SAFEONECHAIN.COM

SAFE ONE CHAIN

BLOCKCHAIN TECHNOLOGY

t.me/SafeOneChain

Value transfer between smart contracts on a distributed blockchain platform on the base of Proof of Stake validation.

ABSTRACT

In a year in which economic turmoil dictates the days circumstances advance more than ever, it is necessary to apply the use of technologies that facilitate their development, which is why SafeOne Chain proposes a technology infrastructure enabled for the development of Smart contracts and Dapps with a PoS consensus system that makes it highly competitive and secure without high energy cost. Blockchain-enabled smart contracts that employ proof-of-stake validation for transactions promise significant per-

SafeOne Chain smart contract framework that targets application suitability and industry best practices.

SafeOnesmart contract development plans for 100% rug/scam secured industry case applications.

t.me/SafeOneChain

INTRODUCTION

 Industrial revolutions have been characterized by bringing with them disruptive products and technologies that have marked and changed people's daily lives, becoming more and more comfortable for their beneficiaries. Such technologies coming always with a price of security issues what SafeOne is aiming to resolve through state of the art POS blockchain technology and insurance industry like vetting Praxis with unique security products like Tokenized insurance policy's and anonymous payment solutions. This is the need of the day and the way to guide us to mass adoption.

Security is the first aim of any governance, in our blockchain con-cept it will have no different position asjust in real life.

Blockchain technology is considered one of these technologies that brings the 4.0 tech era along with IoT and AI technologies, since it offers transactional and communication
methods between P2P peers, passing through a decentralized system and with high standards in security levels and is seen to be higer than till date used systems.

The orchestration and choreography protocols that facilitate, verify and promulgate with computing means a negotiated agreement between the consenting parties, they are called smart contracts. The latter initially find application in various domains such as, for example, financial technology [6], Internet of Things (IoT) applications [33], digital signage solutions [11]. An essential aspect of smart contracts is a decentralized validation of transactions, initially using the so-called proof of work (PoW) [42]. The core technology that enables smart contracts is a distributed public ledger called the blockchain, which records transaction events without requiring a trusted central authority. Blockchain technology spreads in popularity with the inception of Bitcoin [23], a peer-to-peer (P2P) payment and cryp-tocurrency system comprising a limited set of operations at the protocol layer. Bitcoins use PoW for transaction validation which is computationally expensive and electricity intensive. POS is the up to date solution of an modern and economic traditional system.

For example, an inability to automate information logistics between organizations, the lack of privacy protection differentiations between external vs. related internal private contracts, secure and stable virtual machines for blockchains with better- performing proof-of-stake transaction validation [2], formally verifiable smart contract languages, lite wallets that do not require downloading the entire blockchain and mobile device solutions for smart contracts with simple payment verification (SPV) [14].

The latter means that clients simply download block headers when connecting to an arbitrary full node [23]. While SafeOne Chain uses the Ethereum Virtual Machine (EVM) for a current lack of more suitable alternatives, according to [19], the EVM has shortcomings such as previously experienced attacks against poorly handled exceptions and against dependencies such as for transaction requests, timestamps, etc. It is also desirable that a smart contract system achieves the scalability of the industry with the employment of sidechains [10] and outputs of unspent

such as Bitcoins [23] or Colored Coins [36].This whitepaper addresses the gap by specifying SafeOne Chains framework for smart contract systems that answers the question of how to develop a smart contract solution to meet critical customer requirements to enable information logistics between organizations to reduce costs and time. To establish a separation of concerns, we pose the following sub-questions. What differentiating technology performance advantages do SafeOne Chains smart contract solutions offer? What are the critical smart contract requirements that the SafeOne Chains framework satisfies? What are the unique features of the automation of information logistics between organizations that the SafeOne Chains framework aims to support? The rest of this white paper is structured as follows.

First, Section 3 focuses on the concrete advantages of the SafeOne Chains framework in achieving technological performance increases compared to related solutions. Section 4 offers functional and quality objectives in combination with the stakeholders involved for organized smart contract systems. Section 5 shows how the running case supports the SafeOne Chain-framework value transfer protocol. Finally, Section 6 concludes this white paper together with the analysis of constraints, outstanding issues and future development work.

-http://solidity.readthedocs.io/en/develop/

-https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/

-https://bitcoinsmagazine.com/articles/ethereum-classic-hard-forks-diffuses difficulty-bomb-1484350622/

-https://cointelegraph.com/news/ethereum-hard-fork-no-4-has-arrived-as-dosattacks-intensificar

-https://forum.dahub.Org/t/whats-up-with-casper-proof-of-stake-andsharding /6309

THE SAFEONE CHAIN BLOCKCHAIN TECHNOLOGY

The blockchain of the SafeOne BlockChain platform executes the SHA-256 cryptographic algorithm maintaining the UTXO model used by Satoshi Nakamoto in 2009 with the creation of bitcoin, however it offers great changes and added values compared to blockchains such as bitcoin since It integrates a virtual machine (Ethereum EVM) with the difference that it runs a PoS consensus system, which makes it more efficient and scalable in the long term. Owning a virtual machine makes possible the development of

agreement between P2P peers or business-to-business B2B in which all the requirements and conditions are stipulated for the smart contract to be executed correctly, is to clarify and emphasize that a smart contract is only as smart as the person who programmed it.

For the operation of some Smart contracts in blockchains, the use of Oracles is necessary, which play the role of judges (only when future information must be verified), offering the smart contract the pertinent information for the smart contract to be executed. For example, suppose that you want to develop a smart contract in your university or company which will reward the performance above the average of the employees of said entities, in the case of the university you will reward the professors whose stu-

company to the one who stands out among the staff or improves the performance of the company directly / indirectly, rewarding them with a reward of $ 500 USD (equivalent in SafeOne cryptocurrencies) to each person who meets these requirements, for the smart contract to fulfill its function, it must have access to a database or oracle that provides pertinent information on the status of the beneficiaries of the rewards, so these smart contracts are so secure as the blockchain it is associated with and as weak as its oracles or developers can be.

In contrast to Bitcoins, many smart-contract systems are equipped with the Turing-complete language Solidity that resembles JavaScript syntax and targets for enactment, e.g., the Ethereum Virtual [44] machine. Ethereum is the de facto leading smart-contract system despite being plagued by several deficiencies. First, proof-of-work transaction validation diminishes scalability to the point where Ethereum is considered to not be feasible for most industry applications.

LIMITATIONS OF ETHEREUM INDUSTRY ADOPTION

Second, in a recent crowdfunding case study, the Ethereum affiliated Solidity smart contract was hacked because of security flaws resulting from a lack in the state of the art with respect to tools for formal verifications [3]. The security flaw resulted in a loss of ca. $50 million. Consequently, Ethereum performed a hard fork resulting in a schism yielding two separate Ethereum versions. Yet another Ethereum hard fork was caused by a denial of service attack, and more hard forks must be expected for realizing-proof-of-stake [2] transaction validation and blockchain sharing [20]. More reasons limit widespread Ethereum industry adoption [8]. For example, an inability to automate cross-organizational information-logisticsjacking privacy protecting differentiations between external- versus related internal private contracts, secure and stable virtual machines for blockchains with better performing proof-of-stake [2] transaction validation, formally verifiable smart contract languages lite wallets that do not require downloading the entire blockchain, andmobile-device solutions for smart contracts with simple payment verification (SPV) [14].

The latter means that clients merely download block headers when they connect to an arbitrary full node [23]. While SafeOne Chain uses the Ethereum Virtual Machine (EVM) for a current lack of more suitable alternatives, according to [19], the EVM has deficiencies such as earlier experienced attacks against mishandled exceptions and against dependencies such as for transaction-ordering, timestamps, and so on. It is also desirable for a smart-contract system to achieve industry-scalability with employing

compatibility to other blockchain systems such as Bitcoins [23], or Colored coins [36]. Further more, an adoption of features from the Bitcoin Lightning Network [35] yields scalability via bidirectional micropayment channels. While smart-contract systems such as Ethereum attract attention, a widespread industry adoption does not exist for the above discussed reasons.

SafeOne Chains Uniqueness and Comparison with Ethereum his whitepaper addresses the gap by specifying the SafeOne Chains framework for smart- contract systems that answers the question of how to develop a smart-contract solution to satisfy critical customer requirements for enabling cross-organizational information logistics to reduce costs and time? To establish a separation of concerns, we pose the following sub-questions. What differentiating technological performance advantages do SafeOne

ments the SafeOne Chains framework satisfies? What are the unique features of cross-organizational information logistics automation the SafeOne Chains framework aims to support? We will answer this and more questions throughout the document. More reasons limit widespread Ethereum industry adoption [8]. For example, an inability to automate cross-organizational information-logistics, lacking privacy protecting differentiations between external- versus related internal private contracts, secure and stable virtual machines for blockchains with better performing-proof-of-stake [2] transaction validation, formally verifiable smart contract languages, lite wallets that do not require downloading the entire blockchain, and mobile-device solutions for smart contracts with simple payment verification (SPV) [14]. The latter means that clients merely download block headers when they connect to an arbitrary full node [23].

SAFEONE CHAIN PERFORMANCE ADVANTAGE

One of SafeOne Chains main goals is to build a decentralized smart contract system based on UTXO with a proof-of-stake (PoS) consensus model [37] this means that the creator of the next block is chosen at random based on the wealth held in cryptocurrencies within their wallet and the maturity of the same, constantly rotating addresses to ensure decentralization and the participation of the entire network. Therefore, blocks are usually built or minted rather than mined, there are block rewards in addition

amount of funds they bet. This allows the chain to achieve high levels of security without excessive energy consumption, since to participate as an applicable active node for staking it is enough to have a Raspberry-Pi, laptop or 64-bit desktop PC which do not have such a high consumption compared to Proof of Work string mining rigs.

SafeOne supports the Bitcoin and Ethereum ecosystems and aims to produce a variation of Bitcoin with Ethereum Virtual Machine (EVM) support. Following a pragmatic design approach, SafeOne Chain employs industry use cases with a strategy that integrates a 100% safe to invest within the blockchain, which will be reached through state of the art vetting processes and for the rest risk we offer an unique Tokenized Insurance policy. The latter allows SafeOne to promote blockchain technology to a wide range of

globally while proceeding to create a secure and stable network in the long term. The rest is structured as follows. Section 3.1 compares the advantages of Bitcoin UTXO versus the ethereum account model. Next, Section 3.2 discusses the consensus platform for the SafeOne blockchain. Section 3.3 shows the integration of SafeOne Chain contracts into the EVM. Finally, Section 3.4 describes the payment model for SafeOne Chains operations.

SafeOne supports the Bitcoin and Ethereum ecosystems and aims to produce a variation of Bitcoin with Ethereum Virtual Machine (EVM) support. Following a pragmatic design approach, SafeOne Chain employs industry use cases with a strategy that integrates a 100% safe to invest within the blockchain, which will be reached through state of the art vetting processes and for the rest risk we offer an unique Tokenized Insurance policy. The latter allows SafeOne to promote blockchain technology to a wide range of

globally while proceeding to create a secure and stable network in the long term. The rest is structured as follows. Section 3.1 compares the advantages of Bitcoin UTXO versus the ethereum account model. Next, Section 3.2 discusses the consensus platform for the SafeOne blockchain. Section 3.3 shows the integration of SafeOne Chain contracts into the EVM. Finally, Section 3.4 describes the payment model for SafeOne Chains operations.

UTXO VERSUS ACCOUNT MODEL

In the UTXO model, transactions use as input unspent Bitcoins that are destroyed and as transaction outputs, new UTXOs are created. The results of unspent transactions are created as exchange and returned to the spender [1]. In this way, a certain volume of Bitcoins is transferred between different private key owners, and new UTXOs are spent and created in the transaction chain.

The UTXO of a Bitcoin transaction is unlocked by the private key that is used to sign a modified version of a transaction. In the Bitcoin network, miners generate Bitcoins with a process called coinbase transaction, which does not contain any input. Bitcoin uses a scripting language for transactions with a limited set of operations. In the Bitcoin network, the scripting system processes data by stacks (Main Stack and Alt Stack), which is an abstract data type that follows the LIFO principle of Last-In, First-Out. In the Bitcoin client, developers use the is Standard() [1] function to summarize scripting types. Bitcoin client support: P2PKH (Pay to Public Key Hash), P2PK (Pay to Public Key), MultiSignature (less than 15 private key signatures), P2SH (Pay to Script Hash) and OP_RETURN.

With these five types of standard scripting, Bitcoin clients can process complex payment logics. On top of that, a non-standard script can be created and executed if the miners agree to encapsulate such a non standard transaction. For example, using P2PKH for the process of creating and executing scripts, we assume that we pay 0.01BTC for bread in a bakery with the imaginary Bitcoin address "Bread Address". The result of this transaction is:

OP_DUP OP_HASH160 <Bread Public Key Hash> OP_EQUAL OP_CHECKSIG

Operation OP_DUP duplicates the top element of the stack. OP_HASH160 returns aBitcoin address as the main item. To establish ownership of a bitcoin, a Bitcoin address is required in addition to a digital key and a digital signature. OP_EQUAL produces TRUE (1) if the two main elements are exactly the same and otherwise FALSE (0). Finally, OP_CHECKSIG produces a public key and a signature along with a validation for the signature corresponding to the hash data of a transaction, returning TRUE if a match

<Brown signature> < Pan public key>

The combined script with the previous two:

<Pan Sign> <Pan Public Key> OP_DUP OP_HASH160
<Bugging public key hashes> OP_EQUAL OP_CHECKSIG

Only when the unlock script and the lock script have a matching predefined condition is the execution of the script combination true. It means that the bread signature must be signed by matching the private key of a valid bread address signature and then the result is true.

Unfortunately, Bitcoin's scripting language is not Turing-complete, for example, there is no loop function. The Bitcoin scripting language is not a commonly used programming language. Limitations mitigate security risks by avoiding the emergence of complex payment terms, for example, the generation of infinite loops or other complicated logical loops. In the UTXO model, it is possible to transparently track the history of each transaction through the public ledger. The UTXO model has parallel processing capability to initialize transactions between multiple addresses that indicate extensibility. In addition, the UTXO model supports privacy in the sense that users can use Change Address as the output of a UTXO. SafeOne Chains goal is to implement smart contracts based on the innovative design of the UTXO model. Compared to the UTXO model, Ethereum is an account-based system. More precisely, each account experiences direct transfers of value and information with state transitions.

https://en.bitcoin.it/wiki/Script

A 20-byte ethereum account address comprises a noun as a counter to ensure the unique processing of a transaction, the balance of the main internal cryptographic fuel to pay transaction fees called Ether, an optional contract code, and empty account storage by default. The two types of Ether accounts are, on the one hand, external controlled by private key and, on the other hand, controlled by contract code. The old null code account type creates and signs transactions for message transfer. The latter

storage, create contracts or send other messages.

On Ethereum, balance management resembles a bank account in the real world. Each newly generated block potentially influences the overall status of other accounts. Each account has its own balance, storage, and code space base for calling other accounts or addresses, and stores the respective execution results. In the existing Ethereum account system, users perform P2P transactions through remote client procedure calls. Although it is possible to send messages to more accounts via smart contracts, these internal transactions are only visible in each account's balance and tracking them on Ethereum's public ledger is a challenge.

Based on the discussion above, we consider the Ethereum account model to be a scalability bottleneck and see clear advantages of the UTXO model of the Bitcoin network. Since the latter enhances the network effect we wish to offer, an essential design decision for the pending launch of SafeOne Chain is the adoption of the UTXO model.

CONSENSUS MANAGEMENT

There are ongoing discussions about consensus and which platform meets the needs of the respective project requirements. The most discussed consensus topics are: PoW [41], PoS [2], Dynamic PoS, and Byzantine fault tolerance [7] as discussed by HyperLedger. The nature of consensus is about achieving data consistency with distributed algorithms. The available options are, for example, the theorem of Fischer Lynch and Paterson [5] which states that a consensus cannot be reached without a 100% agree-

In the Bitcoin network, miners participate in the hash collision verification process via PoW. When the hash value of a miner is able to calculate and meet a certain condition, the miner can claim from the network that a new block is extracted: For the number of miners M and the mining difficulty D, the Hash() represents the power SHA256 with range of values [0, M] and D. The SHA256 algorithm used by

https://github.com/ethereum/wiki/wiki/White-Paper

http://tinyurl.com/zxgayfr

Bitcoin allows each node to check each block quickly, if the number of miners is high compared to the mining difficulty. The 80-byte BlockHeader varies with each different Nonce. The overall difficulty level of mining is dynamically adjusted according to the total hash power of the blockchain network. When two or more miners resolve a block at the same time, a small fork occurs in the network. This is the point at which the blockchain must make a decision about which block it should accept or reject. In the Bitcoin network, the chain is legitimate that has the most proven work attached. Most PoS blockchains can get their PeerCoin inheritance which is based on an older version of Bitcoin Core. There are different PoW algorithms such as Scrypt, X11, Groestl, Equihash [4], etc. The purpose of launching a new algorithm is to prevent the accumulation of computing power by an entity and ensure that application-specific integrated circuits (ASICs) cannot be introduced into the economy. SafeOne Core chooses PoS based on the latest Bitcoin source code for basic consensus building. In a traditional PoS transaction, the generation of a new block must meet the followingcondition:

ProofHash < coins × target × age

In ProofHash, the stake modifier [40] calculates along with unspent outputs and the current time. With this method, a malicious attacker can initiate a double- spend attack by accumulating large amounts of coin age. Another problem caused by the age of the coins is that the nodes are online intermittently after rewarding rather than being continuously online. Therefore, in the improved version of the PoS agreement, the removal of the age of the coin encourages more nodes to beonline simultaneously.

due to potential coin age attacks and other types of attacks [16]. SafeOne Chain agrees with the security analysis of the Blackcoin team [40] and adopts PoS 3.0 on the latest SafeOne Core.

PoS 3.0 theoretically rewards investors who 14bet their coins longer, without giving any incentive to coin holders who leave their wallets offline. 3.3 SafeOne Chains contract and EVM integration The EVM is stack-based with a 256-bit machine word. Smart contracts running on Ethereum use this virtual machine for execution. The EVM is designed for the Ethereum blockchainband therefore assumes that every value transfer uses an account-based method. SafeOne Chain is based on Bitcoin's blockchain

https://peercoin.net/

https://litecoin.info/Scrypt

http://cryptorials.io/glossary/x11/

http://www.groestlcoin.org/about-groestlcoin/

http://blackcoin.co/

Therefore, SafeOne Chain has an account abstraction layer that translates the UTXO-based model into an account-based interface for the EVM. Note that an abstraction layer in computing is critical to hiding the implementation details of a particular functionality to establish a separation of concerns to facilitate interoperability and platform independence. EVM integration: All transactions in SafeOne Chain use the Scripting Language of Bitcoin, just like Bitcoin. On SafeOne Chain, however, there are three new

– OP_EXEC: This operation code triggers special processing of a transaction (explained below) and executes a specific input EVM bytecode.

t.me/SafeOneChain

– OP_EXEC_ASSIGN: This opcode also triggers special processing as a OP_EXEC. This opcode has as input a contract address and data for the contract. It then follows the execution of the contract bytecode while passing the given data (given as CALLERDATA in EVM). This opcode optionally transfers money to a smart contract .

– OP_TXHASH: This operation code is used to reconcile a strange part of the accounting abstraction layer and pushes the transaction ID hash of a currently executed transaction. Traditionally, scripts only run when trying to spend an output. For example, while the script is on the blockchain, with a standard public key hash transaction, no validation or execution takes place. Execution and validation do not occur until a transaction input references the output. At this point, the transaction is only valid if the input script (ScriptSig) provides valid data to the output script that causes the output script to return non-zero.

SafeOne Chain, however, must accommodate smart contracts that run immediately when merged with the blockchain. As shown in Figure 1, SafeOne Chain accomplishes this by specially processing transaction output scripts (ScriptPubKey) that contain OP_EXEC or OP_EXEC_ASSIGN. When one of these opcodes is detected in a script, it is executed by all nodes in the network after the transaction is placed in a block.

In this mode, the actual bitcoin scripting language serves less as a scripting language and instead carries data to the EVM. The latter changes state within its own state database, after execution by any of the opcodes, similar to an Ethereum contract.

To facilitate the use of SafeOne Chain smart contracts, we have to authenticate the data sent to a smart contract, as well as its creator derived from a particular public keyhash address .

To prevent the UTXO set of the SafeOne blockchain from becoming too large, OP_EXEC and OP_EXEC_ASSIGN transaction outputs are also expendable. OP_EXEC_ASSIGN outputs are spent by contracts when their code sends money to another contract, or to a pubkeyhash address. Departures P_EXEC OR they are spent every time the contract uses the suicide operation to withdraw from the blockchain.

SAFEONE CHAINS ACCOUNT ABSTRACTION LAYER

The EVM is designed to work on an account based blockchain. However, SafeOne Chain, being bitcoin based, uses a UTXO-based blockchain and contains an account abstraction layer (AAL) that allows the EVM to function on the SafeOne blockchain without significant modifications to the existing Ethereum virtual machine and contracts. The EVM account model is easy to use for smart contract programmers. There are operations that check the balance of the current contract and other contracts on the blockchain, and there are operations to send money (attached to the data) to other contracts. Although these actions seem pretty basic and minimalist, they are not trivial to apply within the UTXO-based SafeOne blockchain . Therefore, the AAL implementation of these operations may be more complex than expected.

A smart contract implemented by SafeOne-blockchain is assigned and enforceable by its management and comprises a newly implemented contract balance set to zero. Currently there is no protocol in SafeOne Chain that allows you to implement a contract with a non-zero balance. To send funds to a contract, a transaction uses the OP_EXEC_ASSIGN opcode. The example output script below sends money to a contract: 1: virtual machine version 10000, gas limit for the transaction 100, gas price in SafeOne sa-

0x1452b22265803b201ac1f8bb25840cb70afe3303 ; ripemd–160 hash the contract tx id OP EXEC ASSIGN. The simple script above delivers transaction processing to the OP_EXEC_ ASSIGN operation code. Assuming that there are no gas-free exceptions or other exceptions, the amount of value given to the contract is OutputValue. The exact details of the gas mechanism that we discuss below. By adding this output to the blockchain, the output enters the domain of the UTXO set owned by the contract.

This value of production is reflected in the contract balance as the sum of expendable products. Figure 2. Assign Funds and/or a TX message contract. Although Figure 2 shows sending funds to a contract from a standard public key hash output, the method for sending money from one contract to another is almost identical. When the contract sends funds to another contract or public key hash address, the first one spends one of its own results. The shipping contract involves expected contractual transac-

sense that they must exist in a block to be valid for the SafeOne network. Expected contractual transactions are generated by miners as they verify and execute transactions, rather than being generated by consumers. As such, they are not broadcast on the P2P network. Figure 3. SafeOne block validation showing the Expected Contract

The main mechanism for performing expected contractual transactions is the new operation code, OP_TXHASH which is part of Figure 3. Internally, both OP_EXEC and OP_EXEC_ASSIGN have two different modes. After it is executed as part of the processing of the output script, the EVM is executed. However, when op codes are executed as part of inbound script processing, the EVM does not run to avoid double execution. In contrast, OP_EXEC and OP_EXEC_ASSIGN op codes behave similarly to non-ops and

a given transaction hash.

That is why OP_TXHASH is so important for the functioning of this concept. Briefly, OP_TXHASH is a new added operation code that pushes the SHA256 hash of the current spending transaction into the Bitcoin Script stack. The OP_EXEC and OP_EXEC_ASSIGN opcodes check the List of Expected Contract Transactions During a Spending Attempt. After the transaction passes (usually from OP_TXHASH) to the opcodes that exist in the Expected Contract Transactions List, the result is 1 or it can be spent. Otherwise, the return is 0, or it cannot be spent. In this way, OP_EXEC and OP_EX-EC_ASSIGN using vout can only be spent when a contract, and therefore the account abstraction layer, requires the vout to be expendable, that is, while the contract tries to send money. This results in a safe and robust way to allow contract funds to be spent only through a respective contract in alignment with a normal UTXO transaction. A specific scenario occurs if a contract has more than one result that can be spent. Each node can choose different outputs and therefore use completely different transactions to spend OP_EXEC_ASSIGN transactions. This is solved in SafeOne Chain by a coin selection algorithm critical to consensus.

The latter is similar to the standard coin selection algorithm used within a user wallet. However, SafeOne significantly simplifies the algorithm to avoid the risk of denial-of-service (DoS) attack vectors and to perform simple consensus rules. With this coin selection algorithm critical for consensus, there is now no chance for other nodes to choose different currencies to be spent by a contract. Any miner/node that chooses different outputs must be forked from the SafeOne Chains mainnet, and its blocks become invalid. When an EVM contract in Figure 4 sends money to a public keyhash address or to another contract, this event builds a new transaction. The consensus-critical coin selection algorithm chooses the best results from the group of contracts. These outputs are spent as inputs with the input script (ScriptSig) comprising a single OP_TXHASH opcode.

The results are, therefore, the destination of the funds and a change result (if necessary) to send the remaining funds from the transaction to the contract. This transaction hash is added to the Expected Contract Transaction List, and then the transaction itself is added to the block immediately after the contract execution transaction. Once this constructed transaction is validated and executed, a confirmation check of the Expected Contract Transactions List follows. This transaction hash is then removed from the Expected Contract Transactions List. Using this model, it is impossible to spoof transactions to spend by providing a hash encoded as an input script, rather than using OP_TXHASH.

The abstraction layer described above makes EVM contracts alien to the selection of specific currencies and outputs. Instead, EVM contracts only know that they and other contracts have a balance so that money can be sent to these contracts, as well as out of the contract system to public key-hash addresses. Consequently, contract compatibility between SafeOne Chain and Ethereum is strong and very few modifications are required to port an Ethereum contract to the SafeOne blockchain.

SPEND THE CONTRACT OP_EXEC_ASSIGN TRANSACTION.

Standard transaction types added: The following are the standard transaction types that we added to SafeOne Chain. They are documented here as Bitcoin script templates: the implementation of a new contract on the blockchain requires an output script as follows: 1;virtual machine version [Gas Limit] [GasPrice] [Contract EVM ByteCode]OP EXEC [Sending funds to a contract already implemented on the blockchain requires the following script: 1;virtual machine version [Gas Limit] [Gas Price] [Data to be sent to the contract] rip-emd160 hash of contract transactionid] OP EXEC ASSIGN Note that there are no standard transaction types for spending, as that requires the Expected Contract Transactions List. Therefore, these spending transactions are neither transmitted nor valid on the P2P network.

GAS MODEL

One problem SafeOne faces in adding Turing's completeness to the Bit-coinBitcoin blockchain is relying only on the size of a transaction, which is unreasonable in determining the fee paid to miners. The reason is that one transaction can loop infinitely and stop the entire blockchain for transaction processing miners. As Figure 5 shows, the SafeOne Chain project adopts the concept of Ethereum gas. In the concept of gas, each EVM opcode executed has a price and each transaction has an amount of gas to spend. The remaining gas after the transaction is refunded to the sender.

GAS REFUND MODEL.

When the gas required for contract execution exceeds the amount of gas available for a transaction, then the actions of a transaction and changes in status are reversed. Therefore, any modified permanent storage is reverted to its original state, including any expenditure of contractual funds so that the latter are not spent. Despite a reversal, all the gas from a transaction is consumed and delivered to the processing miner, as the computing resources have already been spent. Although SafeOne Chain uses Ethe-

EVM operating code, to differ significantly from Ethereum. The exact values are determined by comparing the existing prices on Ethereum with the amount of processing and blockchain resources required for each opcode to SafeOne Chain. When creating a financing transaction or contract deployment, the user specifies two specific elements for the gas. The GasLimit determines the amount of consumable gas by executing a contract. The second element is the GasPrice to set the exact price of each gas unit in SafeOne Satoshis.

The latter are currently a smaller unit of the Bitcoin currency that thethe blockchain records. The maximum SafeOne expense of the execution of a contract is equivalent to the multiplication of GasLimit by GasPrice. If this maximum expense exceeds the transaction fee provided by the transaction, the transaction is invalid and cannot be extracted or processed. The remaining transaction fee after subtracting this maximum expense is the Transaction Size Fee and analogous to Bitcoin's standard fee model. To de-

ables.

First, the transaction size fee must match the total size of a transaction, that is, usually determined by a minimum number of coins per kilobyte formula. The second variable is the Gas Price of the execution of a contract. In combination, PoS miners choose the most important and cost-effective transactions to process and include in a block. Consequently, there is a free market fee model with miners and users optimizing the best rate that suits their transaction speed and the price they are willing to pay.

REFUNDS:

Using the UTXO model, funds sent to miners as transaction fees are non-negotiable. It is impossible for a miner to partially refund a fee if the transaction is easier for the miner to process than expected. Still, for the gauze model to be useful, there must be a method of reimbursing the funds to the sender. In addition, it should be possible to reverse the status of a transaction that runs out of gas and return gas fees to miners. Reimbursement of gas fees in SafeOne Chain is enabled by creating new outputs as

We added a new block validation consensus rule to ensure that refund results must exist in the coinbase transaction. Otherwise, miners may choose not to reimburse the gas. The refund is returned to the sender of a transaction fund by copying the output script. For security reasons, this script is currently a standard pay-to-pubkeyhash or paid-to-scripthash script. We plan to lift the restriction after further safety studies. For reference, the OP_EXEC_ASSIGN has the following format for allocating contractual

Inputs:

– Transaction hash for expenses [optional]

– version number

– gas limit

– gas price

– data

– smart contract address

Outputs:

– Expendable

t.me/SafeOneChain

REFUNDS:

Using the UTXO model, funds sent to miners as transaction fees are non-negotiable. It is impossible for a miner to partially refund a fee if the transaction is easier for the miner to process than expected. Still, for the gauze model to be useful, there must be a method of reimbursing the funds to the sender. In addition, it should be possible to reverse the status of a transaction that runs out of gas and return gas fees to miners. Reimbursement of gas fees in SafeOne Chain is enabled by creating new outputs as

We added a new block validation consensus rule to ensure that refund results must exist in the coinbase transaction. Otherwise, miners may choose not to reimburse the gas. The refund is returned to the sender of a transaction fund by copying the output script. For security reasons, this script is currently a standard pay-to-pubkeyhash or paid-to-scripthash script. We plan to lift the restriction after further safety studies. For reference, the OP_EXEC_ASSIGN has the following format for allocating contractual

Inputs:

– Transaction hash for expenses [optional]

– version number

– gas limit

– gas price

– data

– smart contract address

Outputs:

– Expendable

t.me/SafeOneChain

Accordingly:

we give an example EXEC_ASSIGN below:

1

10000

100

0xABCD1234...

3d655b14393b55a4dec8ba043bb286afa96af485

EXEC_ASSIGN

If running the virtual machine results in a no-gas exception, this output is spent on the next block transaction using the redemption script OP_TX-HASH. The vout generated for this transaction is a pubkeyhash script taken from the vin[0].prevout script. In this early version of SafeOne Chain, only pubkeyhash senders are allowed for VM funding transactions. Although other forms can be accepted in blocks to result in the execution of the virtual machine, the msg.sender in the EVM is "0" and any lack of gas or gas

PARTIAL REIMBURSEMENT MODEL

Belonging to the gas model, it is also necessary to reimburse the unspent part for various reasons. On the one hand, users can spend a lot of funds to ensure that their contract is executed correctly. Still, the unused gas returns as a refund from SafeOne Chain.

The return address for the gas is expressed on the blockchain as a vin[0].prevout script of the shipping transaction. The gas is sent to a contract by using bitcoin's standard transaction fee mechanism. Therefore, the new fee model slightly increases this to make the transaction fee:

gas tariff = gas limit ⊠ txfee gas price = vin – vout

txrelay tariff = txfee – gas tariff

reimbursement= gastariff – used gas

There is a proposal to allow miners to evaluate both the tx_relay_fee and the gas_price under a single "credit price" value to determine the priority of the transaction. During the execution of the contract, gas tokens are subtracted from the total fee, that is, multiplied by gas_price. After completing the execution of the contract, the rest of this gas_fee must be returned to the given gas return script by adding an output to the coinbase transaction that miners use to recover their block reward. The added vout

To receive a gas refund, this must be a spent public keyhash vout. Otherwise, the gas refund remains with the miner in an off-gas condition and the funds sent will remain with the contract. Please note that it is currently only possible to have one EVM contract execution per transaction. Therefore, the case never arises when two contract executions attempt to share the transaction fee. This scenario can be enabled after you resolve existing issues with multiple EVM runs per transaction. The current design sup-

Important GAS Edge cases: Miners should be wary of contract gas scripts and return funds. If the last script output causes a block to exceed the maximum size, then the contract transaction can not be placed in this block. Instead, the execution of the gas return script must take place again in the next mined block. Miners must ensure that there is sufficient capacity in the candidate block for the gas return script before attempting to execute the contract.

Not following this rule results in a contract that requires repeated execution, if the refund script does not fit into the current block. If there are no gas funds to return, there is no vout requirement to return the funds. Consensus is critical that the transaction fee includes the gas_fee. A transaction is invalid when adding it to a block results in a negative gas refund, or when the gas_fee is less than the transaction fee. Any transaction output script that has more than one OP_EXEC o r OP_EXEC_ASSIGN opcode is not

cursion and multiple execution issues. Consequently, static analysis is sufficient to determine whether a script is invalid. After SafeOne very blockchain-oriented technicalities, below we conceptually describe the management of smart contract lifecycles. Note that the conceptual presentation in the sequel is supported by scientific literature [12, 13, 24, 18, 26,27, 32].

SMART CONTRACT MANAGEMENT

As previously stipulated, we assume that lifecycle management is essential to securing smart contracts, as proper investigation of potential collaborating parties is carried out prior to enactment. We consider a real-life case of a failed delivery of seafood where a commercial transaction conflict arises from an underspecified conventional contract (CC). An EU company (buyer) orders 12 920 kg of cuttlefish from a South Asian company (seller). In the CC, the responsibility for the quality of the product lies with the seller until the carrier obtains the goods.

Subspecification refers to the quality of the goods which is not specified in the CC and the buyer does not check the goods before the transfer to the shipping company. 15 (carrier).

http://cisgw3.law.pace.edu/cases/090324s4.html

The smart contract alternative resolves the subspecification conflict that exists in the CC. Therefore, in Section 3.1, the Safeone-framework objective model presented reflects the properties of a smart contract lifecycle that is fully formalized in [18, 26, 27, 32]. Below, Section 3.2 provides a small life cycle example for seafood shipping.

LIFECYCLE MANAGEMENT OBJECTIVES

To discuss the objectives, we use the following approach. The Agent- Oriented Modeling (AOM) method [38] is a socio-technical approach to requirements engineering that takes into account that humans who may belong to organizations use technology to collaborate in problem solving. In this section, we use the AOM target model type to capture sociotechnical behavioral characteristics important to the Safeone Chains smart contract system that supports the running case.

Goal models improve communication between technical and non-technical stakeholders to increase understanding of problem mastery. Note that AOM goal models are also instrumental [39] for new agile software development techniques.

An objective model comprises three main elements represented as in Figure 6. The functional requirements we refer to as targets and are represented as parallelograms, the roles we represent as sticky men, and the non-functional requirements. The latter has two variants, namely quality objectives for non-function requirements related to software represented as clouds, and human-related emotional objectives represented as ellipses. The goal model starts with a central root value proposition that is not atomic. Consequently, the value proposition is decomposed into a tree hierarchy into sub-objectives where each sub-objective represents an aspect to achieve its main objective [21] and the lowest sub-objective must be atomic. Goals can have assigned roles, quality goals, and emotional goals that are inherited to lower-level goals.

SafeOne Chains Framework Value Proposition: The root of the objective for the SafeOne Chains framework that we describe in Figure 7 and is the value proposition of the logistics automation of information transfer and inter-organizational value. We divide the complex value proposition into objectives for smart contract lifecycle management [26, 27, 32], i.e. configuration, implementation, enactment, reversal, termination.

These refined objectives are explored further in Section 3.2. Figure 7. SafeOne Chain value proposition with lifeycle management refinement[26, 27, 18]. An essential emotional goal for industry adoption in Figure 7 is reliance on the sociotechnical SafeOne Chains system [34] to reliably perform the intended behavior. In this case, trust refers to the dependencies between humans who use technology to achieve goals.

We consider them economically viable and easy to adopt as additional emotional goals that influence the widespread spread of the industry. The former means that using the SafeOne Chains system results in an economic return on investment, while the latter means that the personal barrier to entry for working with SafeOne Chain is low. There are quality objectives associated with the value proposition that affect all refining parts of the SafeOne Chains system. These quality objectives are derived from a reference architecture [28] for collaboration between organizations and business processes.

The quality objectives below are structured according to [9, 17]. The following quality objectives are not noticeable during the system runtime. Modifiable means that the SafeOne Chains system changes and adapts during its lifecycle to the business context. In addition, it harmonizes heterogeneous system environments among organizations that comprise the periodic updating of commercial software. Integrable systems consist of separately developed and integrated components for which the interface proto-

tween the components of SafeOne Chain must be assured. Next, we specify the quality objectives for SafeOne Chain that are noticeable during runtime.

Interoperable means that SafeOne Chain must interoperate at runtime with systems that support business functions such as planning, logistics, production, external partner systems, etc. Dynamic interoperability challenges are business, conceptual and technical heterogeneity.

Safe refers to resisting unauthorized attempts at use and denial of service while providing services to reputable trusted users. To address security, trust, and reputation issues, SafeOne Chain can come up with several strategies. A blockchain-compatible authentication service verifies collaborating parties, monitors, inspects, and records network events. A system's communication can be encrypted, and so on.

-Highly automated collaboration requires systems to cover the entire lifecycle of the smart contract. Therefore, SafeOne Chain must provide possibilities for a high degree of meaningful collaboration automation that processes tedious and repetitivework while allowing humans to focus on the remaining creative action.

-Flexible collaboration is a highly dynamic process that enacts activities by diverse partners exchanging heterogeneous data [25]. Therefore, SafeOne Chain must allow diverse collaboration scenarios between organizations that harmonize heterogeneous concepts and technologies.

-Usable means that SafeOne Chain should be easy to use for automating information logistics between organizations and breaks down into three areas. Error prevention must anticipate and prevent the collaboration errors that commonly occur. Error handling is the support of the system for a user to recover from errors. Learning ability refers to the learning time required of users to master the SafeOne Chains system. Finally, there are quality objectives that are specific to architecture.

-Integrity is the quality of SafeOne Chain comprising the set of components for the lifecycle management of smart contracts.

-Scalable refers to SafeOne Chains ability to combine more than two collaborating parts into one configuration.

-Applicable means that SafeOne Chain is critical to automating information logistics between organizations and value transfers.

-Portable means that SafeOne Chain supports information logistics regardless of the industrial domain and the heterogeneity of collaboration with respect to the infrastructure of enterprise, conceptual and technological systems. Note that this also includes mobile devices.

-Performant means that computational and communicational stress is low for the automation of information logistics. Therefore, it is important to ensure that all phases of the life cycle of a smart contract are carried out within a desirable response time and without an exponential need for computing power.

LIFECYCLE MANAGEMENT EXAMPLE

We mapped the objective model in Section 3.1 in Figure 8 to project it in the case of shellfish in execution. The modeling notation in Figure 8 is the business process model and BPMN notation [22] and the entire lifecycle is formalized in [26, 27, 18]. The green circle denotes a beginning of the life cycle and the red circle the end of the life cycle. Rectangles with plus signs are so-called threads that correspond to the lifecycle stages in Section 3.1. A thread is a composite activity that hides details of lower-level business processes.

SAFEONE CHAINS SMART CONTRACT LIFEYCLE MANAGEMENT

The starting point for each smart contract lifecycle in Figure 8 is the seafood transport business case that requires the automation of information logistics between organizations. Assuming that there is a collaboration center [29] that serves as a preparation platform for the initiation of smart contracts, a designer creates a template for an enterprise network model (BNM) in which service types are inserted along with roles. The BNM workforce enters the population phase. The roles affiliated with the respective types of services are full of organizations collaborating on the smart contract, i.e. bank2, seller, refrigerator1, carrier, refrigerator2, buyer and bank1. Please note that several candidate organizations may compete for a specific position.

To reinforce the desire to hold a position, potential partner organizations should match a service offering to the type of service a role is affiliated with. A service consumer can evaluate the proposal and decide whether a service offer is acceptable. When all roles are occupied and service types match acceptable service offerings, smart contract negotiation begins. We assume that neither side of the ongoing seafood delivery case has a desire to disagree and bring the setup phase to a sudden end.

Instead, the buyer provides a counteroffer that introduces temperature-related obligations inside the containers where the seafood is stored. We assume that shipping containers are equipped with Internet of Things (IoT) sensors [15] that inform the sender, seller, and buyer in real time when a temperature threshold violation occurs.

The buyer's counteroffer defines in this case that either there is a reduction in the price according to the reduced quality of the seafood. If the change in temperature makes the seafood no longer fit for consumption, the buyer has the right to refuse the purchase of the shipment upon arrival. The counter offer is accepted by all other parties and a consensus is produced, which is the prerequisite for the establishment of a contract.

The smart contract is a coordinating agent from which a distributed governance infrastructure (DGI) must be deduced. Therefore, each party to the ongoing case receives a copy of the local contract from which a set of respective obligations is deduced. For example, an obligation for the carrier is that the temperature inside a seafood shipping container should never be higher than 20°C. Obligations are observed by assigned monitors and enterprise network model (BNMA) agents that connect to IoT sensors. All

to an emerging DGI.

For example, we assume peer-to-peer payment for Bitcoins that the buyer must first buy with Euros. That purchase and payment through the bank1 involves a process comprising compliance steps and reporting as the government imposes regulations on the use of cryptocurrencies. To allow the exchange of information between bank1 and the seller's bank2 , communication endpoints must be established. In this way, the management of the seller's compliance data is automated.

Assuming a breach of the temperature threshold obligation occurs in the fridge1 domain, an assigned BNMA escalates the event and the buyer checks the severity of the breach. If the temperature violation lasts for a period of time resulting in a decrease in seafood quality that still allows for a successful sale for a lower price that the buyer tolerates, one response may be for the latter to request additional cooling by a different company that sneaks into the refrigerator paper1. Assuming that seafood is badly spoiled and cannot be sold in the target country, the buyer triggers a disruptive setback that collapses the transaction. If the seafood shipment arrives with the buyer in the agreed state and the payment to the seller through the bank2 is completed, then the termination stage dissolves the DGI and releases all collaborating parties. Below we give the relationships between the detailed collaboration elements that coordinate the lifecycle management of Figure 8.

VALUE TRANSFER PROTOCOL

An integral part of the SafeOne Chains framework is the notion of a value transfer protocol (VTP) that orchestrates information logistics between organizations and value transfers, in line with the value proposition shown in Figure 7. Accordingly, Section 4.1 describes the relationship of the process types that make up a VTP. Section 4.2 discusses the need for a specific smart contract language with the utility for specifying VTP. Finally, Section 4.3 discusses the characteristics of a VTP-compatible language versus Solidity that uses Ethereum.

INTER-ORGANIZATIONALPROCESSES

The VTP comprises three different types of collaborative processes. Figure 9 shows a simplified BNM in BPMN notation for seafood delivery that introduces Section 3. The BNM assumes that a sequence of threads are placeholders for service types [12, 13] with labels indicating the roles of organizations.

SAFEONE CHAIN BNM.

We assume that the BNM also comprises tasks that connect service-type threads to establish the choreography control flow. For simplicity, Figure 9 depicts unlabeled choreography tasks along with an AND-split and -join. The BNM starts with the seafood seller informing the bank to prepare for an international transaction in currency and then the seafood is cooled before a carrier ships to the destination. In the destination country, seafood is cooled again while a local bank processes the currency transaction between the two countries. Finally, the buyer receives the seafood for local sales. Figure 10. Outsourced service type process view.

For the BNM carrier subprocess, the assumption is that there are several candidate organizations to fulfill the role of shellfish carrier. Figure 10 shows a simplified example for a lower-level refinement in the form of a service type process view [12, 13]. The simplified process in Figure 10 assumes that a carrier receives the seafood from the refrigerator in the home country and charges the seller's bank. Next, three parallel branches require temperature monitoring, preparation of delivery documents and information to the cooling company in the target company to be performed simultaneously. Only a candidate organization can become a service provider operator that promises to adhere to this streamlined process. Note that a collaboration center [30] can offer service type process views to match the service type process with the corresponding service offering organizations. Figure 11. Local carrier contract.

As a third VTP element, Figure 11 shows the local contract that the carrier uses internally. Note that unlike the service type process view in Figure 10, the local contract comprises two additional tasks with the labels informing the buyer and charging the bank2. Therefore, the local contract is a subclass of the service type process view with respect to enactment behavior [12, 13], that is, all process view tasks are experienced externally, while the carrier has the option to insert additional hidden steps in a privacy-guaran-

terest for external viewing, and so on.

SAFEONE CHAINS SMART CONTRACT LANGUAGE

To support the VTP scenario of Section 4.1, the current strength of the smart contract lingua franca does not have the required level of utility with respect to the concepts and properties contained. Instead, the goal is to develop an SafeOne Chains Smart Contract Language (QSCL) and compiler that has comparatively better utility for VTP management. High-level QSCL concepts and properties shown in Figure 12. The VTP scenario in Section 4.1 resembles the eSourcing framework for which there is a dedicated

specified for the semantic web domain. We intend to map the concepts and properties of eSML in the blockchain domain to create QSCL along with a language compiler for a new SafeOne Chains virtual machine. Briefly, as we refer the reader to [31] for more details, the properties of Figure 12 are organized along conceptual interrogatives. A QSCL instance resembles a definition of BNM (Figure 9). Figure 12. Properties and concepts of the future SafeOne Chains smart contract language [31]. The WHO QSCL concept comprises constructs to uniquely define contracting parties along with the resources involved and data definitions. The Where concept specifies the business context and also the provisions of the legal context in which a specific smart contract is maintained.

The What concept allows you to define the exchanged values and service type process views (Figure 10) along with the lifecycle definitions for those process views and also for elementary tasks, respectively. Therefore, in the What part of a QSCL instance, you can define several service type process views comparable to Figure 9.

Finally, conjunction constructs are exchange channels specifically defined for the flow of data between organizations. Monitorability builds allow for a flexible definition of dedicated task monitoring that uses a polling or messaging principle.

COMPARATIVEDISCUSSION

Using smart contract ontology [31], we informally examined the suitability of existing robustness versus QSCL that we built for the SafeOne Chains framework. As a general observation, Solidity is a language with a focus on low-level blockchain manipulation commands with Syntax similar to JavaScript. Still, it's possible to import third-party APIs and make calls to external functions.

The so-called external functions in Solidity are part of a smart contract interface that can be called from other contracts and through transactions. Due to Turing's completeness of solidity, in principle it is possible to define cumbersome supports for all the concepts and properties of the smart contract ontology that QSCL embodies.

However, concepts such as pattern-based design, process knowledge, process matching, etc., are not adopted in any way in Solidity. Regarding the invention of cumber some solutions, a recent publication of conference papers [43] uses Solidity to demonstrate the feasibility of monitoring and executing untrusted business processes in smart contracts. It should be noted that the robustness has historically not been supported by formal means of verification, unlike at the beginning of the QSCL design [31].

Without such formally verifiable expressiveness, it is not possible to know before enactment whether a contract is correct and free of security issues. A Security incident related to Solidity has only recently triggered the development and application of verification tools such as Why, Solidifier or Casper that is likely to lead to a change from proof-of-work to proof-of-stake for Ethereum as a whole.

CONCLUSIONS

This whitepaper introduces the SafeOne Chains framework for a smart contract blockchain technology solution. We show SafeOne Chains specific implementation of transaction processing that uses proof-of-stake validation. In addition, SafeOne Chain integrates the Ethereum virtual machine (EVM) along with Bitcoin's unspent transaction outgoing protocol.

Please note that SafeOne Chains EVM is still consistently backward compatible. In addition, SafeOne Chains framework recognises that smart contract lifecycle management is important to support proper security research by collaborating parties. To support SafeOne Chains lifecycle management, the current lingua franca Solidity lacks suitability.

Consequently, SafeOne Chains emerging framework requires a new smart contract language with an improved utility. The adoption of proof-of-stake in SafeOne Chains constitutes a considerable saving of computational effort over the Ethereum alternative that still uses proof of work.

https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/

http://why3.lri.fr/

https://hack.ether.camp/idea/solidifier: Formal Robustness Verification Programs

http://www.coindesk.com/ethereum-casper-proof-stake-re-write-rules-blockchain/

While Ethereum also plans to adopt proof-of-stake, it's unclear when such a new version will be released. Also the use of unspent transaction outputs is more scalable compared to Ethereum account management. In combination with simple payment verification, SafeOne Chain is already developing a smart contract mobile device solution.

While the non-scalablenon-scalable Ethereum solution does not allow for mobile solutions, SafeOne Chain aims to achieve democratized and highly distributed proof-of- stake transaction validation with its mobile strategy.

The SafeOne Chain framework has a clear understanding of the quality criteria that future developments must satisfy. Regarding functional requirements, SafeOne Chain plans to develop an application layer for smart contract lifecycle management. Most importantly, such lifecycle management is important for investigating collaborating parties to reduce security breaches such as those Ethereum recently experienced, resulting in multiplemultiple hardforks of the latter.

The value transfer protocol for information logistics at SafeOne Chain comprises a business network model for choreographing several collaborating organizations. The latter can provide services with on-premises contracts that must match the specified runtimeruntime behavior of the service type process views in the enterprise network model. With a multi- layered smart contract management layer, collaborating parties protect the privacy of their trade secrets that represent a competitive advantage by hiding extension steps in local contracts. In summary, the SafeOne Chains framework recognizes that smart contracts are sociotechnicalsociotechnical artifacts that must also take into account the quality requirements essential to achieve widespread adoption by users. Continuous real-life industry projects with SafeOne Chains applications result in a continuous collection of empirical requirements. The mobile strategy in support of highly distributed proof-of-stake transaction processing points to a significant breakthrough in the state of the art. Still, SafeOne Chain also recognizes that smart contract lifecycle management requires the development of application layers with a sophisticated front-end user experience that current solutions don't pay enough attention to.

REFERENCES

Text and technology details based on naur exhibit 2022.

1. A.M Antonopoulos. Dominating bitcoins, 2014.

2. I. Bentov, A. Gabizon and A. Mizrahi. Cryptocurrencieswithout proof of work, pages 142-157. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

3. K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy and S. ZanellaB'eguelin. Formal verification of smart contracts: Short paper. In Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, PLAS '16, pp. 91-96, New York, NY, USA, 2016. ACM.

4. A. Biryukov and D. Khovratovich. Equihash: Asymmetric proof of work based on the generalized birthday problem. Minutes of NDSSˆaĂˇZ16, February 21–24, 2016, San Diego, CA, USA ISBN 1-891562-41-X, 2016.

t.me/SafeOneChain

5. B. Bisping, P.D. Brodmann, T. Jungnickel, C. Rickmann, H. Seidler, A. Stuber,¨ A. Wilhelm-Weidner, K. Peters and U. Nestmann. Mechanical verification of a constructive test for flp. In International Conference on Interactive Theorem Proving, pages 107–122. Springer, 2016.

6. O. Bussmann. The Future of Finance: FinTech, Tech Disruption, and Orchestrating Innovation, pp. 473–486. Springer International Publishing, Cham, 2017.

7. C. Cachin. Architecture of the hyperledger blockchain fabric. In Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

8. K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the Internet of Things. ACCESS IEEE, 4:2292–2303,2016.

9. L. Chung, B.A. Nixon, E. Yu and J. Mylopoulos. Non-functional requirements in software engineering, volume 5. Springer Science & Business Media, 2012.

10.K. Croman, C. Decker, I. Eyal, A.E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gun Sirer, D. Song and R. Wattenhofer.¨ On Scaling Decentralized Blockchains, pages 106–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

11.N. Emmadi and H. Narumanchi. Reinforce the immutability of authorized blockchains with keyless signature infrastructure. In Proceedings of the 18th International Conference on Distributed Computing and Networking, ICDCN '17, pages 46:1–46:6, New York, NY, USA, 2017. ACM.

12.R. Eshuis, A. Norta, O. Kopp and E. Pitkanen. Outsourcing of services with process views. IEEE Transactions on Services Computing, 99(PrePrints):1, 2013.

13.R. Eshuis, A. Norta and R. Roulaux. Evolution of process views. Information technology and software, 80:20 – 35, 2016.

14.D. Frey, M.X. Makkes, P.L. Roman, F. Ta¨ıani and S. Voulgaris. Bringing secure bitcoin transactions to your smartphone. In Proceedings of the 15th International Workshop on Adaptive and Reflective Middleware, ARM 2016, pages 3:1–3:6, New York, NY, USA, 2016. ACM.

15.J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami. Internet of Things (iot): A vision, architectural elements and future directions. Future Generation Computer Systems, 29(7):1645 – 1660, 2013.

16. A. Kiayias, I. Konstantinou, A. Russell, B. David and R. Oliynykov. A demonstrably secure proof-of-stake blockchain protocol , 2016.

17. G. Kotonya and I. Sommerville. Requirements engineering: processes and techniques. Wiley Publishing, 1998.

18. L. Kutvonen, A. Norta and S. Ruohomaa. Management of commercial transactions between companies in open service ecosystems. In Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International, pp. 31–40. IEEE, 2012.

19. L. Luu, D.H. Chu, H. Olickel, P. Saxena and A. Hobor. Make smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, pp. 254–269,2016.

20. L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert and P. Saxena. A secure fragmentation protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, pages 17-30, New York, NY, USA, 2016. ACM.

21. J. Marshall. Modeling based on agents of emotional objectives in digital media design projects. International Journal of People-Oriented Programming (IJPOP), 3(1):44–59, 2014.

22. Business Process Model. Notation (bpmn) version 2.0. Object Management Group Specification, 2011.http://www.bpmn.org.

23. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Consulted, 1(2012):28, 2008.

24. N.C. Narendra, A. Norta, M. Mahunnah, L. Ma and F.M. Maggi. Solid conflict management and resolution for collaborations between virtual companies. Computing and Service-Oriented Applications, 10(3):233–251, 2016.

25. A. Norta. Exploration of dynamic collaboration between inter-organizational business processes. Doctoral thesis, Eindhoven University of Technology, Department of Information Systems , 2007.

26. A. Norta. Creation of Smart Contracting Collaborations for Decentralized Autonomous Organizations, pages 3-17. Springer International Publishing, Cham, 2015.

27. A. Norta. Establishing Distributed Governance Infrastructures for the Enactment of Collaborations Among Organizations, pages 24–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

28.A. Norta, P. Grefen and N.C Narendra. A reference architecture for the management of dynamic inter-organizational business processes. Data and Knowledge Engineering, 91(0):52 – 89, 2014.

29.A. Norta and L. Kutvonen. A cloud hub for business process intermedia-tion as a service: a "meetup" platform that supports the discovery of semi-automated partners with background checks for cross-enterprise col-laboration. In SRII Global Conference (SRII), 2012 Annual, pp. 293–302, July 2012.

30.A. Norta and L. Kutvonen. A cloud hub for business process intermedia-tion as a service: a "meetup" platform that supports the discovery of semi-automated partners with background checks for cross-enterprise col-laboration. Srii Annual Global Conference, 0:293–302,2012.

31.A. Norta, L. Ma, Y. Duan, A. Rull, M. Kõlvart and K. Taveter. Properties of the choreography and choreography language of eContractual towards business collaboration between organizations. Journal of Internet Services and Applications, 6(1):1–23, 2015.

32.A. Norta, A.B. Othman and K. Taveter. Conflict resolution lifecycles for the collaboration of autonomous decentralized governed organizations. In Proceedings of the 2015 2Nd International Conference on Electronic Gover-nance and Open Society: Challenges in Eurasia, EGOSE '15, pp. 244–257, New York, NY, USA, 2015. ACM.

33.Aafaf Ouaddah, Anas Abou Elkalam and Abdellah Ait Ouahman. Towards a new privacy-preserving access control model based on Blockchain tech-nology in IoT, pages 523-533. Springer International Publishing, Cham, 2017.

34.E. Paja, A.K. Chopra and P. Giorgini. Specification based on the trust of sociotechnical systems. Data and Knowledge Engineering, 87:339 – 353, 2013.

35.J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2015.

36.M. Rosenfeld. Overview of colored coins. White paper, bitcoil. co. il, 2012.

37.Fr. Sergei. A probabilistic analysis of the nxt forging algorithm. Ledger, 1:69–83,2016.

38.L. Sterling and K. Taveter. The art of agent-oriented modeling. MIT Press, 2009.

39. T. Tenso, A. Norta and I. Vorontsova. Evaluating a new agile method of requirements engineering: a case study. In Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering - Volume 1: ENASE, pp. 156–163, 2016.

40. P Vasin. BlackcoinaˆAˇZs proof-of-stake protocol v2, 2014.'

41. M. Vukoli'c. The search for a scalable blockchain fabric: proof of work vs. bft replication. In International Workshop on Open Problems in Network Security, pages 112–125. Springer, 2015.

42. M. Vukoli'c. The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication, pages 112-125. Springer International Publishing, Cham, 2016.

43. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev and J. Mendling. Monitoring and execution of untrusted business processes using Blockchain, pages 329–347. Springer International Publishing, Cham, 2016.

44. G. Wood.Ethereum: A decentralized and secure generalized transaction ledger.

Ethereum Yellow Paper Project, 2014.