of the provided final software is checked. As after a successful update through the DLM module, the client notifies and changes the version

number it uses, and this new flag is recorded in the blockchain. At a later stage, various parameters can be seen such as - when the version was changed, whether the update was successful, how long it took, and so on. These things are important for the auditor, especially in the event of a problem.

Using IPFS, the recorded information is always available, reliably distributed across multiple computers in the p2p network, and it is kept for years without burdening the size of the EOSIO itself.

Data visibility

Another common problem of SDM processes is the unavailability of process data collected in real time or after its execution. With the proposed distributed SDM system, this is addressed, as all relevant resources during the preparation of the process, as well as during its execution, can be shared with a selected group of process participants. Most often, these are the client himself, the consultants and the software provider, but in some specific processes, this list can also include additional software providers, as well as process auditors. In this way, all involved parties can verify the parameters of the process before it starts, and in some cases, obtain data on the state of the systems before the process starts. During the execution of the process, data such as available resources, system unavailability interval and execution time of the individual steps of the process and the result of their execution can be monitored by different participants securely and, if necessary, in close to real time. In this case, EOSIO and IPFS play the role of a trusted environment for distributing and storing information about the specific instance of the SDM process.

In the previously discussed process of updating a software version, all key data, starting from the initial state of the system, hash codes of the new software, who started the update, how it ended, the time interval in which the update was performed, are stored for years in the blockchain and no one can manipulate this history of the actions that occurred.

System downtime and risk management

One of the most important parameters of SDM processes for the client is the risk to the system and the time when the system will be unavailable for use. The risks in SDM processes are of different nature – exceeding the planned time of unavailability of the systems, exceeding the planned budget, an incompletely functioning system after the process is completed, deterioration of the security of the system operation, and in the worst cases – an

unsuccessfully completed SDM procedure or even data loss. Depending on the type and complexity of the SDM procedure, any of the risks may lead to damage to the client. Collecting anonymized data on the implementation of various SDM procedures at other clients, as well as on the implementation of other similar SDM procedures at this client, can help create a realistic assessment of the risks of a future SDM procedure through the system. The advantage of using a system based on smart contracts is that the collection and use of such data can be done completely automatically without giving access to the collected data to any of the participants in the SDM procedures.

A good example in this area is software updates. When updating software, many companies keep information about how big their database was before the update, what operating system and hardware they used. Based on this information, when updating for the next client, he can get quite precise information about how long the update will take, how long his clients will not be able to use his services, and thus the client could estimate in which period he will do the update itself. Also, if manual steps are used during the update, the client can provide administrators for the given period, as well as very precisely see whether they have completed these steps in the interval specified by the process. This is something that in traditional SDM is a problem area and causes a lot of disputes, mainly related to deleted logs and deleted history of what happened.

Predicting system behavior

Sometimes, slow response from databases or occurrence of warning messages in the system, which are recorded in the log files, are an early indication that the system is at risk and is likely to crash. Very often, the client and technical consultants do not pay attention to these early indications or cannot correctly interpret them. Through the Track LM/KPI Data module, this information will reach the software provider, and thus critical situations can be proactively prevented.

Responsibilities and SLAs

SDM procedures are often complex processes that require the contribution of many participants. These participants are usually financially dependent on the outcome of the procedure and are not necessarily organized in a hierarchical structure. Therefore, the responsibilities for the implementation of steps, especially those that lead to the occurrence of

events related to financial, reputational and other damages, are sometimes difficult to be clearly defined.

The proposed system allows the process to be monitored by many participants, with each step of it having a clearly defined responsible participant. In this way, the implementation of each individual stage of the entire process can be monitored in terms of time and quality of implementation by the other participants. If necessary, the system can automatically monitor the implementation of various financial parameters related to specific SLAs. For example: Payment to consultants could be tied to the successful implementation of the SLA of the upgrade process within the agreed time of system unavailability. At the fixed start of the unavailability interval, the SDM system can check the versions of the working components of the client systems. After the end of the unavailability period, the SDM system can check again whether the system is now available and whether its versions match the planned upgrade versions. If the results of the checks meet the parameters set in the plan, the system can automatically make a payment to the consultants' EOSIO account or to another EOSIO account specified by them.

Typical questions that are asked during the implementation of a traditional SDM system are: Where are the deadlines described?; Who must deliver something by when?; Was the responsible party informed in time?; Is a phone or email used?; Was the responsible party on vacation? etc. give way to automatic and clear rules described in the blockchain. Everyone is informed reliably, in a secure way and deadlines and responsibilities are visible and unambiguously defined. Each process can be linked and secured with cryptocurrency, and penalties could be automatically paid if agreements are not met.

In the last example, the question of the need for auditors even arises, or would this party be redundant in a modern SDM blockchain-based system. From what has been stated so far, the transparency, integrity, security and analysis of the systems is guaranteed in the most reliable way, based on proven cryptographic algorithms and with access provided in distributed p2p networks.

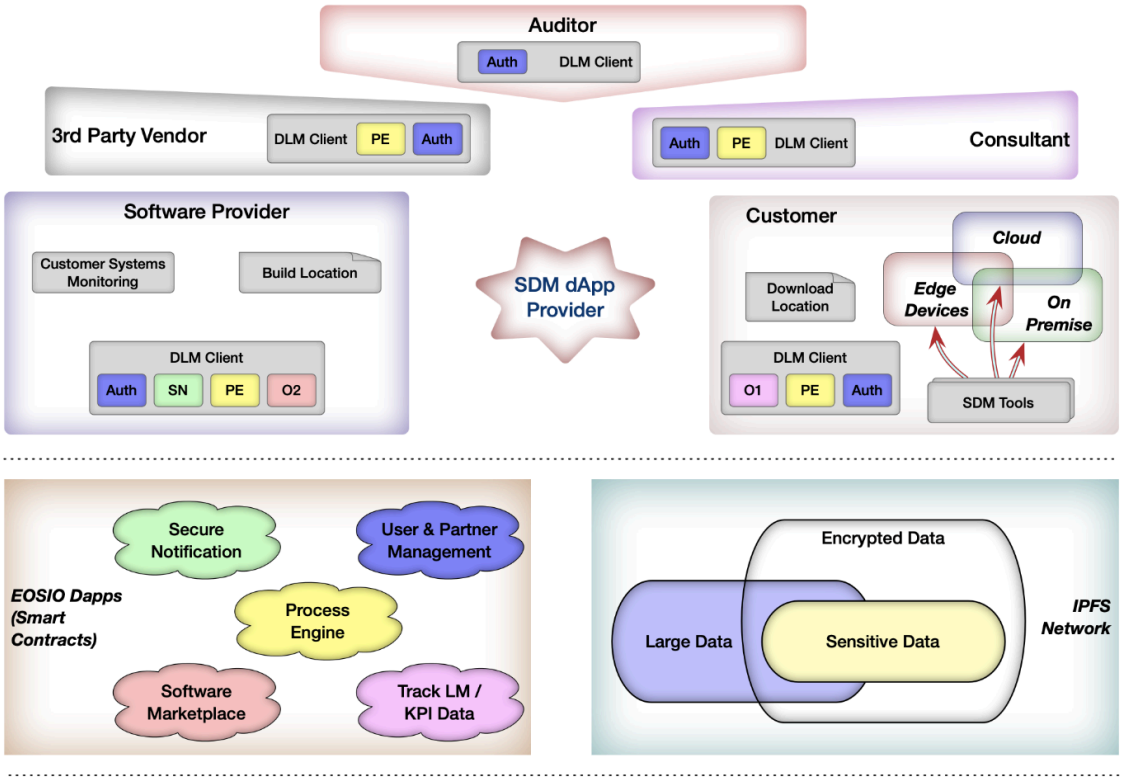

Figure 26 DLT based SDM system based on EOSIO and IPFS (components)

3.4.2 Main modules and communication channels

The following smart contracts of the proposed SDM application are the basis of the dApp: a User and Partner Management contract, a Process Engine, a Secure Notification smart contract, a Software Marketplace, and a module containing customer data for tracking LM/KPI data.

The User and Partner Management smart contract monitors the relationship between different parties, roles, and processes. It is used to validate authorization for EOSIO accounts, as well as to trigger certain changes in the state of processes. One of its main goals is to maintain a model for each process, who the participants are and what roles they perform. It is permissible to have more than one participant in any of the roles. For example, in one SDM process there may be several software vendors, consultants, third-party suppliers, as well as auditors. It can be assumed that customers are usually unique for each process, but even this interpretation can have variants in situations where a group of participants have common requirements and do not have a direct hierarchical relationship with each other. In such cases,

an SDM process can have several participants, defined as clients, but it must be chosen which of the participants is responsible for each individual step when client action is required. For each participant, the system keeps data on the public transport key used, which is to be used by the secure communication module. The data on the processes and interconnected participants is filled in by the software provider or by the client, depending on the type of specific process. One or more EOSIO accounts are maintained for each participant with their associated public keys for accessing EOSIO, as well as the public keys they have provided for exchanging encrypted messages.

Process Engine monitors the processing and data associated with specific multi-step SDM processes. This module can create new SDM processes and change the statuses of processes and user roles during the execution of the processes depending on more complex rules defined by the participants themselves. It is also responsible for automatic payments as a result of manual interaction or triggered automatically when a set deadline is reached.

An example scenario is when a client is looking for suitable consultants to perform technical LM processes in a transparent and verifiable manner. Using a process engine, the parameters of the process that the client needs to perform for a given operation - installation, upgrade or others - can be set. When the parameters are set, different consulting companies can make offers for performing the activity. Different parameters of the offers can be time and price for performing the operation, time of unavailability of key systems, guarantees and KPIs for performing the activity, etc. Based on the offers made, the client can automatically or with the participation of a representative chosen by him decide which of the proposals to use. After this choice, the process is transformed into a normal SDM process for the system, with the role of the consultant being given to the participant who made the best offer. Another situation in which there is a need for a flexible Process Engine is when the specified operation consists of several different sub-operations that need to be performed simultaneously or sequentially. In such a task, the role of the Process Engine is to track the execution of each of the steps and notify the participants who are responsible for the execution of each of the next steps.

When communication or notifications between the participating parties must be encrypted, they use the Secure Notification functionality. Another main task of this smart contract is to provide access to the encrypted data to those participants who must be able to read it. This is done in several ways. The main way to protect the information is by

encrypting this information, which is not intended for general use, with a symmetric key generated for it and making it public in EOSIO or IPFS. Then, the symmetric key itself is signed by the participant who provides the information and encrypted with the public keys of the participants who have the right to decrypt this information. In this way, any participant who has the right to read this information can use their own transport private key to read the key for the encrypted information, as well as use the public key of the information provider to verify that the data is authentic.

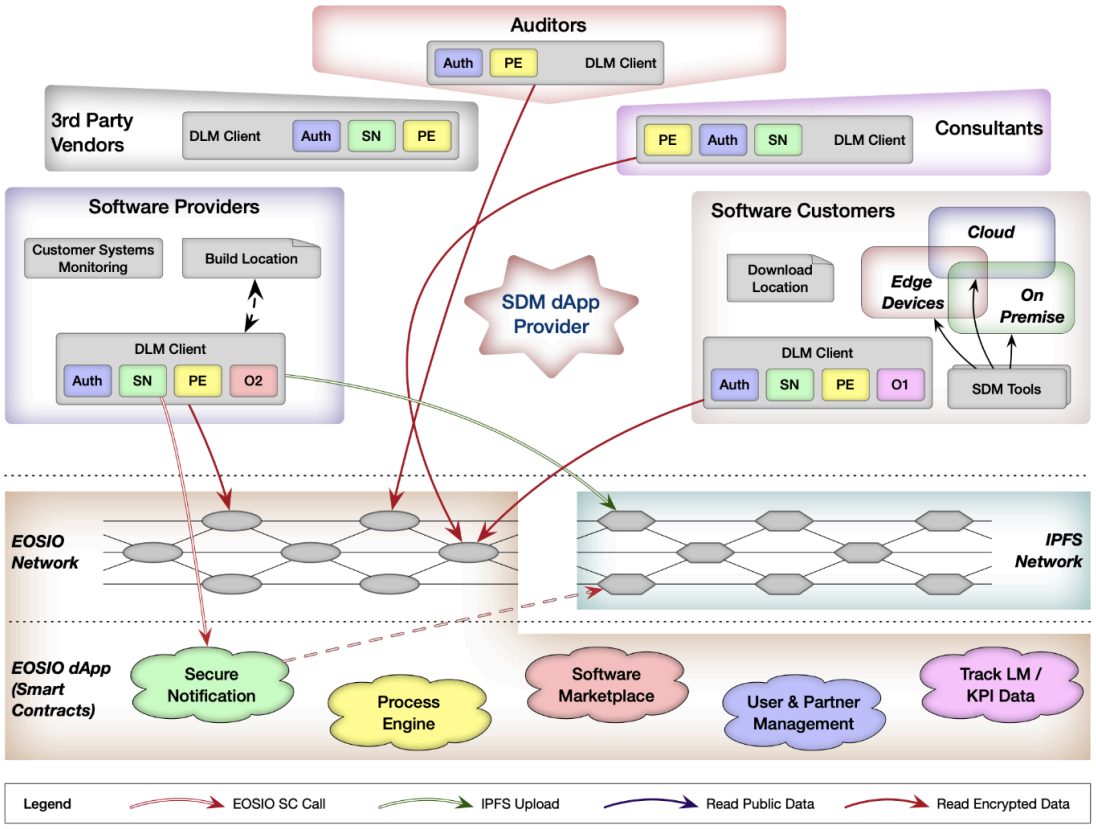

Figure 27 Secure communication

The Software Marketplace smart contract stores information about available SDM artifacts and versions offered by software developers. The Software Marketplace state is updated by oracles running at the software vendor and is used to automatically validate artifacts installed at the customer and check for patches and upgrade recommendations. The information stored in this module is distributed between EOSIO and IPFS. Publicly available data can be stored in unencrypted form. Information about available products offered by software vendors, their versions and the status of their versions (stable, unsupported,

beta-release), etc. Each product or component of a product is most often stored in IPFS due to its size. In some special cases, it is possible to store data about software versions or patches that are not publicly available to everyone in IPFS. For example, a security patch that is specific to customers using a specific version of the product is often preferred to be made available first to the affected customers for a certain period and only when their systems are protected can it be made publicly available. One part of the software market is accessible only to the software provider. This part contains information about the customers using a given software – versions and used components of specific systems for each of their customers. In this way, when starting each SDM process, data from the software market, such as the initial and final version of the product during an upgrade or which new components need to be installed and configured for operation, can automatically be used to start a new SDM process by the software customers. The data in the individual sections of each software provider can be integrated into their CI/CD processes so that they can quickly and flexibly provide up-to-date versions to their customers. This is organized through specially prepared oracles that use their DLM client to connect to the available methods offered by the software market and to maintain the current state of the software offered there. The normal life cycle of software releases is to eventually start as beta, stable releases, recommended releases, and then to versions that are no longer supported. Unsupported versions of software still need to provide their binaries to ensure the security of customer systems that have not yet migrated to the next version supported by the software vendor. Additional tasks that can be performed by this module are tracking the periods for which the software is licensed, as well as notifications when the license period expires or the support period expires for the version of the software that is being used by the specific customer.

Finally, all data from client systems is collected by the smart contract for tracking data either through manual operations or through established oracles running on the client site. The potentially confidential data itself is ultimately encrypted and uploaded to IPFS. The keys to decrypt data in each file are generated and recorded in the Track LM/KPI Data information for each party. The data stored in this module is collected exclusively during the operation of the client systems. These systems can be installed locally at the client, be cloud systems, or even be different (EDGE) devices, the operation of which is monitored and controlled. When collecting information about the operation of many devices, individually installed DLM clients can be used, or several devices can be served by a single DLM client, which provides them with write access to this module. The information collected here is

extremely valuable for software vendors, as it can be used to obtain feedback on any of the SDM processes that are performed through the proposed system. Data on execution time and possible difficulties and delays in real executions of SDM processes allow for more precise predictions about the execution of these processes for future clients, as well as for warning about risks in their execution. A characteristic of this data is that it is owned by the client and access to it must be controlled very carefully. If it is necessary to process this data, it must be anonymized. When storing data in EOSIO/IPFS, access can be given for analysis by software in the blockchain, which collects and analyzes the data, and then offers only summarized results that do not contain a connection to the SDM procedures performed on the systems of a specific client. Another advantage of collecting data on the execution of normal processes is the feedback to the software provider on the quality of the software they offer. Metrics such as system load, the available amount of processing power, memory and network resources can be used to analyze the way in which clients of the software use it. The collected data can be analyzed both with predefined algorithms and can be integrated with artificial intelligence modules to make more accurate assessments.

3.4.3 Covered use cases

The proposed system can be used for various single SDM processes – installation, upgrade, patch, security patch, system copy and migration, provision and improvement of high availability (HA), disaster recovery (DR) capability and many others. But one of the strongest sides of the proposed system is that it also allows for the composition of complex SDM procedures, which consist of many different operations on different systems under different boundary conditions. For example, a complex SDM procedure can include the installation and configuration of new systems at the customer, addition of new modules to existing systems, upgrades and corrections, as well as transition to high availability (HA) and/or disaster recovery (DR) mode. All these processes can be arranged in time to optimize the total time during which the system is unusable during the execution of the various SDM procedures. The main component for modeling complex SDM processes and systems is the Process Engine.

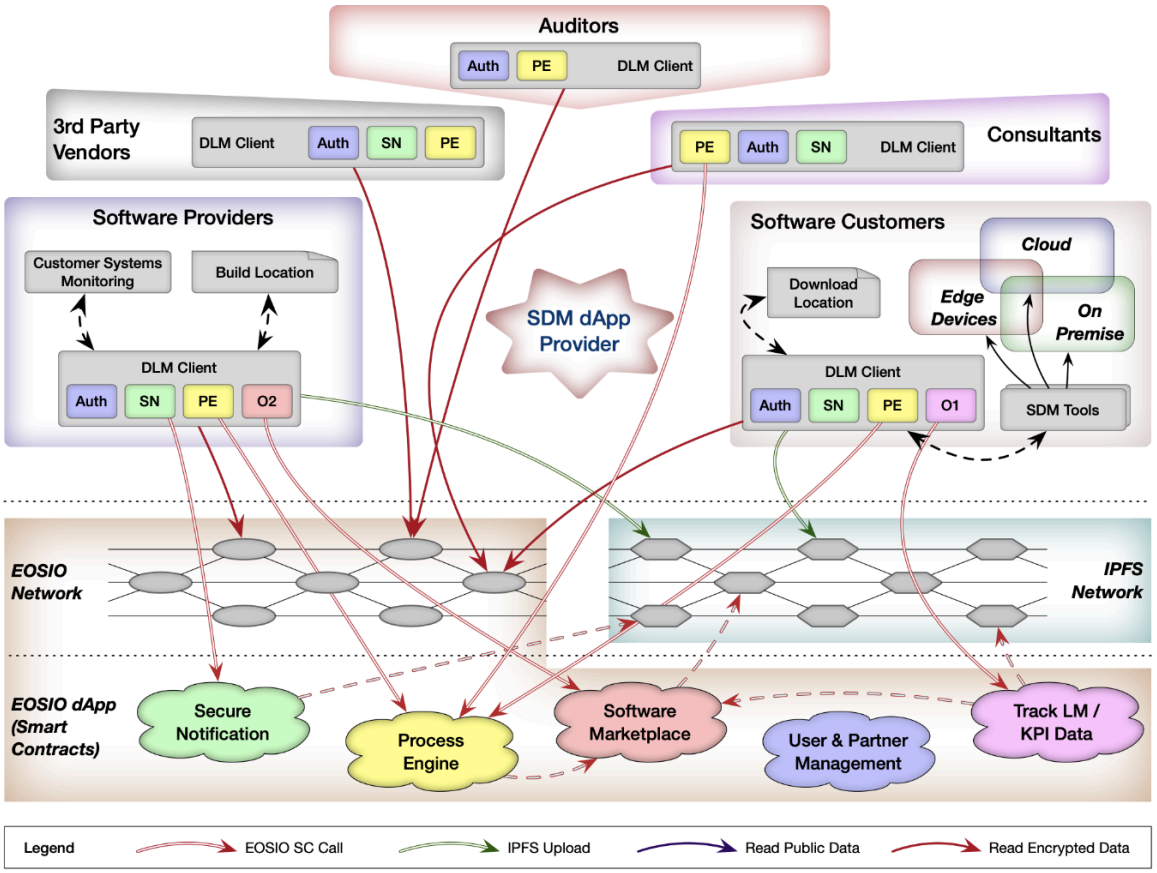

Figure 28 DLT-based SDM system based on EOSIO and IPFS (interactions)

These are some of the atomic SDM processes that the proposed system supports.

When installing software systems and components, the steps and flow of information can be traced through EOSIO and IPFS. First, the client or his technical consultant, through the smart contract for the software market, examines the available versions of the product offered by the software provider. The version that most closely meets the client's requirements is selected from the list. This can be the latest and still beta version, which contains new functionalities that the client particularly cares about. Or the latest published version that is officially supported. Other clients may prefer the most stable known version to be sure that what they are using will not have weaknesses characteristic of the latest versions of a given system. After the client, who is registered, selects a version to be installed, he gains access to binary installation files. During the step of obtaining access, a check can be made for possession of a license for the product. Files that are relatively large are stored only in IPFS and most often encrypted. The catalog in the smart contract for the software market contains

only pointers/hash codes of the files that can be found in IPFS. Code validation in IPFS ensures that the files have not been manipulated by a participant other than the software provider. During the installation itself, the installation software uses the pre-installed DLM client to maintain communication with EOSIO and report the duration and status of each step of the installation process. Later, when the installation procedure is already completed, all logs will be uploaded through the smart contract for collecting customer data, with all data in encrypted form with access only to the customer, the software provider and the participants in the process that the customer has chosen - consultants, auditors or third-party software providers. The main information in the data collected in this way is a flag for the success and execution time of each step of the process. In the case of an incomplete installation, the specific step that caused the failure can be traced and the reasons can be analyzed. In cases where the installation was successful, but the planned time was significantly exceeded, data can be retrieved about the specific step and the reason for the delay in the process. This is important when installing a system that replaces an existing productive system and there is a need to migrate a large amount of data before the new system can replace the old one for productive use.

In short, the installation of a new software system can go through the following stages from planning to the start of its productive use:

1. The software vendor develops the software and offers it to its customers via the Software Marketplace smart contract. This is shown on Figure 29. To use the proposed system, each participant in the processes of the proposed system that offers software to its customers must install and configure its own DLM client. This client must be configured to use a unique account of the software vendor in EOSIO. The next steps, when the software is selected and the license fees paid by the customer, are usually performed without requiring active participation from the software vendor.

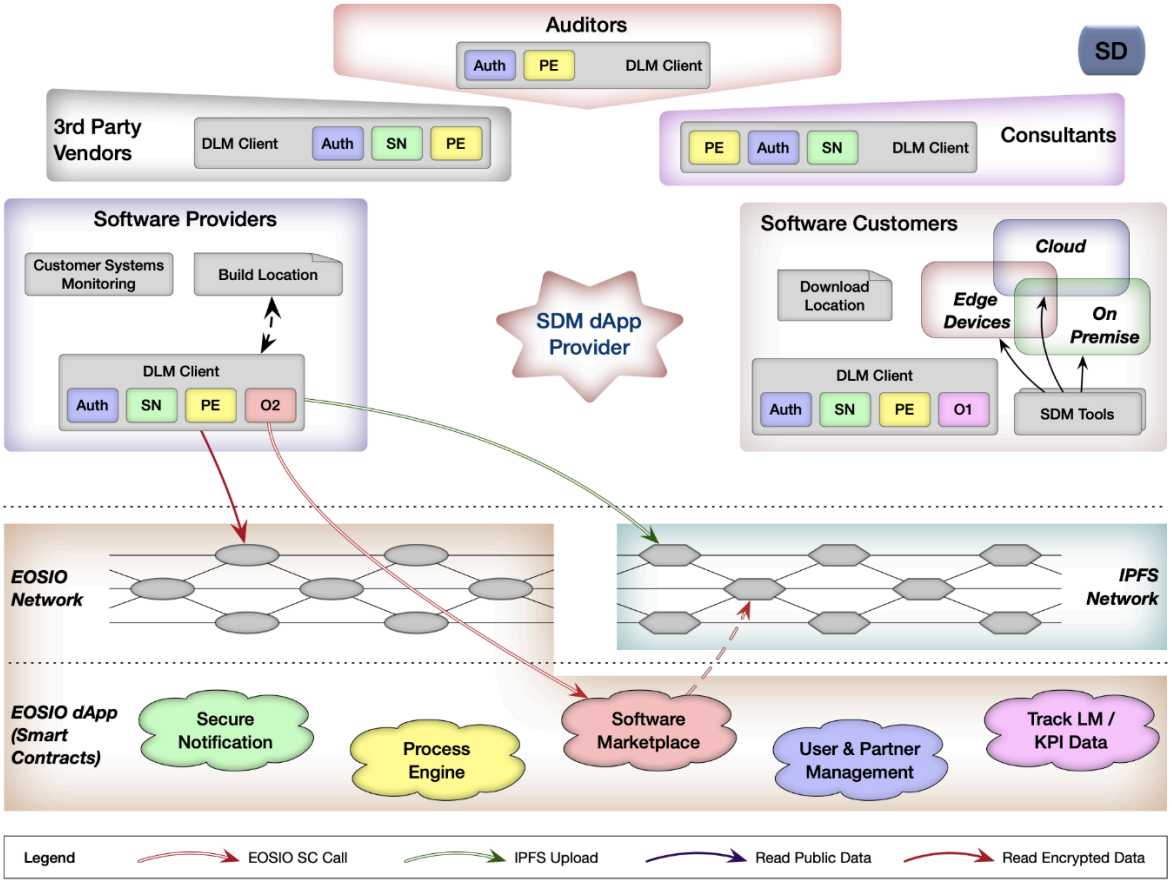

Figure 29 Software Marketplace

2. Each of the remaining participants must have a DLM client installed and configured as part of the preparation for participation in the SDM process. The software client selects and configures the participants in the specific process, with some types of participants being duplicated and others (auditors, third-party representatives) being absent for a specific installation. The selection of participants, such as consultants, can be done by the system itself, which can assist in the selection from several competing consultants. This is shown in Figure 30. At the software client, the DLM agent downloads the installation files licensed by the software provider and provides the connection to the functionality offered in EOSIO and IPFS via the DLM client. A check of the entire procedure can be planned and reviewed by all participants before the installation itself has begun.

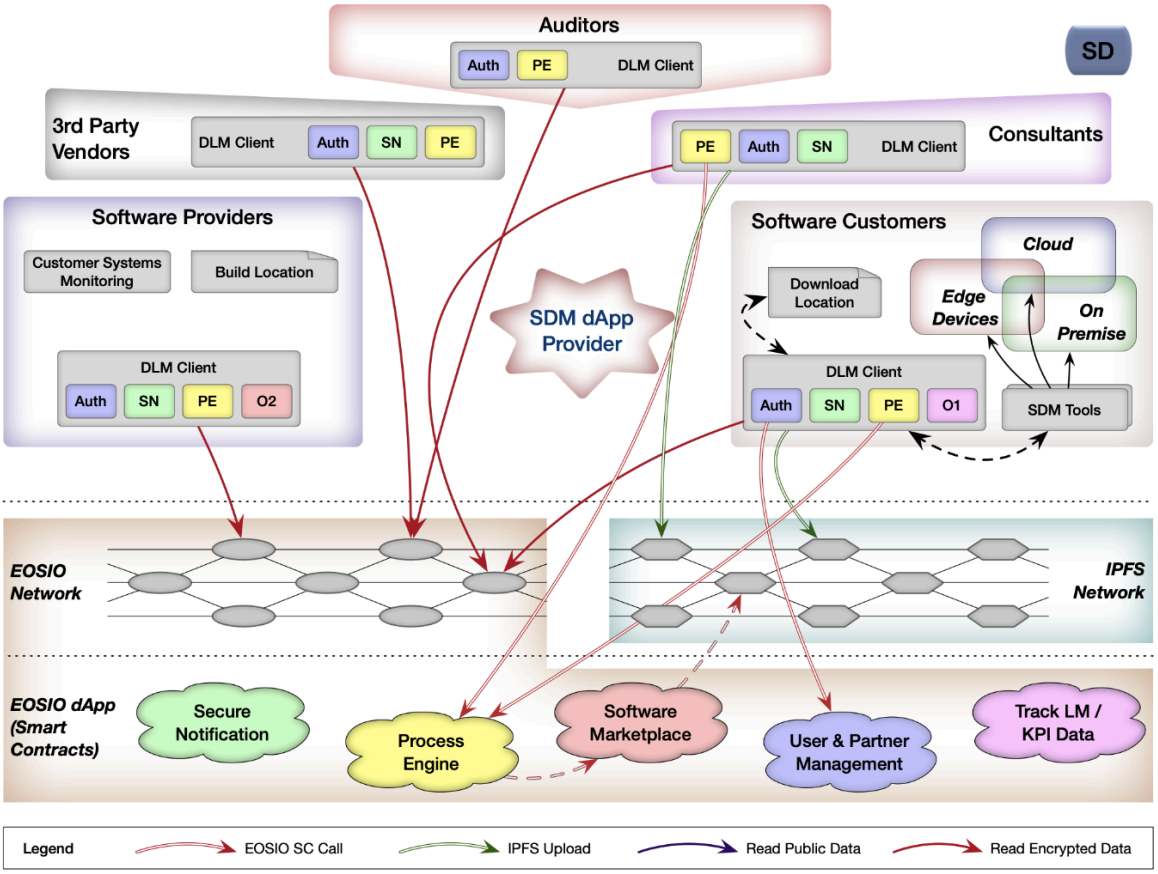

Figure 30 Installation and participants

3. Perform the installation and initial configuration of the software. The entire process can be traced by each of the participants since the installation data is published via the DLM client encrypted in IPFS, and the keys, as well as the file pointers, are shared in the dApp running on the EOSIO platform. In the event of unforeseen events that threaten the successful or timely completion of the process, the other participants (such as software providers or hardware providers) can intervene to avoid risks to the success of the procedure.

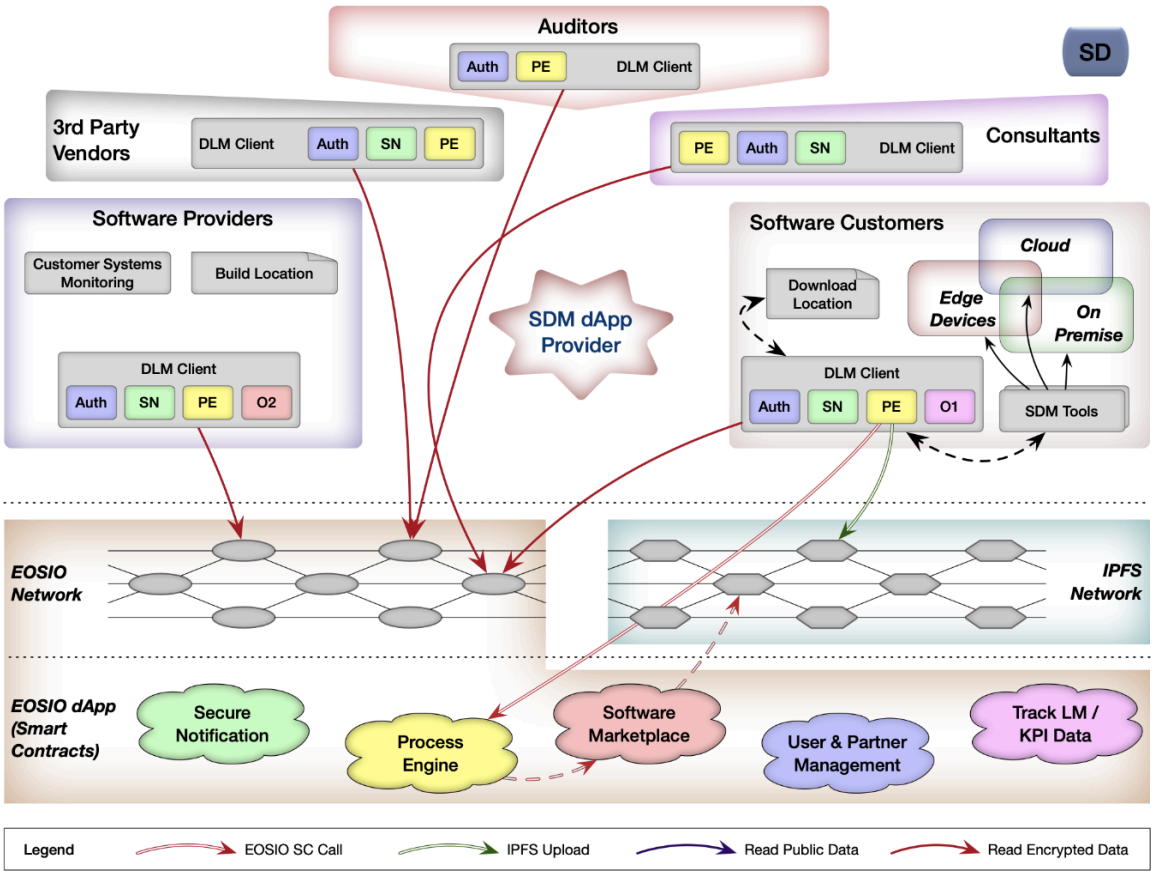

Figure 31 Installation process

4. After the installation is complete, the collected data is stored anonymized so that it can be used to predict future processes. Special modules in the Process Engine create models of process execution and use them to accurately assess execution time and risks when planning future SDM processes.

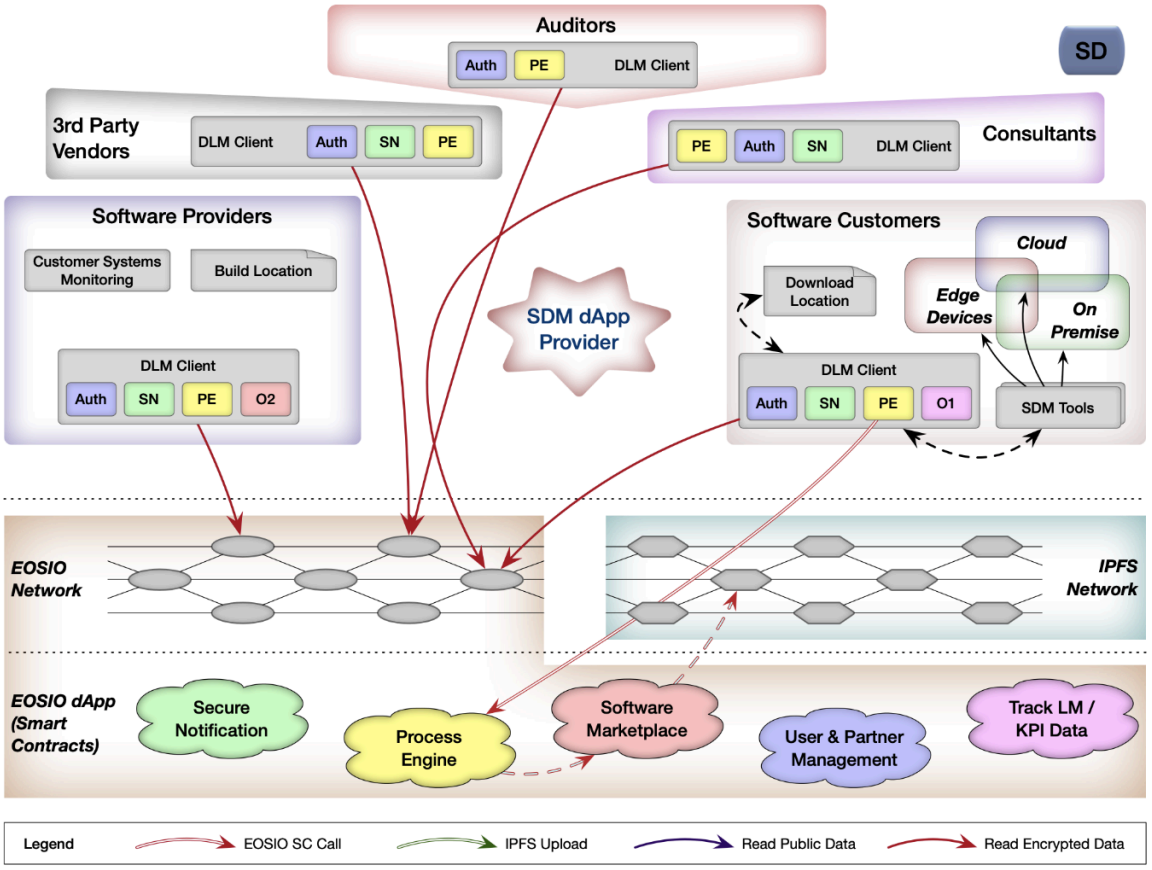

Figure 32 Process Engine

5. The productive operation of the systems continues to be monitored, with the data used to calculate the agreed KPIs being stored for a limited period in IPFS. This allows for analysis of the systems' operation not only by the client and the consultant, but also by the software providers and possibly by the process auditors.

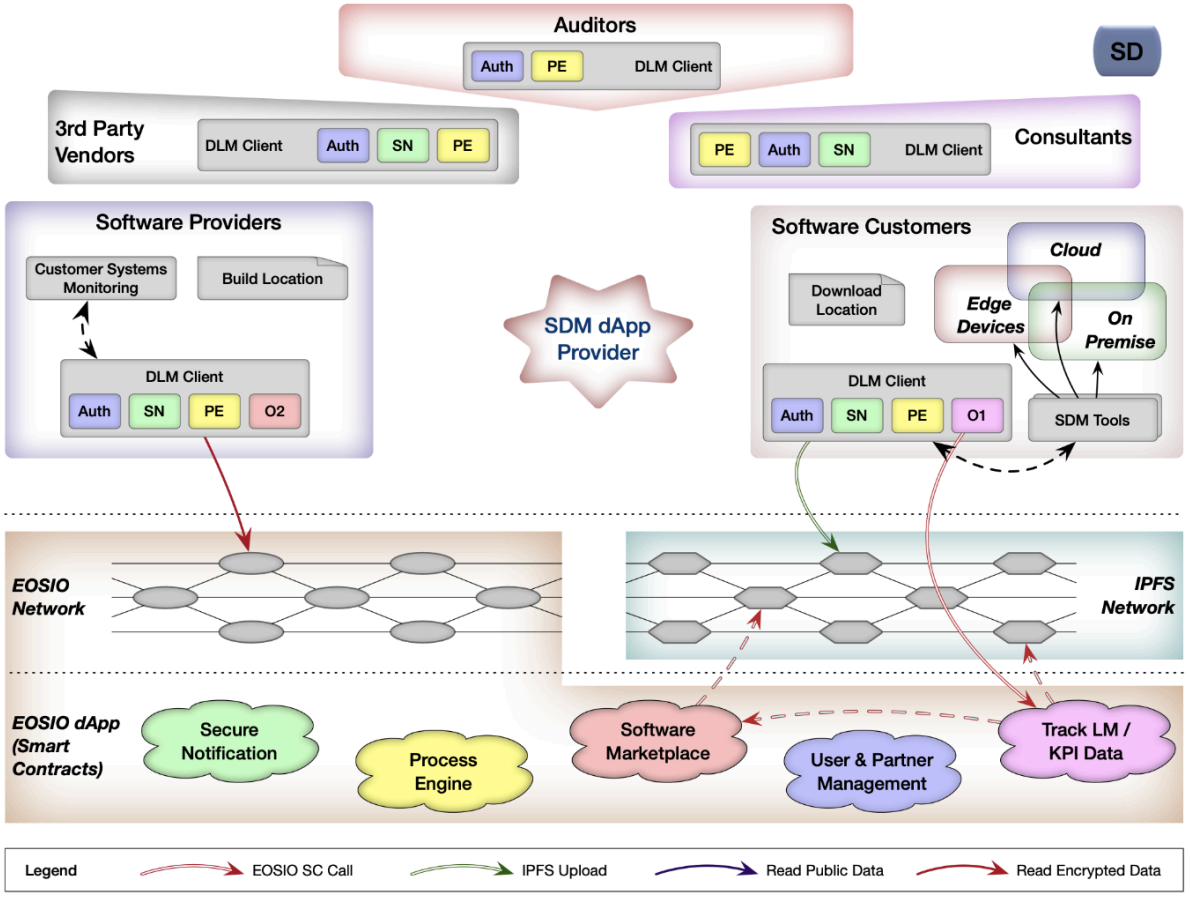

Figure 33 KPI monitoring

The upgrade, update, or software patch processes, although different in nature, can be described by similar processes. One feature of product security patches is that in many situations, the publication of information about a product security issue can be the beginning of an attack on customer systems by unscrupulous individuals and organizations. To avoid this risk, it is important to make security patches available to affected customers before they are made publicly available. This gives customers time to apply the patches to their systems and protect themselves from attacks that exploit their vulnerabilities. The stages through which a patch is applied are as follows:

1. Detecting the need for a fix. In some cases, the need for a fix, such as a fix that improves performance, is detected based on data collected about the performance of the customer's systems. When the data shows unusually high resource usage or extended execution times, the software vendor considers the possibility of correcting the system code to avoid these effects.

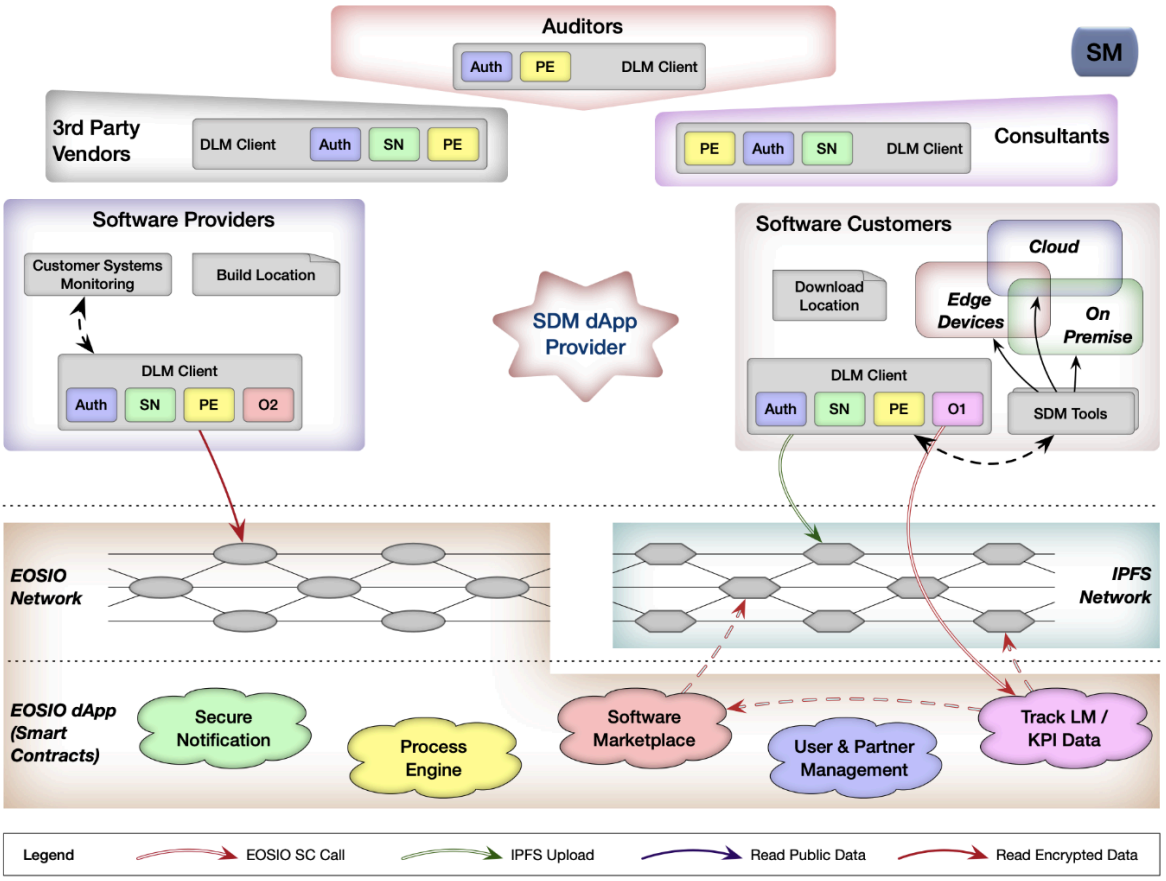

Figure 34 Need for correction

2. The second step is specific to security patches. After preparing a security patch, the software vendor does not automatically publish it to all interested parties but publishes it only on IPFS in encrypted form. Once the patch binaries are available on IPFS, information about them is shared via Secure Notification only with those customers who are currently using the versions of the software to which the security patch applies.

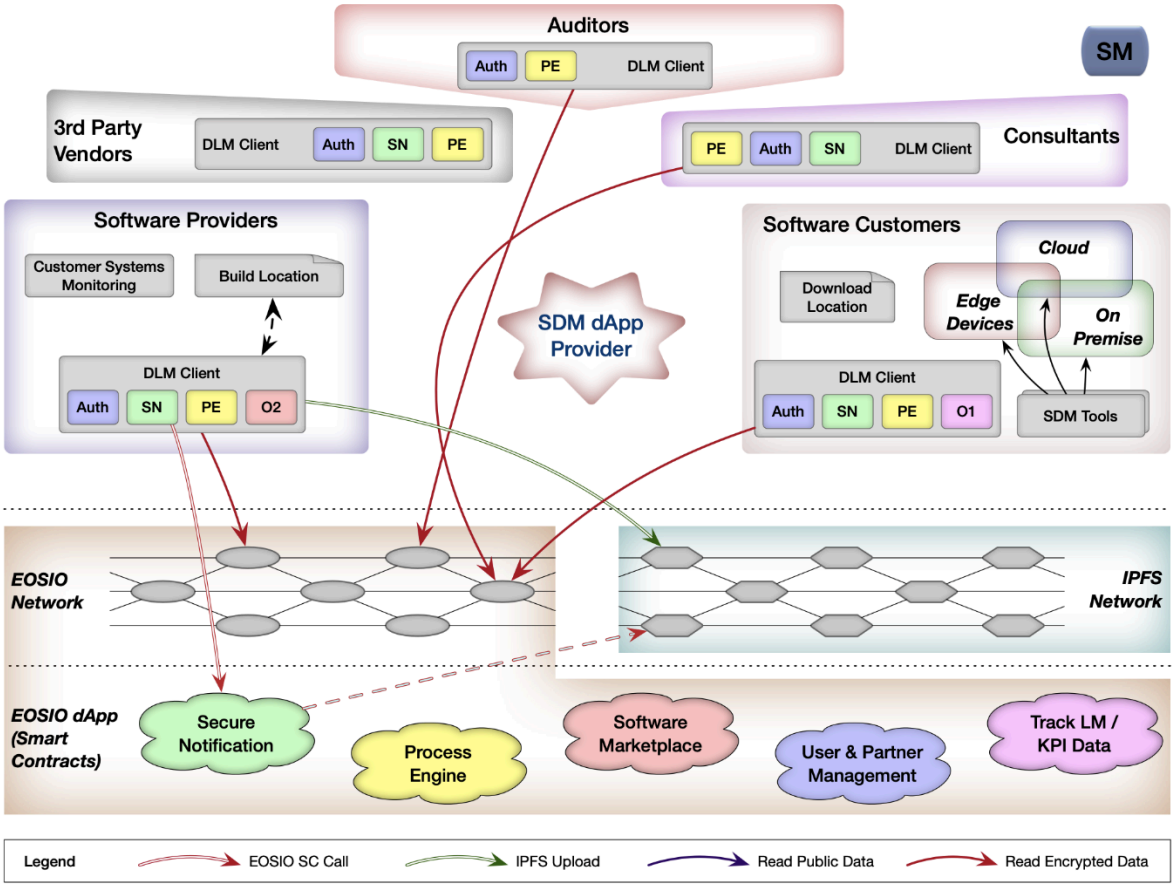

Figure 35 Security patch notification

3. The official release of an upgrade, update, or patch may be made directly or, in the case of a security patch, after a waiting period after notifying the current customers affected by it. When a new version of the product or patch is released, it becomes visible to all EOSIO customers, unless it is encrypted, for access only by the affected customers.

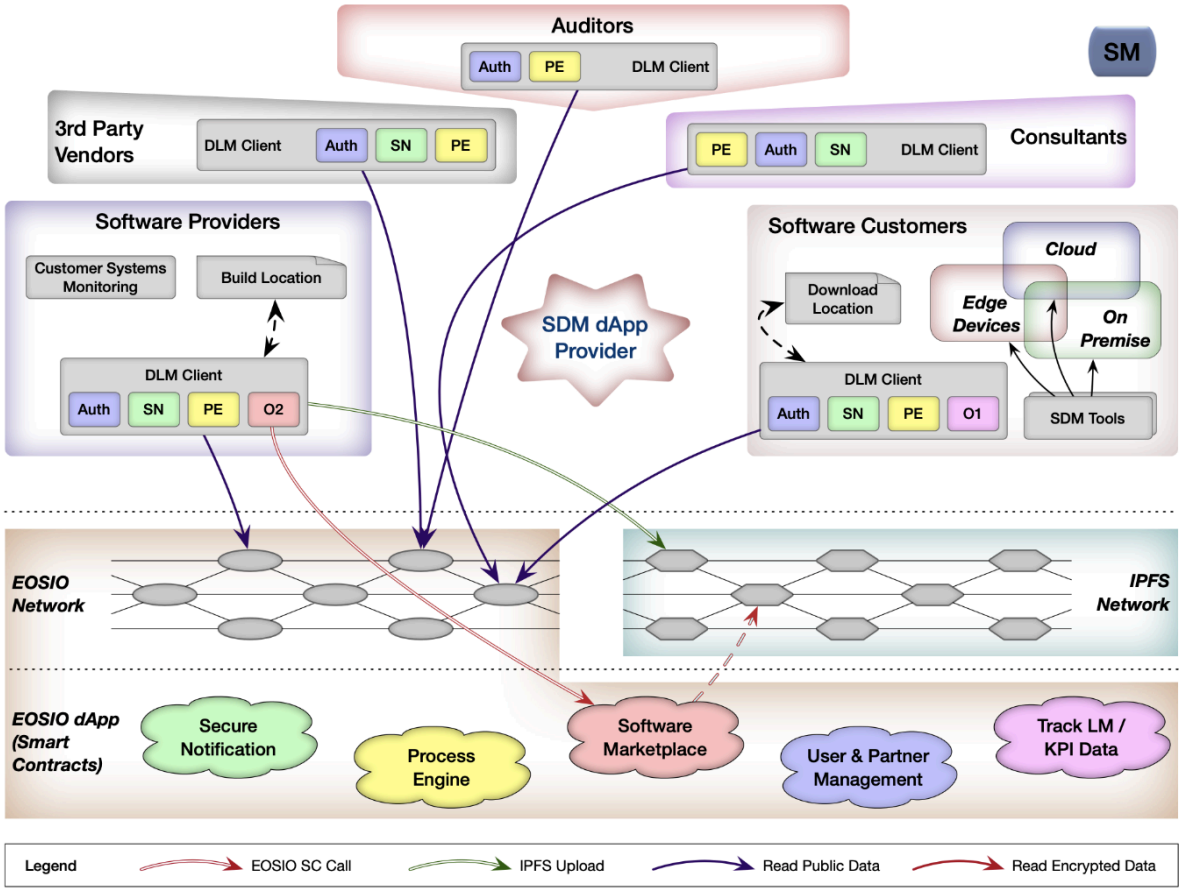

Figure 36 Publish an upgrade

4. Selection of a consultant to perform the SDM procedure. In some situations, the procedure could be performed by several different consulting companies. In such cases, one option provided by the proposed dApp is to collect offers for performing the procedure from different consultants. After announcing the request for offers, the client gives the consultants time to review the proposal and make offers for it. At the end of the selected period, the consulting company is selected automatically (based on pre-set rules for prices, duration and other parameters) or manually by the client. The Process Engine is the module that flexibly sets and processes the selection rules and directs the entire selection process.

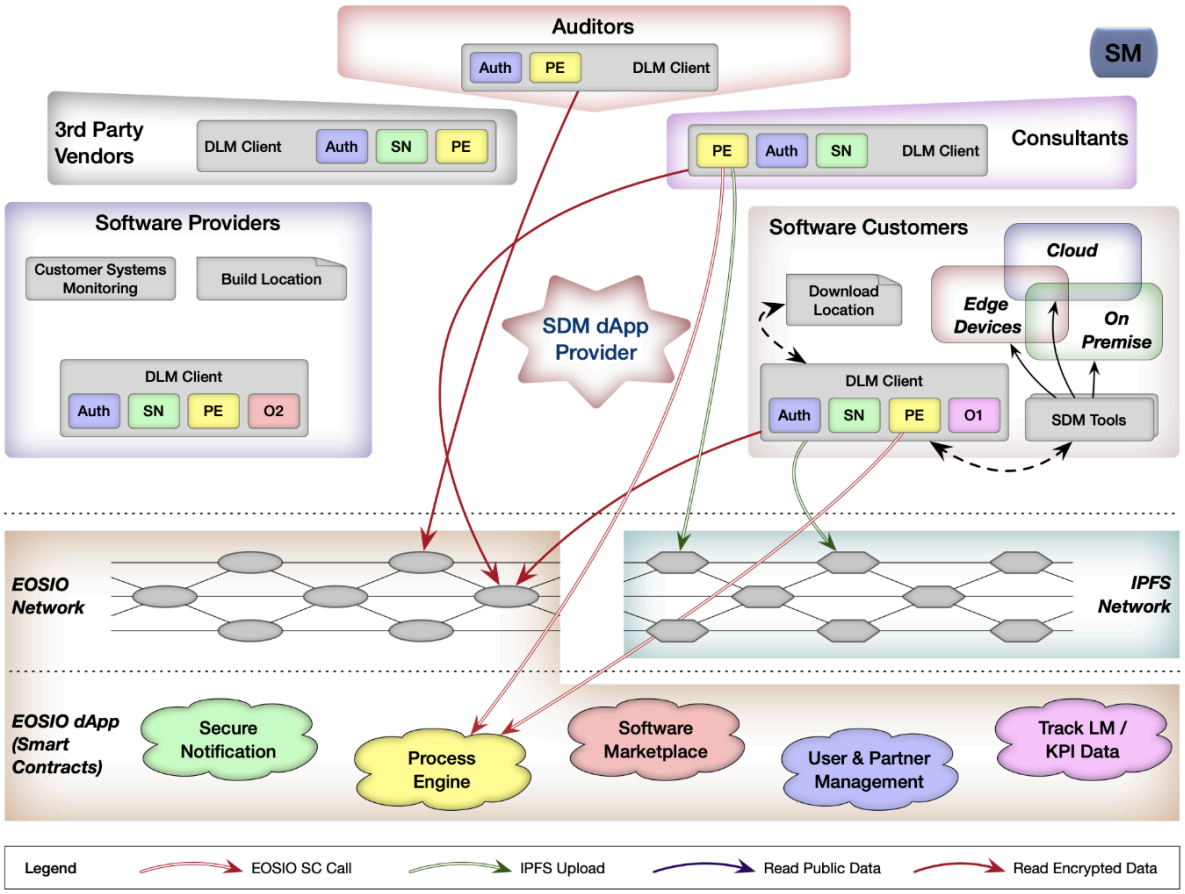

Figure 37 Offers for selection of consultant

5. After selection, technical consultants receive a time window preferred by the client in which productive systems could be unavailable due to the upgrade, update or correction of the software. The procedure can be monitored by all affected participants and, if necessary, the procedure can be corrected or stopped.

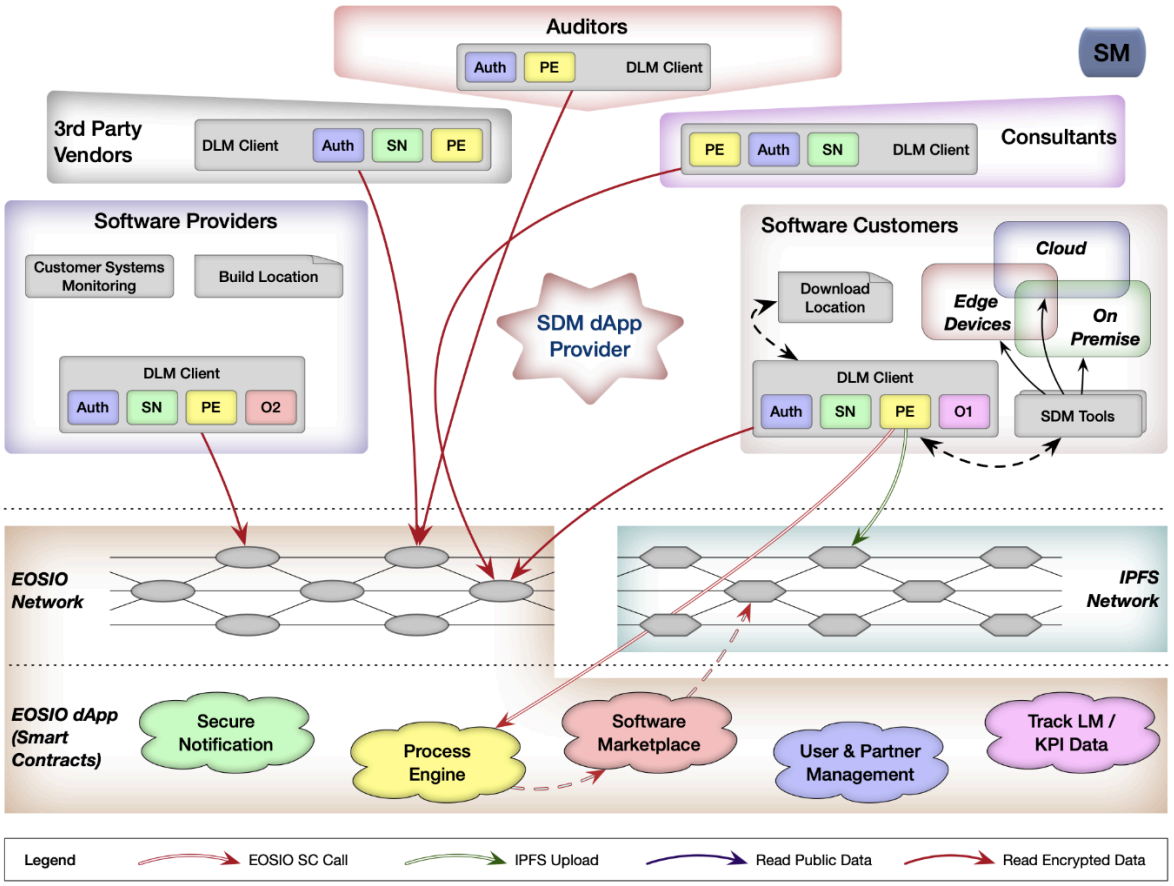

Figure 38 Performing an upgrade

6. The validation of the results of the performed procedure is carried out at different levels. One of them is the automatic verification of the versions of the software components of the client software. The verification is carried out based on the data that is collected periodically by oracle, which shares the important KPI parameters of the client system's operation with the other participants in the procedure.

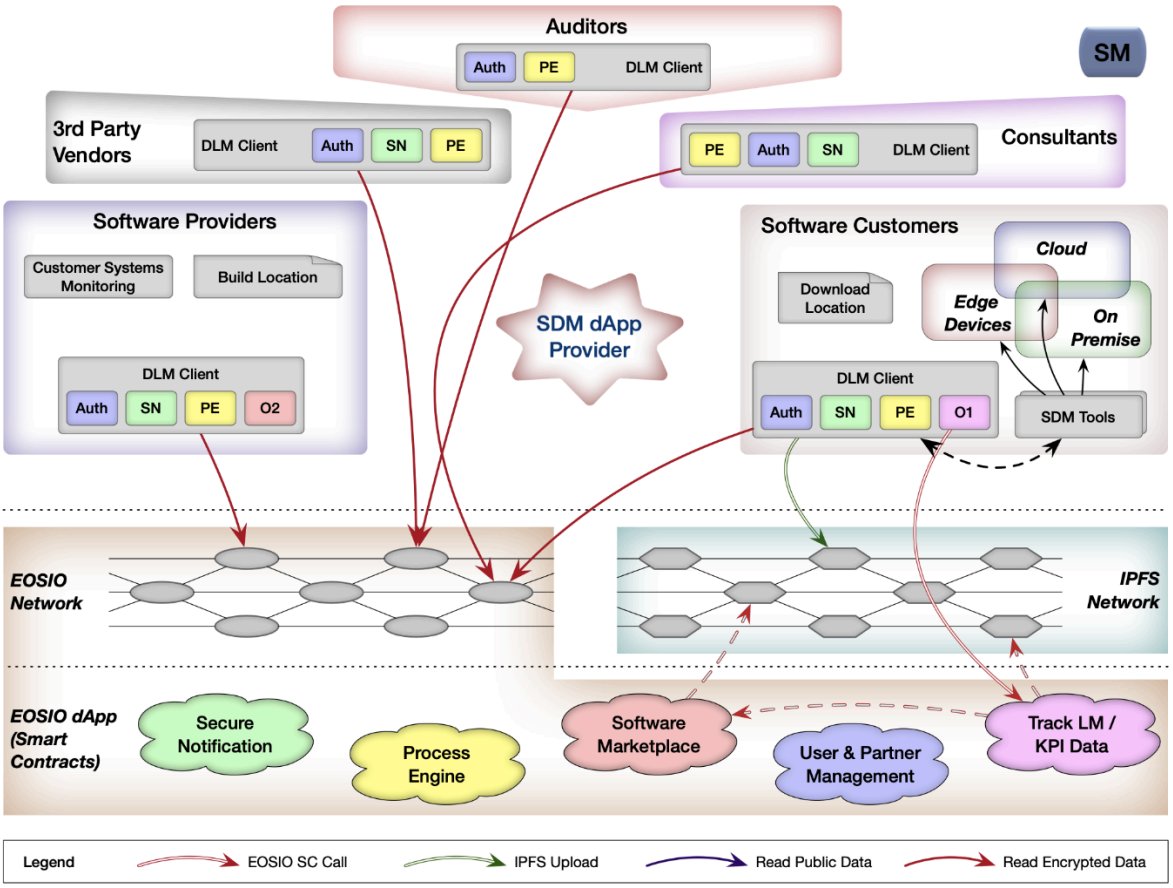

Figure 39 Upgrade validation

Through the five smart contracts and thanks to the key features of EOSIO, all steps of SDM are implemented in a secure and reliable way. A way that provides a quality service with clear responsibilities, predictability of behavior, description and automation of good practices and reliable storage of (historical) information. The next chapter demonstrates in practice how the modules interact and how one of the key SDM scenarios is implemented.

Chapter 4. Publicly verifiable RNG—use case and prototype

Numerous contemporary IT systems depend on the proper creation of secure random and pseudo-random numbers. A critical challenge in this process is ensuring that external entities can independently verify the integrity of these values, guarding against potential manipulation by any participating party. Blockchain-powered smart contract platforms, which have gained widespread adoption across industries, offer a promising solution by enabling the generation of random numbers with full public verifiability. One key aspect of improving the reliability and security of the generated numbers is by adding user-contributed entropy.

4.1. Random and pseudo-random number generation

The generation of secure random and pseudo-random numbers plays a fundamental role in contemporary information systems and forms a critical foundation of modern security infrastructure. Beyond the digital domain, numerous non-IT applications across various sectors also depend heavily on high-quality randomness to support fair, unbiased, and statistically valid processes. Notable examples include:

- **Juror Selection**: Legal systems utilize random number generation to impartially select jurors from a pool of eligible candidates, thereby ensuring objectivity in the judicial process.
- **Lotteries and Gambling**: Randomness is essential in gambling and lottery systems to guarantee fairness, unpredictability, and resistance to manipulation.
- **Statistical Sampling**: In disciplines such as social science, economics, and marketing, random sampling ensures that data collected from populations are representative and unbiased, enhancing the validity of statistical inferences.
- **Randomized Clinical Trials**: In biomedical research, random allocation of subjects to treatment and control groups is vital for eliminating selection bias and ensuring the reliability of trial outcomes.
- **Random Inspections**: Quality assurance protocols in manufacturing frequently use random sampling methods to select products for inspection, thus ensuring equitable evaluation and reducing systematic bias.

- **Randomized Presentation Order**: In academic and professional settings, randomization of presentation order helps mitigate perceived favoritism and preserves procedural neutrality.
- **Random Drug Testing**: To promote fairness and deterrence, organizations apply random selection in the administration of drug tests, preventing targeting and ensuring equal probability of selection.

In all these applications, the integrity of the process is strongly dependent on the quality of the random numbers used. High-entropy, unbiased randomness minimizes systemic bias and enhances trustworthiness, ensuring that outcomes are not subject to prediction or external influence. Reliable random number generation, therefore, underpins objectivity and fairness in both digital and real-world systems.

The significance of the outcome associated with random number generation is directly correlated with the incentive for participating entities to manipulate the generation process in pursuit of advantageous results. In high-stakes scenarios, this creates a heightened risk of intentional bias. Empirical analyses in certain public sectors have revealed patterns in purportedly "random" selections that are statistically implausible, strongly suggesting that the integrity of the randomization process has been compromised.

To preserve the authenticity of random selection, it is advisable to adopt methods that are transparent, independently verifiable, and based on publicly available algorithms and data sources. In this context, Distributed Ledger Technologies (DLT), such as blockchains, offer a promising framework. Their inherent decentralization and auditability make them well-suited for ensuring that randomness remains untampered and verifiable by all stakeholders. The next chapter is dedicated to discussing the processes involved in publicly verifiable random number generation implemented with blockchain technology.

4.2. Publicly Verifiable Random Number Generation via Public Blockchain Technology

Leveraging blockchain technology for the generation of publicly verifiable random numbers offers a transparent, tamper-resistant mechanism that fosters trust and integrity in systems

requiring unbiased randomness. The following outlines the essential components and operational methodology for implementing such a system:

- **Blockchain Infrastructure**: The process begins by selecting an appropriate blockchain network (either public or consortium-based) such as Ethereum, EOS, or a purpose-built platform. The selection is informed by considerations including scalability, transaction costs, and the maturity of developer and user communities.
- **Smart Contract Deployment**: A smart contract is developed and deployed on the chosen blockchain. This contract encapsulates the logic for random number generation (RNG), including functions for seed ingestion, number generation, and result dissemination. The contract must ensure transparency and immutability of the generation process.
- **Entropy Source (Seed Generation)**: Secure random number generation necessitates a reliable entropy source. Potential seed inputs include blockchain-native data (e.g., block hashes, timestamps) or external randomness via decentralized oracles. The robustness and unpredictability of the seed source are paramount, as the integrity of the entire RNG process is contingent on seed quality.
- **Randomness Generation Techniques**: Seeds are transformed into random values using methods such as hash-based derivation (e.g., hashing the previous block), commit-reveal protocols, or oracle-sourced entropy. These mechanisms must be deterministic yet unpredictable prior to execution, ensuring public verifiability.
- **Commitment Phase**: The initially generated random number is cryptographically committed to the blockchain using techniques such as hash commitments. This phase conceals the actual value while guaranteeing that it cannot be altered post hoc.
- **Revelation Phase**: Upon meeting predefined conditions (e.g., reaching a certain block height), the original seed and corresponding random number are disclosed. This step allows independent validation of the result's authenticity and linkage to the original commitment.
- **Verification Process**: All participants can audit the process by verifying that the revealed seed matches the commitment and that the random number was correctly derived using the prescribed algorithm. This openness ensures procedural integrity and fosters user trust.

- **Use Cases**: Such verifiable randomness mechanisms have broad applicability in domains such as online gaming, lotteries, cryptographic key generation, decentralized applications (dApps), and other systems where impartiality is critical.
- **Security Considerations**: To prevent manipulation, adversarial interference, or collusion, multi-party schemes (e.g., distributed commit-reveal) and trusted external randomness providers may be integrated. Proper governance and cryptographic safeguards must be in place to enhance resilience.
- **Cost and Performance Trade-offs**: Blockchain-based RNG systems incur operational expenses due to transaction fees and computational requirements. System designers must balance decentralization, security, and scalability with cost-efficiency when architecting such solutions.

Incorporating Blockhashes and External Entropy in Blockchain-Based Random Number Generation

1. Use of Blockhashes as Entropy Sources for RNG

In blockchain-based environments, blockhashes are frequently employed as sources of entropy to facilitate random number generation (RNG) within smart contracts and decentralized applications (DApps). This method leverages intrinsic properties of the blockchain to achieve determinism and partial unpredictability.

- **Blockhash Definition**: A blockhash is a cryptographic hash derived from a block's header, uniquely identifying each block in a blockchain. It is generated as part of the block creation process and reflects the state of the blockchain at a given time.
- **Block Selection Strategy**: RNG algorithms may define a selection rule for which block’s hash to use. Options include the latest block, a fixed historical block, or a future block yet to be mined. Future blockhashes are particularly valuable because they are not known in advance and are less susceptible to manipulation by current participants (e.g., miners or validators).
- **Hashing Process**: The selected blockhash is input into a secure hash function (most often SHA-256) to derive an intermediate cryptographic value, increasing resistance to pattern prediction or reverse engineering.

- **Number Generation**: The hash output is mapped to a desired numerical range through modular arithmetic, truncation, or other deterministic methods, thereby producing a usable pseudo-random value within specified application constraints.

2. Integration of User-Contributed Entropy and Hash Commitments

To enhance randomness and trustworthiness, external parties may also contribute entropy to the RNG process. This additional input can be securely integrated using **cryptographic hash commitments**, a technique that allows a party to commit to a value without revealing it, while enabling verification at a later stage.

- **Random Number Generation**: A cryptographically secure pseudo-random number generator (CSPRNG) is used to produce a random value. This process may occur off-chain or within a secure hardware enclave.
- **Inclusion of Contextual Metadata**: To bind the generated number to specific operational circumstances, metadata such as timestamps and contextual identifiers (e.g., user ID, transaction ID) are appended to the value prior to commitment.
- **Commitment Creation**: The random number and associated metadata are concatenated into a single message, which is then hashed using a secure cryptographic hash function (e.g., SHA-256) to produce the commitment.
- **Public Disclosure**: The resulting hash commitment is published on a tamper-proof, publicly accessible ledger—such as a blockchain, decentralized database, or timestamping service along with relevant metadata. This step ensures immutability and public verifiability.
- **Reveal and Verification**: At a later point, the original random value and metadata are disclosed. Anyone can recompute the hash of the revealed data and compare it against the previously published commitment. A match confirms that the revealed value is authentic and has not been altered since the initial commitment.

This mechanism ensures that the randomness remains concealed until a designated point in time, while providing a verifiable audit trail. It introduces transparency and strengthens trust in the fairness of the RNG process.

3. Operational Considerations

The cost of executing the hash commitment and reveal operations on-chain can vary depending on the blockchain platform. In architectures like EOSIO, both centralized and decentralized models for covering these costs are supported—either the central authority or the external participants may bear the transaction fees. Such flexibility allows system designers to balance usability, decentralization, and economic feasibility.

Mitigating Strategic Withholding in Hash Commitment-Based RNG Systems

A significant vulnerability in random number generation schemes that rely on external entropy via hash commitments arises when a contributing party withholds the revelation of their committed value after the initialization phase. This deliberate non-disclosure can obstruct the RNG process entirely, rendering the final result unattainable. Importantly, such an action cannot be treated as a neutral failure or ignored, as it may represent a strategic maneuver. By retaining the option to either reveal or withhold their committed seed, the participant effectively gains discretionary influence over the final output, undermining the integrity and fairness of the RNG system.

To discourage such behavior, additional protocol-level safeguards must be introduced—particularly in competitive two-party scenarios where each party has a vested interest in the outcome. The following compensation mechanism is proposed:

- **Dual-Party Commitment and Verification**: When both parties choose to contribute entropy to the RNG process, they may first submit a list of preferred outcomes (e.g., acceptable selections or lottery results). Upon selection, each party is required to reveal their original entropy source (seed), which must be verifiably linked to their previously submitted hash commitment. This process ensures accountability and traceability.
- **Fallback Procedure Upon Non-Disclosure**: If one party fails to reveal their seed within a defined timeframe, the standard generation process is halted. As a contingency, a secondary round of number generation is executed using only the favorable RNG outcomes provided by the compliant party. Given that the parties are

in direct competition, this fallback outcome is expected to be statistically less favorable for the non-cooperative party. This design introduces a disincentive for strategic withholding: it is in each party's rational self-interest to complete the full protocol—even when the expected result may not align with their preferences.

However, this compensation strategy becomes ineffective in multi-party scenarios. When more than two entities are involved, the potential for collusion increases. A subset of parties may coordinate their actions—either by collectively withholding their entropy or selectively revealing inputs—to bias the final output. In such environments, the simple two-party fallback approach is insufficient to preserve fairness, and more robust cryptographic mechanisms (e.g., threshold commitments, multi-party computation, or verifiable delay functions) may be required to mitigate the risks of manipulation and collusion. The subsequent chapter will thoroughly examine the secure application of blockhashes as a component of RNG seed formulation.

4.3 Secure Use of Blockhash as part of the RNG seed

The proposed method involves the utilization of EOSIO blockhashes, which are cryptographic hashes derived from a block’s header. The header of each block contains the hash of the preceding block, creating a secure chain of blockhashes that ensures the blockchain’s immutability, preventing retroactive alterations.

The procedure consists of three distinct phases:

1. **Preparation** (optional, online or offline): This phase may be performed offline, online, or through a combination of both. For instance, a list of potential judges for a case could be digitally signed and prepared offline. This file may be uploaded to publicly accessible storage for transparency. Regardless of its availability, the digital signature must be included in the initialization phase to guarantee the authenticity of the selection process. If hash commitments are part of the RNG procedure, all committed hashes must be uploaded or referenced on the blockchain before the initialization phase to ensure they are accounted for.

2. **Initialization** (online): This step marks the commencement of the selection or RNG process and requires interaction with the EOSIO blockchain. It should include the following components:
    - Authorization of the initiating party
    - References to all hash commitments utilized in the RNG seed calculation
    - References to any relevant offline data that identifies the selected entities without compromising sensitive or personal information (e.g., utilizing distributed file systems such as IPFS)
    - Signatures of documents related to the selection process (either stored offline or on a centralized/distributed file system)
    - Information about the random data's type, quantity, and intended use
    - Details of the RNG algorithm employed, including any limitations
3. **Post-Processing** (optional, offline or online): Post-processing is not mandatory if no hash commitment is involved. The utility of online post-processing depends on the selection method used. Methods based solely on blockhashes as seeds typically do not require post-processing. However, summarizing the RNG results in this phase enhances transparency and facilitates verification. If hash commitments are involved, post-processing entails revealing previously generated random numbers (matching the committed hashes) in a timely manner to finalize the random number calculation. The associated costs of these steps must be agreed upon by the participants.

Considerations for the Chosen Method

Several important factors must be considered when utilizing blockhashes as a seed for Random Number Generation (RNG):

- **Determinism**: Since blockhashes are derived from blockchain headers, and blockchains themselves are deterministic, the level of randomness is inherently limited by the entropy present in the block data. Miners or block producers can influence this data to some extent, meaning it may not be entirely unpredictable.
- **Security**: Although blockhashes can introduce a degree of randomness, they should be applied with caution in cryptographic applications where high levels of unpredictability and security are required. For such purposes, it may be necessary to integrate external randomness sources or utilize more advanced RNG mechanisms.

- **Operational Costs**: Using blockhashes for RNG typically involves operational costs, particularly in blockchain transactions. These costs should be considered, especially in high traffic blockchain environments.
- **Validation**: It is crucial to ensure that the code responsible for selecting and processing blockhashes is transparent and verifiable, allowing independent verification of the randomness in the RNG process.

Our next chapter is dedicated to a comprehensive examination of seed selection methods that utilize blockchain blockhashes.

4.4 Methods for Choosing the Seed Using Blockchain Blockhashes

The proposed method involves the utilization of EOSIO blockhashes, which are cryptographic hashes derived from a block's header. The header of each block contains the hash of the preceding block; several methods exist for selecting the seed for RNG based on EOSIO blockhashes:

1. **Using Current Blockhash**
   Given that blockchain functions are deterministic, always yielding results based on predefined inputs and blockchain state, one straightforward method is to use the hash of the current or previous block as the RNG seed. This approach is simple and efficient, as the hash of the current or last block serves as entropy for the pseudo-random algorithm. However, its primary drawback is that if just one block producer or miner is compromised, they could manipulate the RNG result to favor a specific outcome. This method is insecure without additional guarantees, but it can be made more reliable by incorporating hash commitments to ensure the integrity of the seed.
2. **Using Fixed Delay Blockhash**
   To address the risk of manipulation in the "current/last blockhash" method, a fixed delay can be introduced before selecting the blockhash. The delay should be long enough (e.g., longer than 63 seconds) to ensure that multiple block producers generate a block during this time. This method ensures that no single block producer can manipulate the blockhash, as it involves multiple participants. While simple and

predictable, this method has a potential vulnerability if a small number of block producers work together to manipulate the result. In such cases, the risk of manipulation increases as more block producers synchronize their actions. This method remains effective if the stakes are low enough to deter manipulation by a larger group of block producers.

3. **Using Blockhash with Random Delay**
   To mitigate the risk of manipulation by coordinated block producers, the selection of the blockhash can be made dependent on a random delay. For instance, the target blockhash could be the one generated after 65 even-numbered blocks following the initialization call. The chance of manipulation is significantly reduced, as the final blockhash is less predictable. This method is more reliable, especially when used in conjunction with other randomness guarantees.
4. **Using Blockhash to Collect Randomness During Delay**
   Another variation involves selecting not just one, but multiple blockhashes to increase the randomness. For example, instead of choosing a blockhash to represent the selection, multiple blockhashes could be used, progressively narrowing down the selection or removing options with each new blockhash. This method benefits from increased robustness, as even if a single block producer is honest, the overall selection remains unpredictable. However, this method is more complex to implement and track, and it may require post-processing to summarize the results for transparency.
5. **Using Blockhash and Hash Commitment**
   The most secure method for generating a seed involves combining blockhashes with hash commitments. In this approach, multiple parties contribute entropy via random numbers, which are cryptographically committed prior to the RNG process. The final seed is generated as a hash value of concatenating the blockhash and the committed random numbers. This method is particularly robust when several parties are competing for the selection process, as it ensures transparency and integrity by involving multiple contributors to the randomness.

A comparative summary of the complexity, speed and security of all methods discussed in this chapter is presented in Table 3.

Seed Choosing Method	Complexity	Speed	Security: Mid Stakes	Security: High Stakes
Blockhash: Next Block	Low	Instant	High	Low
Blockhash: Fixed Delay	Low	Fast	High	Low
Blockhash: Random Delay	Low	Fast	High	Medium
Blockhash: Collect Entropy	Medium	Fast	High	Medium
Blockhash + User Entropy	High	Slow	High	High

Table 3 Comparison of seed selection methods

4.5 Choosing DLT for the provable RNG prototype – EOSIO / Antelope / Vaulta

The distributed ledger technology (DLT) used in this prototype has a specific lineage that is important to clarify. The system currently known as **Vaulta** emerged through successive iterations of earlier platforms, namely **EOSIO** and **Antelope**. Each step in this evolution has influenced the technical properties, governance structure, and application domains of the present network.

EOSIO was originally released in 2018 by Block.one as a high-performance blockchain software framework. It introduced a Delegated Proof-of-Stake (DPoS) consensus mechanism, sub-second block times, and a system of account-based permissions and resource staking. These characteristics were designed to support general-purpose smart contracts and decentralized applications at significantly higher throughput than first-generation platforms such as Ethereum. However, the development of EOSIO as a codebase slowed after 2021, leaving many networks that had adopted it without an active steward.

To ensure continuity, community actors including the EOS Network Foundation (ENF), WAX, Telos, and UX Network initiated a coordinated fork of EOSIO in 2022, which was branded **Antelope**. Antelope is best understood as both a direct successor and an open development framework. It preserves the architectural features of EOSIO, most notably the DPoS model and resource-oriented execution environment, while adding performance improvements, security enhancements, and modernized tooling. In this respect, Antelope

became the primary reference implementation for EOSIO-derived blockchains and remains under active community governance.

Vaulta represents the most recent step in this trajectory. In 2025, the EOS Mainnet underwent a rebranding and strategic repositioning to focus on financial applications under the name Vaulta. Technically, Vaulta remains an Antelope-based chain, thus inheriting the low-latency block production, account system, and smart-contract model of its predecessors. Its distinctiveness lies in its institutional orientation: Vaulta emphasizes Web3 banking, cross-border settlement, and tokenized assets. The migration from EOS tokens to Vaulta's native asset was conducted at a one-to-one ratio, ensuring continuity of value while re-aligning the ecosystem with its new purpose. In parallel, the network has pursued partnerships with payment providers and compliance-oriented service firms to reinforce its positioning as a platform for regulated finance.

From a comparative standpoint, **EOSIO** can be viewed as the legacy framework, historically significant but no longer maintained; **Antelope** is the living, open-source foundation on which multiple networks continue to innovate; and **Vaulta** is a concrete implementation of Antelope tailored toward financial services. For the purposes of this work, the prototype for provable random number generation is deployed on Vaulta, yet the technical underpinnings derive directly from the shared Antelope/EOSIO lineage.

4.6 Boundary conditions for the prototype

There are different aspects for the practical use of a distributer application based on DLT. Some of the most important parameters are the price (TCO), speed, reliability and transparency.

1. **Price** for using the proposed provable RNG solution. All distributed applications spend resources for storing data and for using the DLT as a smart contract platform. Reading data from DLT is generally a free operation. To be sustainable any solution will try to minimize the cost of DLT storage and operations while achieving its goals. One way to keep costs low is to move as many of its operations off-line as possible. The prototype is designed to cover only the bare minimum of the operation that are required to generate provable random numbers. The preparation and post-processing steps as described in section 4.3 are completely offline so then do not add to online

costs for using the DLT. The only step that must be online – the RNG Initialization is implemented in a method "**rngstart**" that may be used for any of the first 4 methods described in section 4.4.

2. **Speed** is typically not a defining factor for the quality of the RNG, yet it should not take too long processing time. The four methods that the prototype supports the generation of the random data is almost instant—less than a second for the first method up to a few minutes for the last and most secure method.
3. **Reliability** is important quality of a RNG used in high stake cases. Here the advantage of a Blockchain/DLT is a great advantage sing their "Distributed" property guarantees there are no single points of failure.
4. **Transparency** is another strong point of most Blockchain/DLT-based applications. The data use for the RNG is visible for anyone. The full data used during the RNG process may be validated anonymously and free of charge. This aspect allows multiple parties to validate it independently and guarantee its correctness.

4.7 Prototype description and processing

The prototype is implemented for Vaulta blockchain using the Vaulta Web IDE. The smart contract's name is **get_random** and its main entry point is the **rngstart** action.

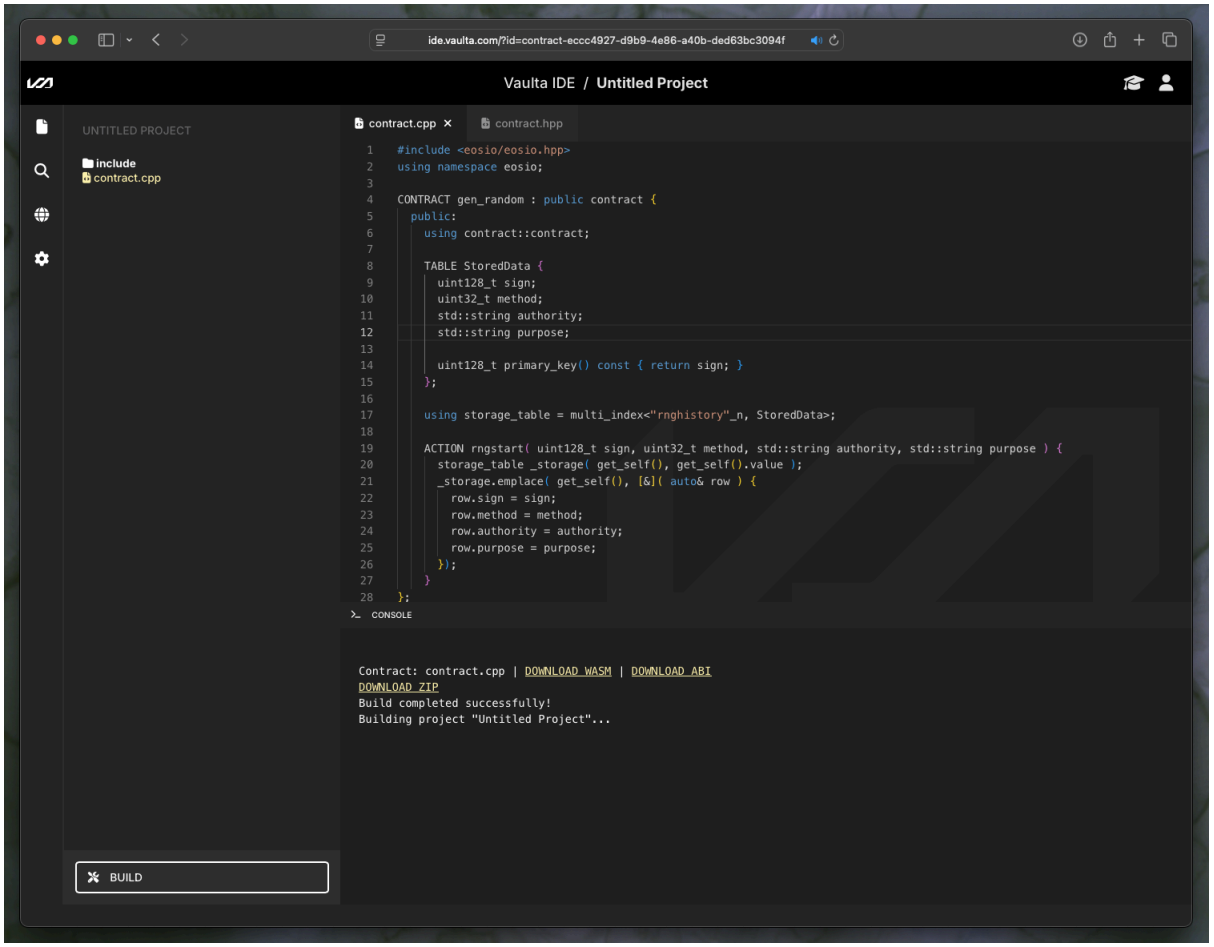

Figure 40 Initial state of the prototype

The **rngstart** action has several parameters in the implemented prototype (Fig 40):

- **sign (uint128_t)** – This is the sole mandatory parameter and serves as a primary identifier for each randomness generation instance. It is the cryptographic digest/signature of the off-chain document that specifies the selection purpose and input data. It may also have other relevant information for the selection process. Because that document may contain personal data subject to regulations (e.g. GDPR), no personal data is written on-chain, only the document's signature is stored. Consequently, for randomness generation the **sign** is used as identifier, which (i) delegated all descriptive details to the referenced document and (ii) provides an authenticity/integrity guarantee for the parties who have access to that document.

- **method (uint32_t)** – This parameter describes the method used to select the blockhash or blockhashes that are used as a seed to generate the random number. If the method is not to be shared it may be described in the document that is identified by **sign**.
- **authority (std::string)** – a description of the organization that is running the random number generation process. It is an optional property that may be used to filter the processes but is not required when the organization does not need to leave trace in the public space.
- **purpose (std::string)** – like **authority** parameter, **purpose** is an optional parameter that may further describe the process within the **authority** structure that use the result of the random number generation.

Using the Prototype. First step after implementation of the prototype is testing it. On Figure 41 we see in Vaulta Wen IDE the option to deploy the contract on real test blockchain. In our case we choose “Jungle Testnet” as one of the original and most popular EOS/Vaulta test blockchains. For deployment of the prototype, we use an account on Jungle Testnet - “dkhsyqa1lmvn”.
As we may see down on the left side of Figure 41 once we successfully build and deploy the smart contract to Jungle Testnet we have the table **rnghistory** created to keep results from all runs of the random number generation.

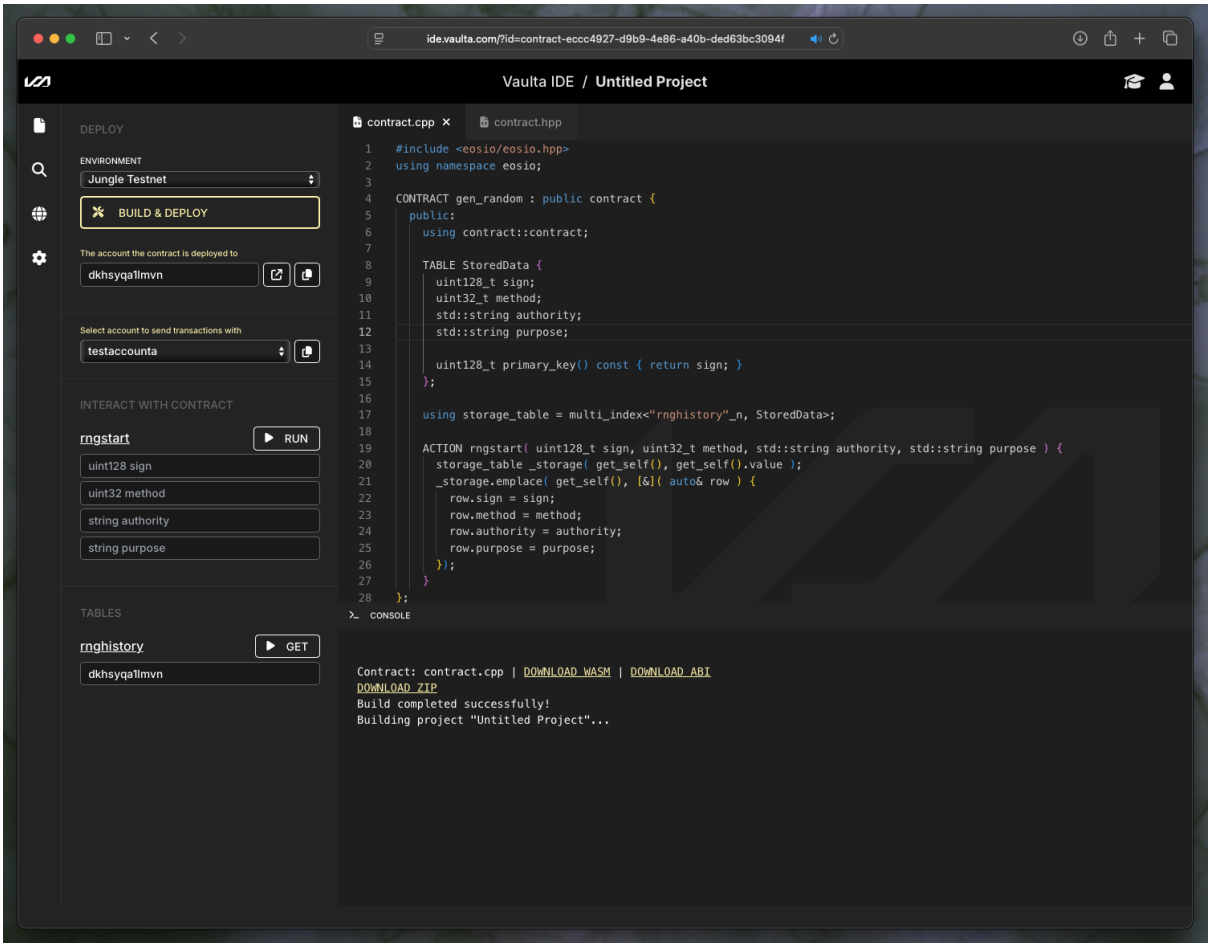

Figure 41 Smart contract is prepared for deployment

Once the deployment is successful, we need to choose an account to test invoking the smart contract methods (Figure 42). In the interface provided by Vaulta Wen IDE we select “testaccounta” as an account to use when calling the contract. This account is only for testing and there is no connection between the account that the contract is deployed with and the account that uses the contract. This way the entity that provides and supports the contract may offer it to the other users.

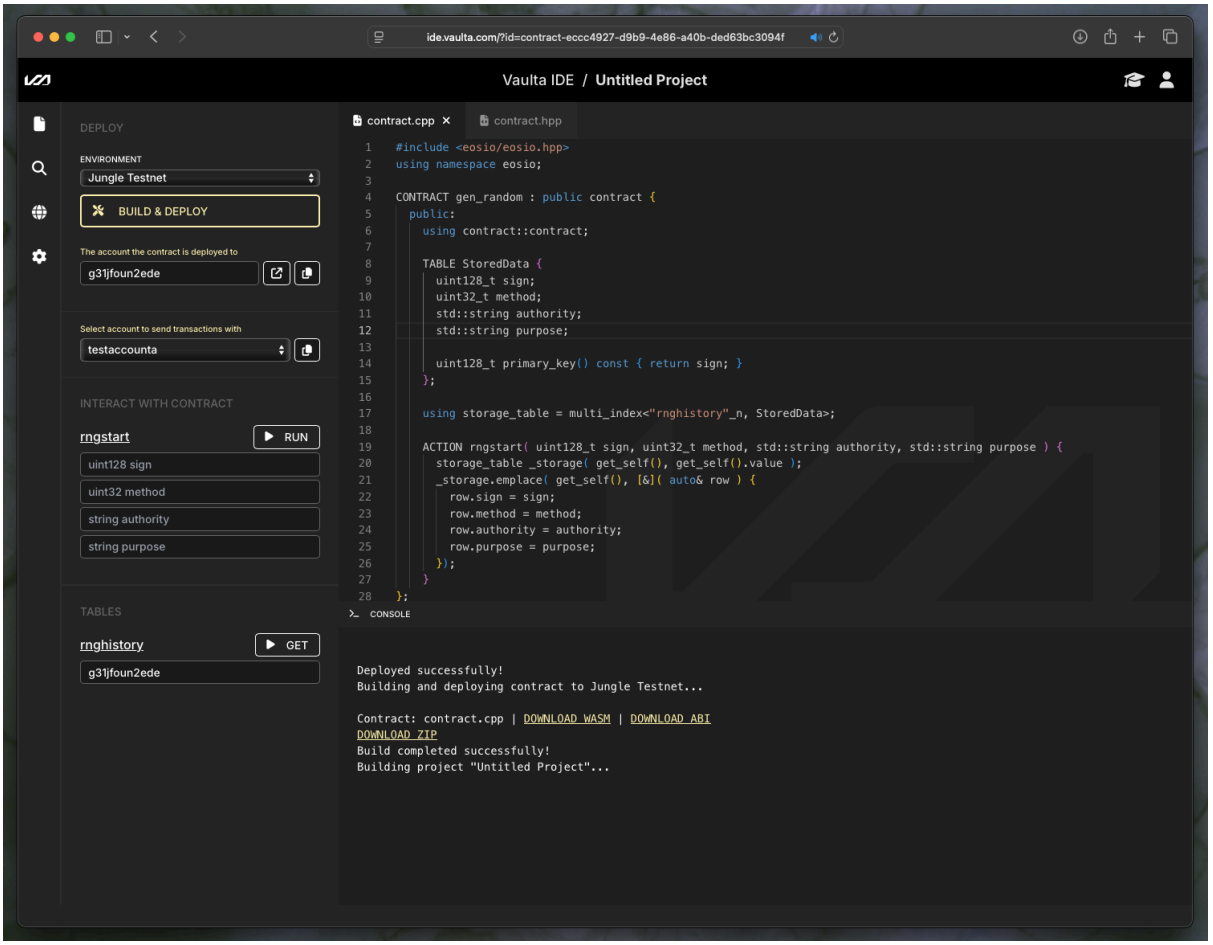

Figure 42 Smart contract is successfully deployed

Next step of testing the smart contract is setting the values pf the parameters in the Vaulta Wen IDE. Here on left-side we see interface to enter values for all parameters of the method **rngstart** (Figure 43). Only the first parameter (**sign**) is required for calling the smart contract. Its value is the digest of the document describing the process of random number generation. Once all parameters are entered the user selects the “Run” button to execute the action and generate the publicly verifiable random number to use as a seed to the determined process of selection. After the invocation the transaction is added to the blockchain and stays there theoretically forever. This way we are sure the same process may never be run again.

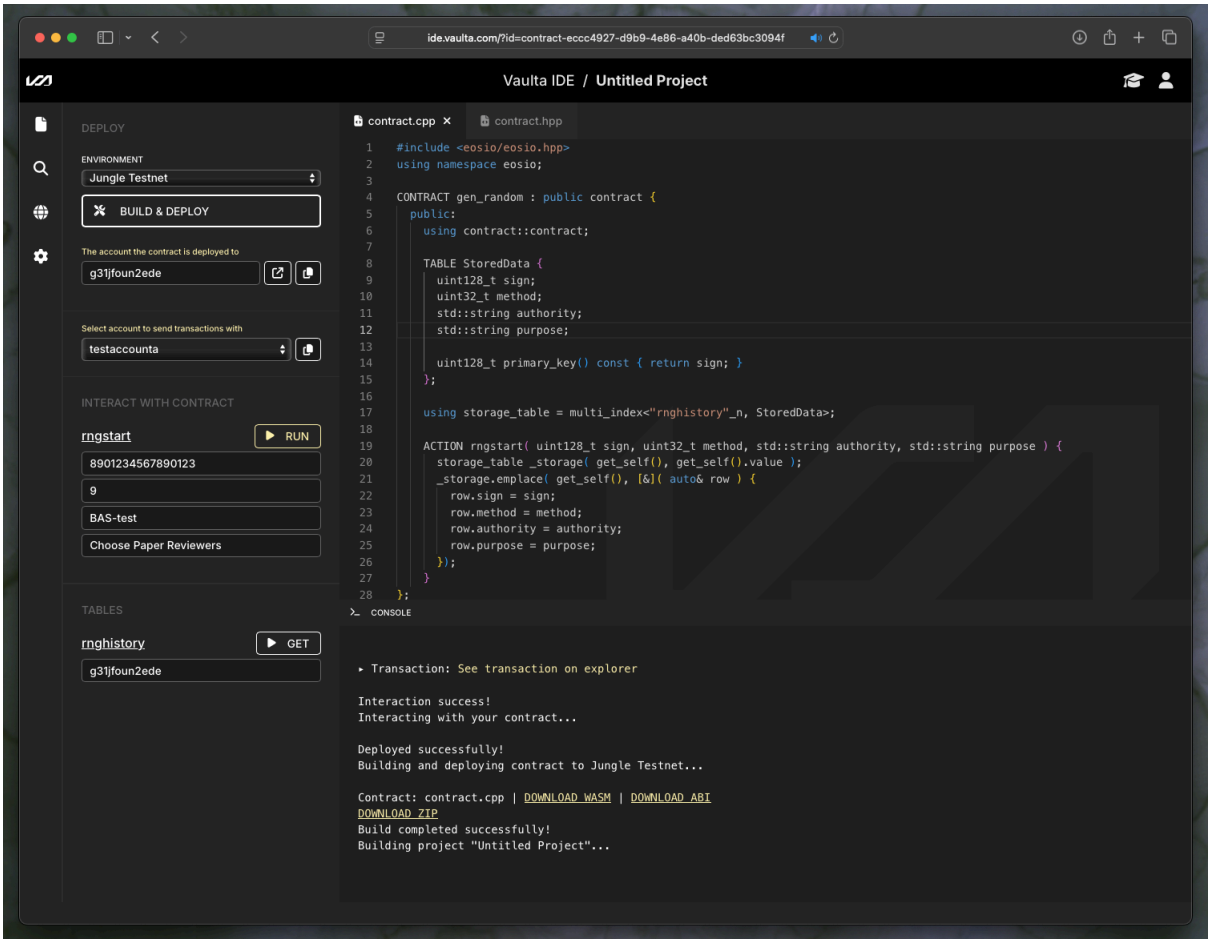

Figure 43 Action rngstart of the smart contract is invoked

Once the transaction is added to the blockchain and become irreversible we may check the value of the blockhash that we will use for seed. First, we need to find the block that the transaction is added to. On Figure 44 we see the information that is publicly visible for the transaction. What we are interested is the **Transaction ID** (for record keeping purposes) and the **blocknumber**. The **blocknumber** of this run of the random number generation is 208481002.

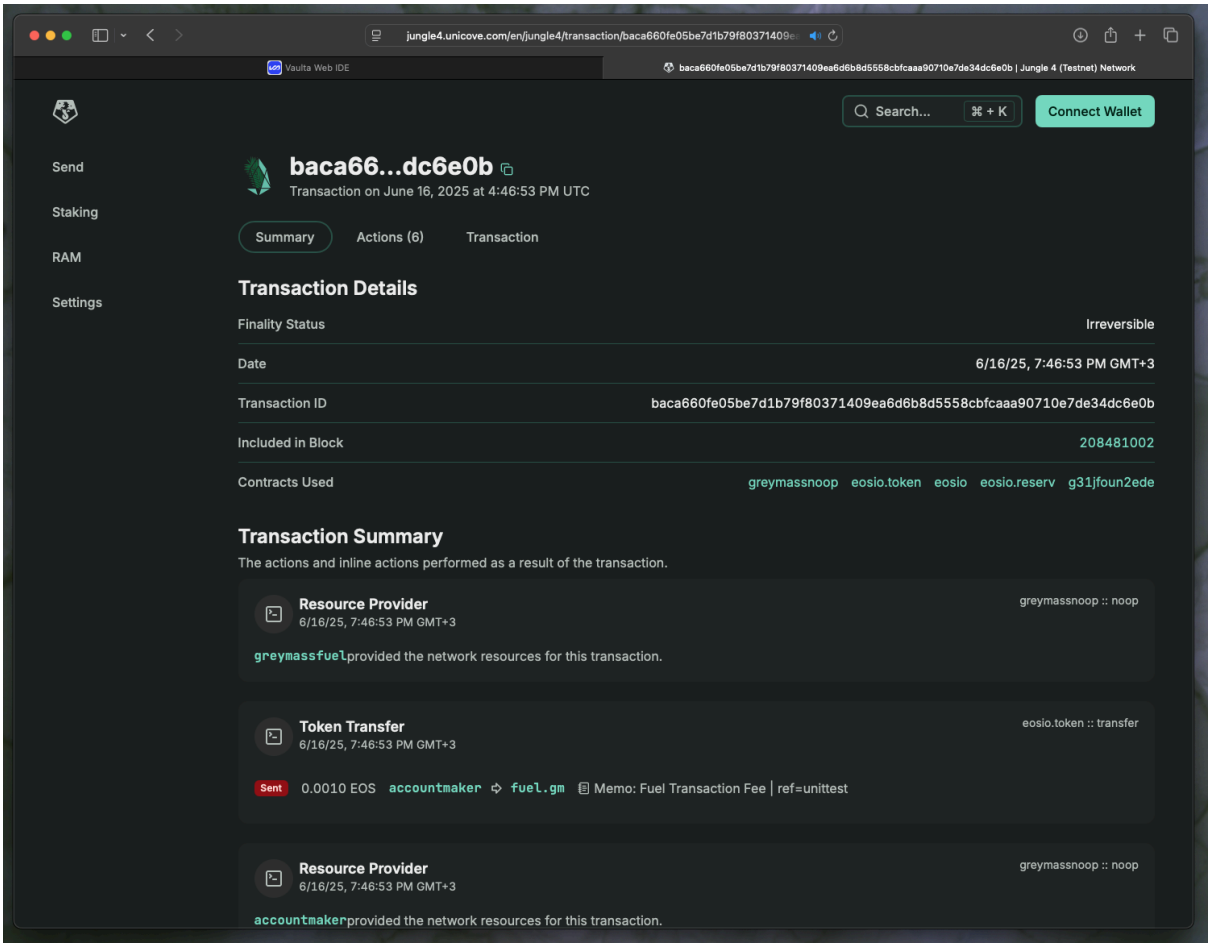

Figure 44 View of the Transaction (top of the page)

To make sure this is exactly the transaction we are interested in we may scroll down to see more info – Figure 45. Here we see that this transaction is invocation of method **rngstart** with parameters that we have set in the Vaulta Web IDE. This confirm that this is the right transaction.

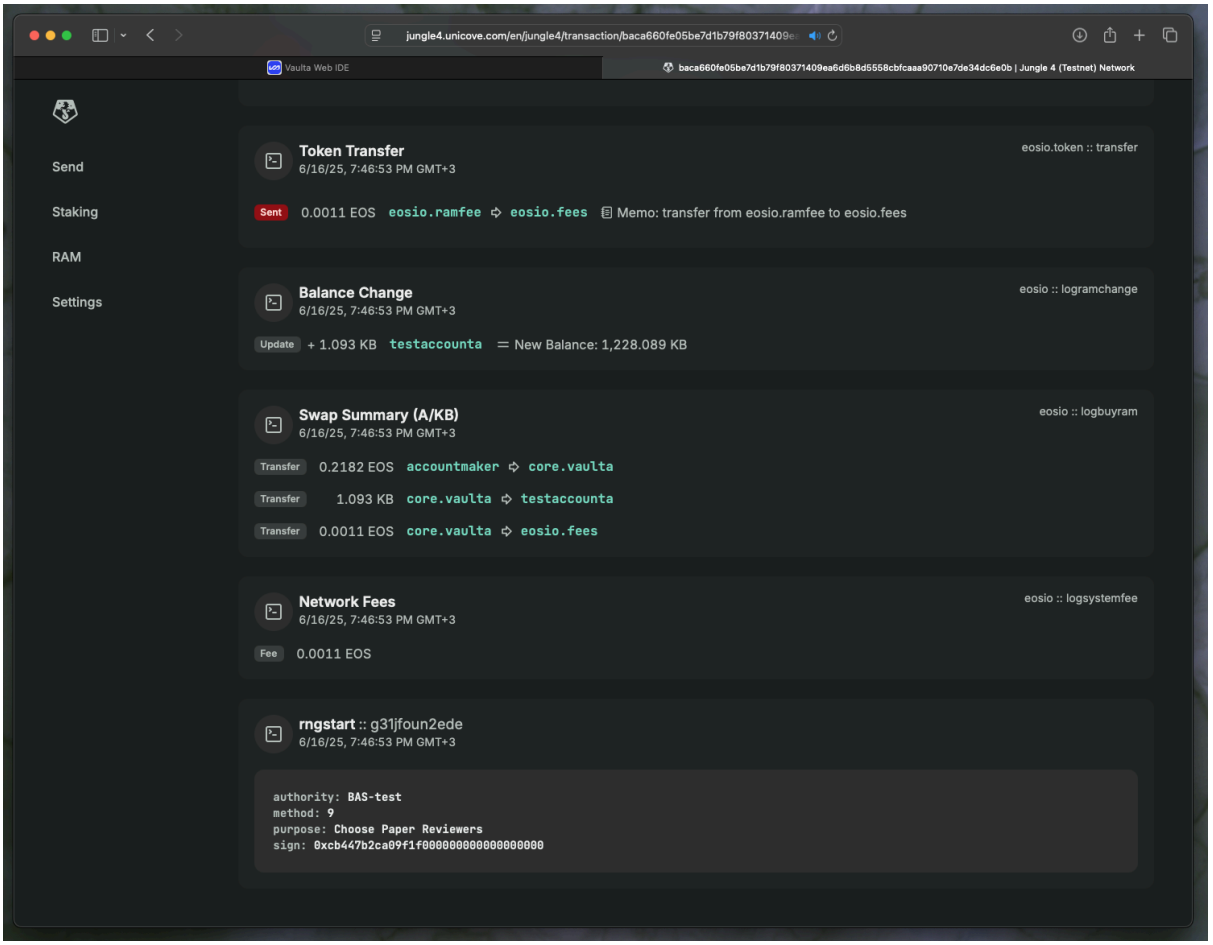

Figure 45 View of the Transaction (bottom of the page)

Next step is to find the blockhash to be used as a RNG seed. For simplicity let's assume we use the current (or next) blockhash as a seed. To find it we may click on the link given in "Included in Block" section and go to the blockchain view (Figure 46). Here in the **Block ID** section, we see that the blockhash is 0c6d2aea5a8e57a3f69bba03de8850b0972c4ed05c6569e99fb54faaa4caee0a. If we use "Current blockhash" method this is the seed, we will use for the RNG. Everyone that is involved in the process and have access to the document that is referenced by **sign** parameter may easily verify it.

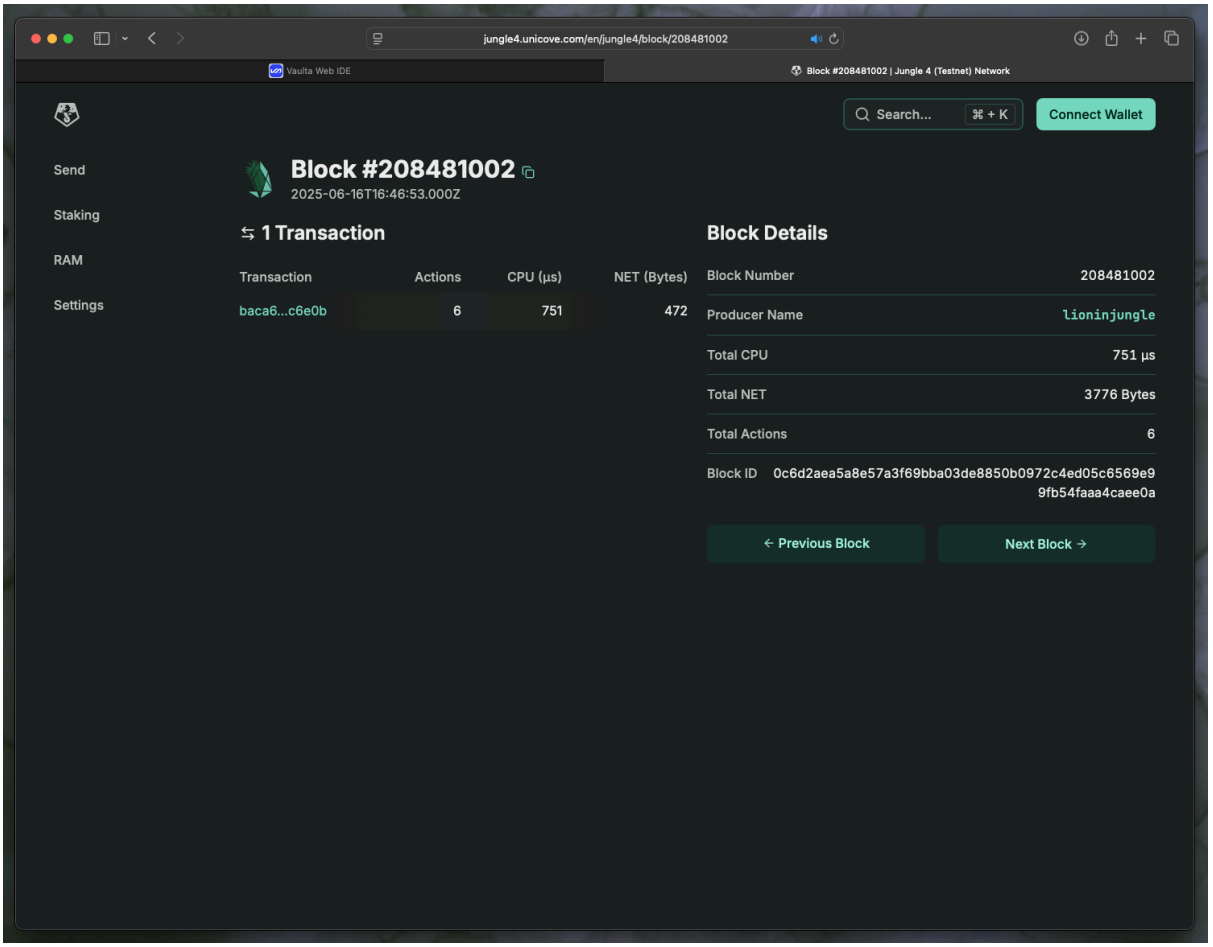

Figure 46 View of the block of the rngstart transaction

And what is we are using the "next blockhash" (or more complicated) method? In this case we would look at the next block – 208481003 (Figure 47). Here we see that the **seed/Block ID/blockhash** is 0c6d2aebec898859b1f4148a2565508f768d7af19ba4c254d59b2f613ff9297a and use it as a seed.

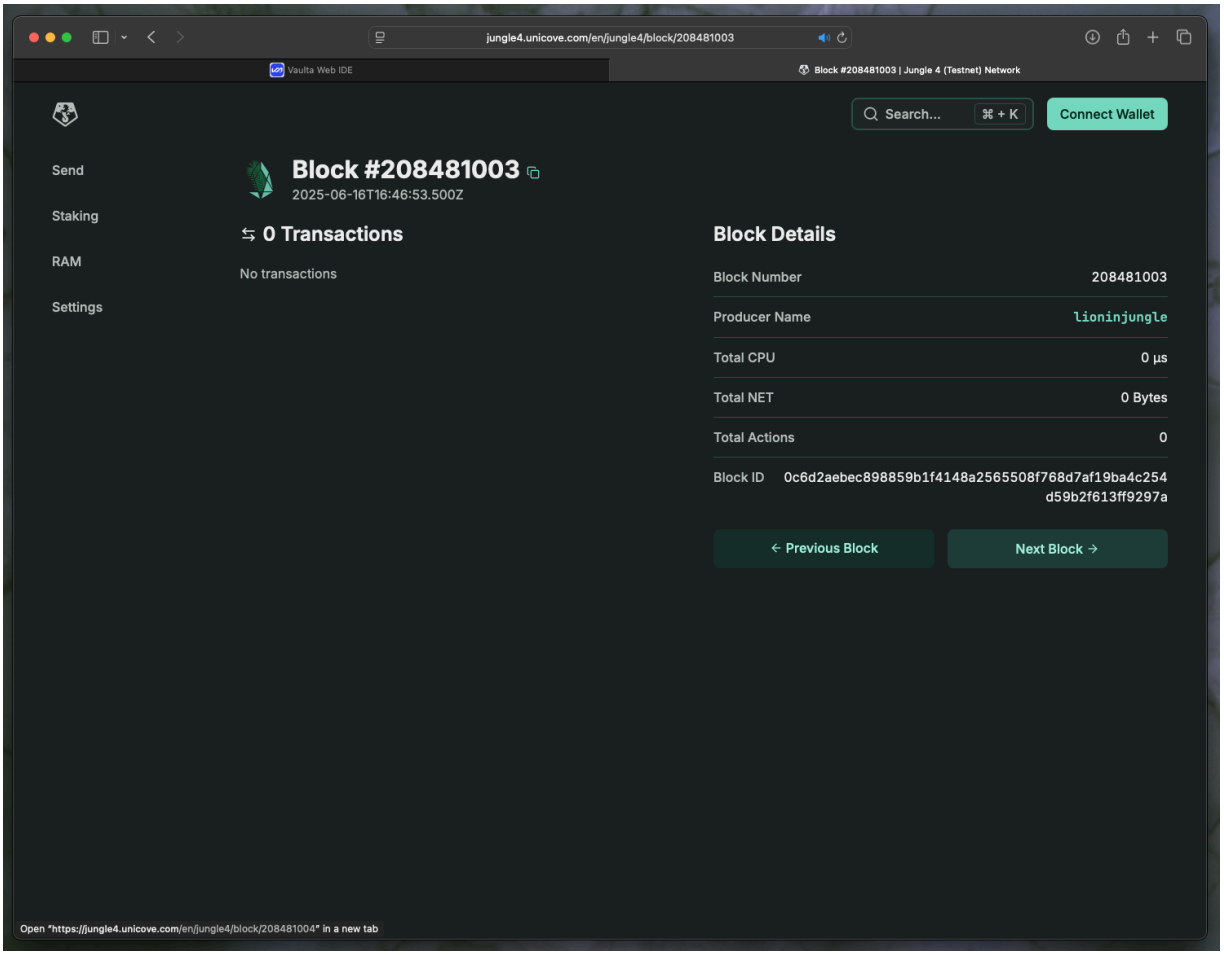

Figure 47 View of the block of the target transaction

Now let's assume we are an auditor that must validate the results got by the RNG process in the past. This the auditor must find the relevant transaction in the **rnghistory** table (Fig. 48). The **Transaction ID** leads to a block and its **Block Number**. This **Block Number** leads to the target **Block Number** and its **Block ID** which is used as a seed.

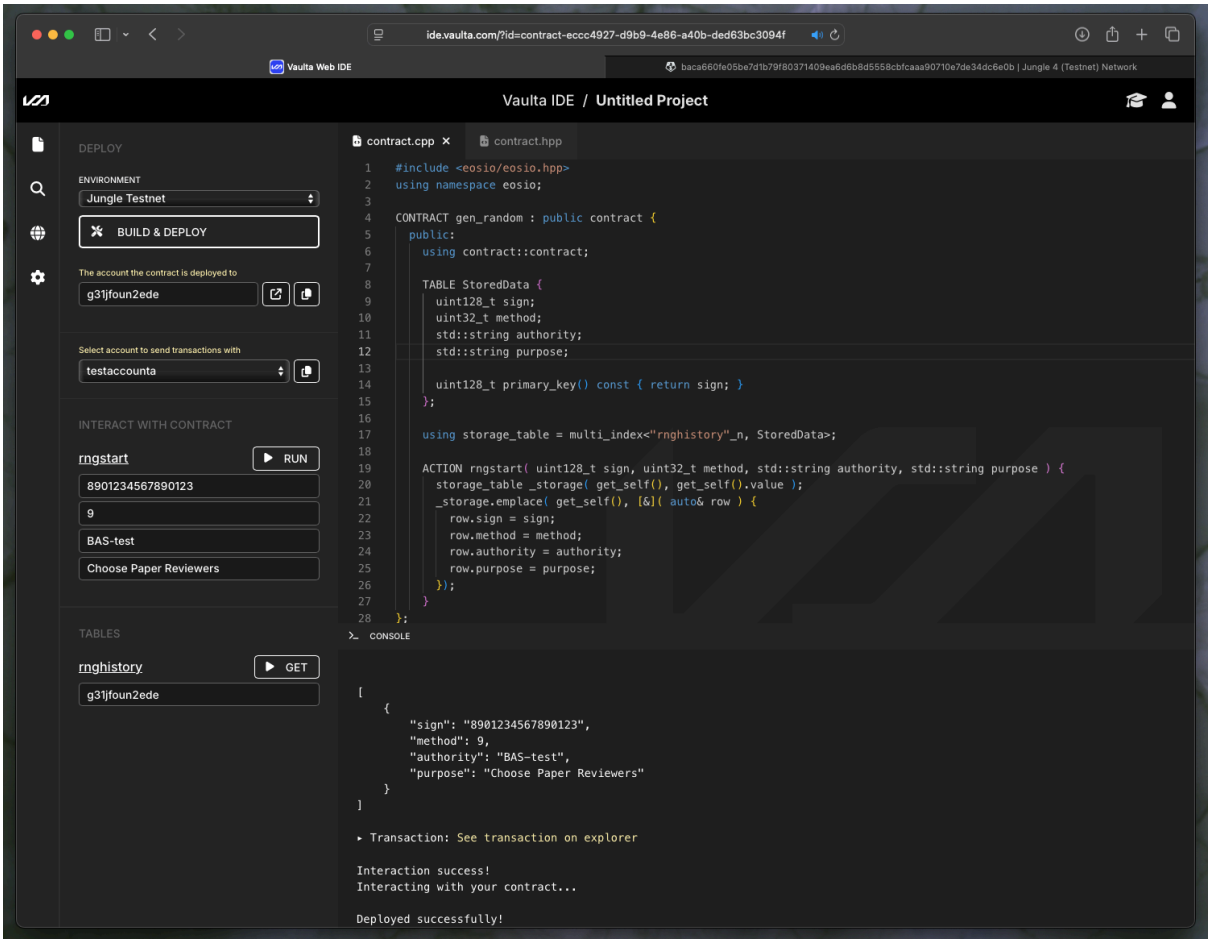

Figure 48 Validation of rnghistory state

Now let's make sure that once the process is run for specific **sign** value no other instances of the RNG process may be ran for the same value. To do so we prepare to run the **rngstart** again but use the same sign value. This may happen if someone is not happy by the result they got and try to re-run it. Once we invoke the **rngstart** using the same sign we get an error explaining that insertion failed due to uniqueness constraint violation. This way we guarantee programmatically that no re-runs may be done once a random number is selected.

As result of testing the prototype we may conclude it works and properly generates provably random numbers using the blockhash as a seed for the random number generator.
This way we validating that using the prototype is that EOSIO blockchain technology is suitable for creation of distributed applications which use the benefits granted by using the blockchain as a smart contract platform. Such application can be very small and efficient while using the benefits granted by EOSIO/Antelope technology by design.

Chapter 5. Approbations and Reports at Scientific Forums

5.1. Approbation of the results

The results of the dissertation have been published in the following articles:

1. Jeliazkov J., Kostadinov H., Using EOSIO Technology for Publicly Verifiable Randomness, Studies in Computational Intelligence, Vol. 641, pp. 87 – 96, 2025, Springer, Scopus, SJR: 0.190
2. Jeliazkov J., Kostadinov H., Decentralized Research Incentivization System, Studies in Computational Intelligence, Vol. 641, pp. 97 – 103, 2025, Springer, Scopus, SJR: 0.190

5.2. Reports at scientific forums

1. Jeliazkov J., Kostadinov H., Incentivizing Research Using DLT-based Smart Contract Platforms, Annual Meeting of the Bulgarian Section of SIAM, 17 - 19.12.2019г.
2. Jeliazkov J., Kostadinov H., Generation of secure public-verifiable random numbers, Annual Meeting of the Bulgarian Section of SIAM, 20 - 22.12.2022г.

Chapter 6. Conclusion

6.1. Scientific contributions of the dissertation

1. An analysis has been carried out of existing solutions for smart contract platforms based on blockchain technology and distributed ledger technologies (DLT).

2. The most popular modern distributed storage technologies have been examined, together with their advantages and weaknesses, when considered as a foundation for a smart contract platform. EOSIO has been selected as the platform for the subsequent applications.

3. A distributed system for incentivizing research and development activity has been described, in which rewards are defined for finding the unique or the best solution to a given problem.

4. A distributed system for managing the lifecycle of enterprise software, using EOSIO technology as a smart contract platform, is described and analyzed.

5. A distributed system has been described for using a set of algorithms for generating verifiably random numbers, built on a smart contract platform based on EOSIO as the underlying blockchain.

6. A prototype running on Vaulta/EOSIO technology for generation of provable random numbers, based on the innovative architecture, has been presented, and the environment in which it operates has been described in detail: interfaces, smart contracts, data models, and communication protocols.

7. The results from the development of the new systems based on smart contract platforms have been summarized, and analyses have been carried out for the different scenarios successfully addressed by the innovative architecture. The advantages have been demonstrated, particularly in the areas of traceability, security, and the definition of responsibilities.

The distributed system for incentivization of research and scientific development based on EOSIO technology is described in Chapter 2. A detailed blockchain architecture is defined that solves the problems of defining and incentivization of finding specific scientific results.

The results of this research are published in Jeliazkov J., Kostadinov H., Decentralized Research Incentivization System, Studies in Computational Intelligence.

The architecture for generation of provably random number generation described with its modules, communication protocols and connections between different distributed modules, has been verified and tested with the prototype described in Chapter 4. The connection between blockchain and IPFS as a storage for the document used to define algorithms and boundaries for generation of provably random number generation is described as well. Serious attention has also been paid to the development environment and the way dapps are written in EOSIO. The results of the architectural design and analysis are published in Jeliazkov J., Kostadinov H., Using EOSIO Technology for Publicly Verifiable Randomness, Studies in Computational Intelligence.

6.2. Summary

We live in a decade marked by the words blockchain and bitcoin. A decade in which cryptocurrency shook up financial markets and changed investment opportunities. A decade that promises that cryptography and blockchain platforms are on their way to eliminating central monopoly institutions and creating a world of democracy in p2p networks, of a fair and reliable process for storing and transferring goods between different participants.

That is why this dissertation is focused on this modern technology. Technology that promises to revolutionize each of the businesses and processes we know today. Technology that makes a request to reduce or eliminate the need for the intervention of highly paid specialists from professions such as a notary or a banker. In which many participants will be able to participate in shared processes, while maintaining the responsibility, ownership and correctness of each participant.

In recent years, various classic businesses have been studied, from finance, through real estate to autonomous cars and insurance, as the topic covered by the dissertation, namely SDLC, is still unexplored. That is why the contribution of this work is unique and contributes to the establishment of blockchain as a new standard in quality and reliable SDM services.

Moreover, the presented blockchain architecture, which solves many problems of data visibility, risk prediction, and defining clear responsibilities, could be expanded and cover other areas of the world around us.

The prototype, which is implemented and described in detail, can serve as a basis for other similar solutions. Smart contracts and communication protocols could change according to the needs of each industry.

The possible development of the studied area and the defined blockchain architecture contains enormous potential for modification in the direction of solving similar problems.

History has shown that when implementing innovation, scientific institutions have always played a leading role, whose function is to investigate, analyze and assist in clarifying the applicability of the new topic in real life. As well as that each study, regardless of its volume and scope, is a step forward in the direction of applicability to facilitate people and businesses to use better and more efficient processes and services.

That is why I would like to thank the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences for the opportunity to be part of their talented team and to make my modest contribution to the establishment of blockchain as a new high-quality standard in revolutionizing classic business processes.

Special thanks also to my colleague Biser Tzvetkov for the fruitful time spent together in discussions and generating ideas. Looking forward to their continued work on the topic and soon to present upgraded developments.

Finally, I would like to thank my scientific supervisor Hristo Kostadinov for his tireless support during my studies, for his invaluable guidance on the stages of implementing scientific activity, for his help in defining goals and demonstrating results, as well as for the method of publishing reports and articles.

Bibliography

1. Imran Bashir (2018), “Mastering blockchain: distributed ledgers, decentralization and smart contracts explained”, Packt Publishing; 2nd Revised edition

2. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

3. Paul A. Tatro (2018) “Blockchain Unchained: The Illustrated Guide to Understanding Blockchain”, Book Counselor LLC

4. Elsden, C., Manohar, A., Briggs, J., Harding, M., Speed, C., Vines, J. “Making sense of blockchain applications: a typology for HCI.” In: CHI 2018. ACM (2018)

5. Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, ‘‘Blockchain challenges and opportunities: A survey,’’ Int. J. Web Grid Services, vol. 14, no. 4, pp. 352–375, 2018

6. Basit Shahzad, Jon Crowcroft, “Trustworthy Electronic Voting Using Adjusted Blockchain Technology.”, IEEE Access 7: 24477-24488

7. Bhardwaj, S., Kaushik, M. “Blockchain—technology to drive the future.” In: Satapathy, S.C., Bhateja, V., Das, S. (eds.) Smart Computing and Informatics. SIST, vol. 78, pp. 263–271. Springer, Singapore (2018).

8. Suhaliana bt Abd Halim, N., Rahman, M.A., Azad, S., Kabir, M.N.” Blockchain security hole: issues and solutions.” In: Saeed, F., Gazem, N., Patnaik, S., Saed Balaid, A.S., Mohammed, F. (eds.) IRICT 2017. LNDECT, vol. 5, pp. 739–746. Springer, Cham (2018).

9. Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An overview of blockchain technology: Architecture consensus and future trends", Proc. IEEE Int. Congr. Big Data (BigData Congr.), pp. 557-564, Jun. 2017.

10. Daniel Drescher (2017) “Blockchain Basics A Non-Technical Introduction in 25 Steps”, ISBN: 978-1-4842-2604-9

11. UNITED STATES NUCLEAR REGULATORY COMMISSION (2002), “System Development and LifeCycle Management (SDLCM) Methodology”, Handbook, Version 2.3

12. R. Turpin, “A progressive software development lifecycle”, Proceedings of ICECCS '96: 2nd IEEE International Conference on Engineering of Complex Computer Systems

13 Gagan Gurung, Rahul Shah, Dhiraj Prasad Jaiswal, “Software Development Life Cycle Models-A Comparative Study”, International Journal of Scientific Research in Computer Science Engineering and Information Technology

14. Shylesh S. (2017), “A Study of Software Development Life Cycle Process Models.” SSRN Electronic Journal

15. Andreas M. Antonopoulos “Mastering Bitcoin”, 2nd Edition, Publisher(s): O'Reilly Media, Inc.

16. Sudhan, A., & Nene, M. J. (2018). “Peer Selection Techniques for Enhanced Transaction Propagation in Bitcoin Peer-to-Peer Network”, 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)

17. Matthew Underhill, “The Bitcoin Book: A Beginner's Guide to the Future of Finance”, Independently published (September 21, 2020)

18. Zhu, F., Chen, W., Wang, Y., Lin, P., Li, T., Cao, X., & Yuan, L. (2017). “Trust your wallet: A new online wallet architecture for Bitcoin”, 2017 International Conference on Progress in Informatics and Computing (PIC).

19. D. Larimer, EOS.IO Technical White Paper

20. EOSIO resource, https://eos.io

21. Zheng, W., Zheng, Z., Dai, H.-N., Chen, X., & Zheng, P. (2021). “XBlock-EOS: Extracting and exploring blockchain data from EOSIO”, Information Processing & Management, 58(3), 102477.

22. IPFS resource, https://ipsf.io

23. Jianjun, S., Ming, L., & Jingang, M. (2020), “Research and application of data sharing platform integrating Ethereum and IPFs Technology”, 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES).

24. Guidi, B., Michienzi, A., & Ricci, L. (2021), “Data Persistence in Decentralized Social Applications: The IPFS approach”, 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC).

25. Buterin V.: A Next-Generation Smart Contract and Decentralized Application Platform (2013). https://github.com/ethereum/wiki/wiki/White-Paper

26. Canessane, R. A., Srinivasan, N., Beuria, A., Singh, A., & Kumar, B. M. (2019), “Decentralised Applications Using Ethereum Blockchain”, 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)

27. C. Dannen, “Introducing Ethereum and Solidity”, Berkeley, CA:Apress, 2017.

28. David Lee Chaum, “Computer Systems Established, Maintained and Trusted by Mutually Suspicious Groups”, University of California, Berkeley, 1982

29. David Lee Chaum, Blind Signatures for Untraceable Payments

30. S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, 1991.

31. Ralph C. Merkle, "A Digital Signature Based on a Conventional Encryption Function." 1998

32. Cynthia Dwork and Noni Naor (1993). “Pricing via Processing, Or, Combatting Junk Mail, Advances in Cryptology”.

33. Adam Back, "Hashcash - A Denial of Service Counter-Measure", technical report, August 2002

34. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols. In: Secure information networks, Springer (1999)

35. Kathleen E. Wegrzyn, Eugenia Wang, “Types of Blockchain: Public, Private, or Something in Between”, Published To: Manufacturing Industry Advisor Innovative Technology Insights Dashboard Insights

36. Parma Bains, “Blockchain Consensus Mechanisms: A Primer for Supervisors”, January 2022

37. Lashkari, B., & Musilek, P. (2021), “A Comprehensive Review of Blockchain Consensus Mechanisms”, IEEE Access, 9, 43620–43652.

38. Yusoff, J., Mohamad, Z. and Anuar, M. (2022), “A Review: Consensus Algorithms on Blockchain”, Journal of Computer and Communications, 10, 37-50

39. Casino, F., Dasaklis, T.K., Patsakis, C. “A systematic literature review of blockchain-based applications: Current status, classification and open issues.” Telematics and Informatics 2019

40. lnes, S., Jansen, A., “Blockchain technology as s support infrastructure in e-government”, In: Janssen, M., Axelsson, K., Glassey, O., Klievink, B., Krimmer, R., Lindgren, I., Parycek, P., Scholl, Hans J., Trutnev, D. (eds.) EGOV 2017. LNCS, vol. 10428, pp. 215–227. Springer, Cham (2017)

41. García-Bañuelos, L., Ponomarev, A., Dumas, M., Weber, I., “Optimized execution of business processes on blockchain” In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 130–146. Springer, Cham (2017).

42. Bocek, T., Rodrigues, B., Strasser, T., Stiller, B., “Blockchains everywhere - a use-case of blockchains in the pharma supply-chain” In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (2017)

43. Shae, Z., Tsai, J., “On the design of a blockchain platform for clinical trial and precision medicine” In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (2017)

44. Toyoda, K., Mathiopoulos, P., Sasase, I., Ohtsuki, T., “A novel blockchain-based Product Ownership Management System (POMS) for anti-counterfeits in the post supply chain”, IEEE Access 5, 17465–17477 (2017)

45. Munsing, E., Mather, J., Moura, S., “Blockchains for decentralized optimization of energy resources in microgrid networks”, In: 2017 IEEE Conference on Control Technology and Applications (CCTA) (2017)

46. Kshetri, N.,” Blockchain’s roles in strengthening cybersecurity and protecting privacy. Telecommun”, Policy 41, 1027–1038 (2017)

47. Aitzhan, N.Z., Svetinovic, D., “Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams”, IEEE Trans. Dependable Secure Comput. 1 (2016)

48. Grumbach, S., Riemann, R., “Distributed random process for a large-scale peer-to-peer lottery”, In: Chen, L.Y., Reiser, H.P. (eds.) DAIS 2017. LNCS, vol. 10320, pp. 34–48. Springer, Cham (2017)

49. Wijaya, D.A., Liu, J.K., Suwarsono, D.A., Zhang, P., “A new blockchain-based value-added tax system”, In: Okamoto, T., Yu, Y., Au, M.H., Li, Y. (eds.) ProvSec 2017. LNCS, vol. 10592, pp. 471–486. Springer, Cham (2017)

50. Zishan Zhao, “Comparison of Hyperledger Fabric and Ethereum Blockchain”, 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)

51. IBM Hyperledger Fabric. Retrieved from https://www.ibm.com/blockchain/hyperledger

52. Androulaki, E., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Laventman, G. (2018), “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”, Proceedings of the Thirteenth EuroSys Conference on - EuroSys ’18.

53. Debajani Mohanty, “R3 Corda for Architects and Developers: With Case Studies in Finance, Insurance, Healthcare, Travel, Telecom, and Agriculture”, Apress; 1st ed. edition (June 29, 2019)

54. Khan, C., Lewis, A., Rutland, E., Wan, C., Rutter, K., & Thompson, C. (2017), “A Distributed-Ledger Consortium Model for Collaborative Innovation”, Computer, 29–37.

55. M. Hearn, “Corda: A Distributed Ledger white paper R3”, Nov. 2016, Available: docs.corda.net/_static/corda-technical-whitepaper.pdf

56. Hewa, T. M., Hu, Y., Liyanage, M., Kanhare, S. S., & Ylianttila, M. (2021), “Survey on Blockchain-Based Smart Contracts: Technical Aspects and Future Research”, IEEE Access, 9, 87643–87662.

Figures

Tables

Appendices

Glossary of abbreviations used

SDLC	Systems Development Life Cycle
SDM	System Deploy and Maintenance
KPI	Key performance indicators
ERP	Enterprise Resource Planning
CRM	Customer Relationship Management
SCM	Supply Chain Management
DLT	Distributed ledger technology
SHA-256	Secure Hashing Algorithm, 256-bits
EVM	Ethereum Virtual Machine
dApps	Decentralized Applications
PoS	Proof-of-Stake
SLA	Service-level agreement (SLA)
DLM	Distributed Lifecycle Management
SLM	Software Lifecycle Management

Glossary of terms used

Edge computing	A strategy of computing at the point where data is collected or used allows IoT data to be collected and processed on-site, rather than sending the data back to a data center or cloud. Together, IoT and edge computing are a powerful way to quickly analyze data in real time.
Cloud	Cloud computing gives users remote access to files, software, and servers through their internet-connected devices: computers, smartphones
On-premise	The software is purchased and maintained by the company that acquired it, which also owns the servers on which the software is uploaded.
High availability	High availability is a label applied to systems that can operate continuously and reliably without downtime. In most cases, this involves copies of the program running on multiple servers.
Ethereum	Popular decentralized open source blockchain with smart contract functionality.
Bitcoin	an open-source payment system based on a P2P distributed network architecture, secured by blockchain technology and operating with the eponymous unit Bitcoin, called a virtual currency or cryptocurrency.
P2P / peer to peer	Peer to peer or also written peer-to-peer, abbreviated as P2P, is a decentralized architecture for distributed computing resources.

Nodes	A computing machine in a p2p network
Merkle trees	A Merkle tree is a hash structure used to efficiently verify data integrity.
Proof-of-Work /PoW	A form of cryptographic proof in which one party (the prover) proves to others (the verifiers) that a certain amount of specific computational effort has been expended.
Bitcoin miner	"Mining" is a term used to explain the validation of transactions waiting to be added to the blockchain database.
Nonce	(blockchain) 32-bit random number that solves the puzzle of finding a hash starting with a certain number of zeros
NFT	NFT is a type of cryptographic token on a blockchain that represents a unique asset
Disaster recovery	Restoring information from IT systems in the event that a server center is down due to a disaster. Typically achieved by replicating information between multiple data centers.
Logs	Files that record events, activities, and messages generated by operating systems, applications, and hardware components, providing a chronological historical record of system behavior
Patching	*Fixing code errors*
SLA	*A service level agreement (SLA) defines expectations between a service provider and a customer and describes the products or services to be delivered,*

	the contact person for end-user issues, and the metrics by which the process is measured.
Blockchain technology	A method of storing information on a computer network that represents a continuously growing list of computer records called "blocks", linked together and encrypted cryptographically.
Systems development life cycle	Development life cycle is the process of planning, creating, testing, and implementing an information system.
Patch	Small code update used to fix bugs, security vulnerability, or enhance performance