Blockchain-based Anonymous P2P Trading System

Sina Rafati Niya, Sebastian Allemann, Arik Gabay, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH,
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
Emails: [rafati|stiller@ifi.uzh.ch], [sebastian.allemann|arik.gabay@uzh.ch]

***Abstract*—Data leaks and privacy scandals have been a growing concern of the last decade. Most traditional, *i.e.*, centralized, online platforms require users to register with their personal data, thereby, potentially exposing the user's identity and data to be used for unintended purposes. This work proposes TradeMap as an integrated architecture, designing and enabling an online end-to-end (e2e) trading market place, while supporting anonymous management features. TradeMap addresses the Swiss Financial Market Supervisory Authority (FINMA) regulations by designing a FINMA-complaint Know Your Customer (KYC) platform. Additionally, TradeMap is based on blockchains and employs Ethereum Smart Contracts (SC). Thus, trust and anonymity between the market place and KYC system relies on zero knowledge proof-based SCs used for user identification processes. With this management approach proposed, the user authentication is only verified within the KYC platform, providing a legally valid and fully anonymous online trading platform.**

***Index Terms*—Anonymous Online Trading, Privacy, Blockchains, Ethereum, Smart Contracts, Know Your Costumer (KYC), Swiss Financial Market Supervisory Authority (FINMA).**

I. Introduction

Trading in any form has been and will be an inevitable part of our lives. During last two decades, especially by expansion of the Internet and eCommerce universally, new paradigms of Business-to-Consumer (B2C) and Business-to-Business (B2B) trading have emerged, all determining end-to-end (e2e) cases. As a result of eCommerce and Information Technology (IT) management in general, improvements have been achieved on e-trading to conform with interactions and the engagement of a vast number of use cases and users globally.

One of the most challenging and crucial aspects of eCommerce platforms has been user privacy. Data leaks from large players, such as Facebook, effected millions of users in 2018 [1], and this is a clear indicator that user privacy and confidentiality is still a major concern. Privacy protection of users in data collection, preserving, and sharing in centralized and distributed platforms require a dedicated design w.r.t. to the data flow, especially when leaking the information can lead to the direct or indirect identification of users.

Several privacy-related solutions offer, *e.g.*, Zero knowledge Proof (ZKP) mechanisms by data encryption, Know Your Costumer (KYC) services, or more recent blockchain-based approaches. However, a combination of eCommerce platforms and these solutions needs to meet legal and technical prerequisites to be addressed by design. While establishing trust between administrators and users of trading platforms by default is getting more difficult due to privacy leaks in centralized approaches, enabling a ZKP-based approach even leads to further challenges. Generally, a user's identification in eCommerce and online trading can be delegated to a third party, *i.e.*, a Certificate Authority (CA) or any other authority that provides KYC services. The KYC process has gained importance, since the emergence of anti-money laundering regulations. Thus, KYC has become a necessary basis for financial institutions to interact with clients. Financial institutions are obliged to identify and authenticate new customers to ensure that customers are not involved in illegal activities. Avoiding even the potential of such activities is key, which prevents the deviation of e-trading platforms from healthy and ethical life to black markets, such as silk road [2], [3].

The KYC process starts, when a financial institution is approached by a potential customer. The customer will hand-in typically basic identity information. Respective checks result in an internal document, which serves as certificate that the KYC process has been properly conducted and confirms, whether the application has been approved or rejected. Up to now, KYC systems are based traditionally on a centralized design. However, past observations show that a centralized entity can be a "hot point" for data leaks leading to high costs in case of errors [4]. Instead and to reduce costs while enhancing user privacy, a decentralization of KYC by relying on blockchains (BC) can decentralize the KYC process.

BCs are defined as distributed digital and public ledgers that record transactions immutably and autonomously without the need of a central control [5]. Through a decentralized consensus mechanism, a public BC is "hosted" by distributed nodes and is accessible to anyone. One of the early BCs is Ethereum integrating Smart Contracts (SC) [4]. SCs contain coded statements, persisted within the BC, allowing the triggering of a contract clause automatically, if the dedicated requirement is fulfilled. SCs can enhance efficiency in daily business interactions.

At a first glance, online trading approaches relying on BCs, either for KYC or for the e2e trading, suffer from several deficits: **(1)** Lacking a sufficient and reliable user identification, **(2)** neglecting regulations, *e.g.*, the Swiss Financial Market Supervisory Authority (FINMA), **(3)** lacking of ZKP-based mechanisms leading to privacy, **(4)** using BCs, which by default do not bring privacy as data stored by them can be accessed publicly, **(5)** neglecting costs of storing large data on BCs, **(6)** overloading an SC, and **(7)** missing the use of standardized interfaces.

Therefore, this work introduces as a solution to those deficits

listed "TradeMap", the ZKP-based design and implementation of an e2e trading market place. It enables anonymous e-trading for customers and businesses, who can sell and purchase goods. TradeMap combines the two main parts as KYC and e-trading, based on BCs, which not only manages the user identification in a FINMA-compliant fashion, but also employs a novel approach to use SCs for managing all user interactions with the platform. As such TradeMap prevents any system administrator of a trading market to reveal or even know the user's identity. The user identification and registration in TradeMap is defined as a messaging flow, where as a replacement of traditional interfaces, the connections between the KYC and the e-trading subsystem are managed by SCs for all identity verification requests, thus, leading to the anonymous management capability.

This paper is organized as follows: Section II overviews related work and leads to the design of TradeMap within Section III. While implementation details are explained in Section IV, Section V evaluates TradeMap, and Section VI draws conclusions.

II. Related Work

While the Technical Report [6] details all relevant related work on KYC systems and blockchain-based decentralized trading platforms, this subsubsection summarizes the key observations, which lead to the TradMap design as fully decentralized, yet anonymous management approach enabling e2e e-trading.

A. Know Your Customer (KYC) Verification

Performing a Know Your Customer (KYC) verification is key to interact with new customers, especially within the financial domain [7]. Since the traditional KYC process carries high costs, it is in the interest of both parties (consumer and supplier) to reach an efficient outcome. Amongst others, the following systems have been introduced to optimize KYC processes:

1) IDnow: IDnow is an identity verification platform which provides a wide range of KYC services which comply towards Anti-Money Laundering (AML) as well as the regulations of the corresponding authority. The goal is to provide tailored verification solutions especially in the financial sector but also in eCommerce, ICO & Crypto KYC. IDnow ensures security, legal compliance and a high quality of KYC processes [8]. Two of their identity verification methods are VideoIdent and AutoIdent.

VideoIdent: With VideoIdent a customer's identification can be verified quickly and comfortably using a P2P video chat with a responsible person. While video chatting the customer will be asked a few questions and asked to hold his/her ID into the camera of the laptop or mobile phone and tilt it so that the responsible person on the other side of the call can verify the ID [9].

AutoIdent: IDnow also provides a way to create your digital identity using artificial intelligence (AI) and Face recognition technology. The process is fully automated and there is no need for any ID verification staff [10].

IDnow provides an API with which a user identification process can be requested. To do so, IDnow needs the personal data of the user to be identified. To request an identification process, there are the following possibilities: REST API, Static link on IDnow, Web form or with a GET request. The results can either be downloaded or sent by email as JSON or as ZIP file [11].

2) CB Financial Services AG (CBFS): Similar to IDnow, CBFS also provides a platform for online identification and conclusion of contracts. With the Identification solution CBF-SIdent, the identification process is done via secure video streaming to verify identities and is provided as a "Software as a Service" (SaaS) or as a licence. Already onboarded customers are granted access to the CBFS's identification and signing service. The login credentials can also be integrated into a company, which allows the user to login directly into for example e-Banking services [12]. Raiffeisen, for example, adapted the technology of CBFS in 2016 to digitally onboard their customers.

3) Shufti Pro: Shufti Pro is a British company which provides efficient and accurate digital KYC verification services. It has the ability to verify people around the globe with their artificial and human hybrid technology within 60 seconds [13]. With face verification, document verification, hand written note verification and 2 factor authentication, Shufti Pro cuts costs and reduces risks at the same time [14].
There are two modes of identity verification with the Shufti Pro solution. When identitiy credentials from an end user for KYC is requested directly, it is called On-site VerificatTra-Tradion. Shufti Pro will then identify the user. The Off-site Verification mode is executed when identity verification of verifiable documents, provided by the customer, is requested. The benefit of the Off-site Verification is that there is no need for user-interaction [15].
Integrating Shufti Pro into your mobile applications, modules or online software allows to easily verify users. The possibilities are RESTFUL APIs, Android and iOS integrations or a hosted verification page [16].

4) Tradle: Tradle is an open-source platform allowing financial institutions to onboard their customers with blockchain-based bots. These automate the KYC processes for financial institutions to make them faster and more efficient providing security audits and building a trust provisioning network. Tradle's technology is integrated in the bank's web site. To use it, click on the link and download the App. In the App the client starts a chat with a bank's representative providing required information for the identification process, such as personal details, a selfie and snapshots of the identity documents. The bank's representative then verifies the data gathered and verifies the client. The information is then digitally signed and securely stored on the blockchain. If the client now would like to open an account at another bank, he can start a new chat within the app and share his verified data with them. The goal of Tradle's system is letting the client

transfer trust, not assets, by ensuring that the transferred data is verifiable [17].

5) KYC-Chain: KYC-Chain enables businesses to handle the KYC processes for individuals and corporations more efficiently, using its B2B managed workflow application. By using the distributed ledger technology, the company ensures the privacy of individuals and corporations and their information, but still reaches a high level of transparency in the blockchain. The identity verification is accomplished with algorithmic validity checks of identity cards [18]. The platform ensures secure sharing of verifiable identity claims, data or documents. With KYC-Chain, the onboarding process of new customers can be carried out in a more efficient and trusted way for businesses and financial institutions [19].

6) KYCstart: KYCstart is a proof of concept (PoC) of the consultancy and accounting firm Deloitte providing KYC-as-a-service using blockchain technology. With "regulated KYC added-value service providers", Deloitte's blockchain-based solution could digitally perform KYC checks for clients who want to be onboarded by several financial institutions at once. The project is aiming for more efficient and cheaper customer onboarding by facilitating onboarding processes for financial institutions and creating digital identities for the customers, which can be shared among permissioned platforms and financial institutions. The benefits of the blockchain technology mean that customers and clients can control whom they want to share their company information with. Blockchain-based smart contracts enable them to keep track of the authorizations [20].

7) Comparison: Table 2.3 displays a comparison of the companies discussed in this chapter as well as the proposed method in this thesis with some selected features.
The first column shows that there are many ways to deploy KYC processes online. The auto identification requires technical tools such as AI and facial recognition which was out of scope for the platform developed in this thesis. Most of the established companies providing KYC-as-a-service don't use the blockchain technology. *Tradle* and *KYCstart* are aiming on having a digital identity of the institutional client on the blockchain and sharing the verified data among permissioned financial institutions. The goal of the method proposed in this thesis is to store user data on the blockchain and additionally to allow financial institutions to send verification requests through a smart contract.

Company	Identification	Blockchain	Access	Usage
IDnow	Video, Auto	x	API	Mobile Web
CBFS	Video	x	SaaS or License	Web
Shufti Pro	Auto	x	API, Integration	Mobile
Tradle	Chat	✓	App, Web	Mobile Web
KYC Chain	n/a	✓	Solution	Web
KYCstart	n/a	✓	n/a	n/a
Proposed Method	Video	✓	Smart Contract	Web

TABLE I
COMPARISON OF REQUIREMENTS

B. Decentralized Trading Systems

1) Blockchain Platform for Industrial Internet of Things (BPIIoT): In their article [21] the authors propose to create a blockchain based marketplace for industrial manufacturing services. The purpose is for the user to be able to interact directly with the manufacturing machines through a dApp, which, according to the authors, has following benefits [21]:

- **Decentralized & Trustless**: Peers do not need to trust each other, given the premise that they trust the underlying consensus algorithm. Also, the intermediate party is eliminated, which on the one hand enhances the trustworthiness of the solution and on the other, lowers the overall costs of any transaction, since the third party would always want to be rewarded for their mediation service.
- **Secure & Auditable**: The complex and secure cryptographic protection provides a safe environment for trading services, without any additional effort to provide it otherwise. Also, the public ledger of a blockchain network is viewable and auditable by any given party; making disputes much less likely, as all information is public.
- **Autonomous**: Since all the involved parties interact with each other autonomously and automatically, not only should the process be noticeably sped-up, but also the need for a trusted third party should be eliminated.
- **Resilient & Scalable**: Due to the fact that a blockchain network is distributed over all full nodes of the network, it has no single point of failure and is therefore resilient to failure as a whole. It is also scalable, given that each additional node in the network also adds computational power to the network.

Although this use case is not exactly identical to the one proposed by this paper, it has similar characteristics, such as the transaction between two parties for a compensation, and therefore provides valuable input for the construction of a well-contemplated smart contract.

2) Exergy: The Exergy power grid by LO3 Energy intends to build a local power grid that allows prosumers and consumers to trade energy directly and process payments through a specifically developed blockchain and the ERC20 compliant tokens used by it. Once set up, the system works without any human interaction whatsoever: The prosumer buys an Exergy compliant measuring device to provide the smart contract with reliable data on how much energy surplus he is able to consistently contribute to the network. He then registers through a mobile app and stakes some native tokens to his measuring device [22].

On the consumer side, the mobile app is also downloaded and the desired amount of energy specified. The consumer then pays the prosumer in fiat currency, which concludes the entire transaction [22].

Although the interaction with the smart contract might be very similar to the use case presented in my paper (*i.e.* seamless interaction with the smart contract through a user friendly front end application), the payment is still executed in fiat currency, which indisputably will involve a third party, such as *e.g.* a bank. The added value to my thesis hence lies in the smart contract and application design.

3) Blockchain P2P Marketplace: In the paper [23] an Android application running on the decentralized Ethereum network is proposed. It is a P2P marketplace that allows users to buy, sell, rent or lend out physical objects using smart contracts and cryptocurrency to process and secure the payment [23]. This use case is very similar to the one presented in this paper, with the exception of it being an Android application instead of running on a web browser.

Objects are added to the marketplace by the seller and thereby listed for the potential buyers to see. The description as well as the price and other necessary information regarding the objects are stored off-chain and are only introduced to the blockchain when a contract is made between two parties. Every transaction creates its own contract, which means a new instance of a smart contract is deployed to the blockchain for each item sold. Seeing as the contract not only stores information about the object, but also the addresses of buyer and seller of an item, the deployment of said contract is triggered by the buyer's decision to buy, and will be voided as soon as the buyer confirms the receipt of the object, as in that moment all the contract's conditions are met [23].

4) Decentralised Sharing App: A very similar project to the Blockchain P2P Marketplace is the decentralized sharing application presented in this paper [24], where objects are registered via a QR code and can then be rented out to any other party that has access to the application. The goal here too, is to eliminate the TTP (*trusted third party*) so as to allow faster and cheaper rentals. The smart contract design choices made in [24] are very similar to the ones in this application, because all the objects and transactions concerning the latter are stored on a single, central smart contract, with no need for a single contract for every transaction.

5) Comparison: Table II shows a comparison of the features these 4 projects offer, as well as the ones they lack. On the basis of the information gained, this projects intends to combine all of these advantages. Since the blockchain is by definition decentralized, as are all the related projects, making this a must-have for this project. All projects with exception of *Exergy* work with cryptocurrencies, making the whole payment process fast and trustworthy. This feature will also be implemented. The environments the projects chose to use for their application varies between *Web Apps*, *Mobile Apps* and *Android Apps*. Here the most compatible solution - a Web App - will be chosen. The unique selling proposition of this project will be the complete anonymity that users will have using the platform. This will be achieved by connecting to the KYC platform developed by [25], meaning no personal data shall ever come in contact with the application developed in this thesis.

TABLE II
COMPARISON OF DISTRIBUTED TRADING APPROACHES

Project	**Uses cryptocurrency**	**Industry**	**Complete Anonymity**	**Environment**
BPIIoT	yes	Industry	no	Web App
Exergy	no	Energy	no	Mobile App
P2P Marketplace	yes	Private Marketplace	no	Android
Decent. Sharing App	yes	Private Marketplace	no	Web App

III. REQUIREMENTS AND DESIGN

Based on the identification of major security requirements to be met, the TradeMap's design is outlined below, including the Swiss FINMA-compliant video identification requirements and relevant SCs.

A. Security Requirements

A major security concern is the topic of *Cross-Site-Forgery-Requests (CSRF)*. Every since Web applications with user authentication have been developed, this risk has to be handled with great care. *CSRF* attacks are basically executed by tricking the browser to send unwanted HTTP requests. That means, that the attacker does not have to steal the user's identity, but only has to place an URL, *e.g.*, in an *img* tag, which the user accesses on the malicious Web site, *e.g.*, by sending a request to delete the user's account [26].

To solve this issue, the use of *JSON Web Tokens (JWT)* is proposed. Using *JWT* allows the system to use a *CSRF token*. The server will require this token to validate any request. Setting a *CSRF token* does not fully eliminate the risk of *CSRF* attacks. Cookies are required, since cookies are only accessible from the domain they were set and, therefore, making them unreadable for the malicious Web site. The *CSRF token* will be stored in the cookie so that *CSRF* attacks will fail, because they cannot read the *CSRF token* value and, therefore, are not able to complete requests [27].

B. Video Identification Requirements of KYC Systems

For the video identification process, the platform must comply with the regulations FINMA has published in 2016 [28]. Hence, the technical aspects of the FINMA publication were considered carefully. In Table III all relevant articles are listed, which had been considered.

To meet the key requirements of (a) all relevant data need to be submitted, (b) consent to use those, and (c) pictures are taken, the following design was prepared: A user friendly form was designed to collect relevant data to be stored in the data base making it available for users at any time. To prevent users from spamming the platform, an email-based verification is executed, once the user registers proving that (a) the user knows her/his email address and (b) the user is in possession of the password to access the respective account.

TABLE III
REQUIREMENTS FOR VIDEO IDENTIFICATION FROM FINMA CIRCULAR 2016 [28]

Text	ID
"The financial intermediary structures the online onboarding process in such a way that the contracting party fills out the details required under Articles 44 and 60 AMLO-FINMA electronically and transfers them to the financial intermediary before the audio-visual identification interview takes place. [...] In addition, the financial intermediary checks the information gathered during the onboarding process against the information contained in the contracting party's identification document."	F-C1
"The financial intermediary must obtain the contracting party's explicit consent to conduct the video identification and audio recording before starting the video interview."	F-C2
"During the video transmission, the financial intermediary takes photographs of the contracting party [...]."	F-C3
"The financial intermediary also reviews the authenticity of the identification documents by using a machine to read and decrypt the information in the MRZ [...] The financial intermediary checks that the decrypted information matches the other information in the identification document and the data provided by the contracting party during the onboarding process."	F-C4
"Only official identification documents issued by the relevant country can be accepted for this process. Moreover, the documents must have an MRZ and optical security features, such as holographic-cinematic marks or printed elements with latent image effects."	F-C5
"The identity of the contracting party is to be verified by means of a TAN or another similar method."	F-C6
"Each identification process must be documented. The photographs of the identification document and contracting party must be filed and archived along with the audio recording of the entire identification process.."	F-C7
"The financial intermediary will stop the video identification process if the picture or sound quality does not enable unambiguous identification of the contracting party, the financial intermediary identifies evidence of increased risks, or there are any doubts regarding the authenticity of the identification document or the identity of the contracting party."	F-C8
"For contracting parties in the form of legal entities or partnerships, the financial intermediary must obtain an electronic extract either from the database of the relevant registration authority or from a trustworthy privately managed directory. The extract may also be submitted to the financial intermediary outside the video identification process."	F-C9

For the video identification itself the usage of WebRTC APIs (Application Programming Interface) is proposed, because the APIs provide secure and interrupt-less RTC (Real-Time Communication) capabilities. Security concerns can be eliminated by using *CSRF* tokens via a valid user id to authenticate. In this way, no not-entitled user can access a channel. As described in FINMA Art. 16, the platform can make use of an email-based verification.

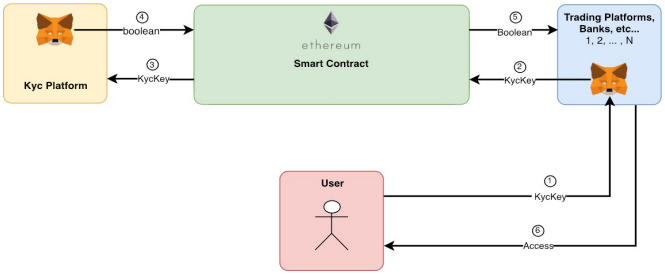

Fig. 1. Trading Platform's Identity Verification Requests to the KYC Platform Through Smart Contract

C. Smart Contracts (SC)

The Smart Contract (SC), which has to be developed brings requirements such as accessibility, anonymity and usability. The accessibility requirement for users could be met by having a browser extension installed allowing the user to create a new wallet. Since sending transactions to the Ethereum network always cost an amount of Ether, it is important that the browser extension has access to the user's wallet to transfer the funds.

Blockchain transactions are transparent and visible for all users in the network. Since the goal is to allow user's complete anonymity, it must be considered to make smart contract requests anonymous. A design proposal is to use a unique hashed key (*KYC key*) with which users could register themselves at connected platforms anonymously while providing consent that the user's data could be shared with the platform. To prevent that the *KYC key* is copied once a smart contract request was processed, it is suggested that the mentioned key be hashed.

As mentioned above, each transaction made to the Ethereum network requires a small payment of Ether amount. This amount is called *gas*. Another aspect, which has to be considered, is the block time of the Ethereum network. It takes some time to process the transaction because every transaction has to wait for a new block to be created. In a system enabling user-interaction with the BC this can decrease the usability due to long transaction times. If the amount of Ether the user is willing to pay per unit of *gas* increases, it could ensure that the transaction is mined quicker. Unfortunately, this results in a trade-off between costs for transactions, the so called *gasPrice*, and block time of transactions meaning the usability decreases when the costs are reduced and vice versa. Therefore, the goal is to find a well balanced *gasPrice*.

Before the video identification, F-C1 needs to be fulfilled. Table IV describes these requirements [29].

D. TradeMap's Design Overview

TradeMap's design relies on the goal of accessing anonymous interaction in the e2e trading. Such anonymity needs to be managed to be legally accepted. for such a management, TradeMap designs two connected systems. On one hand, a FINMA-compliant KYC system, and on the other hand, a secure trading platform that enables BC-based e2e trading. The combined and inclusive platforms are only connected via SC (cf. Fig. 1). This feature enables other trading platforms to connect the designed KYC system without the need for any modifications in their existing system.

As illustrated in Fig. 4, users enter the registration phase (cf. Fig. 2) and verify their email addresses. In the third step beneficiaries (cf. Fig. 3)of the KYC key e.g., managers of the

TABLE IV
REQUIREMENTS FOR VIDEO IDENTIFICATION FROM AMLO-FINMA [29]

Text	ID
When starting a business relationship with an individual, the financial intermediary collects the first name, last name, date of birth, address and nationality from the contracting party.	A-F1
When starting a business relationship with a legal entity, the financial intermediary collects the company name and the company address.	A-F2
The financial intermediary takes note of the electronic copy of the power of attorney and checks the identity of the individuals representing the legal entity.	A-F3
The written declaration of the contracting party about the beneficial owners must contain the first name, last name, date of birth, address and nationality.	A-F4

company will be declared. Terms and conditions which are all set based on FINMA have to be accepted by user in the fourth step. Fallowed by video verification and One Time Password (OTP) activation to approve and register one KYC user. At the end, user KYC key shall be stored in the SC for future use.

The complementary e2e trading is designed to be used at first place as a market place for trading objects and products. In the next step, it is designed to be used as an inventory for the registered products. With TradeMap, history of objects' past owners can be verified. As it is described in Section IV-D, SC preserves all the information needed to identify one object. This feature empowers anonymous management of the whole system against frauds by enabling buyers with pre-purchase checking of objects' past from the BC.

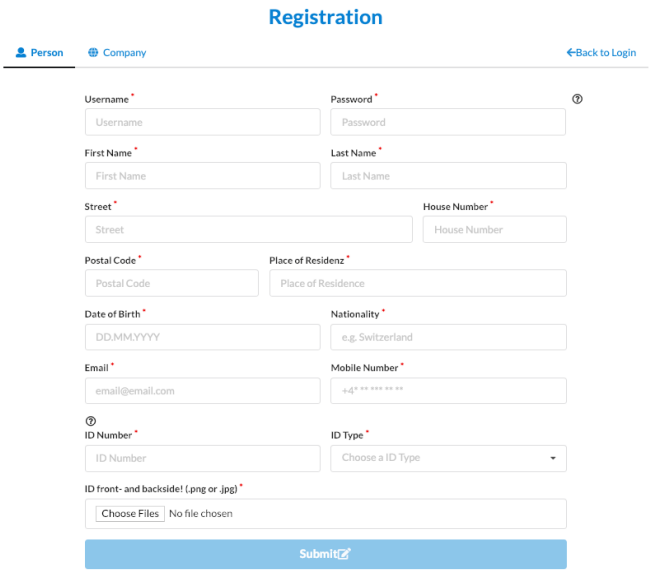

Fig. 2. User Registration Form in the KYC Platform

IV. IMPLEMENTATION

TradeMap consists out of the KYC platform, the e-trading platform, and their related SCs. These together provide the anonymous management features needed to ensure that trust can be provided, too.

A. KYC Platform Implementation

For the video identification process, all requirements of the FINMA circular 2016 [28] such as "The financial intermediary

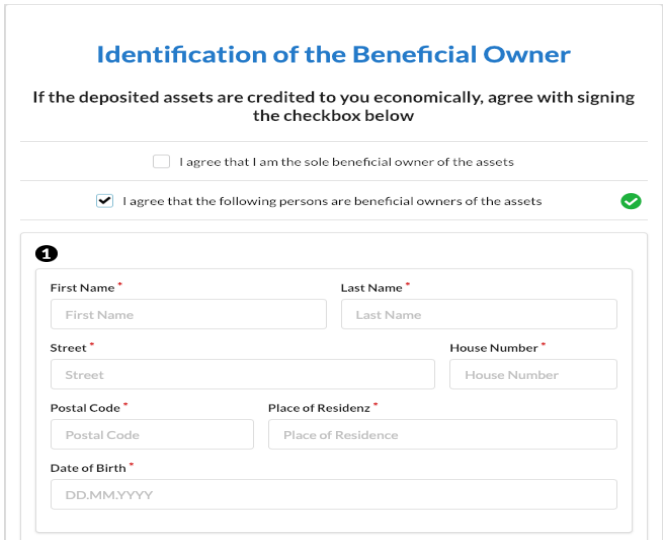

Fig. 3. Beneficial Declaration Within the KYC Platform

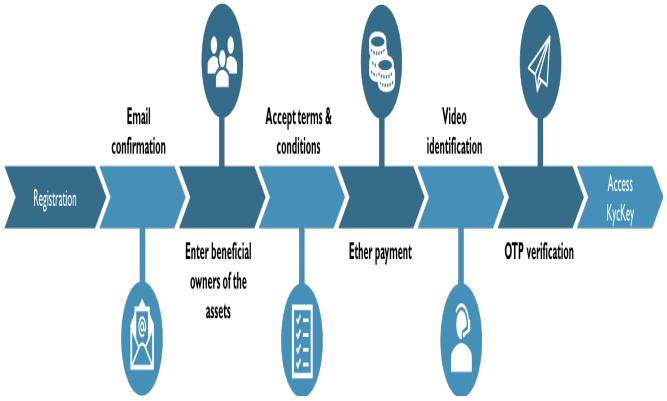

Fig. 4. KYC System Interactions for a Verification Request

structures the online onboarding process in such a way that the contracting party fills out the details required under Articles 44 and 60 AMLO-FINMA electronically and transfers them to the financial intermediary before the audio-visual identification interview takes place. In addition, the financial intermediary checks the information gathered during the onboarding process against the information contained in the contracting party's identification document." and "The financial intermediary must obtain the contracting party's explicit consent to conduct the video identification and audio recording before starting the video interview." have to be met. Also the Anti-Money

Laundering Ordinance [29] was considered. Figure 1 shows each step a user has to go through to claim his KYC key.

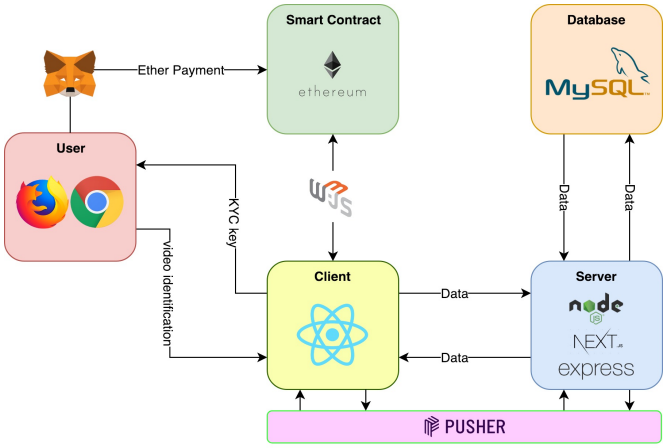

Fig. 5. KYC Platform Implementation Architecture

The main function of the SC is to handle the verification requests for users registering with an external platform using their KYC key. The user will, once verified, have access to his KYC key through the profile page. With this KYC key the user registers at a platform. Figure 4 outlines the respective verification request. Thus, SCs enable the tracking of these requests back, which is important for analyzing unauthorized user accesses from external platforms.

B. KYC Smart Contract

SCs are implemented using Solidity [30]. The KYC platform listens for an event (cf. Listing 1, lines 6-9) and once emitted from an external platform, the KYC platform receives the data of the event. The platform receives a response from the KYC platform in form of a Boolean value (line 10).

Verification Requests: The verification request with a trading platform is shown in Fig. 1. Since the KYC key is confidential and data transmitted through SCs is public, the trading platform hashes the KYC key with *sha3* before sending it to the KYC platform. The function *verify* is triggered by the user's Ethereum address. The hashed KYC key and the external platform's Ethereum address are sent in order to prove which user has registered at which platform (line 11).

The *KycListen* event will be emitted (line 13) and a defined Ether value is transferred to the KYC platform. After that the data is sent to the KYC platform, listening to the *KycListen* event. Once received, the KYC platform checks the database for the KYC hash entry. In turn, the function *answer* (line 16) will be triggered containing a Boolean value (hash of the KYC key was found or not). Only if the KYC platform did sent these data, no error message will be shown (line 27). The *PlatformListen* event is emitted to ensure that the trading platform listens for the Boolean value. Furthermore, the SC transmits the Boolean value to the trading platform and the trading platform checks this value to grants the user access to the platform.

Storing User Approvals for Identity Verification: Since the user's consent was received upon his/her registration at an external platform with the KYC key, within TradeMap it is possible to prove that the user has agreed that his/her data is being requested by tracking back verification requests made. Thus, costs can be saved and the SC is kept simple.

Payment: Users can pay Ethers to the KYC platform for the identification service. This transaction is handled by the SC function *payKYC* (line 21-23). This function transfers Ethers to the *kycAddress* (line 22).

Storage of Hash: In order to prove the user's identity at any time, the hash of the user data is be stored on the SC in a bytes32 array (line 4). Thus, the function *storeHash* takes the hash and pushes it into the array (line 23-24). By calling the function *getHashes*, the array of all hashes stored will be returned (line 25-27). This is necessary, should the KYC platform's database be out of order or in maintenance. Thus, a verification request would not receive any response from the KYC platform.

```
01 pragma solidity ^0.4.17;
02 contract KYCVerification {
03 address private kycAddress =
04     0x4399C3daed9b7cce56b7Edd4157FA3bDe3385d2A;
05 bytes32[] array;
06 event KycListen(
       string kycKey,
       address platformAddress,
       address sender);
10 event PlatformListen(bool confirmed);
11 function verify(string kycKey,
     address platformAddress) public
     payable { emit KycListen(kycKey,
     platformAddress, msg.sender);
     kycAddress.transfer(msg.value);}
16 function answer(bool confirmed)
     public {require(
     msg.sender == kycAddress,
     "You cannot do this");
     emit PlatformListen(confirmed);}
21 function payKYC() public payable {
       kycAddress.transfer(msg.value);}
23 function storeHash(bytes32 newHash)
         public {array.push(newHash);}
25 function getHashes() public
    view returns(bytes32[]) {
27  return array; }
}
```

Listing 1. SC handling interactions between application and blockchain

C. Trading Platform Implementation

Figure 6 shows the three possible data flows and their chronological execution. The green arrows indicate the data flow occurring when an object is being listed on the platform. Step 1 is triggered by the seller (user) clicking the "*Sell*" button in the front end, thereby sending data to the client. The client sends this data to the server (step 2), which finally writes it to the database (step 3). Concurrently to the data being sent to the server, it is also persisted on the BC (step 2, copy). Arrows colored in red show the order in which the components operate to display existing data to a user. At first (step 1), the data is read from the database by the server and forwarded to the

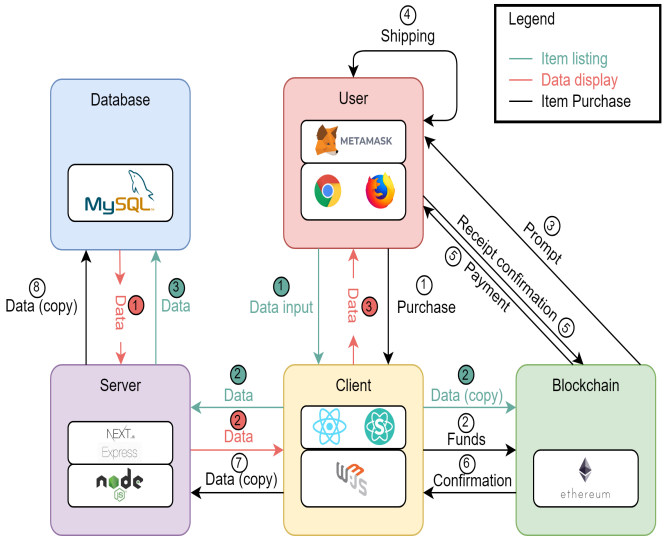

Fig. 6. E2E Trading Platform Architecture

client (step 2). The process in concluded by displaying the data retrieved to the user (step 3).

Black arrows depict the item purchase process, which is triggered as soon as a buyer (user) clicks the "*purchase*" button and pays for an item (step 1). Subsequently, the client sends the funds to the BC (step 2), which will then prompt the seller (step 3) to ship the item to the buyer (step 4). Once the buyer receives the item, he/she confirms the receipt, which will release the funds to him/herself and the seller (step 5). Meanwhile, the BC sends a confirmation back to the client, which passes all relevant data through to the database via the server in order to update it (step 6, 7 & 8).

D. Trading Platform Smart Contract

To manage all transactions and the data desired to be kept on the BC, listing 2 shows the underlying full SC to explain its key functionality.

```
1  pragma solidity ^0.4.25;
2  contract TradingContract {
3  address adminAddress =
4      0xa9C3f40905a01240F63AA2b27375b5D43Dcd64E5;
5  struct Object {
       address owner;
       string uid;
       address buyer;
       string status;
       uint sellerCollateral;
       uint buyerCollateral;
       uint price;}
13 event PurchaseListen(bool confirmed);
14 mapping (uint => Object) public objects;
15 mapping (string => address[]) ownerHistory;
16 function registerObject(string uid, uint objectId
       ,address owner, string status) public payable {
       objects[objectId] = Object(owner, uid,
       address(0), status, msg.value, uint(0),
       uint(0));
       ownerHistory[uid].push(owner);
       emit PurchaseListen(true);}
23 function tradeObject(address buyerAdd,
       uint objectId, string newStatus) public payable
    {objects[objectId].buyer = buyerAdd;
     objects[objectId].status = newStatus;
     objects[objectId].buyerCollateral = msg.value/3;
     objects[objectId].price = msg.value/3 * 2;
     emit PurchaseListen(true);}
30 function confirmTransaction(uint objectId) public
     {require(msg.sender == objects[objectId].buyer,
         "You need to be the registered buyer to
         confirm" + "this transaction");
      address owner = objects[objectId].owner;
35    address buyer = objects[objectId].buyer;
      objects[objectId].owner = buyer;
      objects[objectId].buyer = address(0);
      uint adminFee = objects[objectId].price / 100;
      uint sellerAmount = objects[objectId].price
      +objects[objectId].sellerCollateral-adminFee;
      objects[objectId].sellerCollateral = uint(0);
      objects[objectId].price = uint(0);
      uint buyerCollateral = objects[objectId].
          buyerCollateral;
      objects[objectId].buyerCollateral = uint(0);
      emit PurchaseListen(true);
      ownerHistory[objects[objectId].uid]
        .push(objects[objectId].owner);
      owner.transfer(sellerAmount);
      buyer.transfer(buyerCollateral);
      adminAddress.transfer(adminFee);}
51 function getObject(uint objectId) public view
      returns (address owner, string uid,
      address buyer, string status,
      uint sellerCollateral, uint buyerCollateral,
      uint price ) {owner = objects[objectId].owner;
      uid = objects[objectId].uid;
      buyer = objects[objectId].buyer;
      status = objects[objectId].status;
      sellerCollateral = objects[objectId].
          sellerCollateral;
      buyerCollateral = objects[objectId].
          buyerCollateral;
      price = objects[objectId].price;
      return (owner, uid, buyer, status,
          sellerCollateral,
          buyerCollateral, price);}
64 function releaseSellerCollateral(uint objectId)
      public {
       objects[objectId].owner.transfer
       (objects[objectId].sellerCollateral);
       objects[objectId].sellerCollateral = uint(0);}
78 function recoverItemFunds(uint objectId) public {
       require(msg.sender == adminAddress,
       "You need to be an admin to do this");
       adminAddress.transfer(
       objects[objectId].sellerCollateral
       + objects[objectId].buyerCollateral);}
84 function viewOwnerHistory (string uid) public
     view returns (address[] owners){
      return ownerHistory[uid];}
}
```

Listing 2. E2E Trading Application Management by SC and Blockchain

Admin Address: The constant variable *adminAddress* is set to the administrator's Ethereum address. It is used to allocate a certain percentage of an item's selling price (in this case here 1%) to the administrator's account as a service fee. It also allows fora check, if an entity attempting to reclaim funds for a specific object is truly the administrator.

Data Storage Structure: Between lines 8 and 16 of Listing 2 the struct *Object* stores all necessary data regarding an item. This object is created in the *registerObject()* function (line 16) and assigns all information known at that point in time.

Because this function is called while listing an object, there is no buyer yet, nor is there a *buyerCollateral* nor a final price of the object. This may be edited before the actual purchase.

Storage and History: Lines 16 and 21 illustrate how all objects are stored and how their *ownerHistory* is tracked. Mappings allows to assign a value or struct to a *key*. For this objects mapping a *uint* serves as the key, pointing to an *Object* inside of the mapping. This is done by specifying the object id (uint) in square brackets (line 18). As the arrow inside of a mapping's bracket suggest, the key can point to one and only one struct or value, while there can be multiple keys pointing to a single object or value, determining a *many-to-one* relationship.

With the help of this mapping and by using the *objectId* sent from the client-side, all current information regarding an object, as displayed in the *getObject()* function (line 51), can be retrieved.

Inside the *ownerHistory* mapping, a *string* (which will contain the *uid* of an object) points to an array of *addresses*, which will be appended every time a new owner is assigned to this object. This process occurrs on line 33 in the *registerObject()* function and on line 30 in the *confirmTransaction()* function. Since the string is pointing to an array, the construct of*ownerHistory[uid]* will have all characteristics of an array, allowing for the use of functions such as *.push()* to append the list of owners.

Event: Line 19 defines the *event* that can be emitted to a function caller as a return value inside a function that does not have a actual return. This way the client will know whether or not the transaction has succeeded or not. An event consists out of a single Boolean value *confirmed* and an example is stated in line 35.

Trade Object: The function that will be called during an object's trading life cycle is the *tradeObject()* function. It assigns the address and the status to the object (lines 25 & 26) and distributes the *msg.value* (all funds sent with the transaction) according to the pricing policy. The buyer must submit a payment of 150% of the item's price at the time of purchase, whereof 50% will be held as collateral. Therefore, a third of the *msg.value* is assigned to the *buyerCollateral* attribute, and two thirds to the *price*.

Confirm Transaction: Once a buyer has received an item from the seller, he will click the "*I have received the item*" - button, which calls the *confirmTransaction()* function. At first the function checks, whether the person calling this function is truly the buyer, meaning the *msg.sender*-address has to be equal to the one stored inside an *Object*. Line 31 shows the *require* statement handling this authenticity check. Subsequently, the owner and buyer addresses are assigned to local variables (lines 34 & 35) in order to save them for the funds transfer. The object's attributes are updated accordingly, meaning the buyer is now the new owner, and the object's buyer is set to a zero address. This is completed before any amounts are calculated (lines 36-42) and sent in order to prevent users from calling the function twice and being payed twice doing so.

Release Seller Collateral: In case of an object's deletion before a purchase, the seller will have the right to the collateral held by the SC. The *releaseSellerCollateral()* function does release the collateral, in addition to setting the item's *sellerCollateral* value to zero after the refund.

Redeem Lost Funds: Should anything misfire on the client's side, thereby making necessary functions to redeem pending funds from the SC unavailable, the administrator has the possibility to call the *recoverItemFunds()* function on line 78, transferring all funds regarding a specific object to the administrator's account for further processing. Similarly to the *confirmTransaction()* function, the *msg.sender* is audited on line 113 in order to guarantee, that only the administrator is permitted to call this function.

Owner History: A user can view the complete owner history of a specific object. The *viewOwnerHistory()* function (line 120) returns the array of addresses in the *ownerHistory* mapping of *uid* – provided by the client-side – is pointing to.

E. TradeMap Front End

front end of the TradeMap is designed to include a broad range of objects in different categories (cf. Fig. 7).

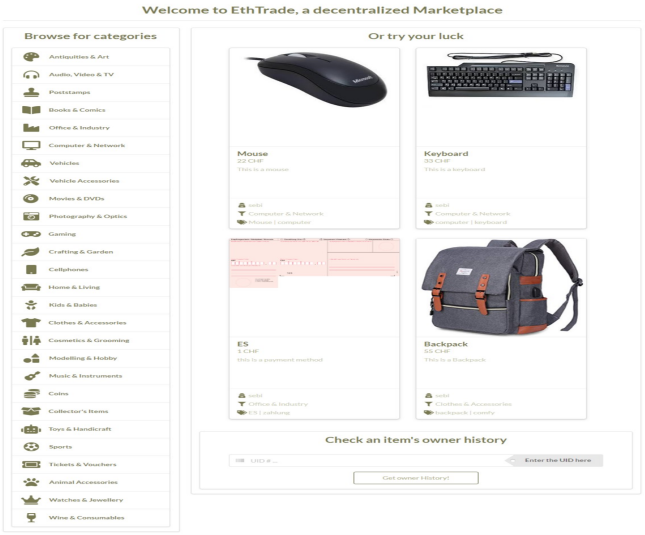

Fig. 7. E2E Trading Platform Front End: Main Page View

Producers or objects owners such as businesses and brands, have to establish an account in the trading system which is only possible after a successful identity verification from KYC platform. Users can register objects as shown in Fig. 8.

Users of the TradeMap are able to view one object's information in detail as of shown in the Fig. 9.

V. Evaluation

TradeMap needs to provide the functionality to provide high user privacy by being anonymous management and participation in the system. As the TradeMap's design outlined in Section III, the main deficits of traditional online e2e trading platforms have been overcome with w.r.t:

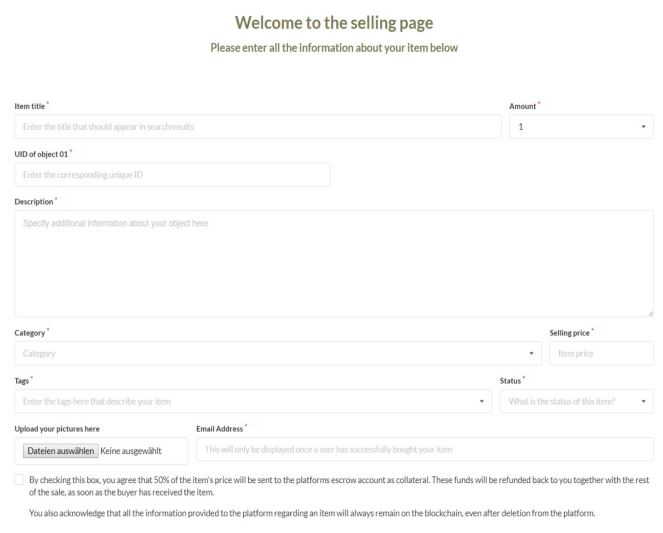

Fig. 8. E2E Trading Platform Front End: Object Registration

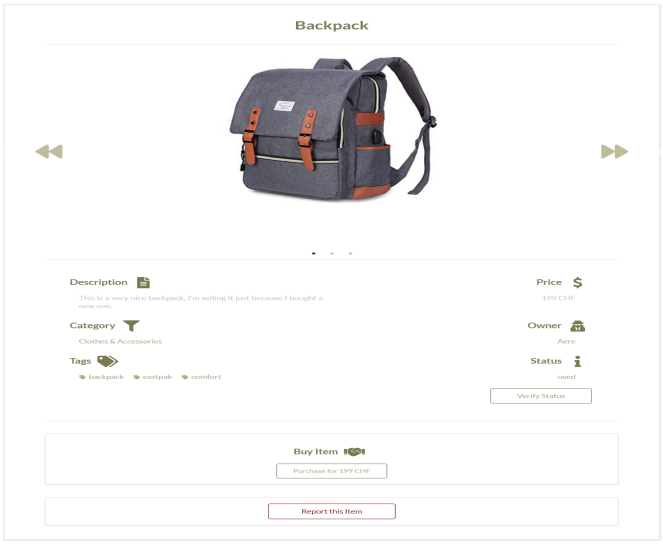

Fig. 9. E2E Trading Platform Front End: Object Detail View

(1) Reliability of user identification approach: The proposed system here defines all the steps to be taken by users and KYC administers for a reliable platform. User information stored in the BC, with use of SC, and only after the KYC process is fully over and user identification was successful.

(2) Adhering to the regulations: In TradeMap, all user identification steps were based on the goal to follow Swiss regulations, i.e., FINMA. Adaptation of such steps in each country may be required to meet the local regulations. However, the main idea of relying on the legal requirements needed step by step consideration on design of KYC system and w.r.t. user-TradeMap interactions.

(3) Lack of ZKP-based mechanisms leading to privacy: The administrators, having full access to the contents of the databases, could potentially abuse their power for unlawful operations/interactions. This becomes more critical in TradeMap design as the database stores confidential data, such as the KYC keys. To confront the consequences, users' passwords are stored in a hashed and encrypted manner, preventing the administrators from logging in with other agents' credentials and executing transactions on behalf of a specific user. Also, a strong password for the authorization is required. These measures also hinder the administrator and external attackers from breaking into user profiles. Finally, regarding the user's KYC key security: Since the administrator has no access to KYC keys of the users, attackers could only access user's profile, in the unlikely event of getting hold of the password and the user name of a user at the same time (via a phishing attack).

(4) User privacy vs public accessibility of BCs: Using BC does not by default bring privacy as data stored by them can be accessed publicly. TradeMap KYC-SC stores the hashed data in the BC. Thus, even by having access to a user KYC key and public key his/her identity cannot be revealed. Trading smart contract is also do not store user specific information. Only the purchasable objects' information is being stored in the BC as an inventory of that object's history.

(5) Costs of operation: On one hand, for an operational set-up, any user should make a one-time payment in Ether. As of June 2019, transferring 0.1 Ether to the KYC platform costs approx. 10.45 CHF. Since the user can enter a personalized value or also choose not to pay, the effective costs cannot be determined in the general case. On the other hand, the administrator has recurring costs, which arise by answering verification requests and storing the respective hash on the SC. Funds needed for identity verification and storing traded products' information on SC needs to be set by system owners, thus these amounts can be set very low numbers that keeping the costs of using this TradeMap for end users to a minimum. Regarding the demanded Gas amounts from Ethereum i.e., the costs of handling functions with SC, answering a verification request will cost the administrator 0.000115 Ether, approx. 0.012 CHF. The costs to store the hash on the SC amounts to 0.000244 Ether, equalling approx. 0.03 CHF. Finally, for processing the verification within the trading platform, the administrator receives a compensation of 0.01 Ether.

Upon summing up these incoming and outgoing flows, they cancel each other out meaning that the administrator has practically no costs or even has earned a small amount. In the long run, assuming that (a) users register at more than one platform and (b) they pay at least a little amount for the video identification, the administrator will be compensated for the KYC processes provided. Thus, TradeMap can be operated cost neutral.

(6) Overloading data storage on SC: A sub-optimal BC-based solution may propose to treat BC as a data base. In contrast, TradeMap stores only the "must have" information in the BC.

(7) Waiting time: To measure the performance of the platform, the waiting time needs an investigation. Ethereum transactions take approx. 15 s until the transaction is mined. This can - potentially - decrease the degree of user friendliness

in the platform. However, users only have to wait once for a transaction to be processed, namely at the payment of the Ethers. Thus, if a user decides to send a payment to the platform, his/her waiting time is approx. at 20 s. The time can vary, since the it depends on how busy the network is. Note, the user also faces in "traditional" KYC or e-commerce systems a wait time upon accessing a video chat or payment transactions to be transferred. Since the administrator is in charge of calling users, it can occur that there many users wait in the calling pipeline. Thus, dependent on the popularity waiting times will differ. But by offering multiple channels multiple administrators could conduct video identifications.

In case that the administrator transmits transactions, the waiting time for him increases: *e.g.*, the storage of hashes takes about 35 s. By answering verification requests additional 16 s could arise. These times are realistic for the administrator, since both transactions are sent from within the administrator page allowing the administrator to keep on working, while waiting for the transaction to be processed. Moreover, a growing amount of hashes stored in the SC array increase the costs. Since hashes are dropped from the array, when a user has to re-register, these costs will compensate each other.

A similar case occurs during a video transmission with a user. The OCR (Optical Character Recognition) scanning and MRZ (Machine-readable Passport) validation takes time to perform. Typically, the administrator has the possibility to trigger the OCR scan, while identifying his counterpart, because the scan takes up to 1 min depending on (a) how the image was cropped and (b) the image quality.

(8) Security: It is crucial to be aware of security concerns, since sensitive data is stored. As the platform is publicly accessible, the need of securing system's and user data is obvious. As explain in the Section III-A the *Cross-site Request Forgery (CSRF)* attacks need to be taken under consideration. These attacks aim to trick the browser to send unwanted HTTP requests. Using *JSON Web Tokens (JWT)* allows the system to use *CSRF tokens*. Consecutively, the server will require this token to validate any request so that *CSRF* attacks will fail. As also experienced by performed tests, this type of attack is prevented as the attacker cannot read the *CSRF token* value and, therefore, are not able to complete requests [27].

The second concern was SQL injection, which determined the system's vulnerability concerning the SQL database. Malicious SQL commands are prevented by escaping the user's input before entering them to the query [31]. Malicious input sent via a from to the server is prevented by the *sqlstring* package for npm, which escapes all input [32].

In addition, security aspects of SCs are prevented by security inspections, in which certified specialists rule out security breaches based on code evaluations before the deploying of the SC onto the main Ethereum network. The only sensitive parameter that is stored on the database is a user's Ethereum address, which on its own does not grant access to an Ethereum account, due to MetaMask's locally orchestrated key handling.

VI. Summary and Conclusions

Designing multi-component systems based on blockchains, can only make IT systems better, if all supported features of traditionally centralized applications, such as KYC ones or eCommerce, are offered unchanged and improvements as of new features and functionality is being added. In this sense TradeMap integrates BC with care, undertaking several steps to meet FINMA regulations of user registrations. TradeMap performs such a registration without storing all user data in the BC. With respect to the trading platform, the permission for verifying the user's identity is only done by a method, which itself does not need any identity disclosure. Thus, before and after the identity verification the trading platform's administrator will not receive any identity-related information. In that sense the anonymous management feature had been achieved for the e2e trading market place.

The goal of this work was to develop an open source and smart contract-based know your customer (KYC) platform which provides video identification for users and enables them to register themselves at a connected platform completely anonymous with their proof of verification (*KYC key*).

Firstly, a secure registration platform was built to enable users to register themselves in a secure and user-friendly way. A database was connected to the platform to store and read the user's data. Since the platform requires authorization from users and contains strictly confidential data, the platform had to be secured with so called JSON Web tokens (JWT). Storing these tokens in the cookies makes the platform secure against CSRF attacks. Also, with help of these tokens users cannot access pages, for which they are not entitled. To prevent users spamming the platform with fake email addresses, email-based verification was implemented to get proof that the user knows his email address and is in possesion of the password.

In a second step, the implementation of the video identification process was approached. To ensure that the video identification complies to the regulations published by FINMA, the requirements had to be identified. All requirements were implemented using a number of external libraries as well as WebRTC to guarantee a interrupt-less and secure onboarding process. The goal of the video identification process is to provide a KYC service to the user, who has to pay for this service. After successful completion the user receives a proof of verification from the platform. This proof of verification, the *KYC key*, is a unique hash, which allows the user to register at external platforms anonymously.

Furthermore, a smart contract was developed to handle verification requests from external platforms. Since the trading platform allows registrations with only the *KYC key*, they need a proof that the user holding the key really has identified himself. In this case the trading platform can request a verification of the KYC key by hashing it and sending it to the KYC platform, which verifies the hashed key and sends back a message saying if the KYC key is valid. Another functionality of the smart contract serves the purpose of giving the users the possibility of validating their KYC key at any time, even

if the platforms database is out of order. By hashing the first name, last name, identity card number and the KYC key of the user and storing it in the smart contract at the end of the identification process, the user is able to enter the mentioned data in a page of the platform to validate his KYC key. The last smart contract functionality handles the user payment. Since it is assumed, that all users are in possession of an Ethereum account, the payment is transacted with Ether.

The great advantage of video identifications is that on-site onboarding at financial institutions becomes unnecessary. This limits the amount of KYC processes to a minimum for the institution and for their customer. Also, the need of physical paper documents is no longer needed since the customer can sign the contracts digitally with the digital identity created in the identification process. Another key benefit is the cost reduction for the financial institution and their customer. On the one hand users only have to identify themselves once until they receive new identification documents. This means that users only have to pay once. The financial institution also only has to onboard the customer once and receives a small compensation for the KYC process for each verification request made from an external platform.

In summary, the prototype developed om this work offers a new approach on how to provide KYC processes more efficient for financial institutions and platforms which require an identification. All the requirements defined at the beginning of the work as well as the requirements published by FINMA are met. The developed system performs reliably in terms of its functions.

REFERENCES

[1] BBC News, "Facebook Scandal 'Hit 87 Million Users'," https://www.bbc.com/news/technology-43649018, April 4, 2018.

[2] Wikipedia, "Silk Road (Marketplace)," https://en.wikipedia.org/wiki/Silk_Road_(marketplace), June 13, 2019.

[3] R. Campbell, "Block Explorer News; The Silk Road: A Story of Bitcoin, Drugs, and the DarkWeb," https://blockexplorer.com/news/silk-road-timeline-bitcoin-drugs-dark-web/, December 1, 2018.

[4] J. Parra Moyano and O. Ross, "KYC Optimization Using Distributed Ledger Technology," *Business & Information Systems Engineering*, Vol. 59, No. 6, pp. 411–423, Dec 2017. [Online]: https://doi.org/10.1007/s12599-017-0504-2

[5] T. Bocek and B. Stiller, *Smart Contracts - Blockchains in the Wings*. Tiergartenstr. 17, 69121 Heidelberg, Germany: Springer, jan 2017, pp. 169–184.

[6] S. R. Niya and B. Stiller, "A Blockchain-based Anonymous P2P Trading System," https://files.ifi.uzh.ch/CSG/staff/Rafati/TMap-IFI-2019.04.pdf, Zürich, Switzerland, June 2019.

[7] NorthRow, https://www.northrow.com/kyc-checks/, November 26, 2018.

[8] IDNow, "IDNow: Regulation," https://www.idnow.io/regulation/identification-kyc/, November 27, 2018.

[9] IDNow.io, "IDnow, VideoIdnent, Seamless Online Identification Using an Agent-assisted Video Chat Process in Compliance with The Money Laundering Act," https://www.idnow.io/products/video-verification/, November 27, 2018.

[10] ——, "Idnow: Auto-identification," https://www.idnow.io/products/idnow-autoident/, November 27, 2018.

[11] ——, "Idnow: Api," https://www.idnow.io/development/api-documentation/, November 27, 2018.

[12] CB Financial Services AG, "Cbfsident," https://www.c-b-f-s.com/services/cbfsident/, November 27, 2018.

[13] shuftipro.com, "Shufti pro," https://shuftipro.com/, November 27, 2018.

[14] S. P. Identity-verification, https://shuftipro.com/identity-verification/, November 27, 2018.

[15] S. P. M. of Verification, https://shuftipro.com/modes-of-verification, January 25, 2019.

[16] S. P. Integration, https://shuftipro.com/integration/, November 27, 2018.

[17] R. De Feniks, "Tradle: KYC on Blockchain," http://www.digitalinsuranceagenda.com/108/tradle-kyc-on-blockchain/, January 24, 2019.

[18] KYC-Chain, "Efficient KYC Management," https://kyc-chain.com/, November 27, 2018.

[19] B. Expo, "Selfkey kyc-chain," https://blockchain-expo.com/global/partners/selfkey-kyc-chain/, November 27, 2018.

[20] A. Lielacher, "Blockchain-Powered KYC-as-a-Service Solution," https://bitcoinmagazine.com/MISCs/deloittes-regtech-offering-blockchain-powered-kyc-service-solution/, January 24, 2019.

[21] M. V.K and A. Bahga, "Blockchain Platform for Industrial Internet of Things," journal of Software Engineering and Applications,Vol.9 No.10, October 2016, 2016.

[22] LO3 Energy, "Exergy Business Whitepaper ," https://exergy.energy/wp-content/uploads/2018/04/Exergy-BIZWhitepaper-v10.pdf, January 4, 2019.

[23] S. R. Niya, F. Schüpfer, T. Bocek, and B. Stiller, "A Peer-to-peer Purchase and Rental Smart Contract-based Application (PuRSCA)," *it- Information Technology*, Vol. 60, No. 5, pp. 307–320, oct 2018. [Online]: https://www.degruyter.com/view/j/itit.ahead-of-print/itit-2017-0036/itit-2017-0036.xml

[24] A. Bogner, M. Chanson, and A. Meeuw, "A Decentralised Sharing App Running a Smart Contract on the Ethereum Blockchain," in *Proceedings of the 6th International Conference on the Internet of Things*, ser. IoT'16. New York, NY, USA: ACM, 2016, pp. 177–178. [Online]: http://doi.acm.org/10.1145/2991561.2998465

[25] S. Allemann, design and Implementation of a Know Your Costumer (KYC) Platform for Blockchains-Integrated Application; https://www.csg.uzh.ch/theses/theses.html?thesisid=253, 2019.

[26] G. Sironi, "Cross-Site-Request-Forgery Explained," https://dzone.com/MISCs/cross-site-request-forgery, January 29, 2019.

[27] garageinsidegarage.com, "Secure JWT Authentication Against Both XSS and XSRF," https://blog.garageinsidegarage.com/secure-jwt-authentication-against-both-xss-and-xsrf-vue-js-django-rest/, January 29, 2019.

[28] FINMA: Circular, "Video and Online Identification," pp. 07–16, https://www.finma.ch/en/~/media/finma/dokumente/rundschreiben-archiv/2016/rs-/finma-rs-2016-07.pdf?la=en, November 7, 2018.

[29] AMLO-FINMA, "Verordnung der eidgen ossischen finanzmarktaufsicht über die bekämpfung von geldwäscherei und terrorismusfinanzierung im finanzsektor," https://www.admin.ch/opc/de/classified-compilation/20143112/index.html, January 24, 2019.

[30] Solidity Documentation, "Solidity," https://solidity.readthedocs.io/en/v0.4.24/, November 27, 2018.

[31] Veracode.com, "What Is SQL Injection?" https://www.veracode.com/security/sql-injection, January 30, 2019.

[32] F. Geisend Orfer, "Sqlstring Package for Npm," https://github.com/mysqljs/sqlstring, January 30, 2019.