IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 October 2025, accepted 22 November 2025, date of publication 28 November 2025,
date of current version 5 December 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3638625

RESEARCH ARTICLE

Self-Sovereign Identity Empowered Automated Teller Machines

FAIRUZ RAHAMAN CHOWDHURY[1], MD MASUM ALAM NAHID[1], FARIDA CHOWDHURY[2], MD MASUM[3], UMIT CALI[4,5], (Senior Member, IEEE), AND MD SADEK FERDOUS[2,6], (Member, IEEE)

[1]Cryptic Consultancy Ltd., EC1V 2NX London, U.K.
[2]BRAC University, Dhaka 1212, Bangladesh
[3]Shahjalal University of Science and Technology, Sylhet 3114, Bangladesh
[4]Norwegian University of Science and Technology, 7034 Trondheim, Norway
[5]University of York, YO10 5DD York, U.K.
[6]Imperial College London, SW7 2AZ London, U.K.

Corresponding author: Umit Cali (umit.cali@ntnu.no)

ABSTRACT To withdraw cash or make a payment, customers can use their bank's interactive Automated Teller Machines (ATMs), debit card, or credit card systems. With these advances, customers no longer have to physically visit a bank to withdraw funds or make a payment. However, there are available vulnerabilities in the structure of current ATMs that put customer funds at risk and open the door to fraud such as various card frauds, skimming devices, shoulder surfing and so on. In recent years, a variety of approaches and suggestions have been proposed to address and eventually eliminate the problem of ATM fraud. In this article, we propose a novel ATM system based on Self-sovereign Identity (SSI) for conducting different types of monetary activities. The proposed system bolsters the security of ATM transactions in multiple ways, hence eliminating the above-mentioned frauds. Our proposed system also introduces a delegated transaction mechanism that allows one user to share credentials in a secure way to another user so as to enable the second user to use those credentials to perform monetary operations at an ATM without even having a registered account in the respective bank. In this article, we present the architecture of the proposed system which is based on a threat model and requirement analysis, discuss the respective protocol flows of several use-cases and analyse its performance. In addition, we examine the usability of the system using the *Cognitive Walkthrough* approach and formally analyse the security of the protocols using *ProVerif*, a state-of-the-art security verification protocol.

INDEX TERMS Automated teller machine, ATM, blockchain, decentralised identifier, decentralised identity, security, self-sovereign identity, SSI, verifiable credential.

I. INTRODUCTION

A gradual evolution of banking services has resulted in a wide-scale adoption of Automated Teller Machines (ATMs) all over the world. Indeed, ATMs have become a cornerstone of modern financial services. An ATM is a computing machine, connected to a specific bank or a network of different banks, that people use to carry out a number of financial tasks, such as cash withdrawals, cash advances, bill payments, balance inquiries and balance transfer between accounts without going to the bank [1]. ATM's ability to be used non-stop 24/7 has led to their immediate and widespread adoption. For foreign travelers, ATMs often offer the first gateway to withdraw cash at reasonable rates. According to World Bank data, the density of automated teller machines (ATMs) was 40.42 per 100,000 adults globally in 2023 [2]. National Cash Systems reported that over 10 billion transactions are processed through ATMs each year in the USA [3]. ATM Global Market Report 2025 states that the worldwide ATM market size will grow from $38.09 billion in the financial year 2024 to $39.54 billion in 2025 at a compound annual growth rate of 3.8% and the value of the global ATM market is estimated to grow up to $45.49 billion by 2029 at a compound annual growth rate of 3.6% [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamad Afendee Mohamed.

Unfortunately, due to ATM's widespread popularity and omnipresence, it has become the target of attackers all over the world, resulting in several attacks such as various card frauds, skimming devices and shoulder surfing [5]. Nearly two-thirds (65.2%) of all ATM fraud cases in Nigeria include some combinations of these attacks [6]. Europe experienced 202 malware and cyber attacks on ATMs in 2020, up from 140 in 2019 [7]. Since March 2021, the UK Police have arrested 22 individuals in connection with over 40 ATM attacks in England, Wales, and Scotland [8]. ATM crimes in America increased by 600% from 2019 to 2022 and global ATM crime database contains nearly 22,000 incidents of ATM crime [9]. In Europe, 250 million euros were reported to be stolen due to ATM frauds in 2019, maximum of 353 million in 2017 [10]. The most recent Bank Crime Statistics report from the FBI indicates that there were 254 robberies from ATMs in 2021, up from 220 in 2020 [11]. It is surprising to realise that Personal Identification Numbers (PINs) serve as the primary form of protection for the majority of ATMs. If an attacker can get access to both a debit/credit card and its corresponding PIN, the attacker can successfully launch an attack. This safety net is thin and easy to overlook. Because of this, a significant number of people, particularly those who regularly use ATMs are understandably concerned about their card security. The need for a better, more reliable, and user-friendly ATM authentication mechanism has increased significantly in recent years [5]. Consumers and commercial groups from around the world are putting pressure on ATM manufacturers to strengthen their security measures.

There have been multiple researches that have explored the possibility of tackling this particular bottleneck of ATMs, unfortunately, the security measures from a user's perspective remained more or less unchanged in the last decade or so. Thinking objectively, one way to tackle the above issue would be to eliminate the need for any card-based authentication and move towards a card-less system. Towards this aim, in this article, we are introducing a Self-sovereign Identity (SSI) empowered ATM system. The domain of identity management has historically been organisation-centric, meaning that the system designed for identity management has mostly served organisations, and the needs of users to manage their identities have been overlooked [12], [13]. SSI has emerged as a potential solution to invert this dynamic by facilitating a dynamic decentralised Identity Management System (IMS) with one major goal: ensuring the control of identity data and activities back to the users [13], [14]. In this article, we introduce a completely novel use-case of SSI where we utilise a blockchain-rooted SSI system to redesign the current ATM protocol for introducing a card-less ATM usage mechanism that offers much better security than the currently available practice.

A. CONTRIBUTIONS

The major contributions of the article are:

- The proposal and architecture of an SSI-integrated ATM system based on a threat model and requirement analysis.
- A Proof of Concept (PoC) implementation of the proposed system utilising readily available SSI libraries.
- Detailed performance, usability, and security analysis to show the practicality and applicability of the developed PoC.

B. STRUCTURE

We present a brief discussion about different aspects of ATMs and SSI in Section II. Section III presents the core architecture along with its threat model, requirements analysis, architecture and implementation aspects. We discuss various data models and algorithms in Section IV. Use-cases and their corresponding protocol flows are presented in Section V. We explain how the developed system satisfies its requirements and present its performance, usability and security analysis in Section VI. We review the related work along with a comparative analysis with the current work in Section VII. Finally, we conclude in Section VIII.

II. BACKGROUND

In this section, we briefly discuss about the core banking system (Section II-A), ATM along with its protocol & cardless ATM transactions in Section II-B and Blockchain & SSI in Section II-C.

A. CORE BANKING SOFTWARE

The core banking software (CBS) is the central engine that runs the core banking and financial operations on a regular basis [16]. It enables the processing of various financial transactions and facilitates the management of the central ledger of the system to keep track of customer data such as account balances, transaction history, loan information, and so on. The CBS acts as the backbone of the system, seamlessly linking various components like ATMs and online banking platforms, ensuring continuous availability and providing real-time essential services like financial transactions and online banking. In our suggested system, we present a unique way for user authentication while keeping the key functionalities of the existing Core Banking System intact. This technique assures that financial institutions can smoothly implement the upgraded authentication mechanism while maintaining their current banking operations. Our design prioritises interoperability, enabling any bank to implement the suggested authentication technique with minimal changes to its CBS infrastructure where the primary goal is to authenticate the users, leaving the rest of the transaction procedure unchanged. To provide a thorough knowledge of the transaction process, we have also illustrated the communication flow between the ATM and the bank's system in Section V-B3. This detailed representation clarifies how the authentication method may work inside the larger financial ecosystem.

B. ATM

Visiting a bank branch was once the most common method of transacting funds. A human teller in a bank branch accepts deposits, processes withdrawals, cashes cheques, and performs other tasks. However, the Automated Teller Machine (ATM) made its debut in the 1967 [17]. This innovation transformed the banking industry by allowing customers to withdraw cash and perform other transactions in an ATM without visiting a bank branch. An ATM is a 24-hour, 365-day-a-year computerised teller machine that provides financial services such as cash withdrawals, cash deposits, bill payments, balance inquiries, and balance transfers between accounts. ATM has several parts, namely, an ATM computer with an ATM Transaction processor, card reader, keypad, output display, modem, and cash dispenser. The ATM Transaction processor is connected to the ATM networks and, through these networks, to the user's bank. A user inserts their bank card in the card reader which initiates the transaction. At that point, the ATM asks for a verification PIN and once inserted and verified by the ATM, a few actions are displayed to the user to choose his/her activity. When the user selects one action, the ATM transmits the EMV transaction code, PIN, and the action request to the user's bank to authorise the transaction. Based on the result of the authorisation, the ATM approves or denies the user request [18]. If a withdrawal is selected and approved, the bank will debit the user's account. This transaction is transmitted via ATM networks and processors. After that, the ATM will start the cash dispensing process. A receipt for the transaction is then printed once the cash has been dispensed from the machine.

1) ATM STANDARD

The ATM system uses a specific standard for its functionalities. ISO 8583 [19] is a global standard for interchanging messages involving credit/debit cards and other financial card transactions. This protocol has been standardised by the International Organisation for Standardization (ISO) and it specifies a message format and a communication flow for exchanging these transaction requests and responses across systems. The vast majority of in-store electronic funds transfer point-of-sale (EFTPOS) [20] transactions and automated teller machine (ATM) transactions use ISO 8583 at some stage in the communication chain. In particular, the ISO 8583 standard is used by the Mastercard, Visa, and Verve networks, among others, for their authorisation messages [21]. Despite ISO 8583's role in defining a common standard, systems and networks rarely make use of it in its purest form. ATM message protocols such as NCR's NDC and Diebold's 911/912 are based on the ISO 8583 standard [22]. Though we employed ISO 8583 in our proposed system, there are other emerging standards, such as the ISO 20022 standard [23], which is intended to provide a more flexible and extensible message format. However, we have utilised ISO 8583 in our suggested system because it is used by the majority of ATMs, including legacy ones [24].

TCP/IP is the protocol utilised for low-lever interconnections in ATM. It is a data network protocol for packet switching that specifies a global standard for communications between two devices. In earlier days, X.25 protocol was used for low-level interaction in ATM [25].

2) CARDLESS ATM TRANSACTION

With cardless ATMs, access to the ATM is enabled through the user's bank's mobile app instead of using a physical bank card. Instead, it utilises account verification methods, such as verification via text messages or a banking app. Cardless ATMs can operate in various ways. Some of those are presented below.

- **Quick Response Codes (QR):** QR codes are square barcodes commonly found on posters or ads, enabling swift access to a system. These codes can also be created for single-use purposes. Certain banking apps permit unlocking an ATM by scanning a QR code displayed on the ATM screen through the bank's app [26].
- **Near-field communication (NFC):** NFC technology, found in payment cards and digital wallets, allows for tap-to-pay features. Certain cardless ATMs offer the option to tap a connected phone or smartwatch with the user's bank's mobile app to perform transactions, akin to mobile payments at a store [27].
- **Biometrics:** Biometric verification methods, such as facial recognition or fingerprint scanning, can unlock the user's smartphone. Likewise, certain banks store users' biometric data, enabling their ATMs to verify users' identities for future transactions [28]. This eliminates the need to present a user's card when withdrawing cash or conducting other activities.
- **Access Codes:** Certain cardless ATMs allow a user to pre-configure the user's transaction using the bank's mobile app before reaching the ATM. The app provides a one time code that is entered into the ATM to complete the transaction [29].

Cardless ATMs offer a convenient, teller-free option for cash withdrawals and transactions. By adhering to cybersecurity best practices, securing user's phone, and ensuring it is charged, using cardless ATMs provides a safe means to access and manage a user's account. However, not all ATMs offer this functionality; only modern ATMs designed to accommodate cardless transactions allow the use of smartphones for access.

C. SELF-SOVEREIGN IDENTITY (SSI)

The notion of Digital Identity is used to represent a user (a person) within an organisation (context or application domain) [30]. It is a crucial component for facilitating personalised online services provided by online Service Providers (SPs). Identity management can be regarded as the mechanism for managing the digital identities of users by their organisations or the users themselves. An IMS is used to manage user identities by their respective organisation.

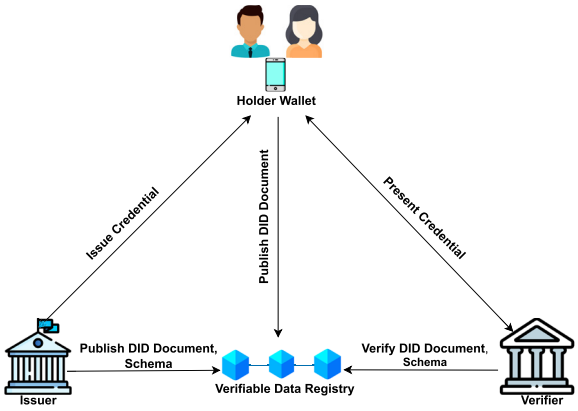


FIGURE 1. Issuer, Holder, Verifier trust triangle.

There are several Identity Management models such as the SILO model, the federated model, the single-sign-on (SSO) model, and so on [31]. Based on these models, there are several IMS such as SAML [32], OpenID [33], OAuth [34], and so on. Historically, all these IMS are organisation-centric. This means that these IMS are used mostly by the SPs to manage their user identities. The wide-scale adoption of online services across many application domains has forced the user to create numerous digital identities in these domains. The consequence is that users find it increasingly difficult to manage these online identities. Even worse, creating digital identities is dependent on centralised organisation (providers) and users have less control over their identity data and have little knowledge of how their identities are being (ab)used by these providers [30]. There has been a long gap for a more user-centric identity management approach.

Recently, Self-Sovereign Identity (SSI) has emerged as a user-controlled identity management solution. The main motivation of SSI is to offer users more control over his/her identity data. In addition, only the user decides how to share their identity data with others [35].

The two constructs, *Decentralised Identifiers* (DIDs) [41] and *Verifiable Credentials* (VCs) [42], developed by the W3C community, act as the central components of SSI [13]. A DID is a user-generated, globally unique identifier linked to an entity's cryptographic keys. Within an SSI ecosystem, an entity can be a user, an organisation or even any object that needs to be identified, and a DID is used to uniquely identify an entity. Entities can generate as many DIDs as required without relying on any other party. Every DID has a corresponding JSON object, called *DID Document* (DID Doc, in short) which contains its linked cryptographic public keys and further metadata such as contact information, service endpoints and so on. A Verifiable Credential (VC) is a digitally signed document containing claims regarding an entity (called the holder) by another entity (the issuer). A claim is a statement regarding a holder made by an issuer. The holder stores VCs from different issuers at their end within a wallet software (a desktop or a mobile app) and releases these VCs, individually or in combination, to different verifiers when required in exchange for an action, e.g. accessing an online service where the verifier is actually an SP. SSI envisions the use of a verifiable data registry, e.g. a blockchain, for storing DIDs and their corresponding DID Docs because of the immutability, persistence, and decentralisation properties of such a registry. The relationship between different SSI constructs and entities is presented in Figure 1.

The holder, the verifier, and the issuer are the three main entities in this paradigm. An issuer stores a DID Doc along with public key and other cryptographic materials to a blockchain. Upon receiving a request from a holder, the issuer creates a VC and uses the corresponding private key to digitally sign the VC. Then the VC is issued to the holder who stores it in their wallet. When a verifier requests a proof of the VC, the holder's wallet generates and returns a proof presentation, containing the VC, with the consent of the holder. Since the proof incorporates the issuer's DID, the verifier can access the DID document from the blockchain, retrieve the issuer's public key, and verify the VC's authenticity by cross-checking the digital signature with the public key, thereby ensuring that the VC has not been tempered with.

Sometimes, an additional optional entity, known as the *Mediator*, is used. A mediator acts as an intermediary between an SSI wallet and other SSI entities and is responsible for forwarding messages back and forth between the wallet and other entities. The messages handled by the mediator are end-to-end encrypted and hence, the mediator has no knowledge of their contents, except that something is exchanged between two parties. Throughout the SSI lifecycle during credential issuance, storage, and verification, entities such as issuers, holders, verifiers and mediators communicate securely using DIDComm Messaging (Decentralised Identifier Communication) [43]. DIDComm provides a standardised, end-to-end encrypted messaging protocol that leverages DIDs for establishing private, authenticated, and interoperable communication between agents and wallets.

The SSI entities (Issuer, Holder and Verifier) engage in the following steps to interact with each other:

- **Establishing a connection:** Before any interactions between two SSI entities, an SSI connection must be established between them. As part of this step, DIDs of both parties are exchanged which are used to retrieve the corresponding DIDDoc of each party and verify it. From this DIDDoc, each party extracts different metadata, including the public key, of the other party. Any two entities may establish a connection using a mediator.
- **VC Issuance:** The issuer issues a VC to the user (holder) which is transferred using the previously established SSI connection and stored in the holder's wallet upon successful verification. The issuer and the holder interact in this step using a mediator.

TABLE 1. Modelled threats.

Threat	Implication
Spoof. (T1)	• Pretending as an account holder when making transactions at the ATM. • Cloning ATM cards to perform transactions.
Tamper. (T2)	• Tampering transaction data.
Repu. (T3)	• Denying any transaction.
Inf. Dis. (T4)	• Disclosing transaction information to a third party (attacker). • Retrieving cryptographic keys from storage.
DoS (T5)	• Denial of banking services so that banking transactions cannot be facilitated.
Elev. (T6)	• Initiating fraudulent transactions.
Repl. (T7)	• Intercepting a previously submitted transaction and resubmits it with malicious intents.
L. Con. (T8)	• Carrying out a transaction without the consent of a user.

- **Proof Request:** When a verifier requests proof, the holder releases the proof presentation of a VC (or a set of VCs) to the verifier using the previously established SSI connection. Upon successful verification of the shared VC(s), the verifier accepts the VC. The verifier and the holder interact in this step using a mediator.

III. PROPOSAL

The primary objective of our proposed system is to get rid of the need for a card and PIN to perform monetary activities at an ATM. To achieve this objective, we are proposing an SSI based solution where no user would be required to insert their card into the machine as well to input their PIN. We showcase the applicability of the proposed solution by developing a Proof-of-Concept (PoC). The PoC itself has been designed and developed based on a well-established research methodology, called Design Science Research (DSR) methodology [65], [66]. In the DSR methodology, problems are solved by developing a PoC through a design, develop, demonstrate and evaluate process. In this section, we present different aspects of our proposal with the help of a threat model (Section III-A), requirement analysis (Section III-B), architecture (Section III-C) and implementation (Section III-D). The demonstration of the developed PoC is presented using different use-cases in Section V while the evaluation of the PoC is presented in Section VI.

A. THREAT MODELING

A threat can be either an intentional act (such as a DoS attack) or an unintentional occurrence (failure of a storage device). The objective of threat modelling is to identify, analyse, and mitigate any potential threats to a system [44]. Most of the time, threat modelling is conducted during the system design phase to find security flaws and understand how design, code, and other decisions affect security [45]. There are several methodologies for threat modelling, such as STRIDE [46], Process for Attack Simulation and Threat Analysis (PASTA) [47], Visual, Agile, and Simple Threat (VAST) [48] and many more. For our proposed system, we have used *STRIDE* which is created by Microsoft [46]. STRIDE is the acronym for each of the six threat categories it deals with: Spoofing (denoted as *T1*), Tampering (denoted as *T2*), Repudiation (denoted as *T3*), Information disclosure (denoted as *T4*), Denial of service (denoted as *T5*), and Elevation of privilege (denoted as *T6*). In addition to these, another security threat, called Replay attack (denoted with *T7*), is considered. Furthermore, we consider one privacy threat: Lack of consent (denoted as *T8*). Table 1 presents the implications of these threats with respect to this research.

B. REQUIREMENT ANALYSIS

To ensure that the proposed system performs its intended operations and mitigates different security and privacy threats, we have formulated a set of functional, security, and privacy requirements. These requirements are presented below.

1) FUNCTIONAL REQUIREMENTS

The functional requirements are as follows.

- F1: Users should be able to carry out regular financial transactions (e.g., balance query, cash withdrawal) using ATMs enabled by SSI.
- F2: The system must ensure the transparency of each financial operation so that every authorised user can review their own monetary activities at any time.
- F3: In addition to regular transactions, we are considering an advanced transaction called a *delegated transaction*. A delegated transaction enables one user to delegate the task of a financial activity (e.g. cash withdrawal) to someone else.

2) SECURITY REQUIREMENTS

The following are the security requirements.

- S1: The system must restrict access to sensitive financial services only to authenticated and authorised users only. This will mitigate T1 and T6.
- S2: The system must utilise the digital signature in different steps to ensure the non-repudiation of the data. The digital signature must be generated using the hashes of the respective data. This will mitigate T2 and T3.
- S3: Transaction data must be securely transmitted and stored in the system to mitigate T4.
- S4: The system must guard against any DoS attack. It will mitigate T5.
- S5: The system should also extensively utilise nonces as a protective measure against any replay attack, thereby mitigating T7.

3) PRIVACY REQUIREMENTS

The only privacy requirement is presented below.

- P1: Each financial activity must be carried out only with the user's consent, mitigating T8.

C. ARCHITECTURE

The architecture has a number of components to be used on the user side and on the bank side. On the user side, an SSI wallet is used whereas on the bank side different SSI components are integrated with the Bank Control Server which simulates the functionalities of a core banking system. The high-level architecture of the proposed system, consisting of the entities and their interactions, is presented in Figure 2. This figure illustrates a secure banking system that integrated digital wallet, DID and blockchain technology. Equipped with a digital wallet, the user communicates with the bank through the ATM or the bank website. While HTTPS communication is used for transaction over the bank or ATM, the mediator leverages didcomm to ensure secure, authenticated, and privacy-preserving communication between the user's wallet and the banking agent. At the core the Bank Controller Server, which acts as a gateway to internal services such as the Fabric Server, which controls transaction validation on a private, permissioned blockchain (Hyperledger Fabric) and the Aries Cloud Agent, which handles secure identity verification and connects the Indy Ledger and Revocation Registry for decentralised identity management. A seamless and secure banking experience is further ensured by the Cache Server and Database, which guarantee effective data management and storage for both identity-related and transactional processes. In this section, we briefly discuss these entities.

- **User:** The user is the principal entity of the system. The main objective of the whole system is to serve users so that they can carry out the financial activities provided by the bank. We have two types of user: general users (denoted as users) and delegated users. A delegated user can carry out delegated transactions on someone's behalf.
- **Mobile Wallet:** Both users are equipped with a mobile-based SSI wallet which is used to establish and save SSI connections as well as to store VCs within the wallet and share them with the verifiers when required. Thus, the wallet acts as the core component by which any user interacts with banks and their ATMs to carry out SSI-enabled financial activities.
- **Bank Controller Server:** The bank controller server (bank controller, in short) is responsible for the financial activities. An SSI agent (SA) is integrated with the bank to ensure that it can facilitate SSI-based banking activities. The bank controller invokes the SA to interact with a public SSI blockchain. A smart contract, Bank Chain-code (BCC), is also implemented within the bank controller to interact with a bank private blockchain (discussed later). We assume that the core banking software (CBS) of the bank is modified in such a way that it can interact with the SSI agent and the bank is connected to the ATM(s) using a method used in the traditional setting. In addition, we assume that the corresponding online service for the bank acts as the interface for the user for different SSI-related activities, discussed later.
- **ATM:** ATM is the interface for the user to carry out SSI-enabled financial activities. Each user uses their mobile wallet to interact with the ATM that initiates any SSI-enabled financial activity.
- **Blockchain Platforms:** We have used two different blockchain platforms (or blockchain in short): an SSI blockchain and a smart-contract blockchain. An SSI blockchain is a blockchain platform specifically used for SSI. This is where the issuer's public DID, DIDDoc and VC schema and other metadata for SSI are created and stored. The SSI agent of the bank and the SSI wallets are connected to the SSI blockchain. On the other hand, a smart-contract blockchain is a private blockchain platform with smart-contract facility. It is used by the bank authority to store transaction details. Since not everyone should be able to view the specifics of a transaction, a private permissioned blockchain is utilised.
- **Mediator:** The Mediator serves as an intermediary, maintaining connections on behalf of an edge agent (wallet). It is also known as a cloud agent. Its functionalities have been discussed in Section II-C.
- **Database and Cache server:** In our system architecture, MongoDB [49] has been used as the primary database to store nonces to effectively mitigate replay attacks. Additionally, Redis [50] has been used as a cache database to improve system performance by reducing response times by fast data retrieval. This combination ensures both security against replay attacks and optimised performance for our system.
- **Revocation Registry:** We have integrated a revocation registry into our system using the Indy Tails Server to ensure a dedicated mechanism for storing and managing VC revocation data within our system.

D. IMPLEMENTATION

We have developed a Proof of Concept (PoC) using our proposed architecture. This PoC utilises a set of protocol flows for different use-cases. We explore these protocol flows and use-cases in Section V. Here, we discuss the technology stacks used for developing the PoC. It should be noted that the bank and ATM functionalities have been simulated during the development process.

There are a number of components in our system: the SSI stack, SSI wallet, bank web application, bank agent, and virtual ATM Android application. In the following, we briefly discuss each of these components.

1) SSI STACK

The core SSI stack used in the PoC is Hyperledger Aries [51]. Hyperledger Aries consists of a set of libraries, supporting the previously discussed SSI constructs, to develop SSI

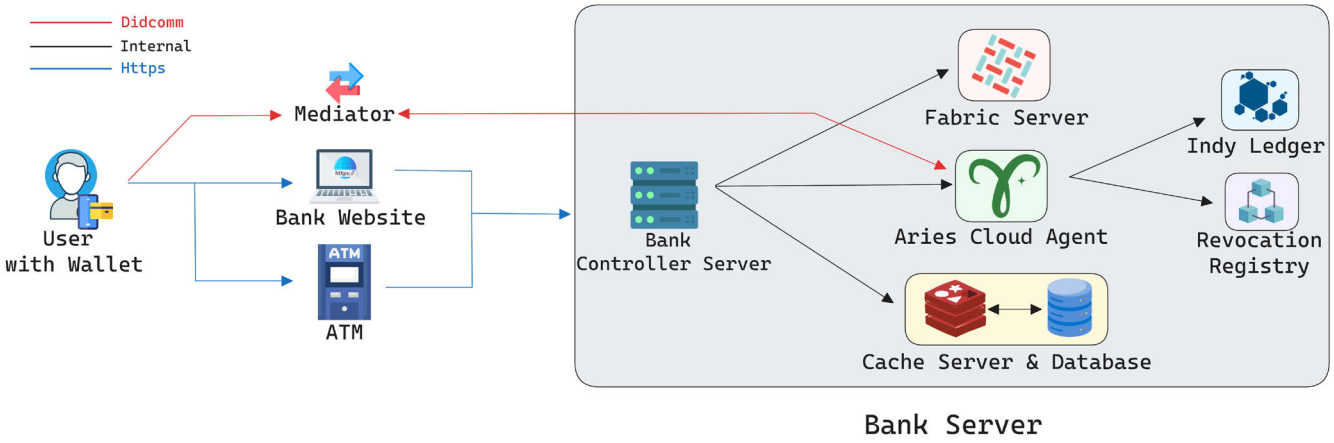


FIGURE 2. High-level architecture.

applications. The libraries are available in different programming languages such as Python, DotNet, Go, JavaScript. For our PoC, we have utilised the Hyperledger Aries Cloud Agent Python (ACA-Py) library [52]. For secure management of wallet keys and secrets, we employed Askar, a storage and key management library maintained under the OpenWallet Foundation [60]. Askar provides encrypted, tamper-resistant storage for cryptographic keys, root secrets, and other sensitive data, ensuring confidentiality and integrity through strong encryption and key-derivation mechanisms. ACA-Py relies on Hyperledger Indy [53], a purpose-built blockchain platform for SSI (denoted as SSI Blockchain in this article), for creating DIDs, storing DIDDoc and credential schemas and so on. Indicio [54] and Sovrin [56] are real-life implementation of Indy which operates a Hybrid Blockchain where anyone can read data from, however, the write access is restricted and costs money. That is why, for our PoC, we have leveraged a test network of Indy blockchain, called BCovrin Indy Network [57], which allows experimentation without requiring transaction fees or manual endorsement. For this purpose, we employed an endorser-capable agent that enabled the submission of various ledger transactions such as DID documents, schemas, and credential definitions without relying on external endorsement. This setup reflects the permissioned nature of production Indy networks, where all write operations are governed by formal operational policies and trusted endorsement mechanisms that prevent arbitrary or malicious transactions and help ensure ledger integrity. Hyperledger Indy Tails Server is a file server built to accept, store, and serve Indy Tails files. When a Revocation Registry is created, an issuer of Indy AnonCred verifiable credentials will generate an Indy tails file, which must be made publicly available to all holders of credentials that are referenced by that Registry. When responding to a proof request with proof that includes claims from a revocable credential, credential holders use the tails file to build a zero knowledge proof of non-revocation about their credentials and include it. A tails file is linked to an accumulator and the factors that make it up [58]. It is a binary file that has a list of factors for an accumulator that were chosen at random. Instead of small numbers like 2, 3, or 7, these factors are huge numbers. Most of the time, there are hundreds of thousands to tens of millions of these numbers in a tails file. A tails file is not a secret. It is available to everyone as standard text and can be downloaded for free by anyone. This file's contents never change. To achieve this goal, we have used indy tail server provided by bcgov [59].

2) BANK SSI AGENT

The SSI agent has been developed using NodeJS [61] which depends on the ACA-Py library to serve different SSI functionalities. It is seamlessly integrated with the BCovrin test network, serving as the SSI public blockchain. Additionally, a tails server is employed to manage the revocation process, providing essential support to the SSI agent's operations. ACA-Py framework serves as a foundation to develop ecosystems of Verifiable Credentials (VC). It uses a number of verifiable credential formats and protocols to function in the second and third levels of the Trust Over IP architecture [37]. ACA-Py runs on servers (cloud, enterprise, IoT devices, and so forth), and is not designed to run on mobile devices. Most notably, ACA-Py supports protocols for issuing, verifying, and storing verifiable credentials using the W3C Standard Verifiable Credential Data Model format using JSON-LD with LD-Signatures [38] and BBS+ Signatures [39], as well as the Hyperledger AnonCreds verifiable credential format [40]. While LD-Signatures are used to semantically sign JSON-LD credentials in a verifiable way, the BBS+ Signature Suite extends this capability by enabling multi-attribute credential signing along with selective disclosure via zero-knowledge proofs.

3) BANK PRIVATE LEDGER

For our PoC, we have used Hyperledger Fabric [62], a state-of-the-art private blockchain platform supporting the development and deployment of smart-contract (known

as *chaincode* in Fabric) in general-purpose programming languages such as Go, JavaScript, Java and so on. We have used JavaScript to write our chaincode. We have used default configurations of Fabric to deploy the chaincode which includes CouchDB database for managing blockchain states and two organisations. It is to be noted that a complex network configuration with multiple organisations with strong privacy features such as channels could be deployed with Fabric. Since it will depend on a specific implementation scenario, we have not considered it in our PoC. Finally, the bank agent also uses Redis cache for caching session cookies and frequently used server static responses to improve response time.

4) BANK WEB APPLICATION

The bank web application provides the UI for the users to interact with the bank agent for different activities. It has been developed with ReactJS.

5) BIFOLD WALLET

For the PoC, we have utilised an open source SSI wallet called Bifold Wallet [63] which is designed to enhance the way we interact with digital identities, making the process both secure and user-friendly. Because of its React Native foundation, it functions flawlessly across a variety of platforms and devices, including iOS and Android. With an emphasis on making verifiable credentials easy and handy for everyone, it is a prime example of a digital wallet. We utilised the BCovrin test network with Bifold wallet for our PoC. This wallet utilises the Indicio public mediator [55] to interact with the SSI Agent of the bank.

6) VIRTUAL ATM ANDROID APPLICATION

We have developed a react-native Android application to simulate the functionalities of an ATM. This ATM application is connected to the bank Agent. Users interact with the ATM application to carry out financial activities related to an ATM.

7) ISO 8583 PROTOCOL

The ISO 8583 standard specifies a message format with a fixed-length header, variable-length message body, and fixed-length trailer [19]. The message body includes transaction-related details, including the amount, transaction type, and account information. The header and trailer include message format, message type, and other control information. The ISO 8583 standard also specifies a protocol for the transmission of messages such as how messages are delivered, acknowledged, and handled. Several communication methods, such as TCP/IP, X.25, and asynchronous serial communication, can be used to implement the protocol. This protocol is commonly used in the financial sector to handle transactions such as credit card payments, ATM withdrawals, and bank transfers. In our system, we have used this protocol to exchange data between the ATM and the bank agent. Specifically, we utilised ISO 8583 data fields including 0, 4, 12, 13, 53, 63, 102, 124, and 125 to facilitate secure and structured exchange of transaction and credential-related information.

TABLE 2. Cryptographic notation.

Notations	Descriptions
A	ATM
A_n	ATM serial number
B	Bank
D_{uid}	Delegation user identifier
G_{uid}	General user identifier for ATM
K_{sh}	Shared key between Bank and ATM
K_U^B	Public key of user for B
K_B^U	Public key of B for user
$K_U^{-1\|B}$	Private key of user for B
$K_B^{-1\|U}$	Private key of B for user
M	Mediator
N_i	Nonce
U_i	General user
U_D	Delegated user
U_{nid}	User's national identifier
$H(M)$	SHA-256 hashing operation of message M
$VC_B^{U_i}$	VC issued by B for U_i
$VC_B^{U_D}$	VC issued by B for U_D
ACN	Account number
DID_B^U	DID of bank for user
DID_U^B	DID of user for bank
PNR	Proof of Non-Revocation
uid	ATM or delegation identifier
url_B	Bank connection URL
url_D	Bank connection URL for U_D
url_S	Bank service URL
$NAME$	User Name
$conID$	User's connection identifier with Bank
$AMOUNT$	Transaction amount
$SENDER$	Sender of the transaction
T	Periodic interval
$\|\|$	Alternative option
$\{\}_k$	Encryption using public key k
$\{\}_{k^{-1}}$	Signature using private key k
"..."	Functionality offered by website and ATM
$[\ldots]_{https}$	Communication over an https channel
$[\ldots]$	Communication on an unencrypted channel
$[\ldots]_k$	Communication channel encrypted with key k
$[[..]]$	Optional data

IV. DATA MODEL AND ALGORITHMS

In this section, we present the data model and algorithms used in our implemented system. First, we introduce the cryptographic notations in Table 2. These notations will be used in the data model as well as our protocol flows discussed in the following sections. Later, we present the data model (Section IV-A) used in the protocol flow and the algorithms (Section IV-B).

A. DATA MODEL

The protocol used in the system is a request-response protocol in which one entity submits a request and the other entity replies with a corresponding response. For different use-cases, different requests and and responses are needed. The data model for these requests and responses is presented in Table 3. We denote the sets of requests and responses as *REQ*

TABLE 3. Data model.

$REQ \triangleq \langle registerReq, loginReq, proofReq_{1-2}, credReq, connectionReq, serviceReq, executeTransactionReq, atmInitiateReq, didExchangeReq, createDelegationReq, receiveDelegationReq, updateUserReq, revokeVCReq, fetchUserReq, symKeyReq, checkRevocationReq \rangle$
$RESP \triangleq \langle registerResp, loginResp, proofResp_{1-2}, credResp, connectionResp, executeTransactionResp serviceResp, atmInitiateResp, updateUserResp, didExchangeResp, createDelegationResp, receiveDelegationResp_{1-2}, revokeVCResp, fetchUserResp, symKeyResp, checkRevocationResp \rangle$
$atmData \triangleq \langle ATMSignature, metadata \rangle$
$atmUserInputData \triangleq \langle option, [[AMOUNT]] \rangle$
$atmInitiateData \triangleq \langle atmInitiateReq, atmData \rangle$
$ATMSignature \triangleq \langle \{H(A_n, N_i)\}_{K_A^{-1}} \rangle$
$BankSignature \triangleq \langle \{H(resp)\}_{K_B^{-1}} \rangle$
$credMetaData \triangleq \langle [[G_{uid}, D_{uid}, PNR]] \rangle$
$delegationData \triangleq \langle credMetaData, amount \rangle$
$option \triangleq \langle Withdrawal \mid\mid Balance\ Check \rangle$
$revocationStatus \triangleq \langle Issued \mid\mid Revoked \rangle$
$transactionType \triangleq \langle General \mid\mid Delegated \rangle$
$userData \triangleq \langle ACN, NAME, AMOUNT, [[metadata]] \rangle$
$VC_B^U \triangleq \langle \{U_{nid}, ACN, G_{uid}, NAME\}_{K_B^{-1\mid U}} \rangle$
$VC_B^{U_D} \triangleq \langle \{D_{uid}, SENDER, AMOUNT\}_{K_B^{-1\mid U}} \rangle$

TABLE 4. Request data model.

$attrReq \triangleq \langle a_1, a_2, ..., a_n \rangle$
$atmInitiateReq \triangleq \langle transactionType, uid \rangle$
$checkRevocationReq \triangleq \langle credMetaData \rangle$
$connectionReq \triangleq \langle Establish\ Connection \rangle$
$createDelegationReq \triangleq \langle Create\ Delegation \rangle$
$credReq \triangleq \langle Get\ Credential \rangle$
$didExchangeReq \triangleq \langle DID_U^B \rangle$
$executeTransactionReq \triangleq \langle atmUserInputData, atmData \rangle$
$fetchDelegationReq \triangleq \langle [[serviceCookie]], D_{uid} \rangle$
$fetchUserReq \triangleq \langle serviceCookie, [[national\_id, G_{uid}]] \rangle$
$loginReq \triangleq \langle Login \rangle$
$proofReq_1 \triangleq \langle NAME, G_{uid}, ACN \rangle$
$proofReq_2 \triangleq \langle D_{uid}, SENDER, AMOUNT \rangle$
$receiveDelegationReq \triangleq \langle Receive\ Delegation \rangle$
$registerReq \triangleq \langle Register \rangle$
$revokeVCReq \triangleq \langle credMetaData \rangle$
$serviceReq \triangleq \langle url_S, [[serviceCookie, inviteCookie]] \rangle$
$symKeyReq \triangleq \langle atmMetadata \rangle$
$updateUserReq \triangleq \langle [[credStatus, connectionStatus]] \rangle$

and *RESP* respectively and these sets are defined as presented in Table 3. Each of these requests/responses consists of several data items or simply a string. In some data models, there are a couple of options (e.g. *option*, *transactionType* and so on in Table 3) divided by the symbol " || " which indicates only one of these options will be available depending on the respective use-case. The *BankServicePage* in the *serviceResp* data model indicates that it is a web page for that respective purpose.

B. ALGORITHMS

The pseudocode for the core functionalities of the system is presented in Algorithm 1 and Algorithm 2 where Algorithm 1

TABLE 5. Response data model.

$attrResp \triangleq \langle av_1, av_2, ..., av_n \rangle$
$atmInitiateResp \triangleq \langle Success \rangle$
$checkRevocationResp \triangleq \langle revocationStatus \rangle$
$connectionResp \triangleq \langle createInvitationResp \rangle$
$createDelegationResp \triangleq \langle D_{uid} \rangle$
$credResp \triangleq \langle VC_B^U \rangle$
$didExchangeResp \triangleq \langle DID_B^U \rangle$
$executeTransactionResp \triangleq \langle Success \rangle$
$fetchDelegationResp \triangleq \langle delegationData \rangle$
$fetchUserResp \triangleq \langle userData \rangle$
$loginResp \triangleq \langle Success, serviceCookie \rangle$
$proofResp_1 \triangleq \langle VC_B^U, PNR \rangle$
$proofResp_2 \triangleq \langle VC_B^{U_D}, PNR \rangle$
$receiveDelegationResp_1 \triangleq \langle createInvitationResp \rangle$
$receiveDelegationResp_2 \triangleq \langle VC_B^{U_D} \rangle$
$registerResp \triangleq \langle Success \rangle$
$revokeVCResp \triangleq \langle Success \rangle$
$serviceResp \triangleq \langle Bank\ Service\ Page \rangle$
$symKeyResp \triangleq \langle A_n, K_{sh} \rangle$
$symKeyUpdate \triangleq \langle updated\ K_{sh} \rangle$
$updateUserResp \triangleq \langle Success \rangle$

represents the pseudocode for the chaincode simulating the bank functionalities and Algorithm 2 outlines the pseudocode for the SSI agent of the bank.

In Algorithm 1, we have presented six functions, each of which presents a specific functionality. Among them, the functions *register*, *login* and *withdrawal* are used to register a user in the bank and to facilitate login and cash withdrawal functionalities for the users respectively. The other three functions, storeData, retrieveData and validateData, are respectively used to store, retrieve and validate data for the users.

On the other hand, the functions in Algorithm 2,, are utilised by the SSI agent of the bank. The SSI agent uses the *createInvitation* function to create an SSI invitation. The *handleInvitation* function is crucial to handle an invitation for creating a secured connection between two parties. Within this function, the DID of the user to be used with the bank (DID_U^B) is retrieved from the invitation request (lines 3 and 4 in Algorithm 2). From this DID Doc, the public key of the user to be used with the bank (K_U^B) is retrieved and the respective pairs of DID_U^B and K_U^B are stored in the bank agent wallet (lines 5 and 6 in Algorithm 2). Then, the SSI agent generates a key pair for the SSI connection with the user (K_B^U and $K_B^{-1|U}$), the bank DID for the user (DID_B^U) (lines 6 and 7 in Algorithm 2). Then, a corresponding response is prepared and returned to the user (lines 8 and 9 in Algorithm 2).

The *issueVC* function is used to provide a VC to a user. There are two types of VC in our system: general VC and delegated VC (used to facilitate the delegated transactions). That is why this function checks the type of VC request to determine the type of VC that needs to be generated and then returns the VC according to the request (lines 28 to 33 in Algorithm 2).

Algorithm 1 Bank Chaincode (BCC)

```
...
function register(attrResp)
    name = attrResp.name
    national_id = attrResp. national_id
    password = Hash(attrResp.password)
    email = attrResp.email
     ACN = Generate unique account number
    uJson = name,password,email,ACN,national_id
    key = ACN
     BCC.storeData(key, uJson)
    return TRUE
end function
function login(attrResp)
    user = BCC.retrieveData(attrResp.national_id)
    password = Hash(attrResp.password)
    if user.password == password then
        return TRUE
    else
        return FALSE
    end if
end function
function retrieveData(key)
    data = getState(key)
    return data
end function
function storeData(key,data)
    putState(key,data)
    return TRUE
end function
function validateData(key,data)
    storedData = getState(key,data)
    if storedData == data then
        return TRUE
    else
        return FALSE
    end if
end function
function withdrawal(executeTransactionReq)
    key = executeTransactionReq.ACN
    user = getState(key)
    amount = executeTransactionReq.amount
    user.balance = user.balance - amount
     BCC.storeData(key,user)
    trxID = Generate a unique transaction ID
    transactionRecord = {trxID,executeTransactionReq}
     BCC.storeData(trxID, transactionRecord)
    return transactionRecord
end function
...
```

Algorithm 2 Pseudocode for the SSI Agent (SA)

```
...
function handleInvite(didExchangeReq)
    DID_U^B = didExchangeReq.DID_U^B
     Retrieve corresponding DID Doc using DID_U^B
     Retrieve K_U^B from DID Doc
     Store DID_U^B and K_U^B in agent wallet
     Generate K_B^U and K_B^{-1|U}, create DID Doc using K_B^U
     Prepare didExchangeResp using K_B^U
    return didExchangeResp
end function
function createInvitation(createInvitationReq)
     Prepare createInvitationResp with url_B and K_B
    return createInvitationResp
end function
function verifyProof(proofResp)
    VC = proofResp.VC_B^U
    PNR = proofResp.PNR
     Verify the VC against the SSI public ledger
     Verify the signature of VC using K_B^U
     Verify PNR against the Tails Server
    if All verifications are successful then
        return TRUE
    else
        return FALSE
    end if
end function
function issueVC(credReq)
    vcType = credReq.type
     Fetch VC metadata from SSI Ledger
    if vcType == delegated VC then
        return VC_B^{U_D}
    else
        return VC_B^U
    end if
end function
function checkRevocation(checkRevocationReq)
    vcMetaData = checkRevocationReq.vcMetaData
    revocationStatus = Retrive VC revocation status
    if revocationStatus == "revoked" then
        return TRUE
    else
        return FALSE
    end if
end function
function revokeVC(revokeVCReq)
    vcMetaData = revokeVCReq.vcMetaData
    success = Submit revocation request to agent and update accumulator
    if success then
        return TRUE
    else
        return FALSE                    ▷ Revocation failed
    end if
end function
...
```

The *verifyProof* function is used to verify a returned proof. In this function, the returned VC is retrieved from the proof response and verified. There are multiple steps involved in

TABLE 6. Register protocol.

M1	$U \rightarrow B : [N_1, serviceReq(url_s)]_{https}$
M2	$B \rightarrow U : [N_1, serviceResp]_{https}$
M3	$U \rightarrow B : [N_2, registerReq]_{https}$
M4	$B \rightarrow U : [N_3, attrReq]_{https}$
M5	$U \rightarrow B : [N_3, attrResp]_{https}$
M6	$B \rightarrow BCC : attrResp$
M7	$BCC \rightarrow B : success$
M8	$B \rightarrow U : [N_2, registerResp]_{https}$

TABLE 7. Login protocol.

M1	$U \rightarrow B : [N_1, serviceReq(url_s)]_{https}$
M2	$B \rightarrow U : [N_1, serviceResp]_{https}$
M3	$U \rightarrow B : [N_2, loginReq]_{https}$
M4	$B \rightarrow U : [N_3, attrReq]_{https}$
M5	$U \rightarrow B : [N_3, attrResp]_{https}$
M6	$B \rightarrow BCC : attrResp$
M7	$BCC \rightarrow B : success$
M8	$B \rightarrow U : [N_2, loginResp, BankSignature]_{https}$

the verification process (lines 17 and 20 in Algorithm 2). The credential definition of the VC is verified against the SSI ledger (line 18 in Algorithm 2). Then, the signature of the VC is verified using the respective public key of the bank (K_B^U) (line 19 in Algorithm 2) and finally performs a check against the tails server to verify revocation status. If the verification is successful, a *TRUE* response is returned; otherwise, a *FALSE* response is returned (lines 21 to 24 in Algorithm 2).

Revocation is the process of revoking or invalidating a previously issued VC. The revocation process is initiated in certain different scenarios, such as expiration, compromised or incorrect information, change of status, issuer's decision and so on. The *revokeVC* function is used for the revocation of the credentials. For this, the VC metadata are retrieved from the request (line 46 in Algorithm 2). Then, a VC revocation request is submitted using the metadata and a TRUE response is returned (lines 48 and 49 in Algorithm 2).

Finally, the *checkRevocation* function is used to check the revocation status of a particular VC using the VC metadata retrieved from the request (line 36 in Algorithm 2). Then, the revocation status is checked from the SSI agent api (line 36 in Algorithm 2). If the VC was previously revoked, a TRUE response is returned; otherwise, a FALSE response is returned (lines 39 to 42 in Algorithm 2).

V. USE-CASE AND PROTOCOL FLOW

In this section, we present several use-cases and their corresponding protocol flows to illustrate how the developed system can be used. Although we have considered common failure possibilities in the implementation, the successful execution path is the main emphasis of the protocol step description. Here, we consider three different use-cases:

- Registration and log-in use-case, representing the registration and login processes of a user in the bank.
- SSI based ATM transaction use-case, representing the steps involved for a user to interact with the bank for withdrawing cash from an ATM.
- SSI based delegated ATM transaction use-case, representing steps involved for a delegated transaction.

A. REGISTER AND LOGIN USE-CASE

To withdraw cash using an ATM, a user must register with a bank. Once registered, the user logs into the bank to avail its different services. These two steps are presented using a few protocol flows that illustrate the interactions between the involved entities along with the data that are transferred during these interactions. We discuss these protocol flows in the following.

1) REGISTRATION PROTOCOL FLOW

The registration process usually requires a KYC (Know Your Customer) verification process where the user provides some documents (ID card, passport, etc.), which are then verified. Once verified, the user is registered with the bank. The technological evolution of this process has enabled one to carry out the KYC verification process online, known as the *e-KYC* [64]. We have assumed a similar e-KYC verification process in the prototype of our proposal.

The registration protocol is presented in Table 6 and illustrated in Figure 3. The steps of this protocol flow are presented below.

i. The user visits the bank's home page URL (url_S) over a secure HTTPS channel, where the user is presented with, among other options, the register and login options (M1 and M2 steps in Table 6).
ii. The user clicks the Register button (step M3 in Table 6) and is presented with an HTML form (Figure 4) which contains different attribute fields, e.g. *Name, Email, National ID, Password*, denoted as *attrReq* in step M4 in Table 6.
iii. The user fills in the registration form and responds back to the bank (denoted as *attrResp* in M5 in Table 6).
iv. After receiving *attrResp*, it is assumed that the bank verifies the submitted data against a KYC database of the respective country. Then, it transfers *attrResp* to the *register* function of Algorithm 1 in BCC (Step M6 in Table 6).
v. The BCC stores the data on the blockchain following the Hyperledger Fabric protocol, omitted here for brevity. If the data is stored successfully, a *TRUE* response is returned from the BCC (line 11 in Algorithm 1 and Step M7 in Table 6). A copy of the confirmation is also sent to the user's email. Finally, the bank returns a successful response to the user and is shown back to the user (Step M8 in Table 6 and Figure 5).

2) LOGIN PROTOCOL FLOW

The login step within a bank allows a user to avail different bank-related services online. In our implementation, this step is required to avail different SSI related services, as explained

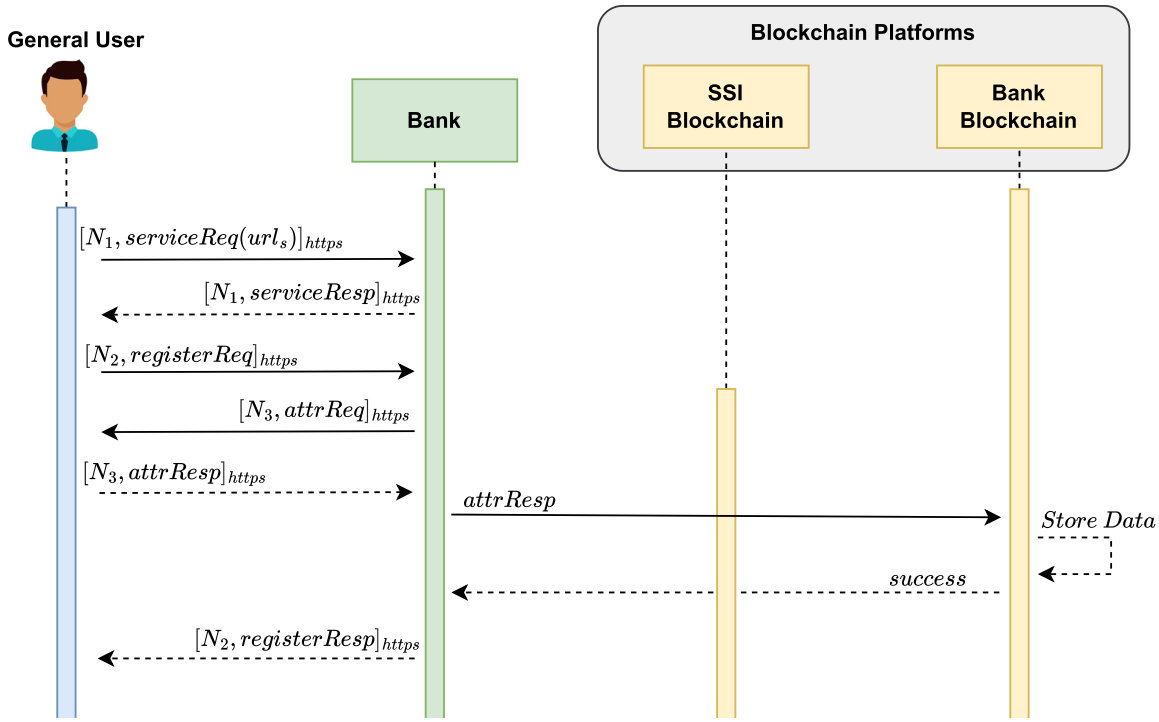


FIGURE 3. Registration protocol flow.

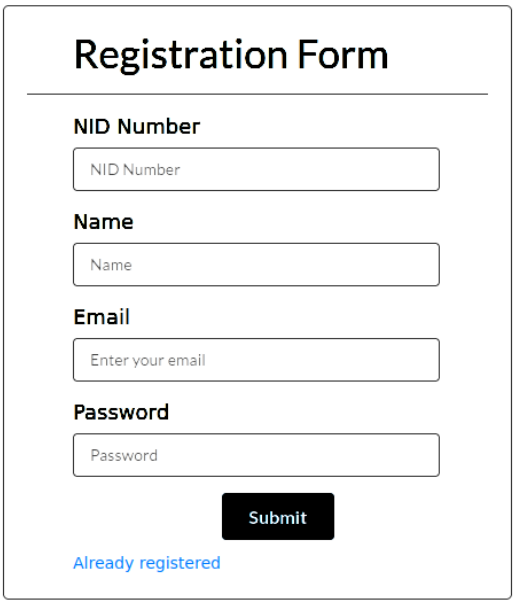


FIGURE 4. Registration page.

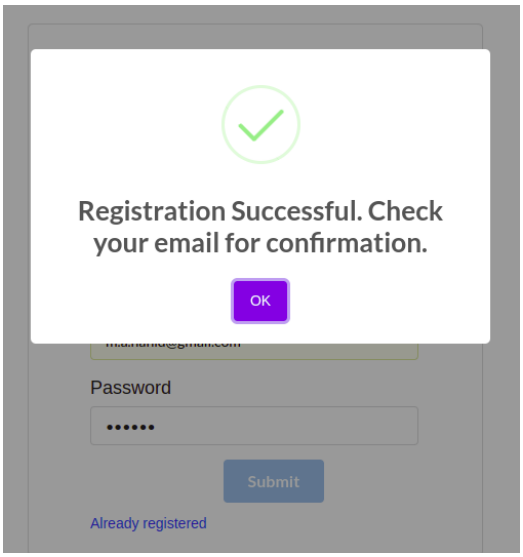


FIGURE 5. Successful registration.

in the subsequent sections. Table 7 shows the corresponding login protocol which is illustrated in Figure 6. We briefly present this protocol in the following.

i. From the home page of the bank, the user clicks the Login button to create a *loginReq* and then to initiate the login process (M1-M3 in Table 7).
ii. The bank returns a web page with an HTML form (Figure 7) listing required attributes, e.g. National ID, and password, denoted as *attrReq* in Step M4, Table 7.
iii. The user fills in the requested information and clicks the Submit button, thereby submitting an *attrResp* (Step M5 in Table 7).
iv. After receiving *attrResp*, the bank transfers it to the *login* function of Algorithm 1 in BCC (Step M6 in

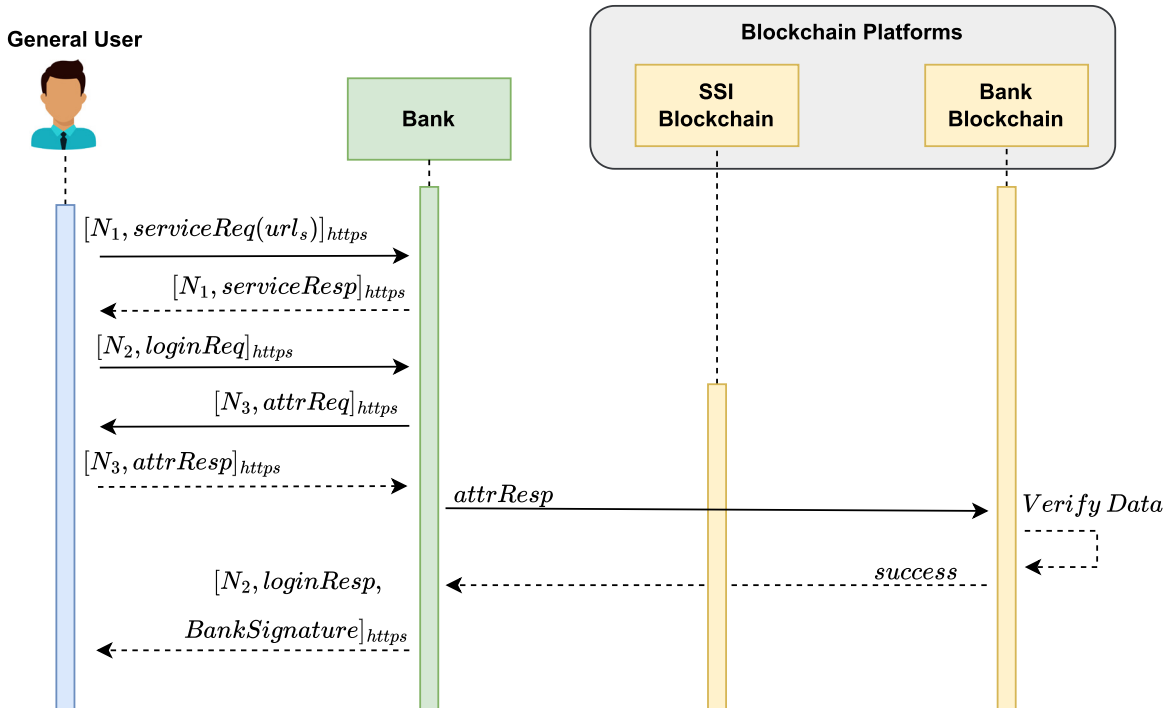


FIGURE 6. Login protocol flow.

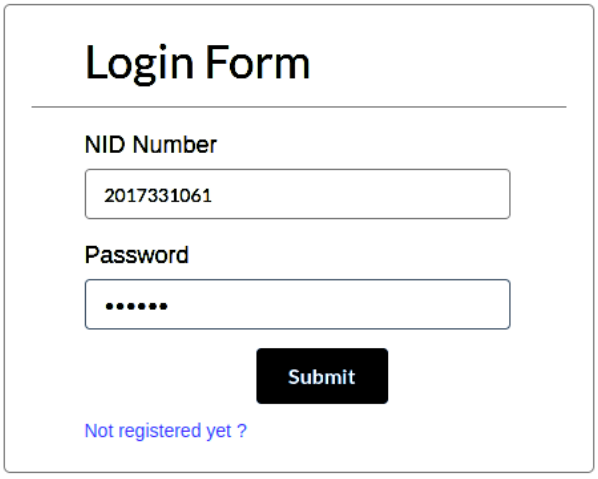


FIGURE 7. Login page.

TABLE 8. ATM register protocol.

M1	$A \rightarrow B : [N_1, registerReq]_{https}$
M2	$B \rightarrow BCC : Store\ ATM\ public\ key$
M3	$BCC \rightarrow B : success$
M4	$B \rightarrow A : [N_1, registerResp]_{https}$
M5	$A \rightarrow B : [\{N_2, symKeyReq\}_{k_b}]_{https}$
M6	$B \rightarrow BCC : symKeyReq$
M7	$BCC \rightarrow B : success$
M8	$B \rightarrow A : [\{N_2, symKeyResp\}_{k_a}]_{https}$
M9	$B \rightarrow A : [\{T, symKeyUpdate\}_{k_a}]_{(periodic)}$

Table 6). This function checks the submitted credentials against the previously stored credentials, and if successful, a TRUE response is returned from the chaincode to the bank's backend (Step M7 in Table 7). Otherwise, a FALSE response is returned.

v. Upon successful login, the bank returns a *loginResp* generated by concatenating the nonce and the content of the service cookie and then signing the hash of the concatenation with the private key of the bank (denoted as *BankSignature* in M8 Table 7)). The digital signature is verified by the user client, thereby ensuring the authenticity of the service cookie. The service cookie is then stored on the client side. This cookie is used in the subsequent interactions to access restricted functionalities provided by the bank website.

It is to be noted that, in this implementation, we have adopted a simple approach for the registration and login. These processes can be easily customised to meet complex business case requirements in a real-life setting.

3) ATM REGISTRATION PROTOCOL FLOW

An ATM also needs to be registered before using the SSI channel for transactions and other monetary activities. The ATM registration protocol is presented in Table 8 and illustrated in Figure 8. A brief overview of the protocol is provided below.

i. When a new ATM is introduced to the network, it must be registered in the bank database. The ATM initiates a communication with the bank by sending an API request containing a nonce and *registerReq* data over an HTTPS channel (Step M1 in Table 8). The *registerReq* contains the public key of the ATM.

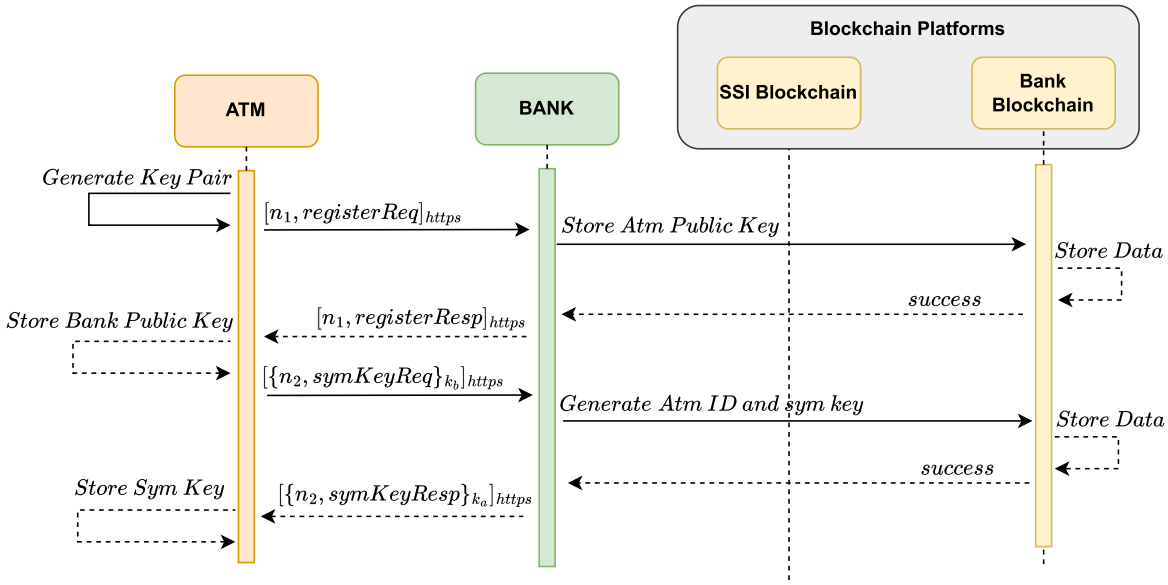


FIGURE 8. ATM registration protocol flow.

TABLE 9. Connection establishment protocol.

M1	$U \rightarrow B : [N_1, serviceReq(url_s, serviceCookie)]_{https}$
M2	$B \rightarrow U : [N_1, serviceResp]_{https}$
M3	$U \rightarrow B : [N_2, connectionReq]_{https}$
M4	$B \rightarrow U : [N_3, attrReq]_{https}$
M5	$U \rightarrow B : [N_3, attrResp]_{https}$
M6	$B \rightarrow BCC : attrResp$
M7	$BCC \rightarrow B : success$
M8	$B \rightarrow U : [N_2, connectionResp, BankSignature]_{https}$
M9	$U \rightarrow M : [B, \{N_4, didExchangeReq\}_{K_B^U}]_{K_M^U}$
M10	$M \rightarrow B : [\{N_4, didExchangeReq\}_{K_B^U}]_{K_B^M}$
M11	$B \rightarrow M : [U, \{N_4, didExchangeResp\}_{K_U^B}]_{K_M^B}$
M12	$M \rightarrow U : [\{N_4, didExchangeResp\}_{K_U^B}]_{K_U^M}$
M13	$B \rightarrow BCC : updateUserReq$
M14	$BCC \rightarrow B : updateUserResp$

ii. The bank forwards the request to the BCC with an instruction to store the ATM's public key (Step M2 in Table 8). If the public key of the ATM is successfully stored, the BCC acknowledges this by sending a *success* message back to the bank (Step M3 in Table 8).
iii. The bank responds to the ATM with *registerResp* and the previous nonce over HTTPS (Step M4 in Table 8). The *registerResp* contains the public key of the bank and a unique identifier for the ATM.
iv. Following the exchange of public keys, the ATM transmits an encrypted request (*symKeyReq*) to the bank, including the ATM number, to initiate the process of exchanging symmetric keys (Step M5 in Table 8). This encryption is carried out using public keys.
v. The bank forwards the *symKeyReq* message to the BCC with a new nonce, indicating the request for a symmetric key. The BCC generates a symmetric key for that specific ATM and stores it on the private blockchain. If the symmetric key request is successfully processed, the BCC acknowledges this by sending a *success* message back to the bank (Steps M5-M7 in Table 8).
vi. The bank transmits a response with *symKeyResp* containing the symmetric key and the previous nonce to the ATM (Step M8 in Table 8). This message is encrypted using the previously established public key. The ATM securely stores this symmetric key, which will be utilised for future secure message exchanges between the ATM and the bank.
vii. The bank generates *symKeyUpdate* periodically (per week) and sends it to the atm to update the shared symmetric key between the two entities.

B. SSI BASED ATM TRANSACTION USE-CASE

In this section, we explore the core functionalities of the proposed system to illustrate how the SSI functionalities have been integrated for withdrawing cash from the ATM. There are three steps in this use-case which are discussed one by one in the following.

1) ESTABLISHING AN SSI CONNECTION

To use the SSI enabled features, the user must establish an SSI connection with the bank at the beginning. The protocol flow for this step is presented in Table 9 and illustrated in Figure 9. A brief overview of these steps is provided below. To continue with the next steps, it is assumed that the user is already logged into the bank and has an authentic service cookie. We also assume that the bank has provided a mechanism for the user to install the Bifold Wallet and the user has installed the wallet.

i. Once the user logs into the bank, a web page (Figure 10) listing available services (e.g *Establish Connection, Get*

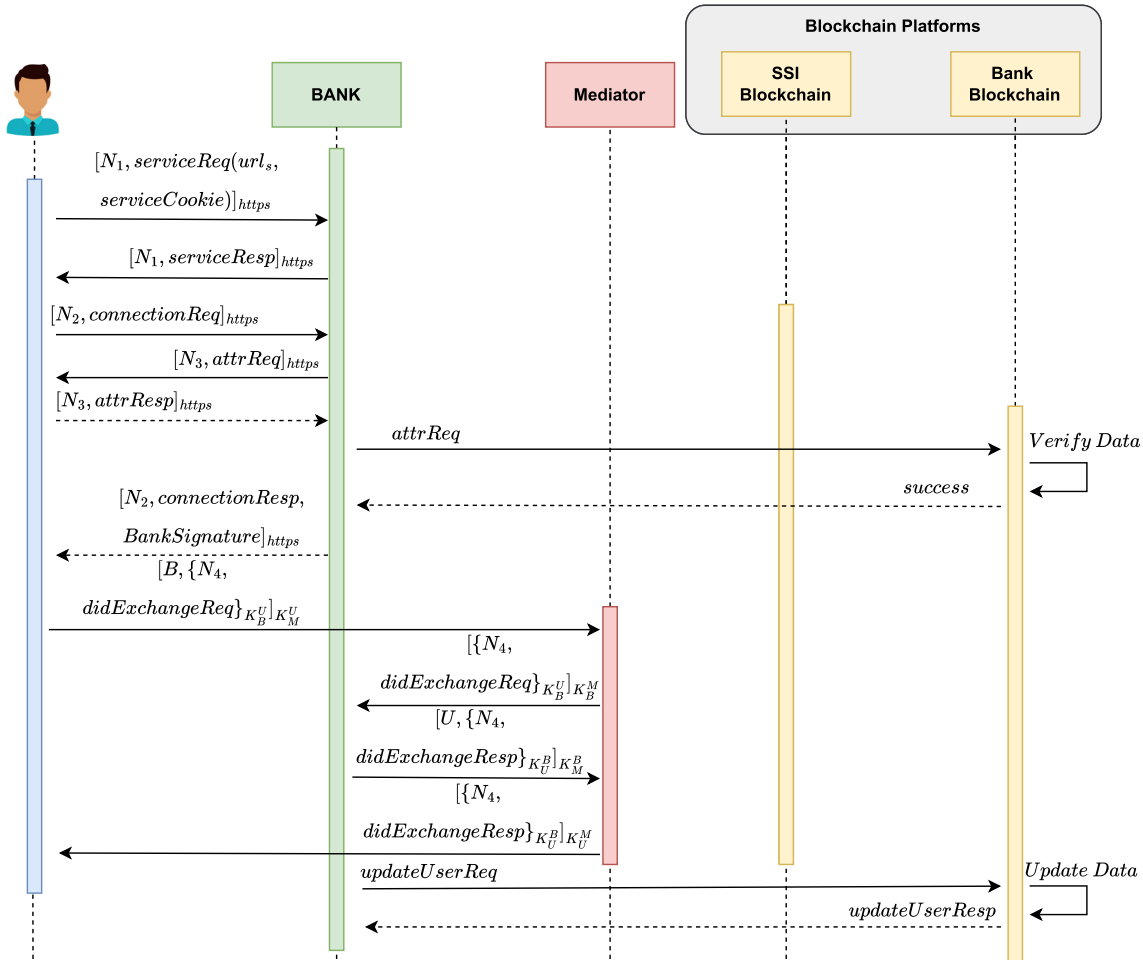


FIGURE 9. Connection establishment protocol flow.

Credential, Create Delegation and so on) is shown to the user (Step M2 in Table 9).

ii. The user chooses the *"Establish Connection"* option, creating a *connectionReq* request (Step M3 in Table 9).

iii. The bank website displays a page asking the user to submit account number and national ID number, modelled as *attrReq* in Step M4 in Table 9.

iv. The user enters the requested information, which is modeled as *attrResp* and submits the information to the bank, step M5 in Table 9. The bank forwards *attrResp* to BCC server, step M6 in Table 9. The BCC server verifies the submitted information against the data stored in the Fabric ledger using the *retrieveData* and *validateData* functions (Algorithm 1). Next, the BCC server returns a TRUE response if the verification is successful, step M7 in Table 9.

v. Once the user's information is successfully verified, the bank invokes the *createInvitation* function from Algorithm 2 to generate a new SSI invitation. This process results in a url_B that encodes the invitation details using the did:peer method, which does not require a blockchain [67]. A QR code is created from this url and displayed to the user as *connectionResp*. A digital signature is also attached to ensure the authenticity of the invitation (Step M8 in Table 9).

vi. The user verifies the bank's signature and scans the QR code using their SSI wallet. The wallet then generates a new key pair (K_U^B *and* $K_U^{-1|B}$) and creates a corresponding did:peer identifier (DID_U^B) to represent the user to the bank. It also constructs a DID Document and a *didExchangeReq* message that includes this new DID and a nonce N_i. This request is sent to the bank via mediator using the out-of-band protocol [68] (Steps M9–M10 in Table 9).

vii. Upon receiving the request, the bank agent invokes the *handleInvite* function from Algorithm 2. It processes the exchange request, stores the connection details in its wallet, and responds with a *didExchangeResp* message. This message includes a new did:peer identifier DID_B^U for the user and is sent back to the user's wallet via the mediator (Steps M11–M12 in Table 9).

viii. Now, the user wallet retrieves the respective data from *didExchangeResp* and by combining the data from

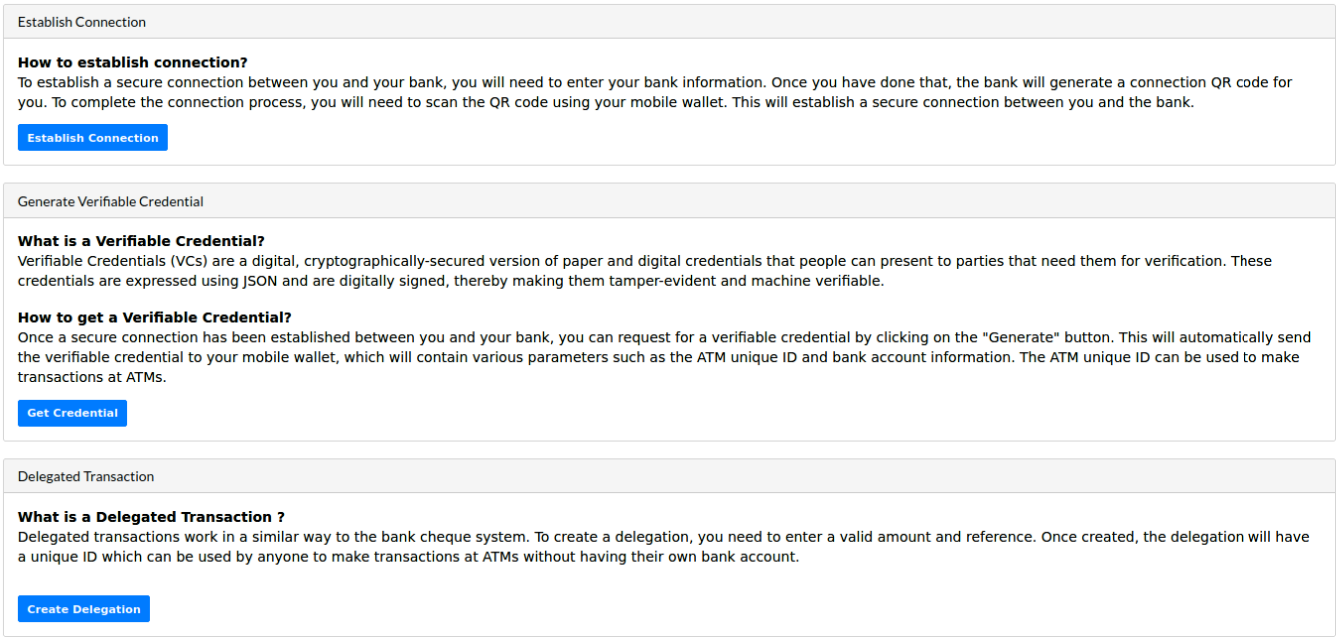


FIGURE 10. Bank's page showing SSI options.

FIGURE 11. Connection QR.

TABLE 10. Acquire VC protocol.

M1	$U \rightarrow B : [N_1, credReq]_{https}$
M2	$B \rightarrow BCC : fetchUserReq$
M3	$BCC \rightarrow B : fetchUserResp$
M4	$B \rightarrow SA : credReq$
M5	$SA \rightarrow B : credResp$
M6	$B \rightarrow BCC : updateUserReq$
M7	$BCC \rightarrow B : updateUserResp$
M8	$B \rightarrow M : [U, \{N_2, credResp\}_{K_U^B}]_{K_M^B}$
M9	$M \rightarrow U : [\{N_2, credResp\}_{K_U^B}]_{K_U^M}$
M10	$B \rightarrow U : [N_1, success]_{https}$

DID_U^B establishes a secure connection between the user and the bank. At this point, the user has established a secure DIDComm channel with the bank and the 'Connection Status', as shown in Figure 11, is changed to 'COMPLETED'.

ix. The bank updates the user's data using *updateUserReq* which holds *connectionStatus*, a boolean value with a *TRUE* value in this case, indicating a successful connection establishment. The bank transfers this request to *BCC* which updates the connection status for that particular user in the Fabric blockchain using the *storeData* function of Algorithm 1. Then, an *updateUserResp* response is returned to the bank's backend (Steps M13-14 in Table 9). This step will prevent a particular user from establishing multiple SSI connections with the bank.

2) ACQUIRING A VC

After successfully establishing an SSI connection, a user can request for a VC. The bank releases the VC to the user via the wallet and the user has to accept the request to store the VC, which is then stored in the wallet to be used for the subsequent step. Table 10 shows the corresponding protocol flow and Figure 12 illustrates the protocols. We sketch out the activities involved in this step in the following. To continue with the next step, it is assumed that the user is already logged in to the bank and has a service cookie and an invite cookie.

1) The user submits a VC request, denoted as *credReq*, along with the service cookie by choosing the "Get Credential" option (Figure 10) from the bank service page (step M1 in Table 10).
2) The bank creates $fetchUserReq$ using the service cookie, *national_id* and G_{uid} and invokes the *retrieveData* function (Algorithm 2) of BCC (Step M2 in Table 10). G_{uid} means a general user identifier for the ATM and is used to initiate general transactions at the ATM.

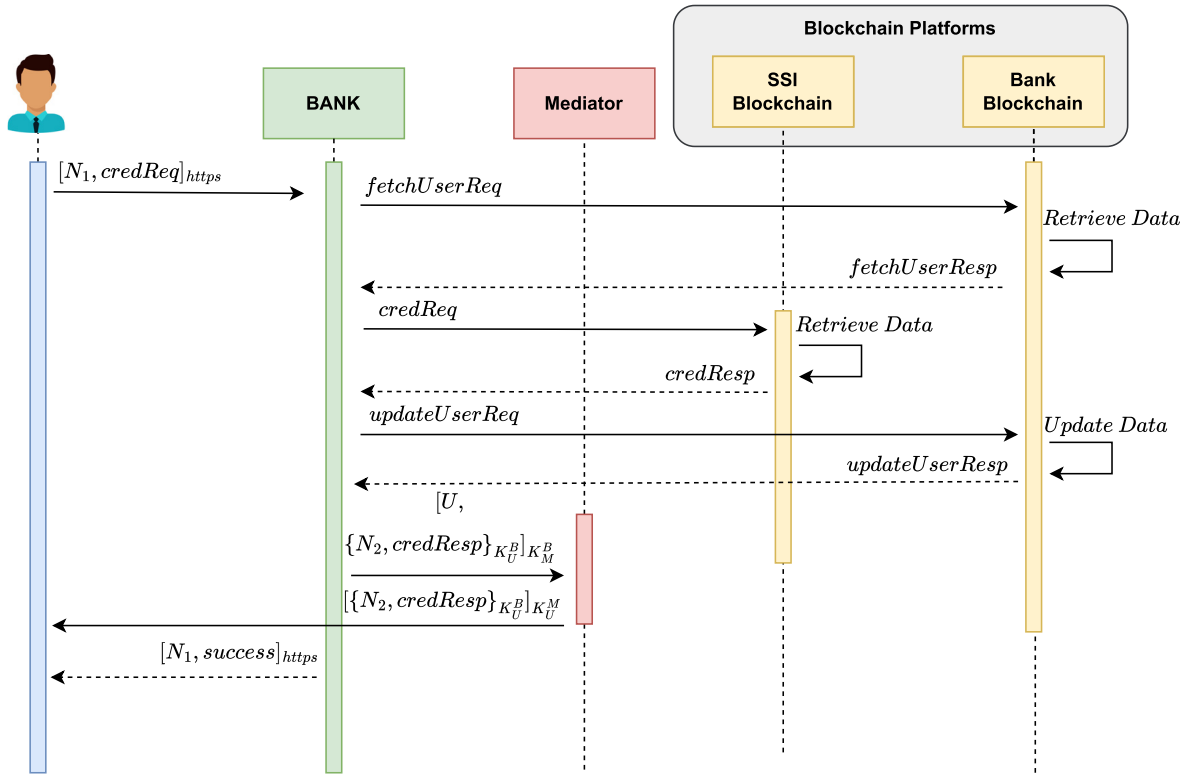


FIGURE 12. Acquire VC protocol flow.

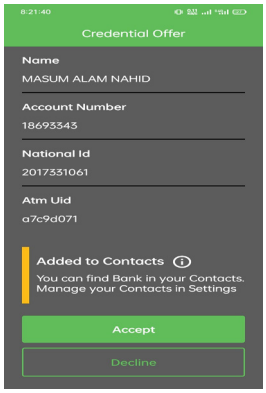


FIGURE 13. Credential offer at user's wallet.

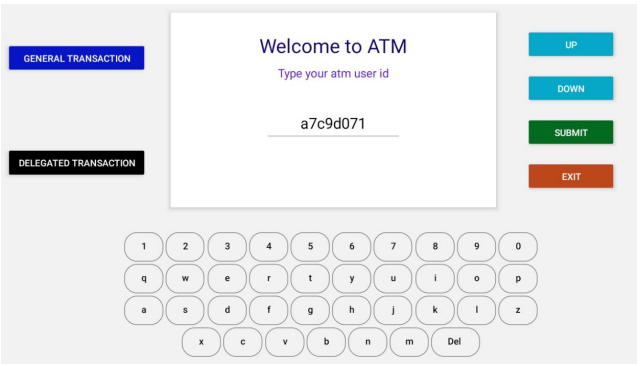


FIGURE 14. ATM main screen.

3) The *retrieveData* function extracts the user information from the Fabric blockchain and sends it back to the bank's backend as *fetchUserResp* which contains user information represented as *userData* (Step M3 in Table 10). This step is crucial as data retrieved from this step are used by the bank to check if there is an active credential for the user (see below).
4) The bank forwards the *credReq* request to the SSI Agent (*SA*) to prepare a VC and the SA prepares a VC (VC_B^U) by invoking the *issueVC* function (Algorithm 2) and returns it to the bank as *credResp*, steps M4-M5 in Table 10. The (VC_B^U) contains *account_number*, *national_id*, *name* and G_{uid} of the user. The G_{uid} is generated using the SSI connection ID (*conID*) and used in the subsequent step.
5) The bank updates the user information on the Fabric blockchain to restrict the user from requesting credentials more than once. The bank makes a *updateUserReq* with *credStatus* (a boolean *TRUE* value in this case, if credential has been successfully issued) and credential metadata, which are stored in the Fabric blockchain and a response is returned to the bank, steps M6-M7 in Table 10.
6) Finally, the bank returns the VC using *credResp* to the user's wallet via *M*, steps M8-M9 in Table 10. Please

note that *credResp* is encrypted with the public key of the user to be used with the bank (K_U^B) and hence, the mediator has no knowledge of the contents.

7) After receiving the VC, the wallet validates the signature of the VC using the public key of the bank for this particular connection (K_B^U). If it is successful, the user's wallet shows the VC to the user (Figure 13) and the user must accept the VC to store it in the wallet to be used in the future for monetary activities at the ATM. During this VC issuance procedure, the bank and the user agent (wallet) use the DIDComm protocol.
8) If the validation is successful, the bank website also displays a success message to the user.

3) PERFORM MONETARY ACTIVITIES AT ATMs

In the final step of this use-case, the user uses an ATM to simulate the cash withdrawal functionality. As mentioned earlier, we have created an Android App which acts as a virtual ATM simulating the functionalities of an ATM. Figure 14 depicts the main screen of the virtual ATM. In the main screen there are 6 buttons, 1 display and a keypad. These buttons are used to select transactions types, choose between options, go forward or backward and submit requests at the ATM. The display at the middle shows the current state of a transaction and other results. Keypad is used to enter alphanumeric inputs like an amount and identifier and so on. To initiate a transaction, the user must first select the transaction type. There are two types of transactions ''General Transaction'' and ''Delegated Transaction''. Then according to the transaction user has to enter the respective value (discussed below).

Table 11 shows the protocol to perform monetary activities at ATMs and Figure 15 depicts the protocol flow. A brief overview of this protocol is discussed below. Like before, it is assumed that the the user has established an SSI connection with the bank and acquired a VC before engaging in the following activites.

i. When a user accesses the ATM, the ATM home screen is shown to the user (Figure 14). From the two transaction types available, the user selects ''General Transaction'' (the blue option in the left of Figure 14). The user then inputs an uid (G_{uid}) for this particular transaction which can be retrieved from the received VC (Figure 13). These activities represent steps M1 and M2 in Figure 11.
ii. The ATM sends an *atmInitiateData*, consisting of *atmInitiateReq* and *atmData*, to the bank (Steps M3 in Table 11). The *atmInitiateReq* contains the transaction type and G_{uid} whereas *atmData* contains a digital signature and the ATM serial number (A_n). The signature is generated by signing the hash of the combination of A_n and a nonce with the private key of ATM. This proves the authenticity of the data sent by the ATM.
iii. The bank retrieves user information from the private blockchain as discussed before, steps M4-M5 in Table 11. The bank retrieves G_{uid} from the user information returned by *BCC* and matches with the one submitted via the *atmInitiateReq* data field and checks the validity of the user. The validity of the user depends on *credStatus* and *connectionStatus*, indicating that the user must complete both (''Establish Connection'' and ''Get Credential'') before trying to do monetary activities at the ATM.
iv. Now, the bank agent prepares $proofReq_1$ to verify the user and sends the proof request to the user's wallet via M, steps M6-M7 in Table 11. The ATM prompts the user to release the proof request within 5 minutes (Figure 16a).
v. The user's wallet parses the requested attributes and matches with the stored bank credentials (VC_B^U) in the wallet and asks for the confirmation to release the proof (Figure 16b). The user will only be able to submit the verifiable presentation if a valid credential exists in the wallet. This process ensures that the users have the autonomy to select the bank with which they wish to engage for transactional purposes.
vi. Once confirmed, the wallet returns the $proofResp_1$ to the bank agent via Mediator *M*, steps M8-M9 in Table 11.
vii. When the bank receives $proofResp_1$, it is verified by the *SA* utilising the *verifyProof* function of Algorithm 2. Afterwards, *atmInitiateResp* along with a signature (*BankSignature*) is returned by the bank. *BankSignature* is generated using the hash of A_n and nonce, which was sent to bank as *atmData* within *atmInitiateReq* and signed with bank's private key, steps M10-M11 in 11. The *BankSignature* proves the authenticity of the response sent by the bank. Upon validating the signature, the ATM displays the monetary options (Figure 17a).
viii. Now, the user can choose their preferred monetary activity. After taking the respective input from the user, the ATM prepares *atmUserInputData* and sends *executeTransactionReq* to the bank in steps M11-M13 in Table 11. *atmUserInputData* consists of the user-chosen

TABLE 11. General transaction protocol.

M1	$A \rightarrow U : ATM\ Main\ Screen$
M2	$U \rightarrow A : atmInitiateReq$
M3	$A \rightarrow B : [\{\{N_1, atmInitiateData\}_{iso}\}_{k_{sh}, A_n}]_{https}$
M4	$B \rightarrow BCC : fetchUserReq$
M5	$BCC \rightarrow B : fetchUserResp$
M6	$B \rightarrow M : [U, \{N_2, proofReq_1\}_{K_U^B}]_{K_M^B}$
M7	$M \rightarrow U : [\{N_2, proofReq_1\}_{K_U^B}]_{K_U^M}$
M8	$U \rightarrow M : [A, \{N_2, proofResp_1\}_{K_B^U}]_{K_M^U}$
M9	$M \rightarrow B : [\{N_2, proofResp_1\}_{K_B^U}]_{K_B^M}$
M10	$B \rightarrow A : [\{\{N_1, atmInitiateResp, BankSignature\}_{iso}\}_{k_{sh}}]_{https}$
M11	$A \rightarrow U : ATM\ Monetary\ Options$
M12	$U \rightarrow A : atmUserInputData$
M13	$A \rightarrow B : [\{\{N_3, executeTransactionReq\}_{iso}\}_{k_{sh}, A_n}]_{https}$
M14	$B \rightarrow BCC : executeTransactionReq$
M15	$BCC \rightarrow B : executeTransactionResp$
M16	$B \rightarrow A : [\{\{N_3, executeTransactionResp, BankSignature\}_{iso}\}_{k_{sh}}]_{https}$
M17	$A \rightarrow U : success$

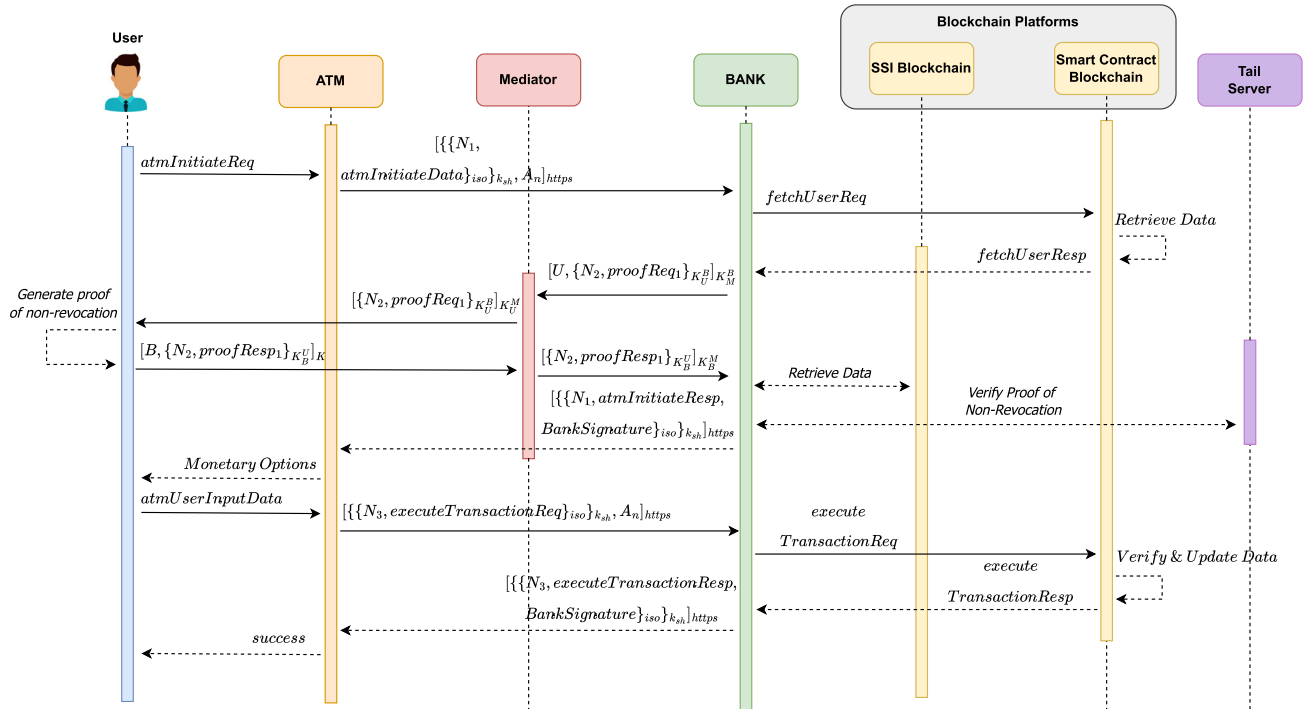
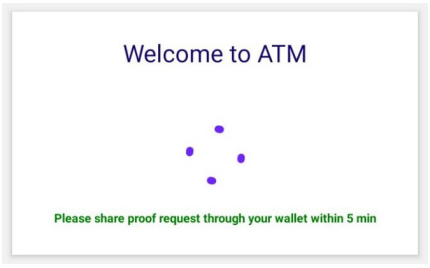
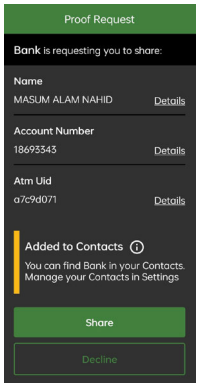


FIGURE 15. General transaction.



(a) ATM prompt for proof request



(b) Proof Request at wallet

FIGURE 16. ATM authentication.

TABLE 12. Create delegation protocol.

M1	$U \rightarrow B : [N_1, createDelegationReq]_{https}$
M2	$B \rightarrow U : [N_2, attrReq]_{https}$
M3	$U \rightarrow B : [N_2, attrResp]_{https}$
M4	$B \rightarrow BCC : attrResp$
M5	$BCC \rightarrow B : success$
M6	$B \rightarrow U : [N_1, createDelegationResp, BankSignature]_{https}$

option (Balance Check or Transaction) and an amount, an optional data required for a withdrawal transaction. If the option is *"Transaction"*, then the user needs to input the amount (Figure 17c).

ix. The bank validates the information received from the ATM using the *validateData* function of Algorithm 1 in steps M14-M15 in Table 11.

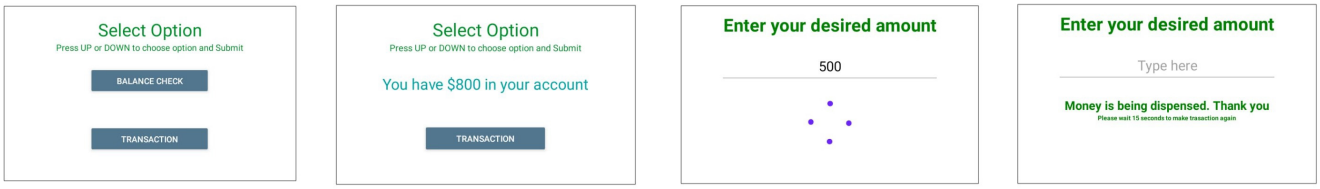
x. Next, the bank prepares *executeTransactionResp* and sends it to the ATM along with a digital signature (Step M16 in Table (11)).

xi. The ATM validates the signature and executes the action chosen by the user. For a "Balance Check" operation, it shows the user balance (Figure 17b) and for a "Transaction" option, the ATM would dispense the requested amount of cash (Figure 17d).

xii. While the protocol assumes a stable network and synchronous operation across entities, real world ATM environments are subject to unexpected disruptions. For example, disconnection between M14 and M17 could result in transaction confirmation being lost or delayed despite execution at the backend. Although not explicitly shown in the diagram, our system includes basic safeguards to manage such failures such as retry logic, validation checks, and the use of digital signatures to detect inconsistencies.

C. SSI BASED DELEGATED ATM TRANSACTION USE-CASE

In this use-case, we explore how a user can authorise another user to withdraw cash from an SSI ATM using the delegation feature. Practically, a user will delegate a VC to another user who can then withdraw cash from the ATM. The delegated user will not have the capability to inquire about the actual balance of the bank account. Instead, they will only be able



(a) ATM Options Page (b) Balance (c) Verifying Amount (d) Dispensing Cash

FIGURE 17. General transaction at ATM.

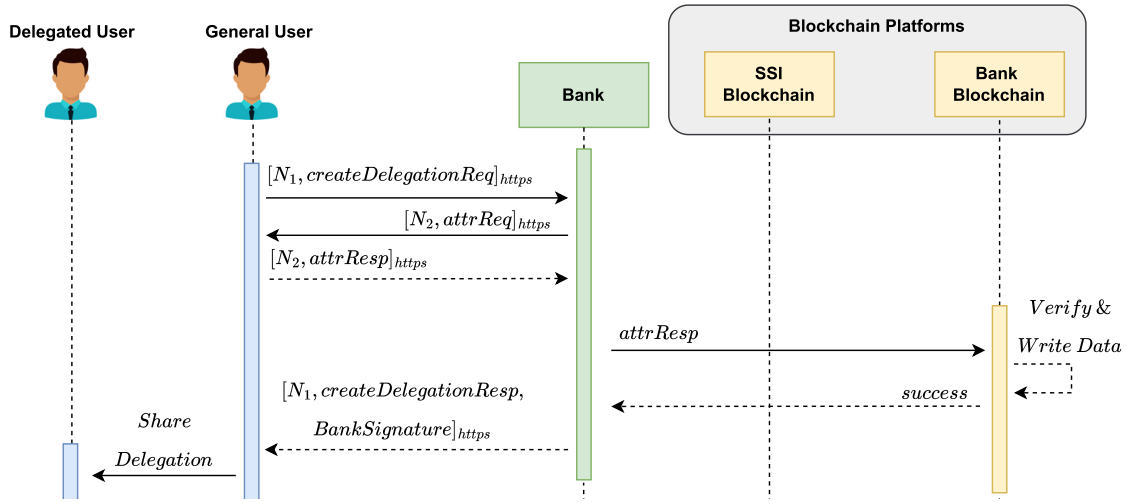
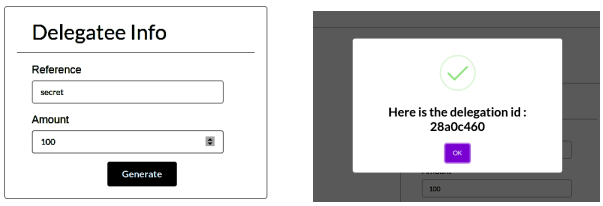


FIGURE 18. Create delegation.



(a) Creating a delegation (b) Success message

FIGURE 19. Creating delegated credential.

to withdraw the amount that the account owner authorises them to withdraw. This is similar to the concept of ATM withdrawal by code feature which is available in many banks such as PNC Bank [69], Islami Bank [70], City Bank [71] in different countries. It provides access to cash at ATMs by allowing users to authorise others, such as family members or friends, to withdraw cash on their behalf using a unique code and it can be useful in emergency situations where immediate access to cash is required.

The delegated transaction use-case has three steps which are discussed one by one in the following.

1) CREATE DELEGATION

The first step in facilitating delegated transactions is to create a delegation. Only a registered user can start this process by logging into the bank's web page. Table 12 shows the protocol to create a delegation, while Figure 18 depicts the respective protocol flow. A brief overview of this protocol is provided below.

i. To create a delegation, a user must be logged in and have the bank's VC (VC_B^U). The user selects the *"Create Delegation"* option (Figure 10), thereby creating a *createDelegationReq* which is sent to the bank (Step M1 in Table 12).
ii. Now, the bank website displays a page showing the required attributes (Figure 19a) to create a delegation since the user needs to submit a reference and a valid amount. This reference will be used in the following steps. This information is used to create *attrResp* which is sent to the bank (Steps M2-M3 in Table 12).
iii. The bank transfers the *attrResp* to *BCC* which validates these data against the data stored in the private blockchain by invoking *validateData* function as discussed previously. If validation is successful, the information regarding the delegation request is stored using the *storeData* (Steps M4-M5 in Table 12).
iv. The bank returns *createDelegationResp* to the user. This includes D_{uid} (a delegation user identifier) and *BankSignature*, step M6 in Table 12. Here, the hash is generated using *serviceCookie* and the nonce and then the hash is signed using the private key of the bank. The

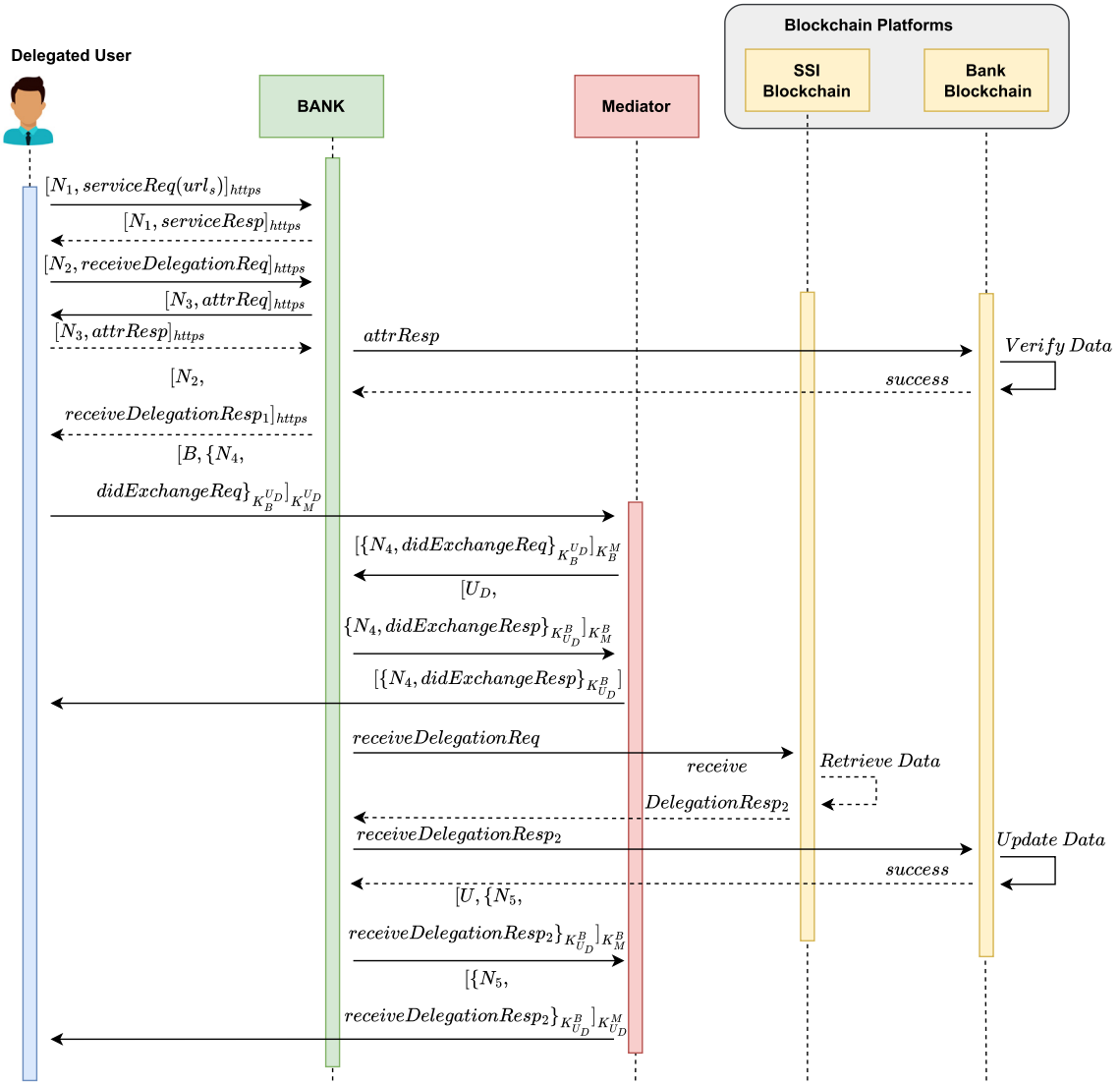


FIGURE 20. Receive delegation.

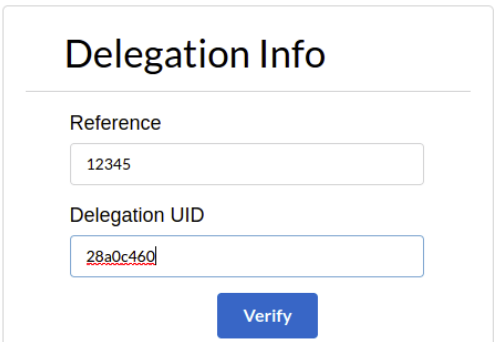


FIGURE 21. New delegation.

user is shown a successful delegation creation message and D_{uid} (Figure 19b), if there is no error.

v. The user supplies the D_{uid} and the reference to the recipient to receive the delegated credential from the bank.

2) RECEIVE DELEGATION

In this step, the delegatee (denoted as U_D) receives the delegation from the delegator. It is to be noted that the delgatee is not required to be a registered user of the bank for receiving this delegation. We also assume that U_D has received D_{uid} and the reference from the sender (delegator) before engaging in the following protocol and has the respective SSI wallet. Table 13 presents the protocol for receiving this delegation and Figure 20 depicts the protocol flow. A brief overview of this protocol is presented below.

i. First, U_D needs to establish a secure connection with the bank to get the delegation. U_D initiates a connection with the bank by submitting a request to the delegation service of the bank access (denoted as $urls_S$) and select the "Receive Delegation" option to create a delegation request (denoted as *receiveDelegationReq*), steps M1-M3 in Table 13.

TABLE 13. Receive delegation protocol.

M1	$U_D \rightarrow B : [N_1, serviceReq(urls)]_{https}$
M2	$B \rightarrow U_D : [N_1, serviceResp]_{https}$
M3	$U_D \rightarrow B : [N_2, receiveDelegationReq]_{https}$
M4	$B \rightarrow U_D : [N_3, attrReq]_{https}$
M5	$U_D \rightarrow B : [N_3, attrResp]_{https}$
M6	$B \rightarrow BCC : attrResp$
M7	$BCC \rightarrow B : success$
M8	$B \rightarrow U_D : [N_2, receiveDelegationResp_1]_{https}$
M9	$U_D \rightarrow M : [B, \{N_4, didExchangeReq\}_{K_B^{U_D}}]_{K_M^{U_D}}$
M10	$M \rightarrow B : [\{N_4, didExchangeReq\}_{K_B^{U_D}}]_{K_B^M}$
M11	$B \rightarrow M : [U_D, \{N_4, didExchangeResp\}_{K_{U_D}^B}]_{K_M^B}$
M12	$M \rightarrow U_D : [\{N_4, didExchangeResp\}_{K_{U_D}^B}]_{K_{U_D}^M}$
M13	$B \rightarrow SA : receiveDelegationReq$
M14	$SA \rightarrow B : receiveDelegationResp_2$
M15	$B \rightarrow BCC : receiveDelegationResp_2$
M16	$BCC \rightarrow B : success$
M17	$B \rightarrow M : [U_D, \{N_5, receiveDelegationResp_2\}_{K_{U_D}^B}]_{K_M^B}$
M18	$M \rightarrow U_D : [\{N_5, receiveDelegationResp_2\}_{K_{U_D}^B}]_{K_{U_D}^M}$

TABLE 14. Delegated transaction protocol.

M1	$A \rightarrow U_D : ATM\ Main\ Screen$
M2	$U_D \rightarrow A : atmInitiateReq$
M3	$A \rightarrow B : [\{\{N_1, atmInitiateData\}_{iso}\}_{k_{sh}}, A_n]_{https}$
M4	$B \rightarrow BCC : fetchDelegationReq$
M5	$BCC \rightarrow B : fetchDelegationResp$
M6	$B \rightarrow M : [U_D, \{N_2, proofReq_2\}_{K_U^B}]_{K_M^B}$
M7	$M \rightarrow U_D : [\{N_2, proofReq_2\}_{K_U^B}]_{K_{U_D}^M}$
M8	$U_D \rightarrow M : [A, \{N_2, proofResp_2\}_{K_B^U}]_{K_M^{U_D}}$
M9	$M \rightarrow B : [\{N_2, proofResp_2\}_{K_B^U}]_{K_B^M}$
M10	$B \rightarrow SA : revokeVCReq$
M11	$SA \rightarrow B : revokeVCResp$
M12	$B \rightarrow A : [\{\{N_1, atmInitiateResp, BankSignature\}_{iso}\}_{k_{sh}}]_{https}$
M13	$A \rightarrow U_D : success$

ii. Now, the bank website displays a page (Figure 21) showing the attributes required (denoted as *attrReq* in the protocol) to receive a delegation. U_D inputs D_{uid} and the reference received from the delegator which are combined to create *attrResp*. Then it was sent to the bank. These steps represent steps M4-M5 in Table 13.

iii. The bank invokes the *validateData* function of Algorithm 1 to validate the data retrieved from *attrResp* against the data stored in the private blockchain. If validation is successful, the BCC provides a *success* message as a response to the bank, steps M6-M7 in Table 13.

iv. Next, the bank creates a new invitation request and sends it to the SSI Agent (denoted as *SA*). The *SA* generates a new SSI invitation by invoking the *createInvitation* function of Algorithm 2. The SSI agent returns the invitation response to the bank which contains an SSI connection url as url_B. The bank generates a QR code using url_B and shows it to U_D as $receiveDelegationResp_1$ (Step M8 in Table 13).

v. U_D scans the QR code using the SSI wallet. The wallet then generates a new key pair ($K_{U_D}^B$ *and* $K_{U_D}^{-1|B}$) and creates a new DID (did:peer) for bank $DID_{U_D}^B$. Then, it creates a DID Doc using the generated key. U_D generates a *didExchangeReq* using $DID_{U_D}^B$ and N_i and sends it to the bank agent via mediator M (Steps M9-M10 in Table 13).

vi. The bank agent stores the information into their wallet and returns a *didExchangeResp* using a newly generated $DID_B^{U_D}$ with N_i via mediator M, in steps M11-M12 in Table 13. Now, the user wallet retrieves data from *didExchangeResp* and by combining the data from $DID_{U_D}^B$, establishes a secure connection between U_D and the bank.

vii. Next, the bank sends the same *receiveDelegationReq* to *SA* to prepare a VC ($VC_B^{U_D}$). The *SA* prepares a VC by invoking the *issueVC* function of Algorithm 2 and returns a $receiveDelegationResp_2$ to the bank which contain $VC_B^{U_D}$ (steps M13-M14 in Table 13).

viii. The bank stores the credential metadata into their private blockchain and also updates *credStatus* of this delegation to prevent any further credential claims in future, steps M15-M16 in Table 13.

ix. The bank sends the $receiveDelegationResp_2$ to U_D's wallet via M (Steps M17-M18 in Table 13). After receiving the VC, the wallet validates the signature of the VC using the public key of the bank for this particular connection. If it is successful, the U_D's wallet stores the VC, which will be used in the future for a one-time delegated transaction at an ATM.

x. The bank website displays a success message to the user if no error occurs, thereby completing this step.

3) DELEGATED TRANSACTION

This is the final step of this use-case in which the delgatee accesses an SSI ATM to perform a delegated transaction. Table 14 shows the protocol for a delegated transaction at an ATM and Figure 22 depicts the protocol flow. A brief overview of this protocol is discussed next.

i. To perform a delegated transaction at an ATM, U_D has to select the transaction type as "Delegated Transaction" (Figure 14) and enter the $delegated\_uid$ which can be viewed from the delegated credential at U_D's wallet, (step M1 in Table 14).

ii. U_D sends an *atmInitiateReq* to the ATM which contains $transactionType$ and uid. The ATM combines it with *atmData* to create *atmInitiateData* and sends it to the bank (Steps M2-M3 in Table 11). The *atmData* consists of *ATMSignature* and A_n. The *ATMSignature* is created by concatenating A_n and nonce, signed with the private key of the ATM. This proves the authenticity of the data sent by the ATM. These represent steps M2 and M3 in Table 14.

iii. The bank requests for delegation information to the BCC by invoking the $fetchDelegationReq$ function of

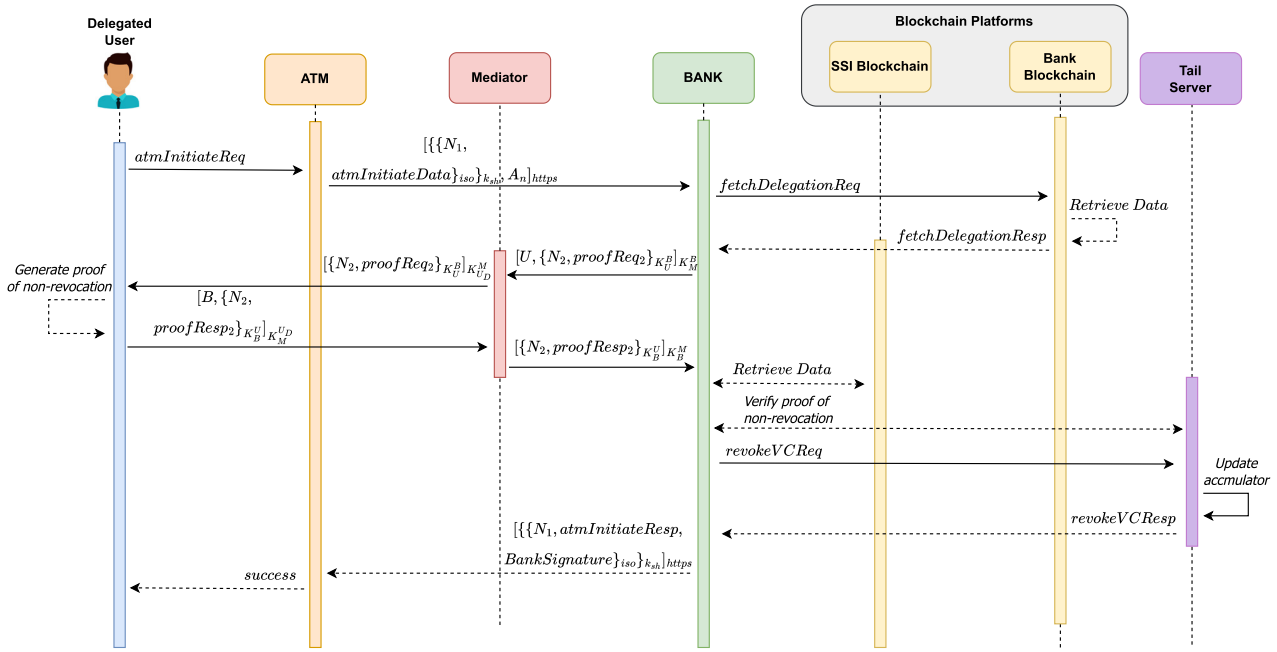
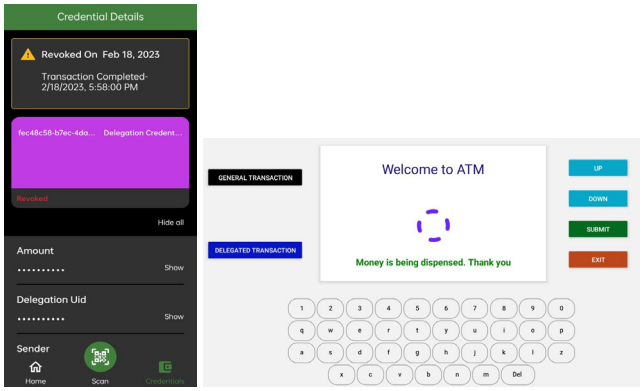


FIGURE 22. Delegated transaction.



(a) Revoked Credential (b) Dispensing Cash

FIGURE 23. Delegated transaction.

Algorithm 1 and *BCC* returns delegation information as *fetchDelegationResp* to the bank (Steps M4-M5 in Table 14).

iv. Using *fetchDelegationResp*, the bank invokes the *CheckRevocation* function from Algorithm 2 to ensure that the VC is not revoked. Then the bank agent prepares $proofReq_2$ to verify U_D and sends the proof request to the U_D's wallet via M. (Steps M6-M7 in Table 14).

v. The U_D's wallet parses the requested attributes and matches with the stored bank credentials $(VC_B^{U_D})$ in the wallet and asks for U_D's confirmation to present the proof. If the credential has already been revoked, the wallet will be unable to generate a valid verifiable presentation, and the proof generation will fail at this stage. After confirmation, the wallet returns the $proofResp_2$ to the bank agent via M (Steps M8-M9 in Table 14).

vi. The bank validates $proofResp_2$ and sends *revokeVCReq* to *SA* to revoke the VC and the *SA* revokes the VC (Figure 23a) by invoking the *revokeVC* function of Algorithm 2, representing steps M10-M11 in Table 14. If the revocation is successful, a *success* message is returned. As part of this process, the revocation registry is updated using the Indy Tails Server as discuss on Section III-D, which maintains the necessary cryptographic data for verifying non-revocation. If the *revokeVC* operation fails (e.g., due to ledger or network issues), the transaction will be marked as failed, and a status of false will be returned to indicate unsuccessful revocation.

vii. Finally, the bank asks the ATM to dispense the amount mentioned in $(VC_B^{U_D})$ with a *success* message (Figure 23b, steps M12-M13 in Table 14), only if the VC revocation is successfully completed. This ensures that the credential cannot be reused and maintains atomicity between revocation and cash disbursement.

VI. EVALUATION AND DISCUSSION

In this section, we present a number of evaluations of the developed system and analyse its advantages and disadvantages along with possible future work.

A. ANALYZING REQUIREMENTS

In this section, we discuss how our system satisfies previously mentioned functional, security and privacy requirements in Section III-B.

1) FUNCTIONAL REQUIREMENTS

The implementation of self-sovereign identity empowered automated teller machine satisfies all functional requirements. For example, our virtual ATM complies with standard ATM protocols, allowing users to perform financial transactions using their mobile wallets, thus fulfilling *F1*.

The wallet of the user records all released VCs and the user can check this list to track their monetary activities. This ensures transparency of their activities, thus ensuring *F2*. Our system also allows users to generate one-time credentials and share them with either account holders or non-account holders to withdraw cash, offering an alternative to the traditional cheque-based system. This capability aligns with the requirement outlined in *F3*.

2) SECURITY REQUIREMENTS

Our solution is designed to enhance the security by limiting the risks of spoofing and unauthorised privilege escalation. The authentication process in ATMs is achieved by presenting a VC, effectively meeting the requirement of *S1*. Furthermore, our system ensures the authenticity and non-repudiation of data through digital signature, thus satisfying *S2*. To prevent replay attacks, our system extensively employs nonces for API requests between various components, effectively addressing the requirement of *S5*. All communication between users, the bank, and ATMs, particularly when it involves sensitive data, occurs exclusively through secure and encrypted channels, primarily using HTTPS. This approach successfully meets the demands outlined in *S3*. As for *S4*, it pertains to the potential susceptibility of all web services to denial-of-service (DoS) attacks. It is important to acknowledge that addressing and mitigating these threats may require additional measures beyond the scope of the work presented here.

3) PRIVACY REQUIREMENTS

No action will be executed unless the respective user has consented to do so. Users can review the information they are sharing when they present a VC via their mobile wallet. The VC is released after the user's consent. This satisfies *P1*.

We provide a thorough mapping of these requirements to particular security and privacy threats to supplement this section, where we have described how each requirement has been met. This mapping is in line with the appropriate architectural components and protocol-level mechanism in Table 15. By doing this, we hope to demonstrate how our system integrates technological protections and design principles at the architectural and protocol layers to prevent possible vulnerabilities.

B. PROTOCOL VERIFICATION

As discussed in the last section, our system satisfies all the security requirements. However, the proposed system uses intricate relationships with several parties using complex protocols, resulting in a complex system, and protecting such a complex system is a daunting task. An approach to examine the security of such a system is to verify its underlying protocols using a protocol verifier. Taking this into account, we have used a state-of-the-art Protocol Verification tool called *ProVerif* [72] to formally examine the underlying protocol. We have examined, using ProVerif, if the created system's protocol meets the secrecy and authenticity objectives. The secrecy objectives determine whether a secret value remains secret while being transferred between two entities. On the other hand, the authenticity objectives verify the legitimacy of two entities while data are sent between them. In the following, we discuss the secrecy and authenticity goals of the protocols used in our proposed system.

1) SECRECY GOALS

We set some private variables in our ProVerif script. These variables stand for private information that we are attempting to share amongst various entities. Below is a quick description of these variables.

- *attrResp & connectionResp:* Registration data from user to bank and bank connection URL for general user
- *didExchangeResp & proofResp*$_1$*:* DID of the bank for a user and the presentation of user's VC to authenticate.
- *atmInitiateData & atmUserInputData:* ATM-related data, Transaction metadata and Transaction related data.
- *createDelegationResp:* Newly created delegation id from bank.
- *receiveDelegationResp*$_1$ *& receiveDelegationResp*$_2$*:* Bank connection URL and VC for the delegated user.
- *proofResp*$_2$*:* Presentation of a delegated user's VC to authenticate.

The ProVerif script analyses the reachability that allows the investigation of which terms are available to an attacker, and hence the (syntactic) secrecy of terms can be evaluated regarding a model. The queries to verify secrecy are presented in the Proverif scripts which are included in the footnote drive link.[1]

2) AUTHENTICITY GOALS

To check the authenticity of communication between entities, we have used some correspondence and injective correspondence assertions in our ProVerif script. Correspondence assertions establish event relationships within protocols, ensuring that if one event occurs, another must precede it. They ensure proper event sequencing without mandating a one-to-one relationship. Injective correspondence is a subset that enforces a strict one-to-one link between events. It is crucial in scenarios like financial transactions, preventing event duplication for system integrity and security. We highlight some of the key stages in the following.

- event *beginRegistration* & *endRegistration* marks the beginning and end of the registration process.

[1] https://drive.google.com/drive/folders/1fwu1_7vjJqNT4cxB0mzdscQfXHTUIGe9?usp=sharing

TABLE 15. Requirement analysis mapping with architectural components and protocol mechanism.

Requirements	Threats	Architectural Components	Protocol level mechanism
S1- Authentication	Spoofing and Elevation of privilege	Atm, Mobile wallet, Bank Controller Server	Presentation and verification of Verifiable Credentials (VCs) for user identity
S2- Data Integrity & Non-repudiation	Tempering and Repudiation	Mobile Wallet, Bank Controller Server, Private Blockchain	Use of digital signatures on credentials and transaction data
S3- Confidentiality	Information disclosure	ATM, Mobile Wallet, Bank Controller Server, Mediator	All communication over secure HTTPS channels using TLS encryption and DIDcom channel using asymetric encryption
S4- Availability	Denial of service	Private blockchain, Bank Controller Server, ATM	System relies on private distributed architecture and usage of basic rate limiting in server apis
S5- Replay Attack Protection	Replay attack	ATM, Bank Controller Server, Mobile Wallet	Use of nonces and timestamps in API requests to prevent replay attacks
P1- User Consent & Privacy	Lack of consent	Mobile Wallet, Bank Controller Server, ATM	User reviews data before releasing the VC via the mobile wallet

```
Query not attacker(attrResp[]) is true.
Query not attacker(connectionResp[]) is true.
Query not attacker(didExchangeResp[]) is true.
Query event(endConnection(connectionResp_1)) ==> event(beginConnection(connectionReq_1)) is true.
Query event(endConnection(connectionResp_1)) ==> event(endAttrRequest(attrResp_1)) is true.
Query inj-event(endDIDexchange(didExchangeResp_1)) ==> inj-event(beginDIDexchange(didExchangeReq_1)) is true.
```

FIGURE 24. Results of connection establishment protocol validation.

```
Query not attacker(credResp[]) is true.
Query event(endIssueCred(credResp_1)) ==> event(beginIssueCred(credReq_1)) is true.
```

FIGURE 25. Results of VC issuance protocol validation.

```
Query not attacker(proofResp_1[]) is true.
Query not attacker(atmUserInputData[]) is true.
Query not attacker(executeTransactionResp[]) is true.
Query event(beginBankInitiate(atmInitiateData_1)) ==> event(beginAtmConnection(atmInitiateReq_1)) is true.
Query event(endBankInitiate(atmInitiateResp_1)) ==> event(beginBankInitiate(atmInitiateData_1)) is true.
Query inj-event(beginProofReq(proofReq_2)) ==> inj-event(endBankInitiate(atmInitiateResp_1)) is true.
Query inj-event(endProofReq(proofResp_2)) ==> inj-event(beginProofReq(proofReq_2)) is true.
Query inj-event(beginTransaction(atmUserInputData_1)) ==> inj-event(endProofReq(proofResp_2)) is true.
Query inj-event(endTransaction(executeTransactionResp_1)) ==> inj-event(beginTransaction(atmUserInputData_1)) is true.
```

FIGURE 26. Results of general transaction protocol validation.

- event *beginAttrRequest & endAttrRequest* invokes when the bank requires attributes from users to register.
- event *beginLogin & endLogin* states the beginning and ending of the login process.
- event *beginConnection & endConnection* denotes a connection establishment process between two agents.
- event *beginDIDexchange & endDIDexchange* invokes the DID exchange process between two entities.
- event *beginIssueCred & endIssueCred* marks the start and end of the issue credential process.
- event *beginAtmConnection & endAtmConnection* states when a connection process with an atm begins and ends.
- event *beginBankInitiate & endBankInitiate* invokes the verification process of a user at the ATM.
- event *beginProofReq & endProofReq* denotes the beginning and the ending of the presentation of VC for verification.
- event *beginCreateDelegation* takes place to start the creation and issuance of a delegated VC.

The correspondence and injective correspondence queries for these events to verify the authenticity objectives are presented in the Proverif scripts which are included in the footnote drive link.[1]

3) QUERY RESULTS

Sensitive information that may be subject to illegal access or manipulation by malevolent actors is identified by the objectives of secrecy and authenticity. All specified secrecy

FIGURE 27. Cognitive procedure.

features are strongly protected against probable adversary behaviors, according to our ProVerif study of the protocol. The integrity and authenticity of the transaction process are also maintained by the protocol's structure, which successfully guarantees that the crucial chain of events cannot be interrupted. True, false, or cannot be proved are the three possible answers that ProVerif returns for its queries. Every query yielded true results once we executed the ProVerif scripts. By demonstrating that the confidentiality and authenticity goals are fulfilled, these results validate the protocol's compliance with these security objectives. We intend to showcase the results obtained from the aforementioned queries in conjunction with visual evidence in the form of screenshots depicting the execution of the scripts utilizing the ProVerif executable. Figure 24 and Figure 25 show the results of the connection establishment and VC issuance protocols, demonstrated in Table 9 and Table 10. The ProVerif result of the general transaction protocol, described in Table 11, is shown in Figure 26. In the footnote, a hyperlink to a Google Drive containing the ProVerif scripts is included, facilitating independent verification of the scripts.[1]

C. USABILITY EVALUATION

The proposed SSI-based ATM introduces a novel way of using ATMs to carry out monetary activities which is different from what we use today. Therefore, it is important to investigate the usability of this system. Usability evaluation is a crucial step in the product development process to ensure that a digital product is user-friendly and meets the needs of its intended audience.

Toward this aim, we have conducted a usability study of our system. In this section, we present the participant recruitment approach, the methodology, and the findings.

1) COGNITIVE WALKTHROUGH

In our evaluation, we have used *Cognitive walkthrough* which is a usability inspection method that relies on an analytical approach and is widely used method to swiftly and accurately decide whether or not a design solution is simple enough for a new or infrequent user to comprehend [73]. While this method should not be used in place of user evaluation, it might be useful for spotting problems early in the implementation phase when we do not have access to a large number of real users. We have selected the cognitive walkthrough approach for the following reasons.

- It allows us to choose and assess standard activities on varied devices, and to help us to identify the common difficulties and accomplishments of our proposed system.
- User onboarding prioritises the user-friendly and educational aspects of a product, focusing on ease of use and learnability. This process, led by experts or semi-experts, aims to ensure a smooth introduction for new users [74].

2) METHODOLOGY

The cognitive walkthrough has five steps [73] which are:

- **Test Plan:** Clearly define the problem or goals and the marketplace or platform that the usability test aims to address. This helps to set the scope and objectives of the test.
- **Task Definition:** Define specific tasks for the participants to perform, ensuring they are realistic, relevant, and clear.
- **Recruit Participants:** Select a group of users or experts to perform a cognitive walkthrough to identify potential usability issues in tasks and overall design.
- **Conduct Evaluation:** Administer the usability test, observe participants, and document issues encountered.
- **Analysis and Report:** Categorise and analyse identified issues based on severity and user impact. Propose improvements to enhance usability, including design changes and feature enhancements.

The outlined steps provide a structured approach to planning, executing, and analyzing usability tests, which can lead to valuable insights for improving the product's overall user experience. Based on these steps, the walkthrough was conducted at the beginning of February, 2023.

3) TEST PLAN AND DEFINITION (TASKS AND HEURISTICS)

For our cognitive evaluation, we outlined seven essential tasks that a typical user must perform. We meticulously observed user behaviors to identify system flaws that might arise later. There were primarily seven tasks involving two types of transactions: general and delegated transactions. Participants were given all the information necessary to complete their assignments in advance. In addition, a series of questions were designed to record their comments and potential improvement suggestions.

a: MAIN TASKS

The experiment had seven tasks with each task having its different sub-tasks. The tasks and their corresponding subtasks (21 in total) are presented in Table 16. The recruited participants independently completed each of the seven tasks and their respective subtasks.

b: HEURISTICS

Because of the importance we placed on the configurability and ease of use, we conducted our walkthrough on a set of heuristics that were created for an evaluation of Tor's usability [75]. We opted to comply with these guidelines because, similar to ToR, effective management of our

TABLE 16. List of tasks and subtasks.

Task	Sub-task
Task 1: Registering an account	• **Sub-task 1.1:** Registering an account to the bank's website by entering various information, e.g. user name, national ID, email, and password. • **Sub-task 1.2:** Checking emails for a confirmation. A successful registration gave each user 1000 units of virtual cash to simulate transactions at the virtual ATM.
Task 2: Establishing an SSI connection	• **Sub-task 2.1:** Logging into the bank's website using registered credentials. • **Sub-task 2.2:** Establishing an SSI connection by scanning a QR on the bank's website.
Task 3: Requesting a VC	• **Sub-task 3.1:** Requesting for a VC from the bank website. • **Sub-task 3.2:** Receiving the VC on the user's mobile wallet and accepting to store it in the wallet. • **Sub-task 3.3:** Checking the newly obtained VC in the mobile wallet.
Task 4: Performing a monetary transaction	• **Sub-task 4.1:** Initiating the regular transaction procedure at the virtual ATM. • **Sub-task 4.2:** Accepting the proof request by the user in the mobile wallet and releasing the requested attributes from the VC. • **Sub-task 4.3:** Checking the balance by selecting the check balance option from the virtual ATM display. • **Sub-task 4.4:** Selecting the withdrawal option to perform a transaction by entering a valid amount to complete the withdrawal. • **Sub-task 4.5:** Checking the new balance.
Task 5: Creating and sharing delegated transactions	• **Sub-task 5.1:** Creating a delegation at the bank's website by logging into the bank and then providing a secret reference key and valid amount from the balance to withdraw. • **Sub-task 5.2:** Sharing the generated unique ID (delegation uid) to a delegated user along with the reference key.
Task 6: Requesting a delegated credential from the bank website by using the delegated transaction information	• **Sub-task 6.1:** Receiving a delegated credential by visiting the delegation page of the bank website without any login and entering the received delegation uid and the reference key. • **Sub-task 6.2:** Scanning the QR code with the mobile wallet at the bank website. • **Sub-task 6.3:** Accepting the delegated VC from the bank by requesting it. This credential will contain the name of the sender, amount, and delegation uid. • **Sub-task 6.4:** Checking the delegated credential in the mobile wallet.
Task 7: Performing a delegated transaction	• **Sub-task 7.1:** Initiating the delegated transaction at the virtual ATM by selecting the transaction type and entering the delegation uid . • **Sub-task 7.2:** Accepting the proof request by sent the bank and releasing the requested attributes of the delegated VC from the mobile wallet. After validating the ATM will dispense the cash. • **Sub-task 7.3:** Checking in the wallet if the delegated VC has been revoked.

TABLE 17. Time (min) taken by each participant while conducting the tasks.

Tasks	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Set Up	2.14	1.53	2.20	1.58	2.07	2.25	2.02	2.21	2.17	2.20
Task 1	2.60	2.38	2.42	2.23	2.27	2.33	2.80	2.92	3.12	3.05
Task 2	3.15	3.06	3.56	3.10	3.28	3.37	4.12	3.87	4.20	4.17
Task 3	2.32	2.41	2.27	1.54	2.15	2.52	3.06	2.94	2.99	3.11
Task 4	5.36	4.85	4.89	4.41	4.35	4.74	5.52	5.75	5.81	5.94
Task 5	1.43	1.53	1.58	1.44	1.42	1.61	2.12	2.03	2.12	2.10
Task 6	2.16	2.07	1.92	2.14	2.02	2.24	2.43	2.57	2.64	2.71
Task 7	1.74	1.80	1.76	1.64	1.82	1.93	2.24	2.14	2.31	2.38

system requires carrying out some complex operations. These guidelines are [75]:

- **G1:** Users should understand the steps necessary to execute a core task.
- **G2:** Users should be able to figure out how to complete these actions.
- **G3:** Users should be able to determine when a fundamental task has been completed.
- **G4:** Users must have the ability to identify, diagnose, and recover from non-critical failures.
- **G5:** Users should not make errors from which there is no recovery.
- **G6:** Users must be familiar with the language used in interface dialogues and documentation.
- **G7:** Users must be comfortable enough with the interface to continue using it.
- **G8:** Users should always be informed of the status of the application.

4) PARTICIPANTS

Ten participants (identified as P1, P2, P3, P4, P5, P6, P7, P8, P9 and P10) were selected for the cognitive walkthrough. The consensus within the field recognises that conducting

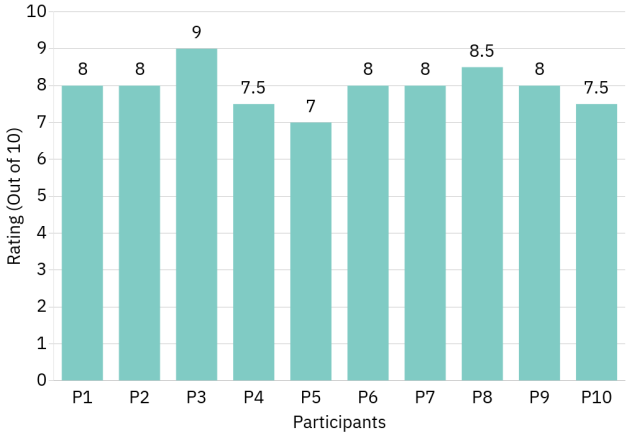


FIGURE 28. User rating out of 10 for Ten participants.

usability evaluation with the cognitive walkthrough method with a sample size of five participants is effective in identifying a nearly equivalent number of usability issues compared to a larger group of participants [76]. An analytical cognitive walkthrough is particularly advantageous due to its reliance on expert users, enabling the simulation of evaluations from the perspective of a novice user. All participants voluntarily participated in the study and were used to ATMs, had some knowledge of financial systems and a basic understanding of SSI technology. The volunteers' ages ranged from 22 to 39 years. Six of them were 4th-year undergrad students of Shahjalal University of Science and Technology, Bangladesh, two were garment industry workers and other two were small-scale retailers. Among the university students, four of them came from the Department of Computer Science and Engineering, one from the Department of Economics, and the last one was from the Department of Chemical Engineering and Polymer Science. The participants in this experiment never used an SSI-based system like the one that we proposed and developed though they are well known with traditional ATMs. All participants carried out these tasks one by one, and the details are described in Section VI-C5. We kept track of the time for each of these tasks completed by each participant outlined in Table 17.

5) FINDINGS

In this section, we summarise our findings from the analysis of our walkthrough.

a: TASK 1- REGISTERING AN ACCOUNT

All participants successfully completed this task without encountering difficulties. It took 2-3 minutes to complete the registration step (Table 17) as the design was easy to understand for the users (achieving G1, G2, G5, G6, G7). Users receive notifications when registration is completed successfully or an error occurs (achieving G3, G4). It took a little time to write the data to the bank blockchain and verify them. However, users are always informed about the application's status (achieving G8).

b: TASK 2- ESTABLISHING AN SSI CONNECTION

Each participant completed this task successfully by scanning the QR code to establish an SSI connection which was easier than the usual login (achieving G7). All the necessary information is described on the website (achieving G1, G2, G4, G6). Each user took less than 4.5 minutes to complete this task (Table 17). User P3 faced a problem when scanning the QR code (failing G5). If scanning is successful, the phone vibrates to notify (achieving G3, G8).

c: TASK 3- REQUESTING A VC

Every participant completed this task easily (achieving G1, G2, G6, G7). They did not face any errors that had no recovery (achieving G4, G5). After requesting a credential from the website, the users received notifications from both the bank website and the mobile wallet that a new credential was issued/obtained (achieving G3, G8). This task took around 1.5-3 min (Table 17).

d: TASK 4- PERFORMING A MONETARY TRANSACTION

Every participant successfully completed this task, although some experienced slight confusion when initiating a connection with the virtual ATM due to differences in the ATM interface (failing G7). Almost every participant had to take a little time to understand how the ATM works. After authentication, it was quite easy for the users to carry out the financial activities without errors (achieving G1, G2, G3, G4, G5, G6, G8). To complete this task, each participant took 4-6 min (Table 17).

e: TASK 5- CREATING AND SHARING A DELEGATION

Delegation is rather a new concept. We had to explain to users how the delegation concept works. Once the users understood the concept, it did not create any problems (achieving G1, G2, G6, G7). It took 1.5-2 minutes to complete this task (Table 17). No participants faced fatal errors and were informed of their activities by notifications (achieving G3, G4, G5, G8).

f: TASK 6- REQUESTING A DELEGATED CREDENTIAL FROM THE BANK

Every participant completed this task with ease (achieving G1, G2, G7, G8). They took 2-3 minutes to complete this task (Table 17). The process of obtaining delegated credentials is the same as usual and hence, the users did not face any fatal difficulties (achieving G3, G4, G5, G6).

g: TASK 7- PERFORMING A DELEGATED TRANSACTION

Each participant successfully completed this task without encountering any issues (achieving G1, G2, G6, G7). The process remains the same as usual, and users are notified upon successful completion (achieving G3, G4, G5, G8). It took 2-3 minutes to complete this task (Table 17).

6) OBSERVATION

Based on the cognitive walkthrough conducted with ten participants, it was found that the proposed ATM authentication approach, which involves acquiring a credential from a bank's website and then using it to authenticate using a mobile wallet to a virtual ATM, was generally usable. Participants were able to complete the essential tasks, such as registering an account, establishing an SSI connection with the bank, requesting and storing VCs in the mobile wallet, and performing transactions at the virtual ATM, without significant difficulties. However, some minor issues were identified, such as the need for clearer instructions and more straightforward navigation in some tasks.

Throughout the usability evaluation, we observed a few instances in which users had difficulties performing tasks or encountered ambiguity. Some participants experienced difficulties in navigating the website and some ATM's interfaces. Some error messages were not always clear or helpful when users encountered errors or problems.

Based on these findings, we choose some improvements to enhance the usability of the website and the ATM which are enhancing information architecture to make it simpler, and noticeable and providing clients with clear, concise messages that instruct them on resolving any problems they encounter. By implementing these changes, the usability of the website has been enhanced, resulting in a better user experience.

Most of the users preferred our system to the traditional banking system as the time delay was not that much of a deal in exchange for the immutable and temper-proof records of identity and transaction data which provides a high level of security and privacy. In our innovative virtual ATM system, we require bank authentication twice and offer two types of transactions, taking 20-30 minutes on average for all seven tasks. This may seem lengthy, but it is faster than traditional bank card and chequebook acquisition. Executing a full transaction, including balance checks before and after the transaction, takes 4-5 minutes, while a cash withdrawal alone with bank verification takes 1.5-2 minutes. Users appreciated the security and efficiency of our system after usability evaluation. Five out of ten users rated our system 8 out of 10, one rated 8.5, two rated our system 7.5, another one rated 7 and one of them rated 9 as shown in Figure 28. It is important to mention again that for usability evaluation, we used the Cognitive Walkthrough approach, which is usually carried out by experts or semi-experts who are familiar with system interfaces. This approach is very good at spotting usability problems that might impact inexperienced users. However, we were unable to include a broader age and literacy group in our study for this system, we have acknowledged this in the challenge section. We are also planning to expand the user study in future research.

D. PERFORMANCE EVALUATION

In this section, we analyse the performance of the developed PoC. To evaluate its performance, we have taken into account two parameters, namely throughput and latency. In this context, throughput is defined as the number of requests that a specific entity is capable of processing within a given timeframe, whereas response time pertains to the duration it takes for an entity to respond to a specific request.

1) EXPERIMENTAL SETUP

The system consists of four main components, namely the bank SSI agent, the ATM, the bank controller server and the wallet. Figure 2 depicts how these components will be connected in a real-life scenario. For this test, we have used a PC with configuration Core™i7-13620H (2.4GHz), 16GB DDR4 RAM, and 512GB SSD.

2) EXPERIMENTAL OUTCOMES

The entire flow has been partitioned into six segments. The process involves establishing a connection, issuing credentials, executing general transactions, creating delegations, obtaining delegations, and executing delegated transactions. In each scenario, we started with an initial load of 100 users. Then we increased the load to 250, 500, 750 and 1000 users, running the simulation again at each level. This study is focused on assessing how well the system performs when faced with a sudden increase in traffic, leading to higher service demands. Figure 29 represents the results from our performance analysis tests. We calculated the latency for each load level. The findings indicate that latency increases as the number of users accessing the system increases. For the initial load of 100 users, the latency times are as follows, 2.21 seconds for establishing a connection, 1.57 seconds for credential issuance, 7.97 seconds for general transactions, 0.81 seconds for delegation creation, 10.38 seconds for receiving delegations, and 13.47 seconds for delegated transactions. As the load increases to 1000 users, the latency times also increase, reaching 28.31 seconds for establishing a connection, 12.65 seconds for credential issuance, 61.02 seconds for general transactions, 11.19 seconds for delegation creation, 123 seconds for receiving delegations, and 164 seconds for delegated transactions in their respective use-cases.

One noticeable pattern in the graph is that as the load on the system increases, the latency for all scenarios tends to increase. This is a common behaviour and suggests that the system experiences higher response times as it becomes more heavily loaded. At the highest load level (1000), there is a noticeable steep increase in latency for all scenarios, specifically general transaction, get delegation and delegated transaction. It is important to note that, while our evaluation highlights the latency and load characteristics of the proposed SSI-based authentication mechanism, we did not include a direct benchmark comparison with traditional ATM card-and-PIN systems. To the best of our knowledge, no published studies provide detailed performance metrics for authentication flows in conventional ATMs. Furthermore, our work modifies only the authentication process; the underlying transaction processing pipeline remains identical to existing

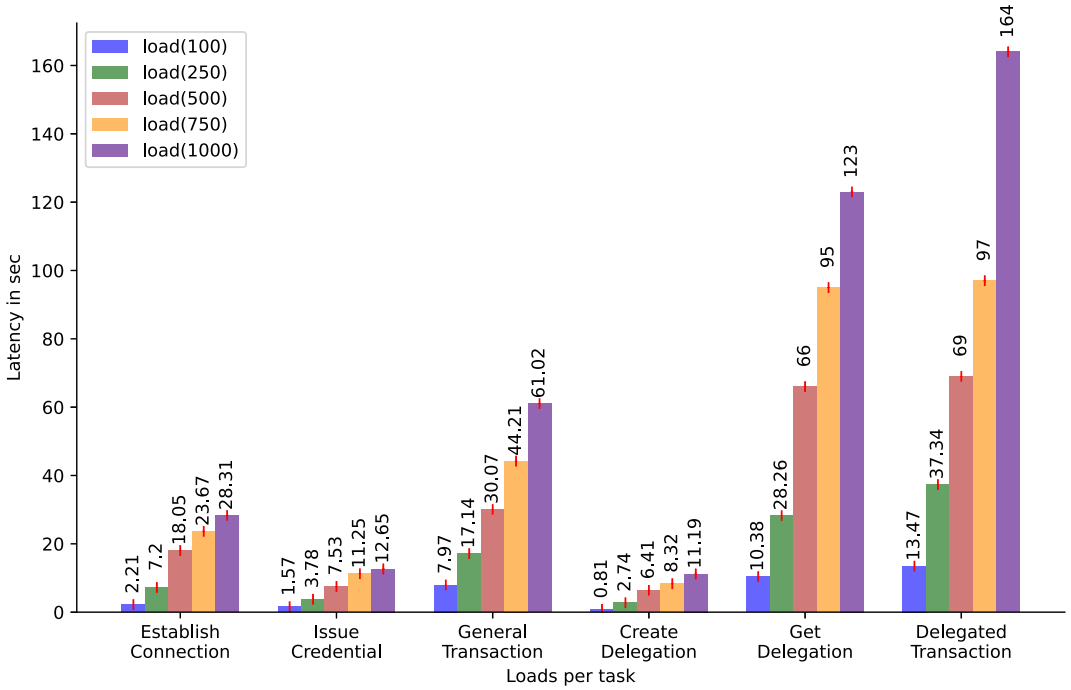


FIGURE 29. Performance evaluation: Latency vs Load (Ramp-up time: 0s).

ATM operations. Consequently, our performance analysis isolates the authentication component, demonstrating that SSI credentials introduce minimal overhead relative to typical ATM session times while offering security and usability benefits. To address these scalability challenges in production environments several architectural strategies can be employed. Horizontal scaling involves deploying multiple acapy agent instances across distributed infrastructure to handle higher transaction volumes efficiently. Load balancing ensures that incoming requests are evenly distributed, preventing any single node from becoming a bottleneck. Adopting a clustered deployment model using containerised or virtual machine-based clusters can improve both availability and parallel processing capabilities. Additionally, asynchronous processing through message queues allows for non-blocking request handling can reduce response time under load. Implementing DID document caching at the application layer can further minimise latency by eliminating repetitive registry lookups. Finally, auto-scaling through container orchestration platforms like Kubernetes [36] enables the system to dynamically adjust resource allocation in response to fluctuating demand, ensuring responsiveness and resilience under real-world conditions.

E. DEPLOYMENT ECONOMICS: COST EFFECTIVENESS AND ROI

Our proposal replaces the physical card with verifiable credential, while reusing the existing ATM network and backend authorization flow. The change is therefore primarily software centric. In the following, we discuss several deployment aspects of the proposed system.

1) ONE-TIME SETUP COSTS (CapEX)

The following update would incur one-time setup costs:

- **ATM software updates:** UI flow for VC presentation, verifier integration, risk rules, and audit logging.
- **Mobile wallet integration:** Integrate a VC wallet into the bank's mobile app, including UX flows and backend hooks.
- **Bank portal integration:** Update online banking portal to integrate with agent APIs (issuance, VC binding/revocation, recovery), update UX flows and expose required backend endpoints.
- **Security & compliance:** Threat modeling, security testing, and staff training.
- **Ledger initialization:** In practice, on-ledger initialization is limited to publishing the issuer DID, schema, and credential definition; where an existing issuer DID and schema are reused, only the new credential definition must be written (revocation registry if enabled).

2) ONGOING OPERATING COSTS (OpEX)

The following would incur ongoing operating costs:

- **Agents support:** Hosting and operating issuer/verifier agents, with monitoring, security maintenance, routine updates, incident response, and user support desk.
- **Ledger access:** Issuance and verification use *read-only* ledger calls to fetch schemas, keys, and revocation

status; there are no per-issuance writes, so ongoing costs are mainly operations and any ledger membership.

3) SAVINGS DRIVERS

Next, we briefly discuss the potential savings for the respective financial institution by adopting our solution.

- **Card lifecycle elimination:** This includes the removal of plastic card manufacturing, personalization/printing, mailing and logistics, and periodic replacements, as well as a reduction in the reissuance of lost or stolen cards.
- **Fraud & chargebacks:** By eliminating magstripe and PAN data, the system removes common skimming and cloning attack vectors, while cryptographic holder-binding of verifiable credentials further reduces the risk of counterfeit transactions and associated chargebacks.
- **Operational agility:** Instant credential reissuance lowers remediation costs after compromise.

4) ROI MODEL

Let I_{one} denote the one-time investment (CapEx), S_{yr} the gross annual savings, and O_{yr} the incremental annual operating cost in steady state. We define the net annual benefit $B_{\text{yr}} = S_{\text{yr}} - O_{\text{yr}}$. We report the annual return on investment (ROI) and the payback period:

$$\text{ROI}_{\text{annual}} = \frac{B_{\text{yr}}}{I_{\text{one}}}, \qquad \text{Payback (years)} = \frac{I_{\text{one}}}{B_{\text{yr}}}.$$

Banks can parameterize S_{yr} with observed card lifecycle costs (unit card cost, issuance volume, replacement rate, logistics) and empirically estimated fraud loss reductions; O_{yr} covers agent hosting/support, revocation, monitoring, and any ledger subscription.

5) MIGRATION AND COEXISTENCE

The VC flow can run alongside cards during transition. No wholesale ATM replacement is assumed; roll-out can be phased by region or ATM cohort, with card fallback retained until adoption targets are met.

F. ADVANTAGES AND CHALLENGES

1) ADVANTAGES

The major advantages of our proposed system are stated below:

- **Enhanced Security:** SSI systems are designed to give users control over their personal information [13]. This can lead to increased security in ATM transactions, reducing the risk of identity theft and fraud. This is because the users do not need to insert any card and input the corresponding PIN and hence, there is no chance of any card-based attack vectors.
- **User Empowerment:** SSI puts users in charge of their identities, providing a sense of empowerment and control over their digital presence.
- **Delegated transaction:** We have implemented this new functionality for ATMs to reduce the hassle of the bank cheque system, thus saving users time and effort. Users can now authorise others to withdraw money from ATMs on their behalf, streamlining the process and minimising inconvenience.
- **Interoperability:** Our system provides an identity verification solution that wraps around the existing ATM system. Therefore, the proposed system could be integrated with different banks allowing it to be used in their respective ATMs.

2) CHALLENGES AND LIMITATIONS

A few possible challenges of the proposed system and limitations are stated below:

- **Implementation Challenges:** Replacing physical ATM cards with SSI-based verifiable credentials necessitates some modifications of existing systems. Banks must upgrade ATM hardware and software to accommodate new authentication mechanisms such as QR-code or NFC-based interactions, which are essential for reading digital credentials. Additionally, integration with legacy financial infrastructures requires careful planning to ensure that new modules work seamlessly with established transaction and fraud detection systems.
- **Adoption Hurdles:** The transition from traditional card-based verification to digital credentials could be met with considerable resistance from both end-users and financial institutions. Many users are comfortable with the familiar process of inserting or tapping a physical card at an ATM, and they may find the idea of managing digital wallets intimidating. At the institutional level, banks often exhibit a cautious approach toward adopting emerging technologies, particularly when the new system lacks industry-wide standards and a proven track record of reliability. The uncertainty surrounding return on investment and the potential risks associated with unproven technology could further slow down the pace of adoption.
- **Regulatory and Legal Challenges:** Integrating SSI into ATM transactions could introduce a complex regulatory landscape that banks must navigate carefully. Financial institutions are subject to strict compliance requirements, including data protection regulations such as GDPR (General Data Protection Regulation) [80] and financial mandates like KYC (Know Your Customer) and AML (Anti-money laundering). The decentralised nature of SSI could create ambiguity in assigning liability, especially in cases of fraud or when a user's cryptographic keys are compromised. Moreover, varying legal standards across different jurisdictions add another layer of complexity, making it difficult for banks to maintain consistent compliance on a global scale.
- **Limitations in Bifold wallet:** The Bifold wallet, which is designed to store and manage digital credentials, currently faces significant limitations that could impact its real-world deployment in financial services. One

major drawback is the absence of robust backup and restore mechanisms, which is critical for ensuring users do not lose access to their accounts in the event of device loss or malfunction. Without a secure and user-friendly recovery solution, the risk of permanent credential loss is significantly heightened, potentially undermining trust in the digital system. However,we collaborated with other authors in a separate project which addressed this issue in detail by developing a secure wallet backup and restore as presented in [15]. If this approach is officially adopted, then this particular limitation could be mitigated.

- **User Study:** Our usability evaluation primarily included only university students, garment workers and small-scale retailers. Even though our primary goal was to obtain some insightful data on task completion time and interaction fluency, it may not adequately represent the difficulties encountered by elderly users or those with low levels of digital literacy. As such, the generalizability of our findings to the broader population is limited. Previous research [77], [78], [79], however, has demonstrated that these populations frequently encounter more challenges using ATMs themselves. Our main objective was to determine how easily those with relevant ATM experience could use our suggested solution.
- **Performance Benchmarking Challenge:** This study does not include direct benchmarks against traditional ATM authentication. To the best of our knowledge, the literature lacks publicly reported performance metrics for conventional ATM authentication flows, limiting one to one comparison.

G. FUTURE WORK

Some of the possible future works are discussed in the following.

- In the proposed solution, we have employed *did:peer* and *did:sov* options. However, there are alternative did options, including *did:indy*, *did:cheqd*, *did:key*, *did:web*, *did:eth*, *did:ion*, and various others that can be explored and tested within the framework of the proposed solution.
- In the proposed solution, the bank assumes the roles of both issuer and verifier. However, we have the option to configure an ATM as an independent agent, serving as the verifier, while the bank retains its role as the issuer.
- Developing an efficient and optimised solution to minimise latency during high loads, ensuring a smooth and seamless user experience.
- We have used a legacy Indy-based verifiable credential labeled as a bank credential. To conduct a comparative analysis, we can explore other verifiable credential types such as Anoncreds, JSON-LD, SD-JWT, etc.
- In future, we also would like to explore how different challenges (as presented in Section VI-F2), particularly adoption hurdles as well as regulatory and legal challenges, could be handled.
- We intend to implement distributed deployment with load balancing and auto-scaling to support higher concurrency. Exploring DID caching, asynchronous processing, and container orchestration will also be key to making the system more resilient and production-ready.
- To accurately represent the demographic of ATM users and ensure inclusive design, we will conduct a large-scale usability evaluation with a more diverse group of participants, particularly elderly individuals and users from rural or low-literacy backgrounds, in our future research.
- We will extend the performance evaluation by incorporating controlled benchmarks against traditional authentication. This will depend on our ability to access reliable baseline metrics after potentially collaborating with banking partners and ATM vendors.

VII. RELATED WORK

Recently, there have been several research works using verifiable credentials and decentralised identities in the financial sectors.

Verifiable credentials, as described by Sun et al. [81], provide a reliable and private way for businesses to authenticate themselves to financial institutions, allowing for safe and decentralised identity verification in multilevel supply chain financing.

Ajayi et al. [82] discuss how Zero Knowledge Proof (ZKP), a feature of Self Sovereign Identity (SSI), can provide a safe framework for managing digital identities and confirming transactions when paired with blockchain's decentralised infrastructure, especially in Central Bank Digital Currencies (CBDCs) and other financial applications. ZKPs offer a state-of-the-art solution by permitting data verification without disclosing private information, protecting privacy and boosting confidence.

To address the credit assessment needs of financial institutions, Ou et al. [83] propose a Customer Self-Sovereign Identity and access-control framework (CSSI) based on SSI technology where customers can securely store assessable assets and credit data on the blockchain. Subsequently, a digital account address is associated with these data. By gaining access to this credit information, with customer authorisation, financial organisations that process loan applications may perform risk management and a thorough assessment of customers' ability to repay loans.

Unprecedented innovations in the financial industry are made possible by the open banking framework, which gives third-party providers access to financial data from all banking institutions. However, several open banking standards are still vulnerable to serious technical threats, such as unreliable data sources, uneven data integrity, and lack of immutability. To eliminate these issues, Rahman et al. [84] propose a layered architecture that covers source validation,

data-level authentication, and tamper-proof storage, offering three different levels of trust assurance in data trustworthiness. Decentralised identity and verifiable presentation are used in the first layer to ensure the validity of the source, while cryptographic signature is used in the second layer to confirm the consistency and authenticity of the material. Finally, the Tangle, a directed acyclic graph distributed ledger, is used in the third layer to provide data immutability.

For secure data exchange in open finance, Ahmed et al. [85] propose the Self-Sovereign Banking Identity (SSBI) framework, which is based on blockchain technology. SSBI uses the Veramo SDK and Ethereum with SECP256K1 elliptic curve support to implement key storage and signature enclaves on users' banking cards. SSBI is a successful strategy for reliable open banking compared to related technologies; nevertheless, the scalability of blockchain is still an issue for practical implementation.

Ahmed et al. mention [86] that conventional Identity Management Systems (IDMSs) depend on centralized authorities, which restrict user control and provide single points of failure. As a decentralized substitute, blockchain-based Self Sovereign Identity (SSI) enables users to completely own and manage their online personas. SSI has been used in the finance industry for confidential data transfer and secure identification verification. However, rather than ATM use cases, the majority of efforts concentrate on online banking or general identification systems. While current solutions still rely on real banking cards, our research investigates using SSI credentials in place of cards to enable cardless ATM transactions that protect privacy.

Multiple studies have also explored the possibility of tackling the existing bottlenecks of ATMs. For example, the authors in [87] and [88] have proposed integrating biometrics and/or one-time passwords (OTP) into the ATM system to increase ATM security. There are other proposals in the literature, e.g. [88], where the authors have proposed integrating blockchain within the ATM ecosystem to increase the security of ATMs. Eranga et al. [89] have claimed that traditional ATM, debit, and credit card systems inherit several flaws, such as lack of ATMs in rural regions, high cost of ATM maintenance, and possible security vulnerabilities in ATM systems, among others. The authors have suggested a new platform called "Promize", an SSI-enabled mobile wallet for its end users which offers a low-cost peer-to-peer cash transfer system based on blockchain technology and eliminates the need for traditional methods such as ATM withdrawals and credit card transactions. Although their approach has merits, it is impractical to think of completely replacing ATM services in the near future. It is important to note that the objective of the Promize platform is to provide an alternative channel that enhances the overall security and efficiency of financial transactions. It is essentially a blockchain-based peer-to-peer money transfer to replace ATMs. However, our solution is designed to operate alongside existing ATMs, leveraging the benefits of SSI and blockchain technology at the authentication layer without necessitating radical changes to current infrastructure. This complementary approach allows financial institutions to gradually adopt innovative technologies, addressing vulnerabilities in traditional systems while preparing for an increasingly digital future. Furthermore, the Promize solution did not adhere to any SSI framework that supports the latest SSI standards. We contend that merely using a blockchain without adhering to any SSI standard is insufficient to claim that it is an SSI-based solution.

The reviewed research works are compared with our proposed system in Table 18 which demonstrates growing interest in blockchain based Self Sovereign Identity for financial use cases. However, most studies remain conceptual or limited to prototype. The comprehensive survey by Ahmed et al. [86] mentions how blockchain based SSI solution can enable users to completely own and manage their online personas and eliminate the single point failure. Several recent research works extend SSI to specific financial contexts. Sun et al. [81] tackle multi-level supply-chain finance with layered identity verification using SSI component - verifiable credentials and signatures, Ajayi et al. [82] and Ou et al. [83] explore ZKP and verifiable credential based approaches for DeFi and institutional access control, and Rahman et al. [84] propose multi-layer architectures for open banking data to cover source validation, authentication and temper proof storage using decentralised identity, verifiable presentation and cryptographic signature. Ahmed et al. [85] is notable for a prototype that integrates banking cards as secure key stores (SECP256K1 on Ethereum), but even that work identifies blockchain scaling as a barrier to real-world deployment. Earlier ATM-focused studies [87], [88] focus on strengthening card+PIN authentication with biometrics and OTPs. These methods improve security to a certain degree, but they frequently ignore two important factors: (i) the user's management and prioritization of their own identification data, and (ii) the decentralization of sensitive data to reduce dependence on a single point of failure. The use of self-sovereign identity and blockchain in both Eranga et al. [89] and our proposed system underscores a shared commitment to empower users with control over their personal data and elimination of single point failure. However, their work presents a peer-to-peer money transfer to replace ATMs, whereas our suggested system presents an identity verification solution using the current ATM infrastructure to reduce card-related concerns.

Across technical dimensions, most prior studies discuss cryptographic building blocks (VCs, DIDs, signatures, and in some cases ZKPs) and propose integration strategies at a conceptual level. Very few provide detailed end-to-end architectures showcasing how SSI components interoperate with legacy banking networks or ATM hardware. Empirical discussion of usability, performance and real deployment effects are rare. Implementations are typically limited to concept or small prototypes and do not address realistic user loads or formal protocol analysis.

TABLE 18. Comparison of reviewed papers.

Aspect	Architecture	Cryptographic Components	Integration Strategies	Usability & Protocol Validation	Performance Evaluation & Scalability	Real-World Deployment
Sun et al. [81]	Multi-level SSI for supply chain finance with layered identity verification	Multi-factor cryptographic proofs with VC and digital signatures	Integrate with supply chain & finance systems	No user study and protocol validation	Performance evaluated without scalability analysis	Conceptual implementation
Ajayi et al. [82]	Blockchain based DeFi identity with ZKPs	Zero-knowledge proofs (ZKPs)	Integration of DeFi platforms with compliance	No discussion available on both	Do not mention, recommended for future study	Conceptual implementation
Ou et al. [83]	SSI framework for finance	Verifiable credential and cryptographic signing	Financial systems for transparent access control	Not empirically evaluated and discussed	No performance evaluation & no analysis of scalability.	Conceptual, early-stage proposal
Rahman et al. [84]	Multi-layer open banking architecture	Hashing, consensus, verifiable IDs	Interoperability with open banking APIs	No user evaluation and protocol validation	Conceptual scalability and performance evaluation	Prototype-level, no large-scale deployment
Ahmed et al. [85]	SSI framework using banking cards, linking physical and digital	Digital signatures and cryptographic bindings	Integrated with customer banking cards & Ethereum	No usability evaluation and protocol validation	Performance analysed (latency, throughput) but no scalability analysis	Prototype implemented on test Ethereum network
Ahmed et al. [86]	Survey of SSI based ecosystems	Digital signatures, hashing, consensus	Conceptual review of integration strategies	Not applicable	Not applicable	Review paper, no deployment needed
Koteswari et al. [87]	Fusion of iris & fingerprint biometrics for ATM authentication	Biometric recognition	ATM system integration	Usability evaluated by survey but no protocol validation	Not analysed	Conceptual study on ATMs without any implementation
Iloduba et al. [88]	Biometric & blockchain hybrid for ATM and bank transaction	Biometric data on blockchain using hashing	ATM & bank transaction systems integration	Questionnaire on atm frauds, no protocol validation	Not analysed	Conceptual, not implemented
Bandara et al. [89]	Blockchain-based, low cost, peer-to-peer money transfer system with mobile wallet	RSA digital signatures, JWT-based authentication, QR code based identity exchange	Integrates with core banking APIs, mobile wallets and bank	No usability evaluation and protocol validation	Performance evaluated with scalability	Live deployment at Merchant Bank Sri Lanka
Proposed system	Blockchain based SSI enabled ATM authentication using Verifiable Credentials	DIDcom channel, anoncreds, selective disclosure, CL-signature	Integrates with core banking APIs, mobile wallets and ATM api	Usability evaluation using cognitive walkthrough and protocol validation using proverif	Performance evaluation (latency) with scalability analysis	Prototype-level, no large-scale deployment

Our work addresses these gaps by presenting a practical SSI-based ATM authentication system that replaces physical cards with SSI credentials and evaluates the solution on multiple important fronts. Unlike the conceptual and prototype studies listed above, we have implemented the full authentication flow and conducted: (i) performance/load testing under realistic concurrent loads (50, 100, 250, 500, 1000 users) to measure latency, (ii) usability assessment via cognitive walkthroughs to capture user error modes and learnability and (iii) formal protocol validation using ProVerif to demonstrate security properties of the authentication protocol. These combined evaluations allow us to demonstrate technical feasibility, quantify scalability behaviour, and surface human-factors issues that prior work has largely left unexamined. Our contribution complements and extends the literature by (a) tailoring SSI to the ATM context (cardless, credential-based authentication), (b) integrating SSI components with banking authentication flows, and (c) providing a comprehensive evaluation (performance, usability, and formal security) that makes a stronger case for practical adoption. Overall, the proposed system demonstrates a proactive approach toward threat modelling, practical implementation, protocol validation and usability evaluation, thus distinguishing itself from the conceptual focus of the related works. These insights collectively underscore the innovative and comprehensive nature of the proposed system, poised to address contemporary challenges in the domain effectively.

VIII. CONCLUSION

This research is motivated by the pressing need to address the security vulnerabilities and inefficiencies inherent in

traditional ATM systems, such as susceptibility to fraud with skimming devices and unauthorised access. The integration of self-sovereign identity and blockchain technologies serves as a pivotal strategy to mitigate the identified security vulnerabilities and operational inefficiencies within the conventional ATM infrastructure. This approach aims to enhance the security, autonomy and trustworthiness of identity verification and transaction processes, thereby fostering resilience and innovation in digital financial services.

Towards this aim, the primary goal of this research is to replace ATM cards and PINs with self-sovereign identities and independently verified credentials, thus improving the safety and simplicity of financial transactions. Additionally, the delegated transaction mechanism makes our system more profound and usable as it can replace the currently widely used cheque-based transaction system. We have presented the architecture of the proposed system which is based on a threat model and a requirement analysis. We have outlined the required protocol flows and examined its security, performance as well as usability aspects. All things considered, this research is meant to be a ground-breaking contribution to the never-ending fight for better financial transaction safety and efficiency.

Despite the numerous conveniences of the proposed system over any traditional system, we must admit that the developed version of our system does have some limitations. A mobile wallet plays a crucial role in our system. The Bifold wallet (the wallet that was used in our PoC) has a fingerprint scanner and a PIN for enhanced protection. More studies are needed on the wallet usability and scalability for its wide adoption. Even with these limitations, it is worth noting that our usability evaluation participants, albeit very small in numbers, were convinced and receptive to this new system due to the security and convenience it brings.

In conclusion, this is a novel approach that presents a new application for SSI. It is justifiable to assert that this new approach has the potential to disrupt the current ways ATM transactions are conducted and to introduce a new line of research in this direction.

REFERENCES

[1] E. O. Koranteng, E. B. Ntiamoah, F. Owusu, and M. Owusu, "A theoretical analysis of Automated Teller Machine (ATM) system on customer satisfaction in the banking industry: The case study of HFC bank," *Eur. J. Bus. Manage.*, vol. 8, no. 3, pp. 183–191, 2016. [Online]. Available: https://www.iiste.org/Journals/index.php/EJBM/article/download/28359/29102

[2] World Bank Group. *Automated Teller Machines (ATMs) (Per 100,000 Adults)*. Accessed: Jul. 11, 2023. [Online]. Available: https://data.worldbank.org/indicator/FB.ATM.TOTL.P5

[3] National Cash Systems. *ATM Statistics*. Accessed: Jul. 11, 2023. [Online]. Available: https://nationalcash.com/statistics/

[4] Global Inf. *ATM Global Market Report 2025*. Accessed: Jun. 10, 2025. [Online]. Available: https://www.giiresearch.com/report/tbrc1674310-atm-global-market-report.html

[5] M. M. Rahman and A. Rani Saha, "Automated teller machine card fraud of financial organizations in Bangladesh," *J. Comput. Sci. Appl. Inf. Technol.*, vol. 312, no. 12, pp. 34–49, Mar. 2018.

[6] J. O. Adeoti, "Automated teller machine (ATM) frauds in nigeria: The way out," *J. Social Sci.*, vol. 27, no. 1, pp. 53–58, Apr. 2011.

[7] O. McEvoy. (2023). *Number of ATM-Related Malware and Hacking Attacks Reported in European Countries From 2014 To 2020*. Accessed: Jul. 11, 2023. [Online]. Available: https://www.statista.com/statistics/707943/attacks-atm-hacking-malware-europe/

[8] L. Garrett. (2022). *'Brazen' ATM Attacks Across U.K. Lead Police to Make 22 Arrests Including in Leicestershire*. Accessed: Jul. 11, 2023. [Online]. Available: https://www.leicestermercury.co.uk/news/local-news/brazen-atm-attacks-across-uk-7833515

[9] D. Tente. (2023). *Industry Concern Over ATM Crime is Peaking*. Accessed: Jan. 11, 2024. [Online]. Available: https://www.atmia.com/news/industry-concern-over-atm-crime-is-peaking/20690/

[10] O. McEvoy. (2023). *Losses Incurred During ATM Related Fraud Attacks (incidents) in Selected European Countries From 2010 to 2019*. Accessed: Jul. 12, 2023. [Online]. Available: https://www.statista.com/statistics/419739/fraud-attacks-atm-losses-in-europe/

[11] Federal Bureau Invest. *Bank Crime Statistics*. Accessed: Jul. 11, 2023. [Online]. Available: https://www.fbi.gov/investigate/violent-crime/bank-robbery/bank-crime-reports

[12] M. S. Ferdous, "User-controlled identity management systems using mobile devices," Ph.D. dissertation, School Comput. Sci., College Sci. Eng., Univ. Glasgow, Glasgow, U.K., 2015.

[13] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. New York, NY, USA: Manning Publications, 2021.

[14] M. S. Ferdous, F. Chowdhury, and M. O. Alassafi, "In search of self-sovereign identity leveraging blockchain technology," *IEEE Access*, vol. 7, pp. 103059–103079, 2019.

[15] M. Farhad, G. Saha, M. A. Nahid, F. R. Chowdhury, P. P. Paul, M. R. Ullah, and M. S. Ferdous, "Secure backup and recovery of SSI wallets using solid pod technology," in *Proc. IEEE 48th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2024, pp. 1101–1111.

[16] M. A. Rahman and X. Qi, "Core banking software (CBS) implementation challenges of e-banking: An exploratory study on Bangladeshi banks," *J. Administ. Bus. Stud.*, vol. 2, no. 4, pp. 208–215, 2016.

[17] ATMIA. *ATM Golden Anniversary Celebration at ATMIA U.S. Conference in Orlando*. Accessed: Feb. 19, 2024. [Online]. Available: https://www.atmia.com/news/atm-golden-anniversary-celebration-at-atmia-us-conference-in-orlando/4519/

[18] National ATM Systems. *How Do ATMs Work*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.nasatm.com/pages/how-do-atms-work

[19] ISO. *ISO8583 Messaging Standard*. Accessed: Feb. 5, 2024. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso:8583:ed-3:v1:en

[20] R. Gyoery and J. Seberry, "Electronic funds transfer point of sale in Australia," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, 1986, pp. 347–377, doi: 10.1007/3-540-47721-7_26.

[21] IR Team. *An Introduction to ISO 8583: What You Need to Know*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.ir.com/guides/introduction-to-iso-8583

[22] atmmarketplace. *(Not) Lost in Translation*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.atmmarketplace.com/articles/not-lost-in-translation/

[23] ISO. *ISO20022 Messaging Standard*. Accessed: Feb. 5, 2024. [Online]. Available: https://www.iso.org/obp/ui/en/#iso: std:57341: en

[24] IR Team. *ISO 20022 Vs ISO 8583: What is the Difference?*. Accessed: Jul. 15, 2024. [Online]. Available: https://www.ir.com/guides/iso-20022-vs-iso-8583

[25] M. Hossain and R. Bari, "Unerstanding of ATM (automated teller machine) in Bangladesh," Ph.D. dissertation, Dept. Comput. Sci. Eng., BRAC Univ., Dhaka, Bangladesh, 2006.

[26] H. E. Feky and M. Ghantous, "Touchless secure QR-based ATM system," in *Proc. Int. Conf. Adv. Intell. Syst. Inform.*, 2022, pp. 321–332.

[27] S. S. Agrawal, P. Oza, M. Biswas, and N. Choksi, "Enhanced secure ATM authentication using NFC technology and iris verification," *Scalable Comput., Pract. Exper.*, vol. 22, no. 2, pp. 273–282, Nov. 2021.

[28] C. M. Manish, N. Chirag, H. R. Praveen, M. J. Darshan, and D. K. Vali, "Card-less ATM transaction using biometric and face recognition—A review," *Int. J. Sci. Eng. Res.*, vol. 11, no. 1, pp. 1393–1400, 2020.

[29] M. A. Imran, M. F. Mridha, and M. K. Nur, "OTP based cardless transction using ATM," in *Proc. Int. Conf. Robot., Elect. Signal Process. Techn. (ICREST)*, Jan. 2019, pp. 511–516.

[30] M. S. Ferdous, G. Norman, and R. Poet, "Mathematical modelling of identity, identity management and other related topics," in *Proc. 7th Int. Conf. Secur. Inf. Netw.*, Sep. 2014, pp. 9–16.

[31] A. Jøsang and S. Pope, "User centric identity management," in *Proc. AusCERT Asia Pacific Inf. Technol. Secur. Conf.*, May 2005, pp. 77–89.

[32] Guevara, Holly. (2022). *How SAML Authentication Works*. Accessed: Jul. 15, 2022. [Online]. Available: https://auth0.com/blog/how-saml-authentication-works/

[33] OpenID. *How OpenID Connect Works*. Accessed: Jul. 15, 2022. [Online]. Available: https://openid.net/developers/how-connect-works/

[34] OAuth. *What is OAuth*. Accessed: Jul. 15, 2022. [Online]. Available: https://oauth.net/

[35] M. Alizadeh, K. Andersson, and O. Schelén, "Comparative analysis of decentralized identity approaches," *IEEE Access*, vol. 10, pp. 92273–92283, 2022.

[36] The Kubernetes Authors. *Kubernetes: Production-Grade Container Orchestration*. Accessed: Jul. 28, 2025. [Online]. Available: https://kubernetes.io/

[37] Trust Over IP Foundation. *Technical Architecture*. Accessed: Jul. 28, 2025. [Online]. Available: https://trustoverip.org/our-work/technical-architecture/

[38] W3C. *Verifiable Credentials Data Integrity 1.0*. Accessed: Jul. 28, 2025. [Online]. Available: https://www.w3.org/TR/vc-data-integrity/

[39] W3C Credentials Community Group. *BBS+ Signature Suite*. Accessed: Jul. 28, 2025. [Online]. Available: https://www.w3.org/TR/vc-di-bbs/

[40] The Hyperledger Foundation. *AnonCreds Specification V1.0*. Accessed: Jul. 28, 2025. [Online]. Available: https://hyperledger.github.io/anoncreds-spec/

[41] W3 Team. *Decentralized Identifiers (DIDs)*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.w3.org/TR/did-core/

[42] W3 Team. *Verifiable Credentials Data Model*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[43] Decentralized Identity Foundation. *DIDComm Messaging V2.1*. Accessed: Jul. 20, 2025. [Online]. Available: https://identity.foundation/didcomm-messaging/spec/v2.1/

[44] Drake, Victoria. (2023). *Threat Modeling*. Accessed: Jul. 12, 2023. [Online]. Available: https://owasp.org/www-community/Threat_Modeling

[45] Y. Weixiong, K. Dozono, R. Lee, A. K. S. Seng, and F. tuz Zahra, "Securing software systems—A survey," *TechRxiv*, May 2020, doi: 10.36227/techrxiv.12319598.v1.

[46] A. Shostack, *Threat Modeling: Designing for Security*. Hoboken, NJ, USA: Wiley, 2014.

[47] T. UcedaVelez and M. M. Morana, *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. Hoboken, NJ, USA: Wiley, 2015.

[48] *Threat Modeling Methodologies: What is Vast?*. Accessed: Jul. 12, 2023. [Online]. Available: https://threatmodeler.com/threat-modeling-methodologies-vast/

[49] *What is MongoDB?*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.ibm.com/topics/mongodb

[50] IBM. *What is Redis?*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.ibm.com/topics/redis

[51] Linux Foundation Decentralized Trust. *Hyperledger Aries*. Accessed: Jun. 10, 2025. [Online]. Available: https://www.lfdecentralizedtrust.org/projects/aries

[52] Aries Cloud Agent-Python. *Aries Cloud Agent Python Code Documentation*. Accessed: Jul. 15, 2022. [Online]. Available: https://aries-cloudagent-python.readthedocs.io/en/latest/

[53] Linux Foundation Decentralized Trust. *Hyperledger Indy*. Accessed: Jun. 10, 2025. [Online]. Available: https://www.lfdecentralizedtrust.org/projects/hyperledger-indy

[54] Indicio. *Indicio Mainnet*. Accessed: Jul. 13, 2023. [Online]. Available: https://indicio.tech/indicio-mainnet/

[55] Indicio. *Indicio Public Mediator: How to Connect*. Accessed: Jul. 13, 2023. [Online]. Available: https://indicio-tech.github.io/mediator/

[56] Sovrin. *A Global Utility for Self-Sovereign Identity*. Accessed: Jun. 11, 2022. [Online]. Available: https://github.com/sovrin-foundation/sovrin

[57] *A Portable Development Level Indy Node Network*. Accessed: Jun. 11, 2022. [Online]. Available: https://github.com/bcgov/von-network

[58] Linux Foundation Decentralized Trust. *How Credential Revocation Works*. Accessed: Jul. 23, 2023. [Online]. Available: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

[59] BCGov. *Indy Tail Server*. Accessed: Jul. 23, 2023. [Online]. Available: https://github.com/bcgov/indy-tails-server

[60] OpenWallet Found. (2025). *Askar: Secure Storage and Key Management Library*. [Online]. Available: https://github.com/openwallet-foundation/askar

[61] NodeJs. *About Node.js*. Accessed: Jun. 11, 2022. [Online]. Available: https://nodejs.org/en/about

[62] *Hyperledger Fabric*. Accessed: Jun. 10, 2025. [Online]. Available: https://www.lfdecentralizedtrust.org/projects/fabric

[63] Linux Foundation Decentralized Trust. *Aries Mobile Agent React Native—Part of the Aries Bifold Effort to Provide SSI Capabilities in a Production Ready App*. Accessed: Jun. 11, 2022. [Online]. Available: https://github.com/hyperledger/aries-mobile-agent-react-native

[64] M. A. Hannan, M. A. Shahriar, M. S. Ferdous, M. J. M. Chowdhury, and M. S. Rahman, "A systematic literature review of blockchain-based e-KYC systems," *Computing*, vol. 105, no. 10, pp. 1–30, Oct. 2023.

[65] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.

[66] A. Dresch, D. P. Lacerda, and J. A. V. Antunes Jr., "Design science research," in *Design Science Research: A Method for Science and Technology Advancement*. Cham, Switzerland: Springer, 2014, pp. 67–102.

[67] Decentralized Identity Foundation. *Peer DID Method Specification*. Accessed: Feb. 24, 2024. [Online]. Available: https://identity.foundation/peer-did-method-spec/

[68] Aries RFC 0434. *Out-of-Band Protocol 1.1*. Accessed: Feb. 19, 2024. [Online]. Available: https://github.com/hyperledger/aries-rfcs/blob/main/features/0434-outofband/README.md

[69] PNC. *PNC Card Free Access*. Accessed: Jul. 13, 2023. [Online]. Available: https://www.pnc.com/en/personal-banking/banking/online-and-mobile-banking/atm-banking.html

[70] IslamiBankBD. *Cash by Code*. Accessed: Jul. 13, 2023. [Online]. Available: https://www.islamibankbd.com/digital-banking/cash-by-code

[71] City Bank. *Send Money To Anyone Using*. Accessed: Jul. 13, 2023. [Online]. Available: https://www.thecitybank.com/citytouch

[72] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, *ProVerif 2.00: Automatic Cryptographic Protocol Verifier—User Manual and Tutorial*. Paris, France: INRIA Paris-Rocquencourt, May 2018, pp. 5–16.

[73] A. R. Lyon, J. Coifman, H. Cook, E. McRee, F. F. Liu, K. Ludwig, S. Dorsey, K. Koerner, S. A. Munson, and E. McCauley, "The cognitive walkthrough for implementation strategies (CWIS): A pragmatic method for assessing implementation strategy usability," *Implement. Sci. Commun.*, vol. 2, no. 1, pp. 1–16, Dec. 2021.

[74] T. Mahatody, M. Sagar, and C. Kolski, "Cognitive walkthrough for HCI evaluation: Basic concepts, evolutions and variants, research issues," in *Proc. EAM Eur. Annu. Conf. Hum.-Decis. Making Manual Control*, 2007, pp. 1–12.

[75] J. Clark, P. C. van Oorschot, and C. Adams, "Usability of anonymous web browsing: An examination of tor interfaces and deployability," in *Proc. 3rd Symp. Usable Privacy Secur.*, Jul. 2007, pp. 41–51.

[76] J. Nielsen. *How Many Test Users in a Usability Study?*. Accessed: Feb. 24, 2024. [Online]. Available: https://www.nngroup.com/articles/how-many-test-users/

[77] W. A. Rogers, A. D. Fisk, S. E. Mead, N. Walker, and E. F. Cabrera, "Training older adults to use automatic teller machines," *Hum. Factors, J. Hum. Factors Ergonom. Soc.*, vol. 38, no. 3, pp. 425–433, Sep. 1996.

[78] F. A. Salman and Z. A. Bakar, "Do older adults adopt the new technologies? ATM interfaces are an example," *Rev. Comput. Eng. Stud.*, vol. 11, no. 3, pp. 30–38, Sep. 2024.

[79] W. A. Rogers, E. F. Cabrera, N. Walker, D. K. Gilbert, and A. D. Fisk, "A survey of automatic teller machine usage across the adult life span," *Hum. Factors, J. Hum. Factors Ergonom. Soc.*, vol. 38, no. 1, pp. 156–166, Mar. 1996.

[80] G. Kondova and J. Erbguth, "Self-sovereign identity on public blockchains and the GDPR," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 342–345.

[81] X. Sun, J. Yang, F. Yang, and C. Li, "A self-sovereign identity authentication scheme for multi-level supply chain finance," in *Proc. 7th Int. Conf. Artif. Intell. Pattern Recognit.*, Sep. 2024, pp. 834–840.

[82] A. A. Ajayi, I. Emmanuel, A. D. Soyele, and J. Enyejo, "Enhancing digital identity and financial security in decentralized finance (DeFi) through zero-knowledge proofs (ZKPs) and blockchain solutions for regulatory compliance and privacy," *Iconic Res. Eng. Journal*, vol. 8, no. 4, pp. 373–394, 2024.

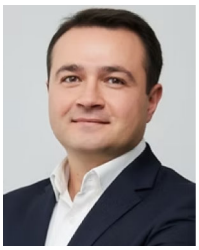
[83] H.-H. Ou, G.-Y. Chen, and I.-C. Lin, "A self-sovereign identity blockchain framework for access control and transparency in financial institutions," *Cryptography*, vol. 9, no. 1, p. 9, Jan. 2025.
[84] A. N. Rahman, B. S. Hantono, and G. D. Putra, "TruChain: A multi-layer architecture for trusted, verifiable, and immutable open banking data," 2025, *arXiv:2507.08286*.
[85] K. A. M. Ahmed, S. F. Saraya, J. F. Wanis, and A. M. T. Ali-Eldin, "A blockchain self-sovereign identity for open banking secured by the customer's banking cards," *Future Internet*, vol. 15, no. 6, p. 208, Jun. 2023.
[86] M. R. Ahmed, A. K. M. M. Islam, S. Shatabda, and S. Islam, "Blockchain-based identity management system and self-sovereign identity ecosystem: A comprehensive survey," *IEEE Access*, vol. 10, pp. 113436–113481, 2022.
[87] S. Koteswari, P. J. Paul, A. Dheeraj, and R. Kone, "Fusion of iris and fingerprint biometric identifier for ATM services: An investigative study," *Int. J. Commun., Netw. Syst. Sci.*, vol. 9, no. 11, pp. 506–518, 2016.
[88] I. Susan. C, M. Sanusi, and E. C. Ubaka, "Secure atm and bank transactions using biometric and blockchain," *Int. J. Adv. Sci. Res. Eng.*, vol. 5, no. 10, pp. 301–313, 2019, doi: 10.31695/ijasre.2019.33580.
[89] E. Bandara, X. Liang, P. Foytik, S. Shetty, N. Ranasinghe, K. De Zoysa, and W. K. Ng, "Promize-blockchain and self sovereign identity empowered mobile ATM platform," in *Proc. Intell. Comput.*, 2021, pp. 891–911.

FAIRUZ RAHAMAN CHOWDHURY received the B.Sc. (Engineering) degree in computer science and engineering from the Shahjalal University of Science and Technology, Bangladesh. He is currently a Research Engineer with Cryptic Consultancy Ltd., U.K. He focuses on blockchain systems, decentralised identity, and advanced security. He enjoys exploring different areas and finding the best solutions.

MD MASUM ALAM NAHID received the Bachelor of Science degree in computer science and engineering from the Shahjalal University of Science and Technology, Bangladesh. He is currently a Research Engineer with Cryptic Consultancy Ltd., U.K. His research interests span a wide spectrum, with a particular focus on blockchain systems, self-sovereign identity, decentralised identity, and security.

FARIDA CHOWDHURY received the joint master's degree in networking and e-business centred computing from the University of Reading, U.K., the Universidad Carlos III de Madrid, Spain, and the Aristotle University of Thessaloniki, Greece, and the Ph.D. degree from the University of Stirling, U.K., on peer-to-peer networking. She is currently a Professor with the Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh. Previously, she was a Professor of computer science and engineering with the Shahjalal University of Science and Technology, Sylhet, Bangladesh. She has published many papers in different journals, conferences, workshops, and symposiums. Her current research interests include the intersection of usability, security, and privacy.

MD MASUM is currently a Professor with the Computer Science and Engineering Department, Shahjalal University of Science and Technology, Sylhet, Bangladesh. He is also the Director of the Computer and Information Center, Shahjalal University of Science and Technology. He has published many research papers in different journals and conferences. His research interests include security, computer networking, and mobile computing.



UMIT CALI (Senior Member, IEEE) received the master's and Ph.D. degrees in electrical engineering and computer science from the University of Kassel, Germany. He is currently a Professor and the Chair in digital engineering for future technologies of the University of York, U.K. He is also a part-time Professor of energy informatics with NTNU, Norway. His career includes roles at IBM International, as a Network Engineer, and Fraunhofer Institute, as a Senior Researcher. He has also held leadership positions, such as the CTO of KREEN Renewables GmbH and a Manager with EnBW, Germany. He has co-founded multiple high-tech startups and is actively involved with IEEE, holding senior roles in energy and blockchain initiatives. His research interests include cyber-physical-social systems, energy informatics, artificial intelligence, blockchain technology, digital twins, and cyber law.

MD SADEK FERDOUS (Member, IEEE) received the double master's degree in security and mobile computing from the Norwegian University of Science and Technology, Norway, and the University of Tartu, Estonia, and the Ph.D. degree in identity management from The University of Glasgow. He is currently a Professor with the Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh. He is also an Honorary Research Fellow with the Institute for Security Science and Technology, Imperial College London, and a Research Associate with the Centre for Financial Technology, Imperial College Business School, London. He is a former ERCIM Alain Bensoussan Fellow with the Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany. He has several years of experience working as a Postdoctoral Researcher in different universities in different European and U.K.-funded research projects. He has published numerous research papers and book chapters in these domains in different books, journals, conferences, workshops, and symposiums. His current research interests include blockchain, self-sovereign identity, identity management, trust management, and security and privacy issues.