A privacy preserving identification protocol for smart contracts

Francesco Bruschi
DEIB
Politecnico di Milano
Milano, Italy
francesco.bruschi@polimi.it

Tommaso Paulon
DEIB
Politecnico di Milano
Milano, Italy

Vincenzo Rana
KNOBS srl
Milano, Italy
vincenzo.rana@knobs.it

Donatella Sciuto
DEIB
Politecnico di Milano
Milano, Italy
donatella.sciuto@polimi.it

2021 IEEE Symposium on Computers and Communications (ISCC) | 978-1-6654-2744-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISCC53001.2021.9631501

***Abstract*—If on one hand the possibility of using pseudonymous identities is an important feature of blockchains and smart contracts, on the other hand official identity can be required in some applications to comply with regulations such as Know Your Customer and Anti Money Laundering. These regulatory compliance issues are usually dealt with either through "custodial" approaches, which however neutralize the decentralization of these systems, or with "whitelisting" mechanisms, that present critical issues with regard to privacy. In this paper we propose a protocol that allows only potentially identifiable users to use a given decentralized applications. The system ensures that users are identifiable only by a competent authority, under certain conditions, and that in any other case they remain pseudonyms. To evaluate performance and costs, we present an Ethereum implementation of the protocol.**

***Index Terms*—Blockchain, smart contracts, decentralized application, self-sovereign identity, zero-knowledge proofs**

I. Introduction

Reliable and guaranteed execution of transparent code is one of the most explored and interesting features of blockchain platforms. Smart contracts [1] are software objects that can encode rules defining financial and social mechanisms such as digital assets ownership, decentralised autonomous organizations, services such as exchanges and prediction and money markets, various dimensions of digital identity. Smart contracts offer different peculiar advantages over other code execution platforms and infrastructures, one of the most notable being *self-sovereignity*, that is the ability, for the user, to interact with the applications and other actors without third-party permissions. Another correlated feature is the ability to participate with *pseudonymous identities*, which implies unprecedented privacy. While self-sovereign pseudonimity is an important possibility, it can also create frictions and adoption barriers in some applicative fields, where "registry" identity is required or plays a role. Imagine for instance a stablecoin [2] token that aspires to become an official means of payment, such as a Central Bank Digital Currency, and the anti-money laundering requirements that it would elicit. On one side, authorities would like to be able to understand who are the actors involved in given financial movements. On the other hand, making it possible for everyone to associate pseudonymous identities (addresses) to real world ones would be problematic with respect to privacy. We can generalize this regulatory requirement dilemma to many smart contract applications. More precisely, in some situations some requirement should be satisfied, for a user of an application:

- the user should be linked to an "official" identity of some kind
- only in case of need, for example by order of the judicial authority, it has to be possible to identify the users involved in a given transaction.

Currently, the most widespread solutions can be described as *custodial*: a trusted third party identifies the user/subject, and then acts on his command, keeping his keys. Authorities can always turn to the keeper of the keys to identify a subject, similarly to what currently happens for instance on social media platforms. This configuration of power and responsibilities, in spite of its simplicity, has of course many drawbacks, and undermines disintermediation. Other solutions, slightly more refined, are based on various forms of "whitelisting": users get identified by a third party that insert them in a *whitelist*, after which users can interact with the system/smart contract with their own keys. This solution has the problem that the identifying entity can trace all the activity of the user on-chain. The problem of users on-chain identifiability is of course intertwined with that of digital identity models. Self Sovereign Identity (SSI) is an ideal digital identity model, based on a set of desirable principles [3]. A possible implementation of SSI relies on Decentralised Identifiers (DID) [4] and Verifiable Claims (VC). A verifiable claim is "a qualification, achievement, quality, or piece of information about an entity's background such as a name, government ID, payment provider, home address, or university degree". Claims can be represented in digital form, and are verifiable in that they are digitally signed.

In this model, individuals bootstrap their identity by first autonomously generating a DID. An example of a DID can be a public key or an Ethereum address, of which the user retains the corresponding private key. Once the user has a DID, he can interact with other entities, that can emit verifiable claims to qualify him. For instance, Bob could go to the registry office, be physically identified, demonstrate control of his DID, and be issued a signed verifiable claim stating that his DID is controlled by a citizen with a particular registry

978-1-6654-2744-9/21/$31.00 ©2021 IEEE

identity. From this moment on, Bob could convince another party of his identity by showing the claim signed by the registry office, and proving control of his DID. This model presents several advantages e.g. over centralized identities "as a service" based on identity providers [5], among which the absence of single points of failure, less infrastructural costs, higher interoperability, self-sovereignty. It is also possible to minimize the information exposed by the user using Zero Knowledge Proofs, for instance providing a proof of being of age, starting from an ID, without disclosing any other information (cfr. Section II-B).

In this paper, we propose a protocol that addresses identifiability while preserving privacy. In particular, we consider an identity model based on verifiable claims, in which a registry office issues an identity certificate to the user. We exploit zero knowledge proofs to let the user guarantee that, in case of need, his identity can be reconstructed by the judicial authority. In this way, user identity isn't exposed until there is a real reason. We show an implementation of the protocol on Ethereum, applying it to guarantee identifiability of users of a given token. We analyse and discuss deployment and usage costs.

II. State of the Art

A. Self-Sovereign Identity and Verifiable Credentials

Being able to represent identity digitally is ever more important. Current models and schemes, such as those federated and based on identity providers have important shortcomings and limitations [6]. In the last decades, different digital identity models have been investigated. One noteworthy result has been the definition of a set of properties/principles that define the so called *Self Sovereign Identity* (SSI). Federates identity models based on Identity Providers are problematic with respect to most SSI principles, since handling identity by users always require intervention/collaboration of the IP. Moreover, IP based identity has high infrastructural costs, it is inflexible (IPs can only guarantee few attributes of the individual) and implies a single point of failure. A much more self-sovereign solution is based on decentralized identifiers (DID) and verifiable credentials (VC). Both concepts have been standardized by W3C. Decentralized identifiers (DID) are identifiers that can be generated autonomously by the individual, and of which the owner can prove control without having to rely on a third party. An example of a DID is a public key: an individual can create a private-public key pair autonomously, and then prove control over it with the tools of asymmetric key cryptography. DIDs can be enriched with other information such as cryptographic material, verification methods, or services that define ways of interaction with the owner of the DID. In this case, the information is called *DID document*, and the DID itself is a resource identifier that resolves to the corresponding DID document. This structure allows, for example, to rotate the public keys that control a DID, without changing the DID itself.

The other components of the identity model are *verifiable credentials*. Credentials, in general, can be thought as assertions made by entities or individuals about other entities or individuals. Think of a driving license: it says that some office in charge says that Bob is allowed to drive a car. When an officer checks a driving license, he tries to understand if the credential was really emitted by the claimed entity, and if the driver is really the subject that is presenting it. How to check these two conditions in the digital realm? Verifiable credentials aim to solve the problem with digital signature and DIDs: the credential is signed digitally, so it provenance can be verified, and the subject is identified by a DID that he controls.

B. Zero-Knowledge Proofs

One of the principles of Self Sovereign Identity is data minimization: only the information essential for an interaction should be disclosed. The classical example is that if there is the need to check whether an individual is of age, other information such as name, birthplace, and even date, shouldn't be disclosed. One solution to this problem is offered by *Zero Knowledge Proofs* (ZKP). ZKPs allow you to demonstrate the knowledge of information with certain properties, without revealing anything about the information itself. For instance, it is possible to create a proof of knowledge of the solution of a given Sudoku, without revealing any hint about the solution itself. One application to identity management can thus be for example to prove that one subject if of age, without revealing the precise birth date present on the ID. In this paper we will exploit Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, or zk-SNARKs, to prove to a smart contract that the user provided some information that, in some circumstances, can lead to his identification.

C. Current approaches to bring identity and kyc to tokens and smart contracts

As described in [7], there are different state-of-the-art approaches to perform user identification (or to "Know Your Customer" (KYC)): some of them, such as IDnow [9], VideoIndent [10], CB Financial: Services AG (CBFS) [11] and Shufti Pro [12] do not consider interoperability with blockchain applications, while others, such as Tradle [13], KYC-Chain [14] and KYCstart [15] directly involve blockchain within their solutions.

On the one hand, IDnow [9] is an identity verification platform, which provides a wide range of KYC services possibly compliant with Anti-Money Laundering as well as with the regulations of the corresponding authority. VideoIdent [10] verifies users using a P2P video chat with a responsible person, simulating a face-to-face setting by asking different questions to the user and requiring him/her to hold and eventually tilt his/her ID document. CBFS [11] utilizes secure video streamings to enable online identification and contracts signing. Finally, Shufti Pro [12] exploits the potential of artificial and human hybrid technologies to provide efficient and accurate digital KYC verification services.

These approaches are sometimes used in relation to blockchain based applications, for instance to perform the KYC of Initial Coin Offerings (ICOs) or Security Token

Offerings (STOs). The classic integration approach is the following: the company or entity responsible for the ICO or STO relies on these approaches to perform a complete KYC and collects a list of addresses (e.g. Ethereum address), usually one per registered user, to be whitelisted in the contract (e.g.: the Smart Contract deployed on the Ethereum blockchain). This kind of integration is however deeply centralized and presents a single point of failure, from the user registration to his/her whitelisting, passing through the KYC process.

On the other hand, some KYC approaches are directly connected with the blockchain technology: Tradle [13] perform identity checks by means of chat and snapshots of identity documents, acquired and safely store thanks to blockchain-based bots. KYC-Chain [14] enables businesses to handle the KYC processes for individuals and corporations by applying a B2B-managed workflow application [12]. Lastly, KYCstart [15] is a Proof-of-Concept of a KYC-as-a-service using blockchains [13]. Other approaches try to exploit the blockchain technology in different way, such as the one presented in [16] that uses a distributed ledger to ensure transparency and cryptography to perform efficient and optimized operation, or the one proposed in [8] that notarizes on blockchain the Zero-Knowledge Proofs committment of users personal data (e.g.: address, phone number, etc.), in order to enable offline verifications.

However, all these approaches do not address nor analyze the possibility of integrating the KYC processes (e.g. claim verification) directly on blockchain.

III. Proposed solution

A. Overview

Our work aims to create an SSI compliant identity management system in which the issuer of the credential and the judicial authority are two distinct entities, and only the latter can identify smart contracts users. It may seem just a shift of power between them, however, as we will explain, some strategies can be adopted in order to prevent abuses from the authority. The actors involved in the process are:

- Users: they control a cryptographic identity (they know a private key that controls a given public key), and are entitled to an "official" identity;
- Registry office: is an entity that can identify users (i.e. can verify a user official identity), and can emit credentials that bind a public key to an official identity
- Judicial authority: an entity entitled, under certain circumstances, to the identification of the agent of some on-chain transactions.

An identifiable user is a subject whose actions can be connected to his "official" identity by the judicial authority, but only given certain circumstances (e.g. when two distinct authorities agree that he is suspected of illegal activities). Our goal is to define a process with the following properties:

1) only identifiable users can actively use the system (submit transactions).
2) the registry office cannot link transactions to actual official identities
3) judicial authority can, under some circumstances, link transactions to official identities.

Whitelisting by the registry office would guarantee property 1, but not 2.

An option would be to ask the user, before being allowed to interact with the system, to submit the verifiable credential to an authorizing smart contract, along with the request to authorize an address. This, of course, would render the identity of the user public. We could then ask the user to cipher the credential using the judicial authority public key. In this way, only the judicial authority could decipher the credential, in case of need. The problem now would be: how to know if the user submitted a valid credential, without deciphering it? The solution we propose is for the user to generate a zero knowledge proof of the fact that the ciphered credential submitted is indeed valid, in that it was emitted and signed by the registry authority. The authorizing smart contract can check the proof, and authorize the user. Interactions among actors are described in Figure 1.

B. Detailed description

The first step for the user is to obtain a valid Verifiable Credential (VC) from the registry office containing his fiscal code (among other claims). The VC can for example be encoded as a JSON Web Token (JWT) and represents the digital corresponding of the paper ID; it is signed by the registry office itself, which is considered a trusted issuer.

After that, the user creates a symmetric shared key along with the judicial authority in order to encrypt his fiscal code; in this way, the ciphered value can be saved on-chain knowing that only these two actors will be able to decipher it. The key is generated using an authenticated version of the Diffie-Hellman protocol, based on elliptic curves:

- all the protocol users agree on an elliptic curve C, a modulus $p \in \mathbb{N}$ and a base $g \in C$
- the judicial authority picks a secret integer $x \in \mathbb{N}$, then publishes $B = g^x \mod p$
- the user chooses a secret integer $y \in \mathbb{N}$, then publishes $A = g^y \mod p$
- the user computes $s = B^y \mod p$
- the user computes the hash of s, and uses it to cipher its credential with a symmetric algorithm
- the user then publishes the ciphered VC
- if it needs to identify the user, the authority computes $s = A^x \mod p$ and obtains the same symmetric key of the user.

In a real-case scenario, some strategies must be implemented in order to make the judicial authority unable to unilaterally identify the user, and distribute the decisions among different actors. For example, the authority's key can be distributed among several distinct entities through some *secret-sharing* techniques [17]; in this way, the owners of the shares must join their parts together in order to reconstruct the key. Another solution could be to consider different independent

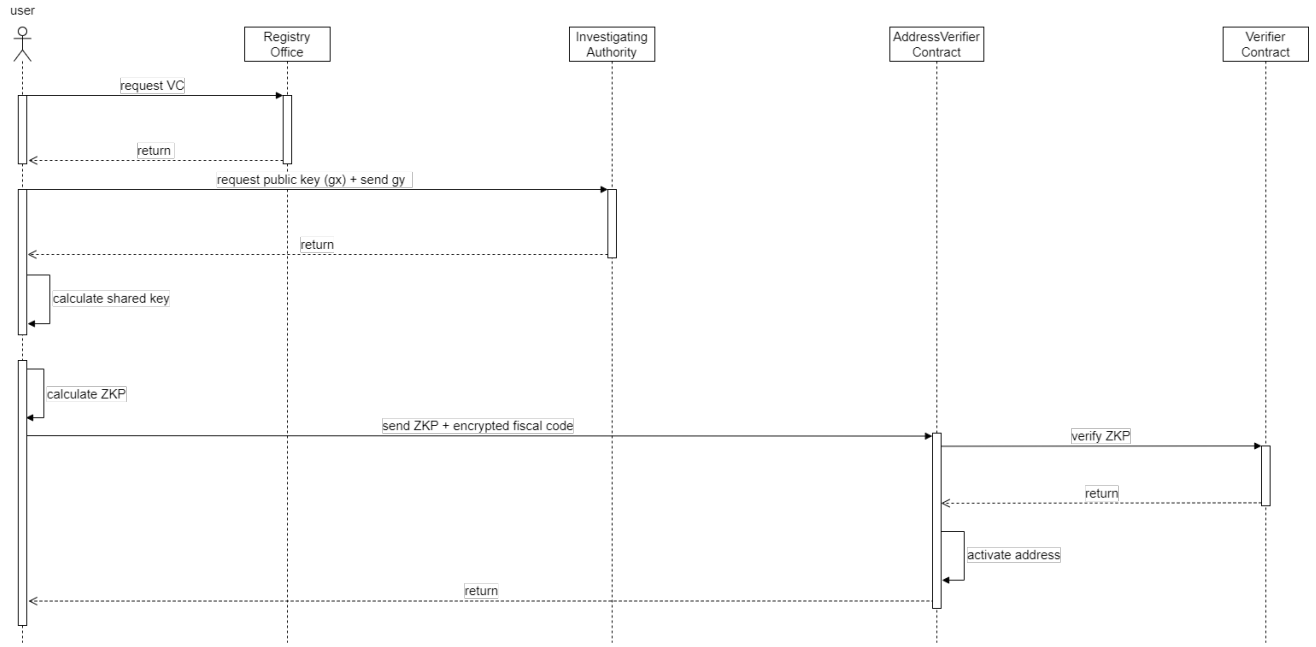

Fig. 1. Interactions among actors

authorities, with different keys, that must agree to decipher the user identity. Imagine for simplicity that there are two such authorities, A and B. The user would create two distinct symmetric keys, and encrypt his credentials with the two keys in sequence. This way, the two authorities would have to cooperate to recover user's identity. Another, more subtle way would be to exploit proxy re-encryption networks [**?**], [**?**].

Now the user has all the elements he needs to generate a Zero-Knowledge Proof proving that:

- the signature is valid and has been signed by a trusted issuer
- the VC contains a fiscal code
- the encrypted value is the fiscal code contained in the VC.

The proof can be checked on-chain by a smart contract, which can in turn enable an address specified by the user. Submission of proof and address by the user implies taking responsabilitiy of its use.

Since the main contract's functionalities are restricted to verified users only, the user can upload said ZKP along with the encrypted fiscal code to the verifier contract in order to activate his address. If the user is suspected of illicit activities, the regulator (and only him) can retrieve his identity by deciphering his fiscal code previously saved on-chain.

Let us now focus on the format for the identity credential. In our case, DIDs are Ethereum addresses; the credential contains, among other claims, a national identification code, the address of the registry office (the issuer) and the address of the user (the subject). For the sake of clarity it's better to specify that user address here is not the same address that he will activate to interact with the smart contract, but the DID of the SSI identity of the user [18].

We defined a particular form of signature: the issuer builds a Merkle tree with the claims, then derives the root and finally signs it with the EdDSA signing algorithm [20]. This eases the generation of the ZKP: the user is able to demonstrate that he owns a valid identity by proving that the tree of his claims contains a leaf with a national identification code. The proofs which show the belonging of a leaf to a particular Merkle tree are known as *Merkle proofs*; they can be built in an efficient way using only the subtrees found in the path from the leaf to the root, thus avoiding to recalculate the whole tree. In the VC exemplified in Figure 2, for example, we only need the second leaf and *subtree2* to reconstruct the root from the *cf* leaf. This allows to minimize the computational cost of the proof generation process.

ZoKrates is a toolbox that helps generating and verifying ZKPs on Ethereum. It provides an higher-level programming language which compiles to the underlying constraint system, making it easier to write ZKP circuits. Moreover, it can generate a Solidity smart contract from a circuit in order to verify the proofs directly on-chain. ZoKrates offers several standard methods to include sha256 hash functions in the circuits, and has also library implementation of EdDSA signatures.

C. MiMC criptography

In the Zero-Knowledge Proofs the total number of field multiplications in the underlying cryptographic primitive poses the largest computation cost. MiMC (Minimal Multiplicative Complexity) is a family of block ciphers and collision-resistant hash functions which can minimize the number of multiplica-

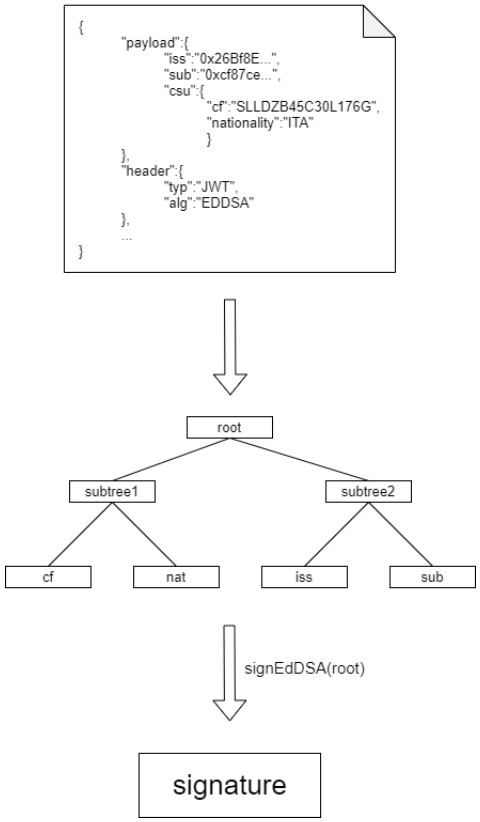

Fig. 2. Example of signed Verifiable Credential

tions. As showed in [19], calculating a MiMC hash in zk-SNARK applications, especially if compared with algorithms like sha256, results in a much lower computation cost. For this reason, we use MiMC functions to both generate the key from the shared secret and to encrypt user's identity credential to keep the impact on system performance as small as possible.

The acronym MiMC-b/k denotes a keyed permutation with block size b and key size k; for our symmetric ciphering algorithm we use the MiMC-n/n type. The cipher starts by iterating (in our case 64) a round function r times, and then by adding the key to the outcome of the last round. Each round involves the addition of the key k, the addition of a round constant and the application of a non linear-function in the form $F(x) := x^{2^t-1}$ (in our case, $F(x) := x^7$). The round constants do not need to be created for every evaluation of MiMC, so they can be generated once and then hardcoded into the implementation. The hash function MiMChash is obtained

by using the MiMC^{P} permutation in the sponge framework. The MiMC^{P} permutation can be derived from the cipher by setting the key to 0^{K} where K is the size of the key in bits.

```
def mimc_cipher(field input, field[64]
    round_constants, field k) -> (field):
  field num_round_constants = 64
  field a = 0

  for field i in 0..64 do
    a = input + round_constants[i] + k
    input = a ** 7
  endfor

  return input + k
```

```
def mimc_hash(field[2] input, field[64]
    round_constants) -> (field):
  field k = 0

  for field i in 0..2 do
    k = mimc_cipher(input[i], round_constants, k)
  endfor

  return k
```

D. ZKP method and smart-contract verification

In ZoKrates, the computations about which to produce the proofs are represented as *circuits*. Circuits can have private and public inputs. A proof demonstrates that the user that created it knows some inputs, some of which are kept secret, that produce a certain output. In our system the verifiable credential has four claims (issuer,subject,nationality and fiscal code), so the ZoKrates circuit takes the following inputs:

- leaf0Key (public) and leaf0Value, leaf1Hash, subtree1Hash (private) - to prove the presence of a fiscal code, derive merkleRoot and compare it with the signed message (in our system the fiscal code is contained in the first leaf, leaf0)
- R, S, M0, M1 (private) - to verify credential EdDSA signature
- user's secret y, fiscal code (private) and regulator's public key gx, user's public key gy, encrypted cf (public) - to check that the encrypted fiscal code is the one contained in the VC
- address (public) - to hold a reference to the address that is being activated, thus avoiding the reuse of one proof.

IV. Implementation Assessment

We implemented the protocol on chain as a set of Solidity contracts. The core the system is `AddressVerifier` contract, which keeps track of verified addresses and stores verification data. By means of its methods it is possible to activate an address, to check the state of an address and, knowing the required cryptographic information, to retrieve the encrypted fiscal code associated with a verified address. In particular, the method `verifyProof` takes care of ZKP verification by calling the `verifyTx` method of ZoKrates `Verifier` contract. The encrypted national identification code is extracted directly from the public inputs of Zokrates circuit; if the verification is successful, it is saved along with the proof into a Solidity mapping, using the verified address as the key.

In Table I and II are reported gas cost and time performance figures.

contract	scope	on-chain size	deploy gas cost
Verifier	verify ZoKrates proofs	3940 bytes	1078328
AddressVerifier	activate addresses	3549 bytes	868587

TABLE I
SIZE AND DEPLOY GAS COST OF CONTRACTS

The total cost for the deployment of both contracts is 1946915 gas units. The great number of points in the verification key makes it impossible to deploy the `Verifier` contract in one go, since it exceeds the maximum contract size of 24KB; to overcome this problem, we deploy a modified version of ZoKrates `Verifier` contract without the verification key and we upload it after the deployment, with a gas cost of 11469194 units. Hence the overall gas cost is 13416109. The complete transaction for the verification of the proof (which is a call to `verifyProof` method of `AddressVerifier`) has a total cost of 6359405 gas units; the straight verification using `verifyTx` method of `Verifier` takes alone 3195378 gas units.

command	execution time
compile	20s
setup	2m 45s
compute-witness	8s
export-verifier	2s
generate-proof	40s

TABLE II
EXECUTION TIMES

The total time to generate the proof (compute-witness + generate-proof) is about 48 seconds on a Windows machine with 16 GB of RAM and Ryzen 5 3600 CPU (3.60 GHz).

V. CONCLUSIONS

Compliance with regulations such as anti money laundering and know your customer can be one of the main obstacles to the widespread adoptions of decentralized applications based on smart contracts. In this paper, we presented a protocol that allows to exploit self sovereign identity credentials to guarantee identifiability of users of smart contracts. The method uses zero knowledge proofs to guarantee that a user can be identified if a set of authorities cooperate. Ad-hoc MiMC primitives were used to minimize proof verification cost and allow on-chain verification. The system addresses and resolves the issues currently present in both custodial and whitelisting based methods. Future works could address verification cost optimization, and exploration of protocols for sharing keys and better controlling user identification.

REFERENCES

[1] Nick Szabo, "Smart Contracts: Building Blocks for Digital Markets", 1996 [online] Available: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[2] Makiko Mita, Kensuke Ito, Shohei Ohsawa, Hideyuki Tanaka, "What is Stablecoin?: A Survey on Its Mechanism and Potential as Decentralized Payment Systems", IIAI International Journal Series, International Institute of Applied Informatics, 2015, Vol. 1, No. 1, pp. 48 – 63, arXiv: 1906.06037.

[3] Christopher Allen, "The Path to Self-Sovereign Identity", 25 April 2016 [online] Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[4] Drummond Reed, Manu Sporny, Dave Longley, Christopher Allen, Ryan Grant, Markus Sabadello, "Decentralized Identifiers (DIDs) v1.0", 27 June 2021 [online] Available: https://www.w3.org/TR/did-core/.

[5] David W Chadwick, "Federated Identity Management", A. Aldini and G. Barthe and R. Gorrieri, editors, FOSAD 2008/2009, Springer-Verlag, Berlin, January 2009, pp. 182 - 196, https://www.cs.kent.ac.uk/pubs/2009/3030/content.pdf.

[6] E. Ghazizadeh, M. Zamani, J. Ab Manan and A. Pashang, "A survey on security issues of federated identity in the cloud computing," 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, 2012, pp. 532-565, doi: 10.1109/CloudCom.2012.6427513.

[7] S. R. Niya, S. Allemann, A. Gabay and B. Stiller, "TradeMap: A FINMA-compliant Anonymous Management of an End-2-end Trading Market Place," 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 2019, pp. 1-5, doi: 10.23919/CNSM46954.2019.9012706.

[8] Y. Jiang, C. Wang, Y. Wang and L. Gao, "A Privacy-Preserving E-Commerce System Based on the Blockchain Technology," 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Hangzhou, China, 2019, pp. 50-55, doi: 10.1109/IWBOSE.2019.8666470.

[9] "IDNow: Regulation" November 2018 [online] Available: https://www.idnow.io/regulation/identification-kyc/.

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

[11] "Cbfsident" November 2018 [online] Available: https://www.c-b-f-s.com/services/cbfsident/.

[12] "Shufti pro" November 2018 [online] Available: https://shuftipro.com/.

[13] R. De Feniks "Tradle: KYC on Blockchain" [online] Available: https://urlzs.com/YL9nU.

[14] "Efficient KYC Management" November 2018 [online] Available: https://kyc-chain.com/.

[15] A. Lielacher January 2019 [online] Available: https://urlzs.com/ehWUM.

[16] N. Sundareswaran, S. Sasirekha, I. J. Louis Paul, S. Balakrishnan and G. Swaminathan, "Optimised KYC Blockchain System," 2020 International Conference on Innovative Trends in Information Technology (ICITIIT), Kottayam, India, 2020, pp. 1-6, doi: 10.1109/ICITIIT49094.2020.9071533.

[17] Adi Shamir, "How to share a secret", 1979, Commun. ACM, vol. 22, no. 11, pp. 612 – 613, doi: 10.1145/359168.359176.

[18] David Chadwick, Manu Sporny, Dave Longley, "Verifiable Credentials Data Model 1.0", 19 November 2019 [online] Available: https://www.w3.org/TR/vc-data-model/.

[19] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, Tyge Tiessen, "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity", 2016 [online] Available: https://eprint.iacr.org/2016/492.pdf

[20] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, ¡https://www.rfc-editor.org/info/rfc8032¿.

[21] Michael Egorov, David Nuñez, MacLane Wilkison, "NuCypher: a proxy re-encryption network to empower privacy in decentralized systems", https://github.com/nucypher/whitepaper/blob/master/whitepaper.pdf