A Blockchain Implementation for Configurable Multi-Factor Challenge-Set Self-Sovereign Identity Authentication

Alex Norta
Tallinn University, Tallinn, Estonia
Johannes Kepler University, Linz, Austria
Dymaxion OÜ, Tallinn, Estonia
alex.norta.phd@ieee.org

Alexandr Kormiltsyn
Dymaxion OÜ, Tallinn, Estonia
alexandrkormiltsyn@gmail.com

Chibuzor Udokwu
University of Applied Sciences Upper Austria, Steyr, Austria
Dymaxion OÜ, Tallinn, Estonia
chibuzor.udokwu@gmail.com

Vimal Dwivedi
University of Tartu, Tartu, Estonia
Dymaxion OÜ, Tallinn, Estonia)
vimal.bncet@gmail.com

Sunday Aroh
Dymaxion OÜ, Tallinn, Estonia
asmelitus@gmail.com

Ignas Nikolajev
Dymaxion OÜ, Tallinn, Estonia
ignas.nikolajev@gmail.com

***Abstract*—Multi-factor challenge-set self-sovereign identity authentication (MFSSIA) is an important part for establishing trust between systems, devices, organizations and humans for the emerging machine-to-everything (M2X) economy. Most systems for identity authentication (IA) are single sign-on (SSO), or have fixed challenge sets of limited degree. Additionally, IA systems are controlled by governments, or corporations that are closely affiliated with government entities. The available systems for self-sovereign IA do not offer the necessary flexible configurability of challenge sets. Based on research publications about a formal MFSSIA protocol, this paper presents a blockchain employing implementation for a running case assuming different smart-contract blockchain systems must be connected for sensitive data exchange. The prototype offers a marketplace for challenge-set creation and the challenge/response-lifecycle management employs decentralized knowledge graphs (DKG) together with oracles for response evaluations.**

***Index Terms*—Blockchain, multi-factor, identity, authentication, self-sovereign**

I. Introduction

Over the last two years, governments have eliminated many fundamental liberties and freedoms of citizens to diffuse identity-authentication (IA) apps for exerting societal control[1]. Once such apps are in place, governments are quickly driven to assign to such IA also social credit-score systems[2] that add further restrictions to members of the public. Furthermore, we observe too the emergence of the so-called machine-to-everything (M2X) economy [13] that is defined as ”the result of interactions, transactions, collaborations and business enactments among humans, autonomous and cooperative smart devices, software agents, and physical systems. The corresponding ecosystem is formed by automated, globally-available, heterogeneous socio-technical e-governance systems with loosely coupled, peer-to-peer (P2P)-resembling network structures and is characterized by its dynamic, continuously changing, interoperable, open and distributed nature. Thereby, the M2X Economy employs concepts such as cyber-physical systems, the Internet of Things, and wireless sensor networks.” Single sign-on IA is, consequently, not adequate for the complex trust-establishment needs in collaborations between diverse systems, devices, organizations and humans.

The foundation for the blockchain-based implementation of the multi-factor challenge-set self-sovereign identity authentication (MFSSIA) application is first a publication [12] in which the requirements are defined and subsequently, a deduced Colored Petri Net (CPN) [10] model formalizes the corresponding MFSSIA lifecycle. This first MFSSIA protocol version we further validate for security flaws [14] to update the requirement sets and deduce again the fortified and formalized MFSSIA protocol. We next explain in [13] why MFSSIA is essential for trust resolution to enable collaborations in a M2X economy, e.g., smart cities, Industry 4.0, e-healthcare, and so on.

The state of the art shows theoretical MFSSIA protocol results exist for which the ideal deployment technology is blockchains [2]. Informally and briefly, blockchains are distributed ledgers of linked blocks that store consensually events in an immutably traceable way. The extension of blockchains with programming languages yield smart-contract systems [9]. For enabling data exchange between smart contracts and the blockchain-external system context, oracles [1] are employed. Given the vast blockchain-technology spectrum, this paper answers the research questions how to implement and deploy MFSSIA with novel blockchain technologies?

This paper answers the question by first giving a hypothetical running case in Section II together with background literature. In Section III, the MFSSIA architecture is described,

[1]https://terviseamet.ee/en/digital-covid-certificate
[2]https://tinyurl.com/scs-italy

followed by Section III-A where we give the blockchain-technology implementation. Next, Section IV provides a prototype evaluation and discussion. Finally, Section V ends this paper with conclusions and future work.

II. Presuppositions

We first present the running case about evaluating cross-blockchain connection in Section II-A. Next, Section II-B gives further literature with concepts that are relevant to follow the subsequent sections.

A. Running case

We assume for the running case in Figure 1, the simple process-aware B2B trade of apples from a farmer coop to a grocery-store chain where individual end customers may purchase the apples for their respective households. As is depicted in the hypothetical running case of Figure 1, apples are one of the trade items for a specific selling price that must be agreed on.

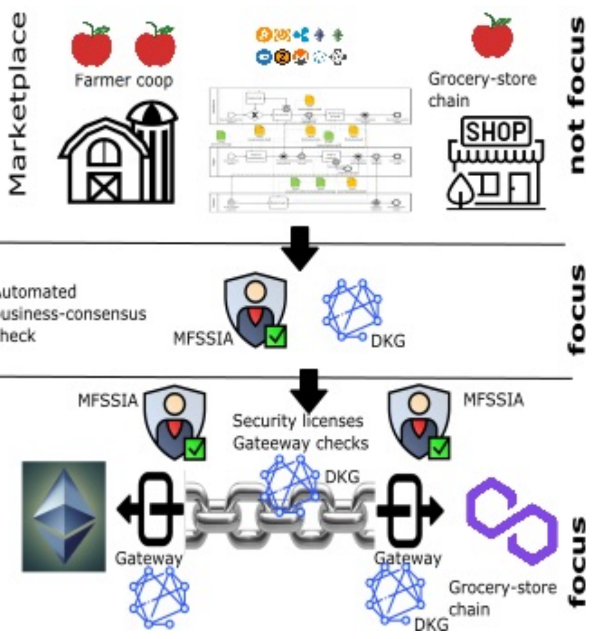

Fig. 1. Conceptual running case for permitting, or denying a connection between different blockchains for sensitive data exchange.

The farmer coop and the grocery-store chain have a high degree of automation including blockchain systems of diverse types for exchanging sensitive business-transaction data. With respect to identification-authentication needs in this business transaction, there are several aspects to be taken into account for establishing trust in such a cross-organizational setting.

For simplicity, we assume that the farmer coop uses a CRM-system that is supported with Ethereum [5] technology, while the ERP-system of the grocery-store chain operates with Polygon[3]blockchain technology in support. For both organizations, it is important to establish trust in the counterparty via responding to issued challeng sets that must be evaluated. In the running case of Figure 1, direct identity authentication of humans and organizations in the marketplace is out of focus, although a possible extension option of the existing dApp. Instead, we assume that the semantics of a marketplace agreement is captures in an instance of a so-called decentralized knowledge graph (DKG) [16] that is explained in the sequel. The price in the marketplace agreement is compared to the values delivered by dedicated oracles [8] that collect such data from the farmer coop and the grocery store respectively. If the three price values match for the apples, we assume a consensus exists about the business agreement and the next challenges must be examined.

The bottom level of Figure 1 focuses on the challenge exploring if a valid security-licence audit exists for the business transaction. Similar to the price-challenge response evaluation, a DKG instance captures the semantic facts about the security-licence audit validity for which an oracle delivers from an off-chain registry the validity period for the specific security audit. If the values match, the final challenge is to check the availability of gateways[4] with the correct configuration to allow for the connection between the different blockchains for sensitive data exchange. This paper reports about the proof-of-concept prototype for automating MFSSIA to resolve the Figure 1 running case.

B. Presuppositions

To expand on the aspect of MFSSIA in trust establishment, Figure 2 shows a very simplified lifecycle between persons. Note that the challenge/response-lifecycle evaluation for identity authentication is further applicable to devices, systems and organizations that may be part of a M2X economy.

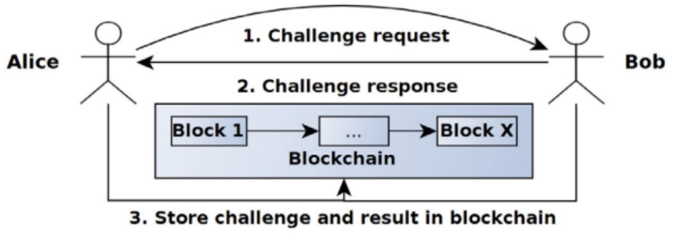

Fig. 2. General validation and authentication procedure for MFSSIA [11].

Thus, either the corresponding entity fails to respond correctly, or is able to successfully complete the challenge by creating a corresponding respective response. The chosen challenge set depends on the use-case, the required strictness level of security and the expected threat level of and for the involved entities. In the hypothetical running case of Figure 1, the configured challenge set culminates in either permitting, or rejecting the cross-blockchain connection between Ethereum and Polygon via a correctly configured gateway.

DKG [15] is an acknowledgement that data is distributed in silos and problematic to combine for knowledge- and business-value extraction. The goal is to build a graph of entities

[3]https://polygon.technology/lightpaper-polygon.pdf

[4]https://ontochain.ngi.eu/content/perun-x

and relationships that are relevant to a specific domain, or organization. Thus an ontology results from gathered and combined data sets for the purpose of knowledge extraction by a reasoner. This created unified view overcomes the problem of distributed data silos to also enable machine-to-machine (M2M) interoperability. In MFSSIA, the distributed data for a complex case, e.g., the business agreement of the marketplace in Figure 1, are combined into a unified view as a DKG instance. Likewise, the challenges combined into a set for response processing are represented as DKG instances to facilitate a semantic processing complementarity.

Since smart-contract blockchain systems are disconnected from the external off-chain information context, oracles [3] are means to manage the data logistics between both domains. The specific differentiation with integrating decentralized oracles is that their users have enhanced trust assurance in that involved workers, i.e., computing entities that provide computing resources, must satisfy a proof-of-contribution (PoCo) [6] consensus protocol for their respective contributions. For example, if the apple trade price the farmer coop and grocery-store chain are prepared to pay respectively is wrongfully delivered, a deposit is lost together with reputation. This situation corresponds to the so-called oracle problem [4] that PoCo[5] tackles.

Expanding on the gateways[6] of the running case of Figure 1 that are out of focus for this paper, the blockchain-scalability problem does not allow for a high degree of transaction processing. Thus, in the running case, we consider so-called state channels [7] that allow for the enactment of complex smart contracts, e.g., elaborate payment transactions.

III. System Architecture

The following discussion addresses the main modules of the MFSSIA system in Figure 3, including the technologies used by third party dependencies. As per the running case in Figure 1, we assumption two external systems must establish a connection. For example, the first system uses Ethereum as an internal blockchain and the second uses Tezos blockchain. We propose iExec as a blockchain for MFSSIA to support smart contracts with decentralised oracles for challenge/response-evaluations. In the current context, there exists a multifactor-authentication module that receives the request from the external systems for authenticating the cross-chain connection. As MFSSIA uses the concept of challenge sets described in further details below, the authentication module has connections with different decentralised oracles that provide information stemming from outside the iExec blockchain for challenge/response evaluation. In our context, the number of oracles is limited according to the running case and includes oracles for checking the business contract-, security licence- and getaway information.

For the knowledge repository, we propose using DKG provided by OriginTrail[7]. We use an Amazon Web Services (AWS) ec2 Linux instance for the DKG test-node setup. Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. The communication between iExec[8] oracles and the DKG test node is performed by the MFSSIA DKG REST client that is deployed to the AWS cloud in another ec2 Linux instance. Furthermore, a SpringBoot with Docker supports the DKG REST client implementation. SpringBoot is a Java framework that simplifies the creation of applications. Docker is an open platform for developing, shipping, and running applications to remain separate from the infrastructure for faster software delivery.

In Figure 3, we provide the component overview for MFSSIA architecture. There are two decentralised applications (dApps) to establish a connection with each other. In the iExec cloud, we have 4 components: smart contracts for multi-factor authentication and three iExec oracles for retrieving the data for business contract-, security licence- and gateway verification. The smart contract for multi-factor authentication uses an RPC interface that is exposed to the external systems. The usage of RPC is a standard way of communication with smart contracts by off-chain applications. Oracles use REST API to retrieve the required information from a DKG node. This communication occurs with the support of a DKG rest client that encapsulates the logic for building and sending the HTTP requests to DKG for receiving and parsing the response in JSON format. The DKG node contains the domain knowledge that includes business contract-, security licence-, gateway- and challenge sets. This domain knowledge is based on the MFSSIA ontology that is mapped to a JSON structure supported by the DKG. The challenge-set marketplace is a separate front-end application deployed outside the AWS infrastructure.

A. Technology Stack of Implementation

The smart contracts for MFSSIA are implemented in Solidity as part of Ethereum. Polygon is designed to render transactions on the Ethereum blockchain much faster and cheaper than on the main network. Specifically, the Polygon network offers a second layer (Layer 2) solution to more flexibly pre-process transactions ahead of the Ethereum network. Two different types of chains can be built in the Polygon ecosystem: offline chains and secure chains. Autonomous chains can have their own consensus models and are therefore less secure than networks using the Ethereum consensus model. We use the iExec Oracle Factory to enable interoperability between Ethereum and Polygon networks as well as the Decentralised Knowledge Graph (DKG) components. iExec integrates cloud-resource sellers and -providers.

As an asynchronous event-driven JavaScript runtime, Node.js[9] is designed to build scalable network applications. Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes

[5]https://tinyurl.com/trustandagents
[6]https://perun.network/
[7]https://origintrail.io/
[8]https://iex.ec/
[9]https://nodejs.org/en/

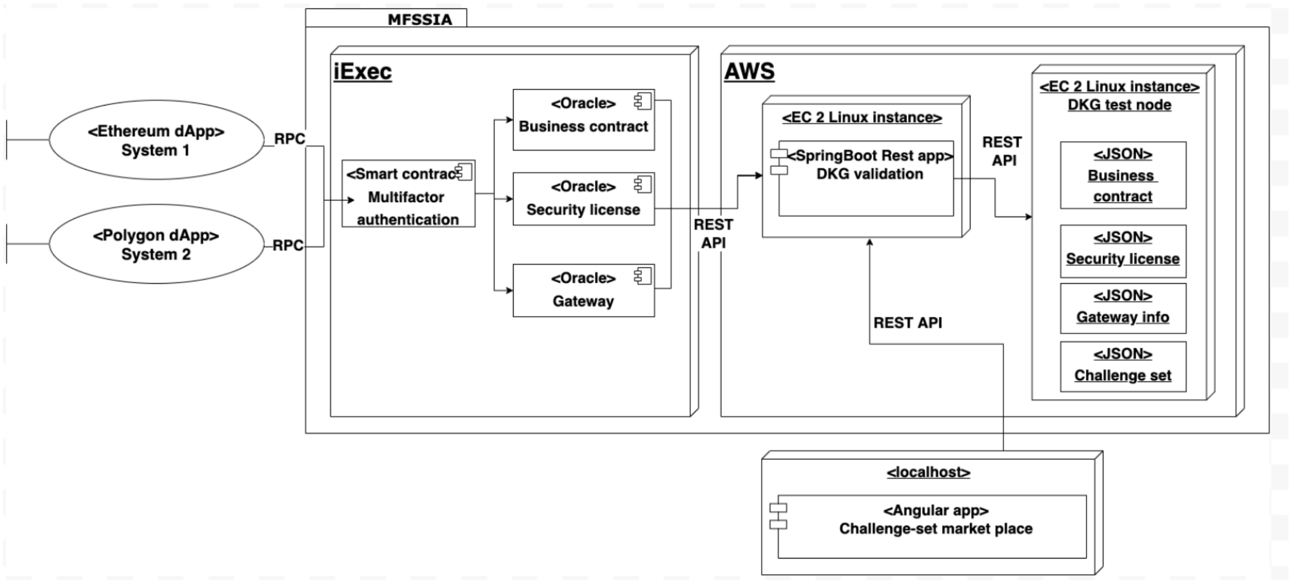

Fig. 3. Architecture of the blockchain-based MFSSIA implementation.

JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command-line tools and for server-side scripting, running scripts server-side to produce dynamic web-page content before a page is sent to the user's web browser. In MFSSIA, we use Node.js for both front- and back-end off-chain components.

Angular[10] is a TypeScript-based free and open-source web application framework led by the Angular Team at Google by a community of individuals and corporations. The Angular framework introduces the single-page application architecture that enables a simple and clear separation of a front- and back-end business logic implementation. The front-end implementation with the Angular framework communicates with the back-end with the HTTP REST protocol. Thus, the back-end component (REST API) can be implemented in MFSSIA in any technology that supports the REST endpoints with the implementation.

The Java Spring Framework[11] (Spring Framework) is a popular, open-source, enterprise-level framework for creating stand-alone, production-grade applications that run on the Java Virtual Machine (JVM). The Spring Framework allows one to set up a Spring-based application with minimal configuration. In MFSSIA, we use Spring Boot for the simplicity of Java code in the implementation of DKG REST API component. Spring Boot renders developing web application and micro-services with the Spring Framework faster and easier due to three core capabilities, i.e., auto-configuration, an opinionated approach to configuration, and the ability to create standalone applications.

For an extensive technology-stack overview, we refer interested readers to the extended technical report[12] for details that also comprises a complete installation guide. There are several publicly available bitbucket repositories available for downloading the MFSSIA source code in MFSSIA-Authcoin[13], the related challenge-set marketplace[14] and the DKG-related code[15].

B. Behavioural MFSSIA system description

There are two types of operations that can be performed in the MFSSIA system. The first one is the deployment of a business contract that contains the challenge sets for a connection and the second operation is the authentication of the connection. We outline the sequential activities and interactions between the users and components in the MFSSIA system that result in these two operations. Figure 4(a) shows the sequence of activities that results in the deployment of a business contract in the MFSSIA system and Figure 4(b) shows the sequence of activities that results in the authentication of a connection with the MFSSIA system.

1) Deployment of business contracts: To create and deploy a new business contract, any of the system users (service provider, or -consumer) accesses the challenge-set marketplace user interface (UI) to create a new business contract. Figure 5 shows the data model of the business contract. First, the user provides information about the actors and their roles in the contract, followed by specifying the duration of the contract. The type of contract is then specified along with the set of contractual obligations. For the evaluation performed in the latter part of this paper, the sample contract deployed contains two roles, *service producer* and *service consumer*. The contract type is product purchase with obligations on *product name*, *quantity* and *price*. The duration for the delivery of the product is specified in *months*.

[10]https://angular.io/
[11]https://spring.io/
[12]https://tinyurl.com/MFSSIAtechstack
[13]https://bitbucket.org/alexnorta/mfssia-authcoin/src/main/
[14]https://tinyurl.com/bitbucketmaster
[15]https://bitbucket.org/alexnorta/mfssia-dkg-authcoin/src/main/

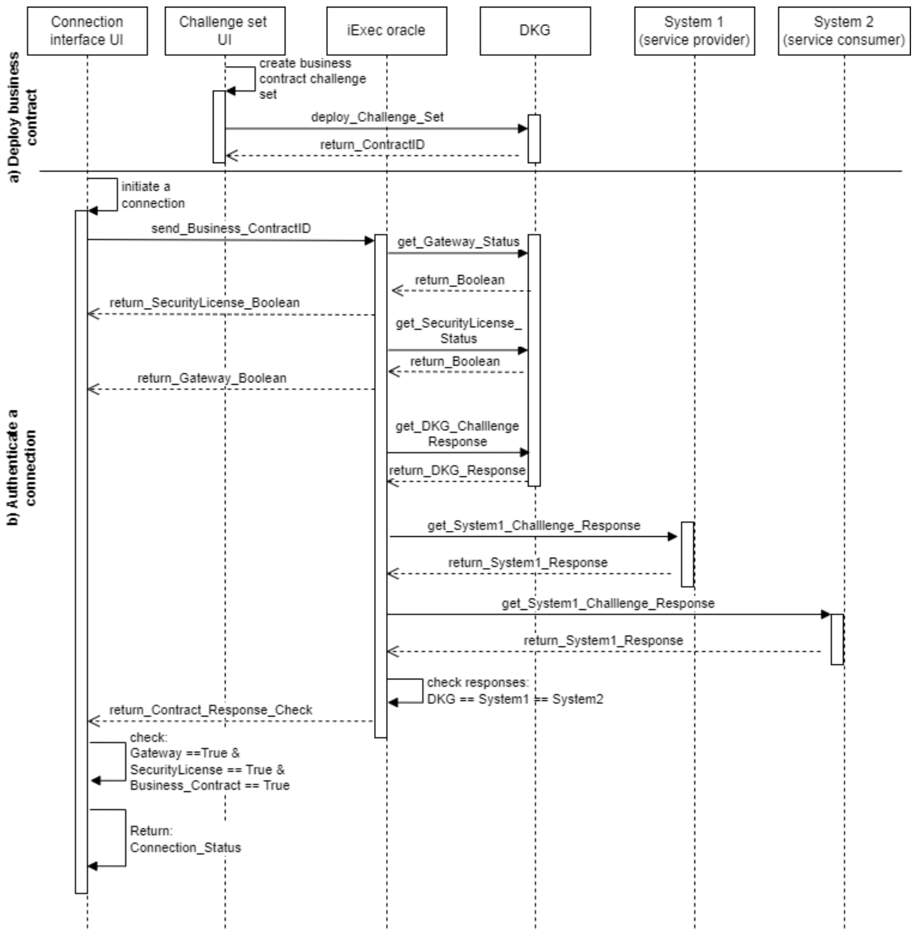
Connection interface UI
Challenge set UI
iExec oracle
DKG
System 1 (service provider)
System 2 (service consumer)
a) Deploy business contract
create business contract challenge set
deploy_Challenge_Set
return_ContractID
b) Authenticate a connection
initiate a connection
send_Business_ContractID
get_Gateway_Status
return_Boolean
return_SecurityLicense_Boolean
get_SecurityLicense_Status
return_Boolean
return_Gateway_Boolean
get_DKG_Challlenge Response
return_DKG_Response
get_System1_Challlenge_Response
return_System1_Response
get_System1_Challlenge_Response
return_System1_Response
check responses: DKG == System1 == System2
return_Contract_Response_Check
check: Gateway ==True & SecurityLicense == True & Business_Contract == True
Return: Connection_Status

Fig. 4. MFSSIA challenge-set deployment and authentication operations.

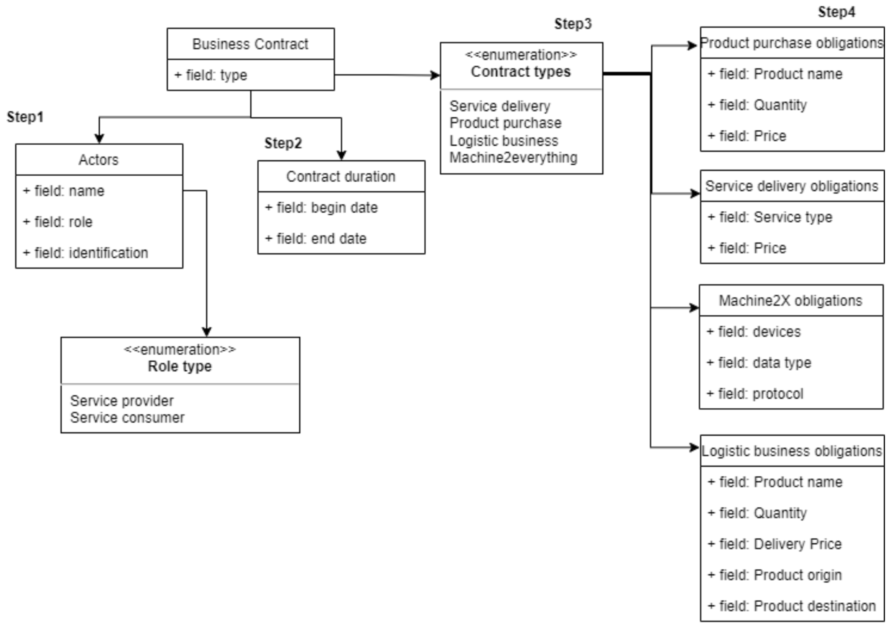

Fig. 5. MFSSIA business-contract data model.

Once the information about the contract is provided, the contract is then deployed as an instance on the DKG network. The *contract ID* number is returned to the user that deploys the contract.

2) Authentication of connection: To authenticate a connection in the MFSSIA system, a connection request is first initiated by either user of the system (service provider, or -consumer) with providing the *business contract ID*. The connection interface transmits the contract ID to the iExec oracle component. The oracle component initiates a request to acquire information about the status of the security license and gateway information between the two parties that wish to establish a connection. The statuses of the security license and gateways respectively for the two systems are then returned back to the connection interface.

The oracle component initiates a request to acquire responses from the DKG component and the separate systems for service providers and service consumers about the properties specified in the business contract. The oracle then evaluates the responses from the three components to check if the returned business-contract properties are the same. The status of the check is returned to the connection interface.

The connection interface performs a final check to verify if the boolean information on the status of the security license, gateway and business are *true*, or *false*. If the final check returns true, the connection interface displays a green notification approving a business-contract authentication between the two systems. Otherwise, the connection interface displays red to disapprove business-contract authentication between the two systems.

IV. Evaluation and Discussion

The running-case demonstration of Figure 1 with the proof-of-concept prototype is available in a video recording[16]. The initial part focuses on evaluating function-test cases. Not included here due to space limitations, the extended technical report contains a table with detailed test-case results. Briefly, the test setup assumes a smart contract ID issuance and the specific price for a type of apples in a set quantity must arrive after a given time interval.

The extended technical report describes both valid- and invalid test cases. Valid test cases ensure that users can perform appropriate actions when using valid data. Invalid test cases are performed to try "breaking" the software by performing invalid (or unacceptable) actions, or by using invalid data. We consider three inputs 1) DKG Input 2) System 1 input, and 3) System 2 input as per Figure 3 . Since we have three variables and each can have valid and invalid test cases, the total number of test cases are eight. The extended technical report describes the output and how many lines of code (LOC) are covered through specific test IDs. Since we test the complete system and not specific modules, therefore the coverage is 100%.

Besides the functional testing, also the entire MFSSIA prototype is evaluated. The purpose of this evaluation is to understand the scalability factor of the MFSSIA authentication, therefore, determining the practical use-cases of the system. The MFSSIA system comprises three components that return data to the authentication interface, i.e., DKG, System 1 and System 2. Each of the systems have their individual time delays. Thus, the scalability factor determines the minimal amount of time required for providing valid inputs (responses) for the authentication. The key performance indicator (KPI) for the evaluation is the time lag required for each component (DKG, System 1 and System 2). The time lag is counted from a negative number (-2 mins) to positive number (+2 mins). The negative time implies that challenge-set responses are provided before the connection is initiated. The positive time implies that challenge-set responses are provided after the connection request has been initiated from the MFSSIA connection interface. The expected results are either a successful connection, or an unsuccessful connection. A given case of time lag is benchmarked successfully if the connection interface returns the expected connection results for given inputs.

To reproduce the evaluation, the following input data are considered as expected input: *Contract ID* = 1234, *Price* = 10000, *QUANTITY* = 2, *DELIVERY INTERVAL* = 3, and *PRODUCT NAME* = apple. If the System 1, System 2 and DKG from the marketplace provide the same expected input data then the expected output is green, otherwise the expected output is red. If the output produced by the connection interface, for a given time lag, matches the expected result, then the test result is marked positively for the measured time interval. Otherwise, the test result is marked negatively (-) in the extended technical report.

The test results show that MFSSIA has a latency that does not suggest a use for systems with fast processing time, e.g., for an identity-authentication application in high-frequency financial-trading applications. Instead, a very suitable usecase would be for situations such as B2B-commerce where the identity authentication of the individuals, organizations and systems employed takes a long time due to bureaucracy. For example, if a container of clothes is ordered by a European retailer on Alibaba from a factory in China, for both parties a trusted identity authentication is time consuming, if not impossible. Instead, with MFSSIA where both parties commit to an automated means of identity authentication, the trade risk, invested time and expenses can be reduced considerably.

V. Conclusions and Future Work

The MFSSIA project has its roots in several research publications and a position paper about the M2X economy where it is evident that trust management is important to establish and maintain collaboration between humans, machines, systems, organisations, devices, and so on. In this prior research, we denoted the protocol for the MFSSIA with so-called goal models, Colored Petri Nets and studied the security of the formally defined protocol for further fortification. Consequently, a rather diagnostic understanding has been established

[16]https://tinyurl.com/2ap46em8

this way for the MFSSIA system prior to the proof-of-concept implementation this paper reports about. Decentralised knowledge graphs and oracles using proof-of-contribution are important to integrate into MFSSIA. We use the former to define challenges, responses, and context-definition files for automated processing; for the response evaluation we require the latter as oracles to fetch relevant information from the application context.

For MFSSIA, we have created the foundation for a challenge-set library that we consider to be a marketplace allegorically similar to an app store for trading apps. The vision is to have security experts create challenges under DKG consideration that absorb many other existing systems for identity authentication and verification, e.g., various ID solutions that are either national, or self-sovereign, document verification systems, etc. The evaluation of MFSSIA shows that the latency in identity authentication renders the system applicable for cases where high speed is not required. Instead, less time-critical application scenarios are appropriate such as onboarding devices into existing IoT-systems, or for B2B-commerce situations where an identity authentication may otherwise require several days with costly bureaucratic processes.

There are several limitations of the currently existing first prototype implementation and a lot of future work is required. First, future work should aim to establish a token economy related to MFSSIA system use. In principle, utility tokens have many uses such as for challenges, issuing responses, evaluating these responses with oracles, and so on. All of these points justify the use of utility tokens. Furthermore, aiming for a challenge-set marketplace suggests that NFTs lend themselves for managing the ownership of challenges that can be assembled into sets in a context-dependent way. An NFT expresses ownership of a challenge as a digital twin that facilitates ownership trading and allows for royalty collection.

Second, we aim to develop and grow the challenge-set marketplace with standards that integrate seamlessly with the identity authentication of systems and devices. Further MFSSIA use cases in the Web3 domain are planned to re-establish a viable business- and revenue model for creative-media content production in our age of centrally controlled platforms such as NetFlix. We also aim for further use cases with various ONTOCHAIN projects such as for IoT[17], the management of non-fungible tokens (NFT)[18], reputation-automation integration[19], and so on.

Acknowledgment

The research and development reported in this paper about MFSSIA[20] is funded by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 957338 (ONTOCHAIN: Trusted, traceable and transparent ontological knowledge on blockchain). This project is also partially funded by the Estonian "Personal research funding: Team grant (PRG)" project PRG1641. Finally, partial funding is also acknowledged from the NGI Explorers Program, a European initiative funded under the Horizon Research and Innovation Framework (H2020), Grant agreement number: 825183, Call-identifier: H2020-ICT-31.

References

[1] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, and Davor Svetinovic. Trustworthy blockchain oracles: review, comparison, and open research challenges. *IEEE Access*, 8:85675–85685, 2020.

[2] Omar Ali, Ashraf Jaradat, Atik Kulakli, and Ahmed Abuhalimeh. A comparative study: Blockchain technology utilization benefits, challenges and functionalities. *IEEE Access*, 9:12730–12749, 2021.

[3] Giulio Caldarelli. Real-world blockchain applications under the lens of the oracle problem. a systematic literature review. In *2020 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*, pages 1–6. IEEE, 2020.

[4] Giulio Caldarelli. Understanding the blockchain oracle problem: A call for action. *Information*, 11(11):509, 2020.

[5] Chris Dannen. Solidity programming. In *Introducing Ethereum and Solidity*, pages 69–88. Springer, 2017.

[6] Shifeng Ding, Gangxiang Shen, Kevin X Pan, Sanjay K Bose, Qiong Zhang, and Biswanath Mukherjee. Blockchain-assisted spectrum trading between elastic virtual optical networks. *IEEE Network*, 34(6):205–211, 2020.

[7] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 949–966, 2018.

[8] Jonathan Heiss, Jacob Eberhardt, and Stefan Tai. From oracles to trustworthy data on-chaining systems. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 496–503. IEEE, 2019.

[9] Tharaka Hewa, Mika Ylianttila, and Madhusanka Liyanage. Survey on blockchain based smart contracts: Applications, opportunities and challenges. *Journal of Network and Computer Applications*, 177:102857, 2021.

[10] Kurt Jensen and Lars M Kristensen. Colored petri nets: a graphical language for formal modeling and validation of concurrent systems. *Communications of the ACM*, 58(6):61–70, 2015.

[11] Benjamin Leiding, Clemens H Cap, Thomas Mundt, and Samaneh Rashidibajgan. Authcoin: validation and authentication in decentralized networks. *arXiv preprint arXiv:1609.04955*, 2016.

[12] Benjamin Leiding and Alex Norta. Mapping requirements specifications into a formalized blockchain-enabled authentication protocol for secured personal identity assurance. In *International Conference on Future Data and Security Engineering*, pages 181–196. Springer, 2017.

[13] Benjamin Leiding, Priyanka Sharma, and Alexander Norta. The machine-to-everything (m2x) economy: Business enactments, collaborations, and e-governance. *Future Internet*, 13(12):319, 2021.

[14] Alex Norta, Raimundas Matulevičius, and Benjamin Leiding. Safeguarding a formalized blockchain-enabled identity-authentication protocol by applying security risk-oriented patterns. *Computers & Security*, 86:253–269, 2019.

[15] Bastien Vidé, Joan Marty, Franck Ravat, and Max Chevalier. Designing a business view of enterprise data: An approach based on a decentralised enterprise knowledge graph. In *25th International Database Engineering & Applications Symposium*, pages 184–193, 2021.

[16] Shuai Wang, Chenchen Huang, Juanjuan Li, Yong Yuan, and Fei-Yue Wang. Decentralized construction of knowledge graphs for deep recommender systems based on blockchain-powered smart contracts. *IEEE Access*, 7:136951–136961, 2019.

[17]https://ontochain.ngi.eu/content/ados

[18]https://ontochain.ngi.eu/content/piswap

[19]https://tinyurl.com/ontochainreputation

[20]https://bit.ly/360Bbj0