future internet

MDPI

Article

A Blockchain Self-Sovereign Identity for Open Banking Secured by the Customer's Banking Cards

Khaled A. M. Ahmed *, Sabry F. Saraya, John F. Wanis and Amr M. T. Ali-Eldin

Computer Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt; drsfsaraya@mans.edu.eg (S.F.S.); jfzaki@mans.edu.eg (J.F.W.); amr.ali-eldin@mans.edu.eg (A.M.T.A.-E.)

* Correspondence: khalida.whab@gmail.com

Abstract: Open finance is evolving and extending open banking. This creates a large context that implies a financial and identity data exchange paradigm, which faces challenges to balance customer experience, security, and the self-control over personal identity information. We propose Self-Sovereign Banking Identity (SSBI), a Blockchain-based self-sovereign identity (SSI) to secure private data sharing by utilizing trusted customer's banking cards as a key storage and identity transaction-signing enclave. The design and implementation of the SSI framework is based on the Veramo SDK and Ethereum to overcome the limitation of signing curve availability on the current banking Java Cards needed for Hyperledger Indy. SSBI uses the elliptic curve SECP256K1 for transaction signing, which exists for several payment cards in the market. SSBI enables automated financial services and trust in the service provider communication. This work analyzes the flow and framework components, and evaluates the usability, integration, and performance in terms of throughput, latency, security, and complexity. Furthermore, the proposed approach is compared with related solutions. The presented prototype implementation is based on a test Ethereum network and signing transactions on the banking card. The preliminary results show that SSBI provides an effective solution for integrating the customer's banking cards to secure open banking identity exchange. Furthermore, it allows the integration of several scenarios to support trusted open banking. The Blockchain layer settings need to be scaled and improved before real-world implementation.

Keywords: digital identity; Blockchain; self-sovereign identity; banking card; open banking

Citation: Ahmed, K.A.M.; Saraya, S.F.; Wanis, J.F.; Ali-Eldin, A.M.T. A Blockchain Self-Sovereign Identity for Open Banking Secured by the Customer's Banking Cards. *Future Internet* **2023**, *15*, 208. https://doi.org/10.3390/fi15060208

Academic Editors: Christoph Stach, Clémentine Gritti and Claude Chaudet

Received: 8 April 2023
Revised: 16 May 2023
Accepted: 5 June 2023
Published: 8 June 2023

1. Introduction

In the modern transformation of digital banking, the customer does not have control of the sharing of their personal and financial data with the increasing number of authorized financial service providers. Open finance [1] is an emerging field within open banking, in which financial institutes, regulated websites, and financial applications obtain access to transactional details, for example, customer financial data such as savings, insurance, and customer credit (or part of it) from the user's bank accounts and payment services, usually via the Application Programming Interface (API) and based on customer consent. The purpose is to not only recommend cheaper services, but also to provide advice and customized product recommendations, similar to open banking but with a kind of 'read' data permission [2]. Open finance comes with 'write' permissions to execute cost savings on behalf of the customer, for example, transferring an additional USD 150 to the user's saving account with the aim of tailoring the best financial decision for the customer.

The banking sector has a significant stake in the digital trust era, including via customers' personal data, Know Your Customer (KYC) procedures, policy regulations, and secure authentication. There is a strong potential for banks to invest in identity management and expand payment card services and digital payment credentials with digital identity capabilities based on the strong position of trust and security measures. Banks, as identity

actors, can act as an identity issuer, deliver digital identity services, and provide credentials and authentication services.

In such a sensitive data-sharing context, many challenges appear, such as banks trying to adapt to a higher volume of transactions, data polices, and governmental regulations to ensure proper security and privacy controls for users to manage their own data between financial pillars. To avoid privacy leakage in such a paradigm, recalling for instance the Facebook–Cambridge Analytical data scandal [3], customers want to control and grant permission to each sharing transaction. In our proposed prototype, SSBI provides self-sovereign identity (SSI) features based on Veramo version 3 APIs [4] and strengthens the security of managing customer private keys using industry-trusted banking cards, and related form factors, assuming at least one banking account is available before the registration phase. Banks and other parties cannot share the data without permission from the customer in a decentralized framework. These data can be limited to credentials and proofs instead of disclosing the detailed personal identity of financial information. This attempts to address the issues of centralized or cloud-based identity systems, including leakage of immutability, traceability, and third-party control over individual data.

In this paper, we propose a framework to enable users to control their identities and privacy in the context of open finance and open banking, using a Blockchain-based SSI platform to improve the trust between customers and involved financial parties. SSBI integrates the banking card to improve the security of signing identity creation and sharing transactions, and considers a seamless user experience. In addition, this work is validated against a typical use case scenario and possible results are discussed to evaluate the proposal in contrast to related solutions.

During the demonstrated implementation, we designed and implemented an SSI framework based on the Veramo SDK with an underlying Ethereum Blockchain to extend the model, as suggested in [5]. Our contributions include the integration of the customer's banking cards to hold the identity secret keys, the signing of pairwise DIDs (Decentralized Identifiers), and verifiable proofs to balance the security and customer experience. This improves the user's trust by controlling the identity enrolment and claims of several financial services provided by open banking institutions. Our work overcomes the limitation of signing curve availability on banking Java Cards that are required to sign Hyperledger Indy [6] transactions, as stated by [7]. The Ethereum signing curve exists for several market banking cards, which we have utilized for this work, to issue ethr DID and VCs (Verifiable Credentials). Furthermore, by storing the private key and performing the transaction signing on the trusted smart card, the level of trust is increased compared to manipulating these keys on a mobile device's memory.

This work applies the proposed framework to a typical open banking case, and describes the integration and the method used to implement the card personalization flow to ensure feasible deployment of the identity applet for the banking Java Card. SSBI implements the SDK calls to integrate with a mobile app, which in turn can be part of a mobile banking app and the Internet banking portals. The proposed work allows identity key recovery by splitting the secret key seeds and passphrase between the trusted bank (card issuer) and the identity owner (the customer).

This paper is organized as follows: Section 2 presents the underlying background. In Section 3 we present related work in contrast to SSBI. Section 4 describes the proposed solution, design objectives, platform layers, and architecture, and breaks down the components. Section 5 provides the proposed flow and related phases. Section 6 describes the implementation and the tools used, in addition to discussing key recovery and integration of customer banking cards for SSI operations. Section 7 presents the discussion of the security and performance evaluation, in addition to comparative analysis. Section 8 provides the conclusion and discussion of future work.

2. Background

The user-centric model, and now the SSI approach, are consequences of the revolution of identity management from centralized silos to a federated model. These imply the principals of returning the control of identity data to their owners, as stated by Christopher Allen and Kim Cameron [8]. The self-sovereign identity principals [8] include:

- Users must have an independent existence;
- Users must control their identities;
- Users must have access to their own data;
- Systems and algorithms must be transparent;
- Identities must be long-lived;
- Information and services about identity must be transportable;
- Identities should be as widely used as possible;
- Users must agree to the use of their identity;
- Disclosure of claims must be minimized.

SSI systems based on Blockchain have the means to deploy the concept of identity and benefit from the rich characteristics provided by Blockchain, such as immunity, cryptographic data storage, distributed key management, and distributed storage [9]. SSI returns the control over identity information storage and selective disclosure rights to the identity owner. It enables the decentralized architecture to minimize the probability of data leakage that may be caused by compromising centralized identity repositories. In addition, combining SSI with Blockchain provides data consistency, availability, and privacy, and reduces correlation by creating a unique identifier for each service.

The aim of Blockchain is to store cryptographic transactions in a public ledger based on a decentralized consensus mechanism, which is hard to break or modify, according to the investigation results from [10]. Our proposed platform utilizes the Ethereum public ledger to record the identity transaction hashes, to prevent hacking, denial of service attacks, and identity theft or loss. In addition, it enables smart contracts, which allow multiple benefits such as automation of approved financial services and integration with decentralized applications.

uPort [11,12] is an identity system implementation based on Decentralized Identifier (DID) specifications and W3C [13], and relies on Ethereum Blockchain to deploy several smart contracts to manage identities, attributes, and attestations of attributes. It allows selective disclosure and sharing of identity attributes between the user and related parties. Public data are stored in the Interplanetary File System (IPFS) [14]; the public key is bound to a hash link that points to public profile data in the IPFS of Ethereum Blockchain. Only users who own the private key and the data hash on the IPFS, which ensures the integrity, can modify this link. This work does not use the uPort app since it was decommissioned in June 2021 and replaced by the Veramo framework [4].

Veramo [4,15,16] is a new and evolving iteration of uPort. It provides open source libraries for a modular API for SSI and verifiable data, creating and managing interoperable DID and Verifiable Credentials (VCs) [17] without relying on third-party vendors or centralized systems. It supports many platforms, such as Node, browsers, and React Native. Veramo is fitted to our framework and banking use case as a middle layer to control DID and VCs.

3. Related Work

KYC2 [18] is a framework for client onboarding based on the SSI model and Hyperledger Indy [6]. The model was discussed in terms of the SSI concept and General Data Protection Regulation (GDPR) [19] rules. It advised the need for key management to secure the secret agent keys and recovery conditions. This work proposes using banking smart cards for cryptographic operation handling and deploys the SSI concept based on Ethereum Blockchain; specifically, the curve exists for several market banking Java Cards and was validated in our proof of concept.

The authors in [20] provide a survey and overview of the SSI concept and available solutions in terms of the architecture and actors in the approach to identifier-registry and claim-registry models. Blockchain enables these decentralized registries, although it is not a necessity, and the storage of both identity and claim registries can also be decentralized. The paper presents on-chain and off-chain storage with the advantages and disadvantages for each option. In our work, we store private information off-chain while the integrity and authenticity proofs are stored on-chain.

SCARAB [21] is a model used for decentralized and secure access control, in which the accountability issue of SSI Blockchain-based systems is resolved by registering every data access request in public mode. It introduced on-chain secret verifiable sharing to collective management using the Byzantine protocol, and also introduced identity skipchains to control access policies to manage the identity and enable user's self-sovereign identity management [21]. SCARAB stores everything on-chain, which affects the scaling and performance, making sensitive identity data available for the public, and meaning that it is not possible to erase it.

BBM [5] is a Blockchain-based SSI model for open banking and enables secure communication between involved parties. In contrast to BBM, it is proposed here to use the banking card, which is trusted and conveniently used by customers for daily purchases, to act as tamper-resistant storage for the identity private key, and to perform the cryptography required for verifiable credential issuance and sharing with banks and third-party agencies. Instead of relying on additional devices, such as hardware tokens, we utilize the banking card to balance security and usability. This work proposes the implementation of a prototype to assess the feasibility and possibility of integration with the existing payment card issuance flow.

The SSI framework has been used to prevent banking scam calls [7]. The authors proposed an SSI model based on Hyperledger Indy [6] and utilization of the user's banking card with the aim of preventing banking scam calls. In contrast, in our framework, we leverage the use case and overcome the limitation of the signing curve on Java Card ED25519, which is required for Indy transactions. The proposed framework is based on Veramo, which relies on Ethereum Blockchain, where the required SECP256K1 signing curve exists for several banking cards used for this work, to issue ethr DIDs and VCs. This fits within the open banking use case; the proposed framework allows smart contracts that automate several online financial services.

In [22], the authors proposed a protocol to reach a high level of assurance (LoA) for a mobile-phone-based SSI identity wallet with the main focus on achieving a high LoA. It considered key tamper-resistant storage requirements to achieve the high LoA via combining a software wallet key part and a YubiKey 5Ci token [23] for the hardware key part. This protocol is applicable for our implementation, since we rely on the banking card for the tamper-resistant storage of the key materials to secure the SSI wallet instead of the mobile secure element, which does not exist in some market mobile devices. In contrast, in our work we leverage the banking use case and balance the usability and strong security without relying on an additional device for banking customers. Furthermore, our implementation supports encrypted strong authentication and VC manipulation for the open banking services, and not only the identity wallet.

The authors in [24] proposed "Casper", a mobile identity wallet that enables inter-banking Know Your Customer (KYC) based on SSI and Rahasak Blockchain to manage smart contracts. In contrast, in our work, we attempt to combine high LoA as a market transition step to build on the customer's trust in the ecosystems of banks and financial institutes. This increases the trust and identity control via the sharing of a similar payment experience using banking cards and modern wearables. Our work can be extended to several business values, such as smooth KYC, password-less login to the Internet and mobile banking, and loyalty programs. We rely on Ethereum Blockchain, and benefit from the Veramo SDK to handle DID and VC-related smart contracts, in addition to the key

recovery mechanism, by combining a secret part at the customer's bank and the passphrase that is held by the customer.

PriFob [25] is a privacy-aware fog-enhanced Blockchain online global credential management solution. It uses a reliable encryption scheme with public Blockchain, allowing digital signatures and zero-knowledge proofs. Furthermore, it deploys two consensus algorithms (Proof of Authority (PoA) and Signature of Work (SoW)) for efficient verifiable credential handling. PriFob has been evaluated in terms of security and performance metrics, such as throughput and latency, based on the simulation and emulation of the framework. It showed better performance compared to other Blockchain-based solutions. In our work, we introduce the usage of the banking card to improve the security of key management and to achieve a high level of assurance, and aim to present the SSI benefits to customers, financial agencies, and banks for integration in the specific field of open banking. Similar to PriFob, our work is based on Ethereum Blockchain, and our framework can be improved to enhance the performance by applying the adopted reliable encryption and efficient consensus algorithms.

4. Proposed Solution

This section provides an overview of the components and actors used to build a demonstrative implementation, Figure 1 shows the SSI-based Blockchain identity systems components. The aim of utilizing banking cards is to balance the trusted industry security behind the day-to-day usage of personal payment cards and the user's experience, and to provide ease of use without relying on additional hardware tokens. As an extension, the proposed architecture supports smart cards with biometric fingerprints and other form factors such as wearable tags and banking bracelets. This section describes the designed flow, framework components, and interaction between the parties. This implementation shows the feasibility and attractive values for adoption by banks and other financial institutes to secure and improve open banking integrations, and to provide convenience to customers, who can control their identity and sensitive information sharing.

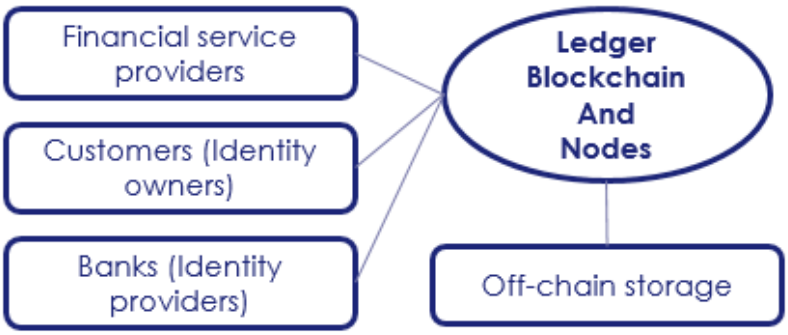

Figure 1. Blockchain SSI-based identity system components.

This work proposes an architecture based on the SSI and Blockchain implementations, Ethereum with Veramo and smart contracts. Specifically, a decentralized framework is used that stores private information off-chain while the integrity and authenticity proofs are stored on-chain. Our proposal provides a friendly interface as a mobile application or via web access integrated with a payment card for a seamless user experience, which allows third parties to verify the validity of identity data. The Veramo SDK layer provides public profile data to be stored on the IPFS; this means the user has to interactively manage the sharing of their private identity with other parties using web links or QR codes. Mnemonics and QR code technology allow presenting the secret keys in mnemonic terms to help with memory retention. Seed phrases are analogies for mnemonic words to ease portability and retrieve the key parts.

Furthermore, new banking cards in the market are equipped with biometric fingerprints [26], as shown in Figure 2, which improve the user experience by eliminating PIN entry for each identity transaction.

Figure 2. NFC banking card sample equipped with a fingerprint.

4.1. Requirements and Design Goals

The SSI concept emphasizes that personal data are subject to governance and are not dependent on any external factor, as listed in the below characteristics [27]:

- Complete control over private data;
- Ensuring the privacy and security over users' data;
- Trust in any central institution is not required;
- Data portability;
- Data integration;
- Personal data transparency.

This concept is important as it reduces the chance of data leakage incidents since information is not stored in central locations [28]. A high level of assurance is assessed based on specifications [22] compiled from two standards that control digital identity: (1) The ISO 29115 [29] standard; this standard covers the four main phases of (I) enrolment, (II) electronic identification mean management, (III) authentication, and (IV) management and organization; and (2) NIST (National Institute of Standards and Technology (NIST), 2020) [30]. We include these standards' requirements in our work to ensure a high level of assurance by extending the banking card usage as a hardware-based second factor authenticator.

4.2. Design Objectives

The design of SSBI aims to address and satisfy the following list of requirements within our proposed implementation:

- Prevent financial data leakage when third-party agencies promote financial products or KYC services, and eliminate user's fraud calls and phishing messages by trusting DID and VCs.
- Trusted identity owner since we consider the active account bank user, which means the KYC process is to completed in advance with a unique DID.
- Achieve a high level of assurance using Veramo key management combined with tamper-resistant secret key storage, by relying on a payment card to prevent theft and unauthorized usage, with potential enhancement to use a smart card with embedded biometric fingerprint authentication.
- Ensure validity of identity and proofs by asserting the revocation status on-chain using Veramo APIs.
- Strong and multifactor authentication against attacks, such as replay, Man-in-The-Middle, and theft attacks, through the usage of a banking card, PIN, and biometric card.
- User-controlled identity and data sharing. The user is able to manage one or more identities, including their attributes, using an open system that prohibits third parties

to restrict its usage. The user is able to select and authorize related parties to access these identities by enabling access to subsets of identity attributes. User consent can be given, and the related party is able to request access to the identity attribute of an end user. The attributes must be retrievable without a direct communication channel between the user and the party.

- The identity data owner is able to revoke access at any time.
- A secure and seamless user experience is achieved by combining the trusted issued banking cards for cryptographical identity operations. Furthermore, we propose new form factors such as stickers, tags, and biometric payment cards.
- An identity and secret key recovery mechanism is proposed where the key part is saved by the issuing bank while the passphrase is memorized by the user.

4.3. SSBI Layers

The framework consists of four layers, as presented in Figure 3: a banking applications layer, a Veramo identity layer, a Blockchain layer, and a customer data layer, as shown in Figure 3. Veramo APIs provide identity-related functionality, and we store data hashes on Blockchain (smart contracts) while customer-sensitive data are kept off-chain to maintain users' privacy. Ethereum Blockchain is used for strong cryptography DIDs. Public keys and VC transactions are stored on the Ethereum Blockchain; however, the VC itself is stored off-chain, such as in a database on a PC or in the mobile device's memory.

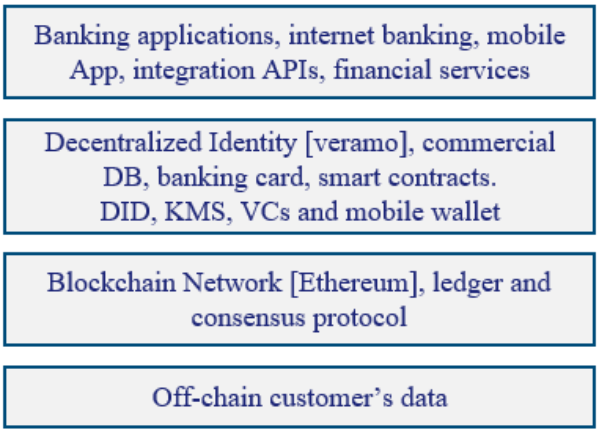

Figure 3. Proposed SSBI layers.

The banking application layer provides a means for third-party agencies and financial service providers to integrate and communicate through a list of APIs to obtain access to customers' account data and business functionalities.

The Veramo layer consists of backend software to manage communication, DIDs, DKMS, and VC-related functionality, in addition to the mobile application interface and database storage. Non-sensitive data, such as the public key, are stored on Blockchain. An Android mobile application is used to connect with agencies or other financial service providers through a QR scan.

The Blockchain layer and the SSBI framework using Veramo relies on Ethereum Blockchain and four smart contracts: the controller contract, the proxy contract, the registry contract, and the application contract, which provide consistency, security, and smart contract features.

The proposed architecture for SSBI, main actors and system functionalities are presented in Figure 4.

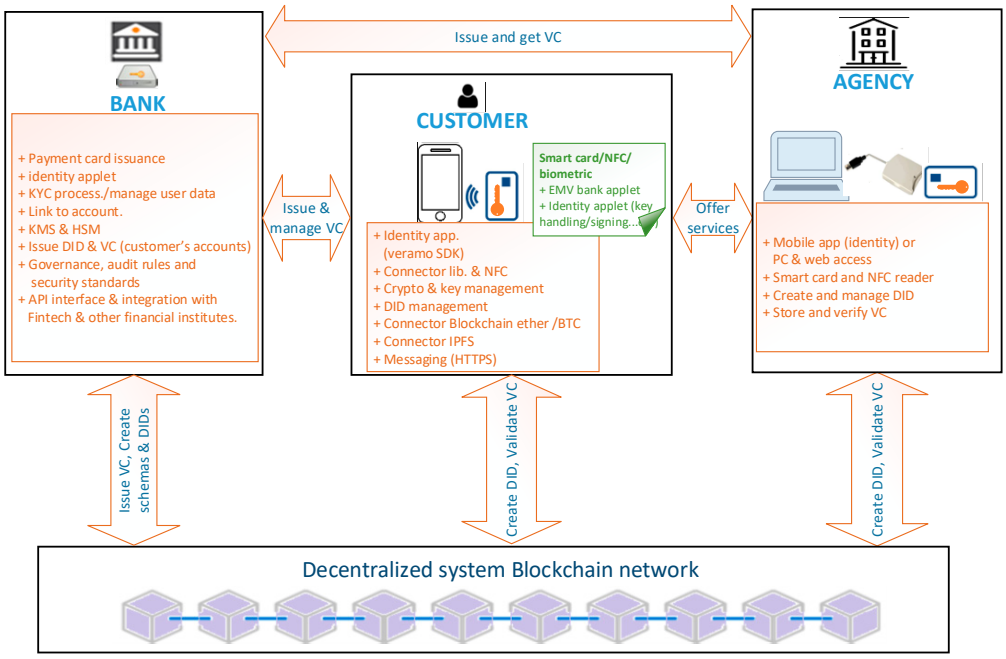

Figure 4. Proposed architecture customized for SSBI, extended from the model in [7].

4.4. Sub-Systems (Actors and Features)

The objective is to provide an easy-to-use open banking service, powered by SSI to reduce the risk of fraud, provide transparency, and provide verifiable and publicly resolvable identity information for an efficient KYC and open banking paradigm, as follows:

1. The contributing parties are based on Blockchain, which underlies Ethereum, to provide a decentralized infrastructure that enables smart contracts to manage identity information and provide immune storage for VCs and proofs to allow cryptographical verifications.
2. The banking card is used by the customer as an extended feature to enable secure signing operations and maintain secure private key storage.
3. An information management platform with web and mobile app interfaces allows identity data management flow, creation, revoking, verification, and selective data sharing.
4. DIDs [13] (Decentralized Identifiers) are linked to their owner, and represent the user, customer, financial institutes, third-party providers, and verifier parties. In our case, the Ethereum address is able to register, issue, query, and verify proofs and VCs.
5. Smart contracts [31–34] interact with Blockchain directly and execute the on-chain application logics among the system including the Ethereum *DIDRegistry.sol* to manipulate the DIDs, the Ethereum *EthereumClaimsRegistry.sol* to track claims, VCs, and proofs, and *Verifier.sol* to implement verification and check validity of VCs and proofs.
6. System admin and auditor roles.

4.5. SSBI Components

The main platform components are:

- Mobile app and agent operations;
- Server-side application to handle communication between parties, access to Blockchain, and smart card connections;
- Applet to install on the banking card;
- Veramo layer for DID and VC management.

Figure 5 shows the UML diagram presenting the designed classes to implement the SSBI platform. The off-chain class handles the storage of verifiable credentials and identity attributes, in addition to allowing the interface with agent application calls and providing one instance per entity. For the customer, this application is deployed as a smartphone app,

although it can be implemented as a PC application. The identity operations include DID operations and connectivity with Blockchain during registration, modification, and DID revocation. The entity class has an interface to the pairwise DID class. The JavaCardApplet class provides several functions to perform cryptographic operations, such as signing DIDs and VCs, and holding and generating keys, in addition to state and PIN management. The connector provides handling and exchange of the APDU command (main application protocol data unit) between the app and the card through NFC connections or a USB smart card reader.

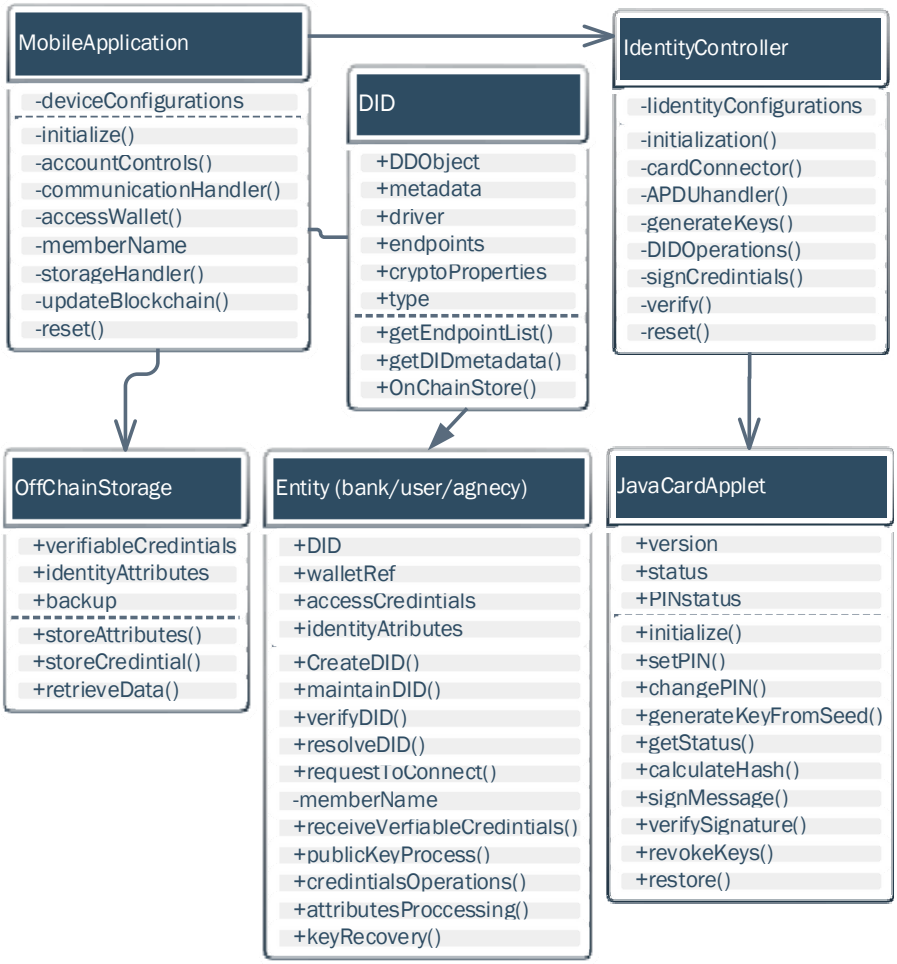

Figure 5. UML class diagram adapted from [7].

According to Lopez [35], the content of a certified document must include:

- URI for a unique identity credential (a set of claims by the issuer about a subject);
- such a credential is associated with a DID.
- URI to identify the issuer (issuer DID).
- URI for credential type.
- URI for protocols and terminology to help parties read the credentials.
- Cryptographic proof of the issuer.
- Claims data, and the statement or characteristics about a subject (metadata).
- Issuance date.
- Expiration date.
- Location of the credential status (such as a smart contract reference).

The format of VC and certificate exchange has to be in a protected and secure digital channel such as JSON-LD, as proposed by European Blockchain technical specifications [36] and according to the DID standards of W3C [17], as follows:

- The context of data description inside JSON-LD to ensure common understanding between parties.
- Identifiers—according to W3C, a DID should be used to identify issuers, credential subjects, persons, things, and verifiable IDs or attestations.
- Types, such as W3C (Verifiable Credential), in addition to other types.
- Issuer—the public institution, government entity, or an organization that is trusted to issue a specific type of VC, typically a bank or financial institute for our case.
- Credential subject, which is a DID to identify a person, thing, or organization.
- Claims, which constitute a verifiable ID or attestation for a semantic statement to a certain value about an attribute of a DID subject such as (.dateOfBirth).
- Issuance date is the date the claim becomes valid, and it is not mandatory that it be the same as the issuance date of the VC. Expiration date is the date when the claim should become false or expire. Status is the current claim status, such as active, suspended, or revoked.
- Proofs—at least one proof mechanism should exist to allow the verify operations.
- Optional extensibility—as per W3C, extensions of the data model can be used to add more attributes.

Moreover, Lopez [35] considers it a mandatory feature to allow public querying and resolving in the Blockchain of all involved parties' DIDs.

4.6. DID and Registry Using Veramo SDK

We propose managing DID and related identity operations using Veramo, which is one of the popular DID platforms implemented for Ethereum, and is based on smart contracts. Veramo is the next generation of uPort [11,12], which uses a more powerful and modular architecture and can run on backend servers, and web or mobile devices.

The Veramo platform consists of identity management and messaging protocols that are interoperable with any distributed ledgers for decentralized web and SSI models [4,16]. The platform uses the DID standard ERC1056 [34], which is an Ethereum pattern designed to create and update identities based on a lightweight identifier that utilizes smart contracts applicable for off-chain usage. The underling DID method allows any key pair account, for instance, a SECP256K1 public key, or any Ethereum smart contract to be considered as a valid identifier without additional registration. Attributes such as 'service end point' are retrieved by resolving an ERC1056 smart contract deployed in the network and listed in the registry repository. Any identity is mapped to a single Ethereum address, which is controlled by its owner. However, the advanced ownership model is managed by multi-signature contracts. EIP-1812 [37] is used by Veramo as a standard for claiming registry. It allows off-chain management of VCs by storing claims off the chain, and verifies them using on-chain smart contracts that implement either a stat channel or off-chain libraries. This is in contrast to other standards such as ERC-735 and ERC-780 [32], which rely on storing on-chain claims, and, as such, do not comply with GDPR rules regarding the storage of personal information on public databases. ERC1056 is compliant with W3C DID recommendations and allows identities to have multiple associated attributes and delegations for on-behalf identity operation.

5. Proposed Flow

5.1. Typical Initialization on the Mobile App

As presented in Figure 6:

1. The bank instructs the customer to download the mobile app from an authorized store.
2. The customer creates a DID and inserts his basic information to generate a secret key on the banking card and chooses the recovery mechanism.
3. The customer performs the first KYC for enrolment and creates a bank account.
4. The bank issues a new payment card and links it to the customer account. In addition, the bank installs an identity applet to serve the identity and open banking data sharing.

5. The bank confirms second factor authentication, e.g., to verify a one-time password (OTP) on the registered mobile number.
6. The customer performs initial configuration, sets a PIN, and inserts seeds for the master key.
7. The bank deploys an identity applet for an agency smart card and links it to the agency ID.
8. The agency performs identity card applet initialization, sets the PIN, and inserts master key seeds following the bank's instructions, and installs the corresponding application.

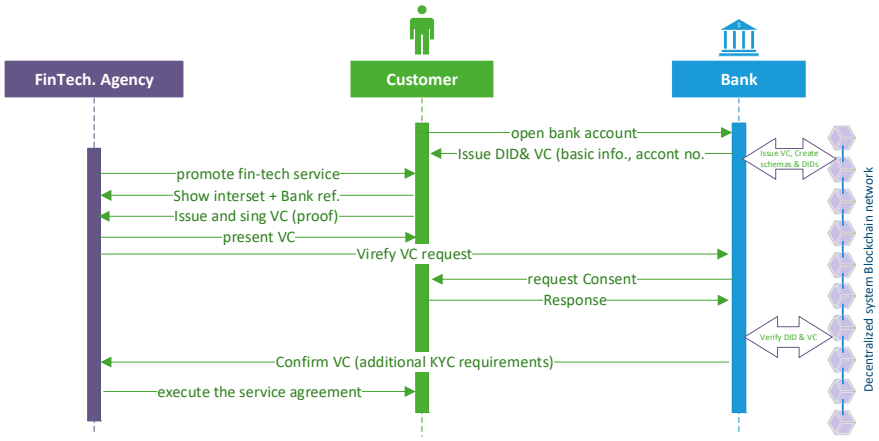

Figure 6. UML representation of the initialization phase.

5.2. Authorization Phase

As presented in Figure 7:

1. The user uses the mobile application to scan the QR code for the Fintech agency DID.
2. The user triggers verification of the DID (ether DID).
3. The user triggers connection with the fintech agency and creates a new pairwise DID.
4. The fintech agency requests a VC.
5. The customer selects which information to share.
6. The bank authenticates the user and picks the corresponding schema.
7. The bank signs the user claim using a Hardware Security Module (HSM) private key.
8. The bank sends the VC to the customer.
9. The user shares the VC, and it is signed/confirmed by requesting that the user presents an NFC banking card (from Bank A) and PIN or the biometrics for the card authentication.
10. The fintech agency validates the submitted VC by querying the Ethereum distributed ledger (DL), validates the digital signature to verify the user credentials, and validates the bank signature.
11. [Optionally request additional VC/proofs.]
12. The fintech agency signs a proof with the proposed financial service details (KMS/HSM key).
13. The user confirms and generates a signed proof by signing the transaction using his/her banking card.
14. The fintech agency executes a manual digital contract or, in automated mode, triggers a smart contract to the user.

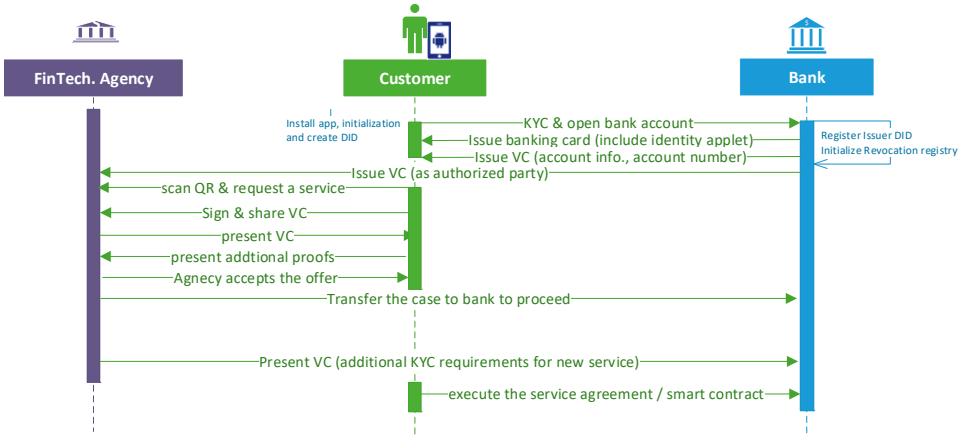

Figure 7. UML representation if the authorization phase.

6. Demonstrative Implementation

We implemented a mobile phone application to issue and manage the user's DID, proofs, and VCs based on the Veramo SDK and connector to utilize the banking smart card for secret key management through NFC communication. This is instead of saving the key encrypted on mobile memory. In addition, we use a webserver to act as a registry to handle the storage of DIDs and VCs, and to allow communication with other framework parties such as banks and financial agencies. The customer's bank issues the VC as proof of the account and initializes the identity applet with a secret key as part of normal banking card issuance. We simulate the interaction and basic flow functionality using a web mock service for indirect actors.

The mobile application for Android and IOS is based on React Native communication with an NFC banking card to set the PIN, generate secret keys, and sign the transaction to create pairwise DIDs and sign VCs using the connector library and mobile NFC reader.

6.1. Set of Tools Used in Our Demonstrated Implementation

- **Veramo** [4]—SDK to handle verifiable data and related SSI functionalities;
- **React Native** [38]—a framework for creating the mobile and web applications;
- **Node.js** [39]—the runtime environment;
- **Express.js** [40]—a web framework used to create the API;
- **Infura** [41]—used as access point for Ethereum networks;
- **Firestore** [42]—used to save data in a cloud database;
- **TypeScript** [43]—programming language;
- **Expo** [44]—used for building and debugging the application;
- **Mocha** [45]—used for unit testing.

6.2. DID and DID Document

SSBI uses the DID method *did:ethr*, which is a representation of the Ethereum address linked to the ERC1056 smart contract, and its DID document [13] stored off the chain, such as in the IPFS.

Sample DID Document

```
{
    "@context": [http://www.w3.org/2019/did/v1],
    "id": "did:ethr:0xdf0d . . . de1aebd143e",
    "publicKey": [
    {
      "did:ethr:0xdf0d . . . de1aebd143e#key-1",
      "type": "EcdsaSecp256k1VerficationKey2019",
```

```
        "publicKeyHex": "035bc987531e4823 . . . 46821d763ea36",
        "controller": " did:ethr:0xdf0d . . . de1aebd143e"
      }
   ],
   "authentication": [
   "did:ether:ysktz-uyzt-82re-d#key-1"
   ]
}
```


6.3. Claims and Verifiable Credential

The VC is formatted as JSON-LD, which contains the following information in the form of a Uniform Resource Identifier (URI), to define the means and protocol for parties to communicate: credential type, issuer, date, the credential and its subject, cryptographic proof of the issuer, optional metadata, and revocation condition.

Sample of Input Data for a VC


```
[
{
    "@context": "http://schema.org",
    "@type": "bankingAccount",
    "agent": { "did": "did:ethr:0xdf0d . . . de1aebd143e" },
    "bankingCredential": {
        "issuer": { "name": "Bank A" },
        "description": "customer valid account",
        "accountName": "User A",
        "accountNo": "012345678",
        "status": "Active",   //or “Revoked”
        "time": "2021-11-29T08:00:00.000-07:00"
    }
}
]
```


The application allows authorities, such as authorized banks in our case, to issue claims and VCs and store them on the IPFS. The authorized entities, such as central banks, regularly monitor banks and apply rules policies. Based on this, we consider the bank as a trusted authority to issue VCs. Off-chain there are two GUI interfaces: the first allows the bank to create pairwise DIDs and issue VCs; the second enables verifiers and other open banking agencies to validate the customer's VCs.

The following is a list of SSBI subset functionalities to handle DIDs and VCs:

- Create_DID;
- Create_VC;
- Resolve_DID;
- Verify_VC;
- List_existing_DIDs;
- List_existing_VCs;
- Retrieve_VC;
- Send_signed_VC.

The on-chain component is part of a front-end user interface (Android or iOS app hosting identity operations) and React Native application (customer and open banking agents). The functionality includes creation of DIDs, removal of DIDs, and VC manipulation for issuance and verification, in addition to underlying connectivity with the banking card and wearables to perform key management operations and Ethereum transaction signing. Sample snapshots to show SSBI sub-functionalities presented in Figure 8. The flow was validated and tested using Ethereum Blockchain. The off-chain part is based on Java

script and a Node.js application that communicates with the client through web3.js and API implementation. The encrypted data are stored in an SQLite3 database [46] and are accessible via a JDBC driver. The open banking API layer includes financial operations such as Account API, Payment, Money transfer, and Card life cycle.

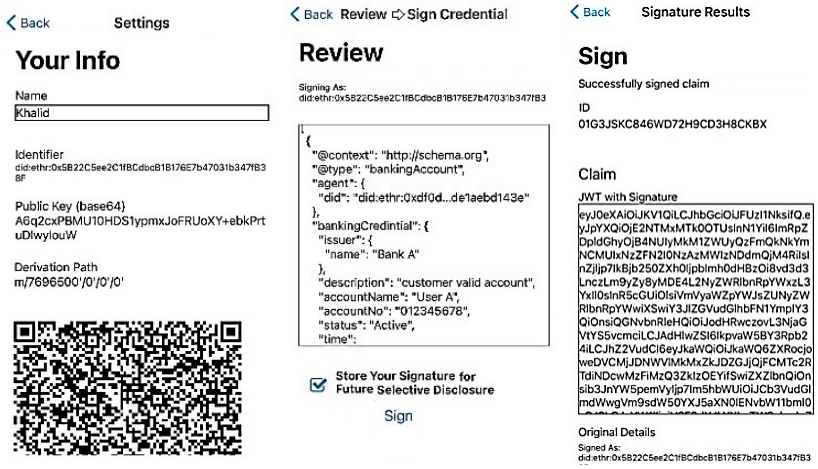

Figure 8. Sample snapshots present SSBI sub-functionalities to register and *create_DID* and fill in banking VC details, followed by signing and storing the VC by calling the *Create_VC* API with the underlying presence of the customer card to perform Ethereum transaction signing.

6.4. Key Recovery

The bank issues the payment card including the applet and initial key with seeds, and personalizes it for the user based on the Primacy Account Number (PAN) and the bank (issuer) private key to derive the seed and corresponding key. The operation is managed through the bank's HSM and strong key management system (KMS). The trusted bank complies with, and is audited for, security standards such as the Payment Card Industry Data Security standards PCI-DSS [47], policies, and monitored operations. During the initialization phase, the customer injects his passphrase. The pairwise DID key is generated based on both the key and the phrase. To recover the key in the case that it or the payment card is lost, the bank issues a new card with the same derived key and the customer inserts his passphrase to recover the key. The DID is then restored and all corresponding VCs and proofs can be requested again or restored from the encrypted local database.

An alternative option is to apply the mechanism suggested in the BBM model [5], by storing a recovery network in a smart contract; then, during the recovery on the new device, the bank verifies the issuance of a new card based on triggering the recovery smart contract.

6.5. Using Customer's Banking Smart Card to Secure Identity Private Keys and Sign Blockchain Transactions

6.5.1. Java Card for Banking and Identity Operations

SSBI introduces the utilization of the customer banking card to store identity secret keys and perform transaction signing to balance security and usability. The majority of banking payment cards support signature implementation for the SECP256K1 curve-based ECDSA, which is compatible with the Veramo platform and Ethereum Blockchain.

Banking customers commonly use payment smart cards for day-to-day purchases and money withdrawals. This relies on strong industry standards and security rules, such as EMV [48] and PCI-DSS [47], with advanced deployment of strong cryptographic keys and certificates. In addition to usually maintaining a secure PIN code or biometric verification to perform payment transactions, customers trust the issuer (banks) and rely on security rules that control this industry. We extend this level of trust and user experience to apply the financial identity sharing paradigm to open banking by using the same card for private key storage and signing the underlying Blockchain transaction for DID and VC operations, as suggested by [7].

Java smart cards are designed to be tamper-resistant and follow Java Card Platform specifications [49], where the developed applet can be installed once and run on several card's virtual machines; this applet is deployed on the card's operating system layer. In contrast to many studies that recommend a hardware wallet [50,51], we utilize the banking card to manage identity secret keys. This was also proposed by [7], where the applet is installed separately from the banking applet, with programmed functions to initialize master keys, generate sub-keys, and sign transactions securely on the card for DID and VC operations.

6.5.2. Java Card and Identity Libraries

The libraries required to interact with the banking Java Card for identity cryptographic operations include:

- APDU handling to construct and exchange the APDU commands according to context calls such as verify_PIN and get_status;
- Applet client for the interaction with the card applet methods, building the logic command, and the input data, in addition to handling the response from the applet, e.g., the signed hash;
- Card connector to establish the hardware handler with the smart card and manage the secure channel between the reader and the card;
- Java Card identity applet, which is the installed and personalized applet (cap) file on the card;
- Hashing and key operations, where the corresponding key methods include generating the keys and setup recovery methods.

6.5.3. Generating Identity Applet Master Key

During the initialization phase, the master key for the identity applet on the banking card is generated with the following steps:

- The applet performs pseudo-random code generation;
- The user inserts mnemonic words;
- The root seed (bank key part) + phrase (customer part) are generated;
- The applet performs the HMAC-SHA512 function;
- The resulting hash is used to generate the master private key and master chain code for the HD key root [52].

6.5.4. Initialization

This section describes the logical flow to initialize and issue the banking card for the identity applet next to the normal payment applet.

On the card issuer (bank) side:

1. Include applet installation (*.cap file) during the pre-personalization step;
2. Personalize the payment applet and link the card to the customer account;
3. The bank performs chip key rotation to secure the card and prevent unauthorized changes to installed applets

On the customer side:

1. The customer receives the payment card (instantly or by mail);
2. During the card activation process, the customer is then instructed to install the identity software application from a safe repository using a web link provided by the bank;
3. As part of initializing the identity application, the customer connects his card, usually via an NFC interface, to set the owner PIN, generate seed words, back it up for recovery in the future, and generate the master private key.

6.5.5. DID and VC Signing

In operation mode, the logical flow to sign DID and VC cryptographic operations using the Java Card is as follows:

On the customer side:

1. Identity application—build transaction message;
2. Request user to present banking card to NFC mobile device;
3. PIN authentication or biometric verification;
4. Transaction data signed on the card and the APDU responds with hash;
5. The application handles the Blockchain transaction broadcasting and waits for mining to complete;
6. Using the Veramo SDK, generate the DID or sign the VC according to the requested identity operation;
7. Store the created DID or VC in the case of proofs;
8. Present the VC to the user and ask for consent to share with a financial third party.

The applet supports three different categories of APDU commands that follow the ISO 7816 [53] smart card standard: initially, external authentication techniques such as PIN setting and verifying instructions; secondly, status and key generation; finally, the identity and Blockchain commands, such as signing and verifying transaction hashes.

7. Discussion and Evaluation

The implementation in this work is based on leveraging SSI, Blockchain, and tamper-resistant key management, with the aim of simplifying and securing the banking identity in decentralized mode. In the design, we considered security and easy integration with existing banking platforms, with benefits arising from security standards such as PCI-DSS and from the use of HSM. Each customer will be able to control and gain from sharing his identity and financial data without relying on a centralized system or third parties. The Veramo SDK and Ethereum Blockchain allow the management of DIDs, which are sets of characteristics and claims that uniquely identify persons, entities, or simply things. In addition, users can share claims and attested proofs with others with minimal attributes required for each case. This ensures high privacy requirements are met with the concept of zero-knowledge proof and selective disclosure, which exists in our work by relying on the Veramo framework. Our work aims to control and maintain the trust between participants in the open banking context based on SSI implementation, and achieves privacy, security, and a high level of trust, while balancing the user experience, by utilizing the customer banking cards to securely manage the private keys and on-chain identity operations. In addition, it is able to manipulate the required characteristics, such as key recovery and portability, via the use of a client mobile app, enables interoperability by relying on DID and VC specifications, and addresses the ability to be integrated with the banking ecosystem.

In contrast to using hardware tokens, such as in the work presented in [22], to achieve a high level of assurance we rely on existing banking cards, which are usually part of every banking customer's life, and provide the required level of trust and security (controlled by PCI-DSS and related standards), while remaining portable and easy to use. This can be extended for many other valued services, such as fast KYC and password-less access to banking mobile applications and online banking, on top of secure interaction with open banking and financial services.

7.1. SSBI Evaluation

In this section, we evaluate SSBI according to the stated design objectives, and in terms of the fulfillment of the requirements and features of the framework.

Prevent financial data leakage in open banking: SSBI helps to prevent identity and financial data leakage during open banking operations. Third-party agencies can promote financial products and perform KYC services on behalf of banks via the trusted SSI platform where customers and agencies can verify each other. By providing trust in DIDs and VCs, this eliminates fraudulent calls and phishing.

Trusted identity issuer: The proposed solution capitalizes on having banks as a trusted party. SSBI considers the active bank account customer, where the KYC process is performed in advance with a unique DID.

High level of assurance: SSBI combines Veramo key management with a tamper-resistant secret key storage, which is a payment card, to prevent unauthorized access to an identity.

Digital identity verification: The approach relies on Veramo SSI APIs to ensure identity proof, VC validity, and revocation status through the access to Ethereum Blockchain.

Strong authentication and secure key handling: Multifactor authentication is enabled through the usage of a banking card for sensitive key operations and transaction signing. Moreover, banking cards use PIN verification and have recently begun to use embedded biometric fingerprint authentication.

Identity ownership and SSI: The user is able to control several identity proofs and related attributes, request and store the VC and proofs from the identity issuer (banks in our case), and control the sharing of the VC or selective attributes with other parties.

Ability to revoke the proofs: The identity owner is able to revoke the proofs and VC at any time, by calling the corresponding Veramo API to revoke the Ethereum smart contract.

Seamless user experience: SSBI combines the trusted issued banking cards, which are used in day-to-day payments and money withdrawals, for identity cryptographic operations. It is more convenient for the customer to use modern form factors such as payment stickers and miniTags. The mobile app communicates through NFC to simplify the process.

Key recovery: Identity and private key recovery are achieved by combining two key parts and performing the generation action, where the key part is saved by the issuing bank while the passphrase is memorized by the user.

7.2. Security Analysis

The proposed SSI framework uses the Veramo (uPort) SDK and integration with a banking card. To evaluate the security of Veramo, we applied the threat model from [54]. SSBI does not rely on the official uPort mobile app; however, we utilize the same SDK functions:

1. The private key in uPort is stored on the mobile secure enclave, where the SSBI stores the key on the banking card, which in turn cannot be extracted because the threat score is reduced significantly.
2. To counter the potential threat of stealing or gaining unauthorized access to the smartphone, SSBI isolates the private key store on external devices (the banking card), which prevents attacker identity operations.
3. uPort does not include an additional layer of protection such as a PIN or biometrics. We implement access control to identity the app using a PIN or by presenting the smart card.
4. Recovery mode in uPort is available with a social mechanism; however, we applied a different mechanism by involving the trusted bank to store the keys (HSM), while the user is responsible for the passphrase. The key cannot be regenerated without having the two components.
5. The action time stamp exists and can cause false revocation. This is possibly mitigated by forcing the SSBI app to fetch a governance timeserver.
6. Denial identity actions due to stealing or losing the mobile device can be mitigated via the recovery mechanism.
7. uPort app deletion means erasing the private keys. In contrast to the SSBI app, the deleted key is still safe inside the user's banking card.

7.3. Security Challenges of Using the Banking Card for Identity Transactions

SSBI introduces the banking card to sign identity transactions. This raises additional security concerns related to the applet and APDU data exchange. Here, we present a set of these concerns alongside their corresponding mitigations:

1. Malware attack by capturing the PIN or passphrase during customer entry—this is possibly mitigated using a secure keypad instead of a standard device keypad.
2. Presenting the banking card to sign the identity transaction is vulnerable to a change in the data package sent to the card by an attacker—this can be mitigated by establishing a secure messaging layer [55] based on a session key and securing all APDU data exchange.
3. The identity applet installed on the banking card is vulnerable to unauthorized updates or re-installation—this is mitigated by the default use of access to a chip key, which is a 3DES key securely managed by the chip manufacturer and shared only with the issuer bank. Additionally, the issuing script performs key rotation to lock the Java Card state to avoid further installations.
4. Another layer of authentication could be a smart card with an embedded biometric fingerprint sensor to achieve a better user experience and security.

7.4. Performance Analysis

To analyze the general performance of the SSBI prototype implementation, we implemented different test scenarios and stress testing with recorded customer groups. The complete flow of the payment card issuance and applet installation was performed using the desktop personalization tool at an early step, followed by card initialization, setting the customer PIN, and key generation. During the Create_DID process, the card is presented to perform the signing operation, with the average time to respond being 3.2 s, assuming that the customer presents the physical card to the mobile app to eliminate the human action time of fetching the card. Moreover, to be able to perform real-world stress testing on the system throughput and latency, we isolated the card signing time during the stress testing and prepared a set of pre-signed transactions (10, 25, 50, and up to 300) to push the network.

The testing environment was as follows:

- Number of organizations: two banks and one agency;
- Access to Ethereum network: Infura;
- Consensus mechanism: Proof of Authority (PoA);
- Nodes: four nodes (one validator and three peer nodes);
- Customer device: Samsung Galaxy x4 and iPhone 6;
- The system ran on: Intel(R) Core(TM) i7-10850H CPU @ 2.70 GHz;
- OS: Windows 10;
- State database: Firestore;
- Mocha tool for unit testing;
- Caliper benchmarking tool;
- Solidity to write smart contracts.

To perform the transaction on the Ethereum network, the computation cost is expressed in the term of gas expenses. The gas amount is the parameter of computation complexity. There is a significant cost of the operations that deploy smart contracts such as VC and DID creation. The function of SSBI was validated for general comparison and the evaluation results were gathered for the system throughput and latency, followed by assessment of the complexity and consistency of the Blockchain interaction operations. The test was conducted based on a peer network of four nodes: one node was configured as a validity node to perform PoA consensus and sign the transaction; the remaining controller nodes sent the transactions to the validation node.

7.4.1. Throughput and Latency

The throughput and time per transaction achieved for different sets of recorded customers, from 100 to 500, are shown in Figure 9. In addition, the performance when sending a concurrent set of calls, of 10, 25, 50, and up to 300 transactions per second, was assessed, as presented in Figures 10 and 11.

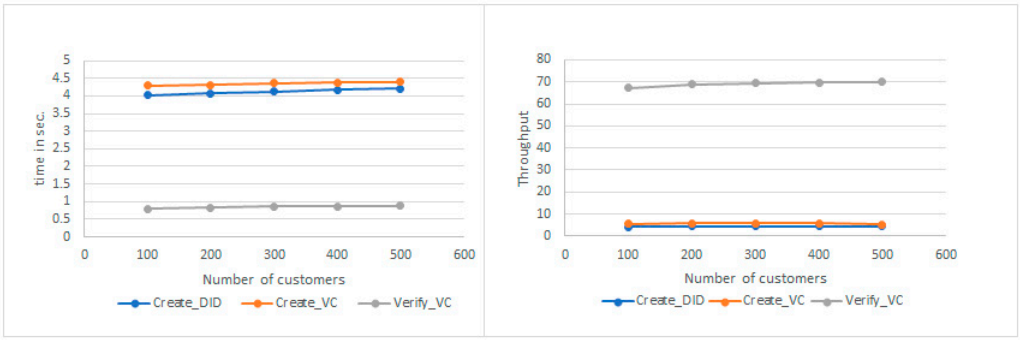

Figure 9. Transaction processing time (**left** side) and throughput (**right** side) for create and verify operations for different recorded sets of customers.

To test the throughput performance, we used sets of customers of 100, 200, and up to 500 as a small-scale implementation. The send rate varied from 10 to 300 transactions per second. The signing operation by the Java Card depends on the speed of APDU command exchange and the chip characteristics to process the SECP256K1 signing function. For the sample card used for our testing, we observed an average time for the card crypto-operation of ~3.2 s. The broadcasting, mining, and publishing to the Blockchain are all included in the transaction time duration. We were able to measure the time including the signing by the card for the case of one transaction at a time, while for stress testing, we pre-signed a set of 10, 25, 50, and up to 300 transactions to assess the performance. It was observed that key generation on the card takes a relatively long duration, of about 10 s; however, it is done only once during the registration phase for each customer, and hence it is still acceptable behavior.

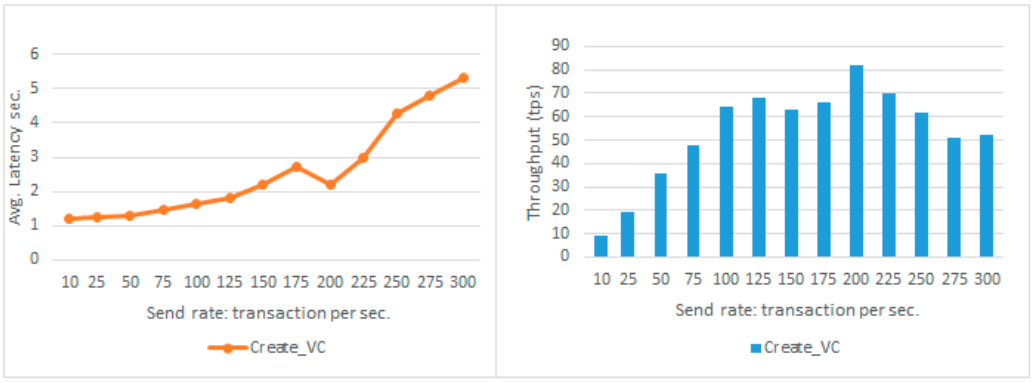

Figure 10. Latency (**left** side) and throughput (**right** side) for Create_VC (write operation) under different sending rates of 10, 25, 50, and up to 300 tps.

The observed behavior for sending rates of 10, 25, 50, and up to 300 transactions per second (tps) for Create_VC calls, which in terms of Blockchain access is a write operation, is shown in Figure 10, which indicates the throughput is relatively increasing. However, after 125 tps the throughput starts to slightly decrease, then significantly increases at 200 tps, before gradually decreasing again. The average throughput of write transactions is 53.07 tps.

The latency increases up to the sending rate of 175 tps, then drops at 200 tps, before the trend gradually increases again. The observed average latency for the write operation was 2.55 s.

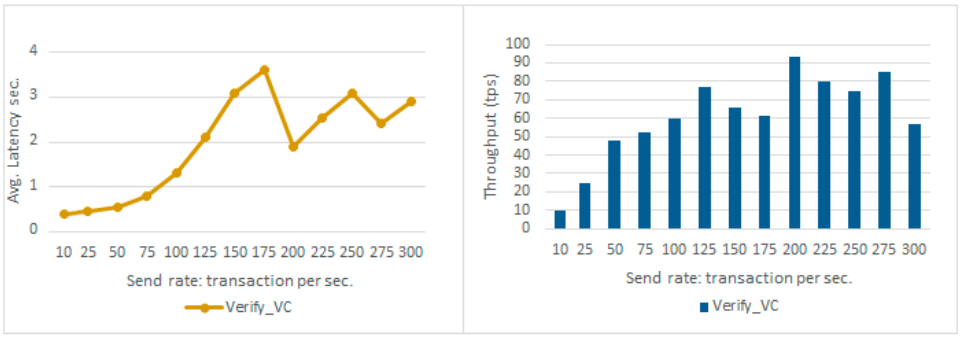

Figure 11. Latency **left** side) and throughput (**right** side) for Verify_VC (read operation) under different sending rates of 10, 25, 50, and up to 300 tps.

For Verify_VC calls, which are read operations, as shown in Figure 11, the transaction throughput increases until starting to decrease after 125 tps; furthermore, it oscillates and relatively increases at 200 tps and 275 tps, and then the trend goes down. The average throughput of read transactions was 60.69 tps.

The transaction latency increased with higher sending rates up to 175 tps, then dropped at 200 tps, before the trend gradually increased again, although it started to decrease after 250 tps. The observed average latency for the read operation was 1.93 s.

Throughput significantly varies depending on the function performed, such as Create_VC, which implies smart contracts; the throughput is much lower than that of read operations such as Verify_VC. We tested the performance with a set of customers ranging from 100 to 500 and sending rates of 10, 25, 50, and up to 300 tps. The latency and throughput for this testing show that optimum performance happened at 200 tps, with a throughput of 82 tps in writing operations and 93 tps in reading operations, while the latency observed was 2.2 s for writing and 1.9 s for reading. However, these preliminary evaluation metrics are highly impacted by the capacity and Blockchain setup configuration, such as the size of the network and the number of nodes, in addition to the expected load (transaction sending rate). Although the system showed a stable throughput, which complies with our preliminary proof-of-concept implementation, it does not satisfy the real-world banking use case.

The customer expects a much faster transaction. Hence, we will validate other options in future work to improve the throughput and speed, such as the proposed solution in [25] and other private Blockchain deployments, with the aim of achieving a maximum of 15 s per transaction. As per our discussions with SMEs from different banks, the expected time for customer convenience should not exceed 15 s per transaction at peak load, and the system should be capable of concurrently handling 1000 transactions in the pilot phase, with the ability to scale based on pilot performance analysis.

7.4.2. Proof of Authority PoA Consensus Mechanism

Proof of Authority (PoA) consensus is considered an efficient deviation from Proof of Stake (PoS), where the assigned validators sign and confirm the transactions. PoA includes a governance penalty system to eliminate malicious behavior and can provide faster transaction rates compared to PoW, excluding the mining time. The PoA mechanism reduces the utilization of computational resources, which is preferred in our case as it leads to a relatively stable transaction time

In our work, we use Proof of Authority (PoA), which is efficient and reasonable for verifiable credential handling, since this kind of platform requires only a small number of sealers. To ensure audit trails of access to records in this financial service ecosystem, it is necessary to have the participation of banks and regulated authorities.

7.4.3. Consistency

Distributed ledger consistency implies that no two nodes are in agreement, to ensure that they decide differently [56]. This is hard to achieve in Blockchain-based applications for both agreement on the valid blocks and the order of those blocks. Our proposed work relies on uPort and Ethereum, which has a default solution by hardening the puzzle difficulty. This leads to the generation of one block every several minutes, which is enough to propagate the new block through the Blockchain network.

7.4.4. Complexity

To assess the complexity of the SSBI system, we considered the work of [57], in which the authors suggested using the push–pull mooring effect to extend the complex theory, explaining this to the field experts, and proposing that they use the solution. We attempted to adapt the mechanism for our use case and conducted an interview with senior banking business managers to propose the idea and explain the features proposed to extend the usage of payment cards for identity operations. The interview included a three-slide presentation, a guide on how to use the demo, a presentation of the demo, and a request for feedback regarding risks and their intention to use.

The collected feedback from four representatives in four different banks, in addition to three expert engineers who worked in banking card issuance consultants, is summarized as the following:

- The idea seemed to be promising and extends the trust in banks for identity operations.
- The integration with bank systems was theoretically possible but still needed to be validated by a bank architect.
- A question was asked about the permission of the payment scheme to add the identity applet to the payment app, which should be accepted. However, it needs consultation regarding the specific payment scheme.
- After trying a demo using the card signing to share the identity with another bank, five participants out of seven, saw it as being easy to use but difficult to register.
- Bank representatives saw it as an opportunity for KYC enrolment and updates.
- Card issuance representatives thought that it has good potential and value for the customers.
- A general comment was made about linking Blockchain to crypto-currency and an explanation was provided that the scope is different.
- The average complexity level as collected from participants was 2 out of 10.

7.5. Analysis and Comparison

KYC2 [18] advised the need for key management, secret handling, and recovery options. This work proposes using banking smart cards for identity cryptographic operations and deploying the SSI concept based on Ethereum Blockchain; in particular, the curve exists for multiple-market banking Java Cards, which we validated in this proof of concept.

SCARAB [21] stores everything on-chain. This affects the scaling and system performance; furthermore, it stores identity-sensitive data on public servers, with no possibility to erase them.

BBM [5] is a Blockchain SSI model for open banking; however, SSBI uses the banking card, which is trusted and available to use by the customer for daily purchases. This work extends the use of the card to store and manipulate the identity transaction signing and allows the sharing of VCs with third parties through the Veramo SSI platform. This achieves the balance between usability and assurance without additional hardware tokens. This work attempted to implement a prototype, and to assess its feasibility and possibility of being integrated with the existing banking card issuance process.

The SSI framework prevents banking scam calls [7]. We leverage its use case and overcome the limitation of the signing curve on Java Card ED25519, which is required for Indy transactions. Instead, the framework is based on Veramo and Ethereum Blockchain, for which the required SECP256K1 signing curve exists in several banking cards, to issue

ethr DIDs and VCs. This fits within open banking use cases, and the proposed framework supports smart contracts to enable automated online financial services.

In [22], the authors proposed a protocol to achieve a high level of assurance (LoA) by combining a software wallet key part and a YubiKey 5Ci token [23] hardware key part. In contrast, in SSBI we leverage the banking use case and balance the usability and strong security without additional devices. Moreover, the framework supports strong authentication and VC exchange for open banking services, and not only the identity operations.

The authors in [24] proposed "Casper", a mobile identity wallet that enables inter-banking Know Your Customer (KYC) based on SSI. In this work, we combine high LoA using banking cards to improve the identity control and share a payment experience that is similar to that of using banking cards to support extension to several business values, such as smooth KYC, password-less login to mobile banking, and much more. We rely on Ethereum Blockchain, and the Veramo SDK to handle DIDs and VCs. In addition, we use a key recovery mechanism that combines a secret part from the bank with a known passphrase held by the customer.

PriFob [25] is a global online credential management solution that provides privacy, which is fog-enhanced and based on a publicly permissioned Blockchain. The efficiency and performance improved due to the adaptation of PoA and SoW consensus protocols. Although it is not applied directly for the banking use case, we focus on improving security and the customer experience by involving banking cards in the flow of identity sharing. However, the reliable encryption scheme and consensus algorithm in PriFob can be utilized in our future improvements to achieve better scalability.

A comparative analysis between SSBI and the previous work is summarized in Table 1.

Table 1. SSBI comparative analysis.

	KYC2 [18]	**SCARAB [21]**	**Onename.io [58]**	**PriFob [25]**	**Casper [24]**	**SSI Bank Scam Calls [7]**	**BBM [5]**	**SSBI**
Banking KYC exchange	Yes	No	No	No	Yes	No	Yes	Yes
Utilize banking card for identity operations	No	No	No	No	No	Yes	No	Yes
Recoverability	No	No	No	?	No	No	Yes	Yes
Flexibility	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Portability	Yes	Yes	No	Yes	Yes	No	Yes	Yes
Smart contract	Yes	No	?	Yes	No	No	Yes	Yes
Interoperability	Yes	Yes	No	Yes	No	Yes	Yes	Yes
Privacy protection	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Integrity and confidentiality	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Accountability	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Selective disclosure	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Achieve high LoA	No	No	No	No	No	Yes	No	Yes
Blockchain	Hyperledger Indy	Public ledger	Blockstack	Publicly permissioned Blockchain	Rahasak	Hyperledger Indy	Ethereum	Ethereum
Consensus	Plenum	Byzantine protocol	Stacks	PoA/SoW	PoW	Plenum	PoW	PoA

8. Conclusions and Future Work

This work is a step toward adopting and emphasizing the SSI identity-sharing concept for the open banking paradigm. The proposed platform relies on usability and customer trust in using a payment card as tamper-resistant secure key storage for identity cryptographic operations and transactions performed on Ethereum Blockchain. SSBI provides a mobile app and a web-based trace component to manage communication and reporting to entities and controllers including third-party agencies, banks, and financial institutes. Per-

sonal data are stored in the customer's mobile device while proofs are stored on Blockchain. Verifiable credentials are shared between the customers and the prover entities.

SSBI benefits from the trusted banking card industry and security measures such as PCI-DSS standards. The payment card secures the issuance process and the audited life cycle ensures attestation, genuine smart chip keys, and certified installed applets. We assume that the customer holds at least one active bank account in the initial phase. The platform eliminates fraudulent banking calls and financial identity theft due to the strongly encrypted credentials and proofs.

We reviewed the related solutions, proposed a solution, and described the framework architecture, layers, and components. In addition to the design objectives, we also describes the flow, including the initialization and authorization phases. This work provides a demonstrative proof-of-concept implementation based on banking Java Cards to sign the identity transactions on Ethereum Blockchain. We presented the integration of the banking card, personalization, applet development, and the flow to integrate with the banking issuance process. We presented a performance evaluation and security analysis, in addition to comparisons to related solutions.

The preliminary evaluation of our implemented prototype, which exploits the use of a banking card with a Blockchain-based SSI solution, is highly impacted by the capacity and Blockchain configuration, such as size of the network, the number of nodes, and the expected load. The system showed a stable throughput, which complies with the requirements of our preliminary proof-of-concept implementation, but does not satisfy the real-world banking use case. The customer expects faster transaction execution.

Hence, in future work, we will validate other options to improve the performance in terms of throughput and speed, with the aim of achieving a maximum of 15 s per transaction and a 1000 tps scale, which can fit with pilot real-world implementations. Furthermore, we will integrate the proposed approach with an existing bank system to implement practical settings and real test cases, in addition to validating end-to-end encrypted data sharing and cost analysis for identity operations.

Author Contributions: Conceptualization, K.A.M.A. and A.M.T.A.-E.; methodology, K.A.M.A.; software, K.A.M.A.; validation, J.F.W. and S.F.S.; formal analysis, K.A.M.A., A.M.T.A.-E. and J.F.W.; writing—original draft preparation, K.A.M.A.; writing—review and editing, K.A.M.A., J.F.W. and A.M.T.A.-E.; supervision, S.F.S. and A.M.T.A.-E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Open Finance. Available online: https://fastpayltd.co.uk/blog/what-is-open-finance (accessed on 25 March 2021).
2. Remolina, N. *Open Banking: Regulatory Challenges for a New Form of Financial Intermediation in a Data-Driven World*; SMU Centre for AI & Data Governance Research Paper No. 2019/05; SSRN: Rochester, NY, USA, 2019. [CrossRef]
3. Confessore, N. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. *The New York Times*. Available online: https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html (accessed on 17 May 2022).
4. Veramo. Available online: https://veramo.io/ (accessed on 22 December 2022).
5. Dong, C.; Wang, Z.; Chen, S.; Xiang, Y. BBM: A Blockchain-Based Model for Open Banking via Self-sovereign Identity. In Proceedings of the International Conference on Blockchain, Third International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, 18–20 September 2020; Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany, 2020; Volume 12404, pp. 61–75. [CrossRef]
6. Hyperledger Indy. Available online: https://www.hyperledger.org/projects/hyperledger-Indy (accessed on 10 April 2021).
7. Ahmed, K.A.M.; Saraya, S.F.; Wanis, J.F.; Ali-Eldin, A.M.T. A Self-Sovereign Identity Architecture Based on Blockchain and the Utilization of Customer's Banking Cards: The Case of Bank Scam Calls Prevention. In Proceedings of the 2020 15th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 15–16 December 2020. [CrossRef]
8. Allen, C. The Path to Self-Sovereign Identity. 2016. Available online: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html (accessed on 13 February 2022).

9. Tobin, D.R.A. The Inevitable Rise of Self-Sovereign Identity. Sovrin Foundation Technical Report. 2018. Available online: https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf (accessed on 15 March 2022).
10. Xu, J.J. Are blockchains immune to all malicious attacks? *Financ. Innov.* **2016**, *2*, 25. [CrossRef]
11. uPort.me. Available online: https://www.uport.me (accessed on 10 April 2021).
12. uPort Whitepaper. Available online: https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf (accessed on 10 April 2021).
13. Reed, M.S.D. Decentralized identifiers (dids) v0.12. In *Community Group Report*; W3C: Cambridge, MA, USA, 2019.
14. IPFS. Available online: https://ipfs.io (accessed on 10 April 2021).
15. Veramo Agents. Available online: https://veramo.io/docs/veramo_agent/introduction/ (accessed on 10 March 2022).
16. Veramo Specifications. Available online: https://identity.foundation/didcomm-messaging/spec (accessed on 22 December 2022).
17. W3C. Available online: https://www.w3.org/TR/vc-data-model/ (accessed on 15 April 2021).
18. Soltani, R.; Nguyen, U.T.; An, A. A New Approach to Client Onboarding Using Self-Sovereign Identity and Distributed Ledger. In Proceedings of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1129–1136. [CrossRef]
19. GDPR, E.U. GDPR. Available online: http://eugdpr.org/eugdpr.org.html (accessed on 12 June 2021).
20. Mühle, A.; Grüner, A.; Gayvoronskaya, T.; Meinel, C. A survey on essential components of a self-sovereign identity. *Comput. Sci. Rev.* **2018**, *30*, 80–86. [CrossRef]
21. Kokoris-Kogias, E.; Alp, E.C.; Siby, S.D.; Gailly, N.; Jovanovic, P.; Gasser, L.; Ford, B. Hidden in Plain Sight: Storing and Managing Secrets on a Public Ledger. *IACR Cryptol. ePrint Arch.* **2018**, *2018*, 209.
22. Abraham, A.; Schinnerl, C.; More, S. SSI strong authentication using a mobile-phone based identity wallet reaching a high level of assurance. In Proceedings of the 18th International Conference Security Cryptography, SECRYPT 2021, No. Secrypt, Online, 6–8 July 2021; Volume 1, pp. 137–148. [CrossRef]
23. Yubikey. Available online: https://www.yubico.com/at/product/yubikey-5ci (accessed on 15 March 2021).
24. Bandara, E.; Liang, X.; Foytik, P.; Shetty, S.; De Zoysa, K. A Blockchain and Self-Sovereign Identity Empowered Digital Identity Platform. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; Volume 2021. [CrossRef]
25. Baniata, H.; Kertesz, A. PriFoB: A Privacy-aware Fog-enhanced Blockchain-based system for Global Accreditation and Credential Verification. *J. Netw. Comput. Appl.* **2022**, *205*, 103440. [CrossRef]
26. Biometric Card. Available online: https://www.thalesgroup.com/en/markets/digital-identity-and-security/banking-payment/cards/emv-biometric-card (accessed on 12 April 2022).
27. Abraham, A. Self-Sovereign Identity—Whitepaper about the Concept of Self-Sovereign Identity including Its Potential. 2017. Available online: https://technology.a-sit.at/en/whitepaper-self-sovereign-identity (accessed on 10 March 2021).
28. Ogawa, A. What Is the Self-Sovereign Identity? The New Potential of Blockchain. Info-Com T & S World Trend Report, No. 346. 2018.
29. *ISO/IEC 29115:2013*; Information Technology—Security Techniques—Entity Authentication Assurance Framework. International Organization for Standardization: Geneva, Switzerland, 2013.
30. *NIST SP 800-63*; Digital Identity Guidelines. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020.
31. Solidity. Available online: https://solidity-by-example.org/app/ (accessed on 15 May 2021).
32. EIP-780. Available online: https://github.com/ethereum/EIPs/issues/780 (accessed on 15 March 2021).
33. Solidity Language. Available online: https://docs.soliditylang.org/en/v0.8.4/solidity-by-example.html (accessed on 15 March 2021).
34. eip-1056. Available online: https://eips.ethereum.org/EIPS/eip-1056 (accessed on 14 January 2021).
35. López, M.A. *Self Sovereign Identity: The Future of Identity: Self-Sovereignity, Digital Wallets, and Blockchain*; Technical Report; Inter-American Development Bank: Washington, DC, USA, 2020.
36. Infrastructure, E.B.S. EBSI's Technical Specification. Available online: https://ecas.ec.europa.eu/ (accessed on 10 July 2022).
37. EIP-1812. Available online: https://eips.ethereum.org/EIPS/eip-1812 (accessed on 1 March 2022).
38. Reactnative. Available online: https://reactnative.dev/ (accessed on 18 March 2021).
39. Nodejs. Available online: https://nodejs.org/en/ (accessed on 15 March 2021).
40. Expressjs. Available online: https://expressjs.com/ (accessed on 15 March 2021).
41. infura.io. Available online: https://infura.io/ (accessed on 15 March 2021).
42. Firestore. Available online: https://firebase.google.com/docs/firestore (accessed on 10 April 2022).
43. Typescript. Available online: https://www.typescriptlang.org/ (accessed on 10 May 2021).
44. Expo. Available online: https://docs.expo.dev/ (accessed on 13 November 2021).
45. mocha.js. Available online: https://mochajs.org/ (accessed on 12 August 2021).
46. Sqlite. Available online: https://sqlite.org/index.html (accessed on 12 April 2021).
47. PCI. Available online: https://www.pcisecuritystandards.org/ (accessed on 10 May 2022).
48. EMV. Available online: https://www.emvco.com/ (accessed on 15 April 2021).

49. Java Card Plaftorm. Available online: https://download.oracle.com/otndocs/jcp/java_card_kit-2.2.2-fr-oth-JSpec/ (accessed on 19 March 2022).
50. Fritsche, J.E.M.R.V.; Palma, L.M. Recommendations for implementing a Bitcoin Wallet Using Smart Card. Dep. Informática e Estatística—Univ. Fed. St. Catarina (UFSC), Campus Univ. Trindade Cx.P. 476/CEP 88040—Florianóp.—SC—Brazil 2018. Available online: https://repositorio.ufsc.br/bitstream/handle/123456789/192174/TCC%20Ricardo%20Fritsche%20Final.pdf?sequence=1 (accessed on 5 March 2021).
51. Bamert, T.; Decker, C.; Wattenhofer, R.; Welten, S. BlueWallet: The Secure Bitcoin Wallet. In *Lecture Notes in Computer Science*; Springer International Publishing: Berlin/Heidelberg, Germany, 2014; pp. 65–80.
52. BIP-32. Available online: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki (accessed on 18 March 2021).
53. *ISO/IEC 7816-8:2021*; Identification Cards—Integrated Circuit Cards—Part 8: Commands and Mechanisms for Security Operations. International Organization for Standardization: Geneva, Switzerland, 2021. Available online: https://www.iso.org/obp/ui/#iso:std:iso-iec:7816:-8:en (accessed on 1 March 2021).
54. Grüner, A.; Mühle, A.; Lockenvitz, N.; Meinel, C. Analyzing and comparing the security of self-sovereign identity management systems through threat modeling. *Int. J. Inf. Secur.* **2023**, 3. [CrossRef]
55. Hölzl, M.; Asnake, E.; Mayrhofer, R.; Roland, M. A password-authenticated secure channel for App to Java Card applet communication. *Int. J. Pervasive Comput. Commun.* **2015**, *11*, 374–397. [CrossRef]
56. Kertesz, H.B.A. Consistency analysis of distributed ledgersin fogenhanced blockchains. In Proceedings of the European Conference on Parallel Processing, Lisbon, Portugal, 1–3 September 2021.
57. Sun, W.; Dedahanov, A.T.; Shin, H.Y.; Li, W.P. Using extended complexity theory to test SMEs' adoption of Blockchain-based loan system. *PLoS ONE* **2021**, *16*, e0245964. [CrossRef]
58. OneName.io: The Bridge Between Physical & Digital Identity & Blockchain for the Billions. WordPress.com 2015. Available online: https://rywalk.wordpress.com/2015/02/13/onename-the-bridge-between-physical-digital-identity (accessed on 12 August 2021).