Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Payments in openCBDC with Self-Sovereign Identitites – from the verifiable to the private

Scientific Students' Association Report

Author:

Martin Farkas

Advisor:

dr. Imre Kocsis

2022

Contents

Kivonat i

Abstract ii

1 Introduction 1

2 Background 3

2.1 Central Bank Digital Currencies 3

2.1.1 Motivations for CBDC development 4

2.1.2 Design choices 5

2.1.3 Architecture 5

2.1.4 Infrastructure 6

2.1.5 Access Technology 7

2.2 OpenCBDC 8

2.2.1 Core design features 8

2.2.2 Threat model and security properties 10

2.2.3 Client 10

2.3 Self-Sovereign Identity 11

2.3.1 ToIP protocol stack 11

2.3.1.1 Verifiable Credentials 11

2.3.1.2 Digital Agents 12

2.3.1.3 Decentralized Identifiers - DIDs 13

2.3.2 Axes of identity 13

2.4 The Hyperledger Aries/Indy/Ursa stack 14

2.4.1 Hyperledger Ursa 14

2.4.2 Hyperledger Indy 14

2.4.3 Hyperledger Aries 14

2.4.4 Aries Cloudagent Python 14

3 Motivation 16

3.1 Self-Sovereign Identity and Central Bank Digital Currency 16
3.1.1 A secure and long-term identity layer of a CBDC 16
3.1.2 Aiding policy decisions 16
3.1.3 Emerging technologies around SSI 18
3.2 Open ecosystem 18
3.2.1 Benefits of a CBDC ecosystem 18
3.2.2 Generalisability of a private cash-like CBDC 19

4 Maximum and minimum CBDC privacy **20**
4.1 Spectrum of privacy 20
4.1.1 The needs of the public 20
4.1.2 The responsibilities of the Central Bank 20
4.2 Privacy Classes 21
4.2.1 Better privacy with SSI 21
4.2.1.1 Roles of the Central Bank 21
4.3 Payment over openCBDC with Hyperledger Aries 22
4.4 Privacy goals 22

5 Auditable payment protocol **24**
5.1 Actors and Components 24
5.2 Preparation to issue Verifiable Credentials 24
5.3 Opening openCBDC accounts 25
5.4 Facilitating transactions 25
5.5 Implementation in development 28

6 Private Payment protocol **29**
6.1 Notation, and terminology 29
6.2 Actors and Components 29
6.3 Data structures 30
6.3.1 Data kept by the citizen 30
6.3.2 Data maintained at the CB 30
6.3.2.1 Data structure associated with each citizen VC 30
6.3.2.2 Data kept by only the CB 30
6.3.2.3 Data shared between the sentinels and the CB: 30
6.3.3 Data used during transactions 30
6.4 Prerequisites for a receiving spend- and receive authorizations 31
6.5 Preparation for a transaction 31
6.6 Facilitaiting transactions 32

6.7 Updating the commitments after a transaction 34

7 Results and Discussion **36**

8 Conclusion and Future Work **38**

Bibliography **39**

Kivonat

Az elmúlt években a világ központi bankjai a Digitális Jegybak Pénzek (CBDC) lehetséges bevezetését kutatták; digitális valutákét, amelyeket közvetlenül a központi bank fedez. A CBDC modellek és kiszolgálási architektúrák széles skáláját vizsgálták meg, a klasszikus bankszámla jellegű digitális pénztől a Bitcoin-stílusú pénzügyi eszközökig, valamint a központosított megoldásoktól a decentralizált rendszerekig. A nyilvánosságra hozott kísérletek nagy száma ellenére azonban a nyílt forráskódú prototípusok rendkívül ritkák, valamint az identitáskezelés és a titoktartás kulcskérdései többnyire nyitva maradnak, egyszerű hivatkozással arra, hogy ezek a jegybank politikai döntéseitől függenek. Ugyanakkor a digitális személyazonosság-kezelés saját forradalmán megy keresztül. Az olyan kezdeményezések, mint a Self-Sovereign Identities (SSI) és a Decentralized Identities azt a közös elképzelést vallják, hogy egy személy sokféle identitását valamilyen személyes „pénztárcához” kell kötni, és erre kell hitelesítő okiratokat kiadni. Az elosztott főkönyvek fontos szerepet játszanak ezen megközelítésekben. Azt azonban csak nemrég ismerték fel, hogy az SSI-nak a CBDC biztonságának, auditálhatóságának és adatvédelmének fontos elemét kell képeznie – a konkrét lehetséges megközelítésekre vonatkozó elemzések egyelőre gyakorlatilag nem léteznek. Ebben a munkában az első hozzájárulásom azoknak a fizetési identitás-támogató képességeknek a feltérképezése és prototípusának elkészítése, amelyeket a vezető SSI technológiai halmaz, a Hyperledger Aries és az Indy hozhat a gyakorlatilag egyetlen nyílt forráskódú CBDC fizetési rendszerhez, az MIT Digital Currency Initiative-jából származó openCBDC-hez. Az implementált funkciók közé tartozik az „Ismerd meg ügyfeled” követelmények támogatása, valamint a hitelesítés és a fizetésre való authorizáció. A második pedig, hogy bár a CBDC-k titoktartást tekintve általában nem készpénzszerűek – mivel digitális eszközökről van szó –, el kell ismerni a közvélemény igényét a készpénz által biztosított titoktartás szintjére. Javaslom, hogy egy speciális CBDC eszköz egy teljes CBDC ökoszisztémában, amely mesterséges készpénzszerű „terhekkel” rendelkezik – mint például a maximálisan tartható összeg, a pénz alacsony sebessége, valamint a fizető és a kedvezményezett fizikai közelsége – elősegíthetné a készpénzszerű titoktartást. Ebből a célból, egy SSI protokol rendszerre támaszkodva, egy fizetési protokollt és egy prototípust készítek az openCBD-hez, amelyet a Chaum-i DigiCash ihletett, amely egyrészt lehetővé teszi a felhasználók számára, hogy tetszőleges számú Bitcoin-szerű pszeudonímát használjanak a Bitcoinban ismert adatvédelmi szint mellett., másrészt biztosítja ezen (paraméterezhető) „terhek” működését, amelyek a készpénz fizikai tulajdonságait hivatottak emulálni.

Abstract

In recent years, central banks around the world have been researching the possible introduction of Central Bank Digital Currencies (CBDCs); digital currencies which are directly backed by the central bank. A wide range of CBDC models and hosting architectures have been explored, from classic bank account like money to Bitcoin-style bearer instruments, and from centralized solutions to compositions of decentralized systems. However, despite the large number of publicized experiments, open prototypes are extremely scarce and key questions of identity handling and privacy are mostly left open with a simple reference of these being dependent on central bank policy decisions. At the same time, digital identity handling is undergoing its own revolution. With markedly different emphases, "flavors" and relationships, but initiatives as Self-Sovereign Identities (SSI) and Decentralized Identities share the common idea that the many identities of a person should be tied to personal "wallets" and credentials should be issued against these. Decentralized ledgers form an important part in securing these approaches. However, it has only been recognized that SSI should form an important component in CBDC security, auditability, and privacy – as of now, analyses on the specific possible approaches are practically nonexistent. In this work, my first contribution is mapping out and prototyping the payment identity support capabilities that the leading SSI technology stack, Hyperledger Aries and Indy can bring to the practically only open source CBDC payment processor solution, openCBDC from the Digital Currency Initiative of MIT. Importantly, the use cases include Know-Your-Customer requirement support and authentication and authorization for payment. Second, while CBDCs will not generally be cash-like in their privacy – as they are digital assets – the need of the public for the level of privacy provided by cash will have to be recognized. I propose that a special CBDC asset in a full CBDC ecosystem, which has artificial cash-like encumbrances – as maximum holdable amount, low speed of money and physical proximity of payer and payee – could be allowed to function with cash-like privacy. To this end, drawing on the presence of an SSI stack, I propose and prototype a payment protocol for openCBDC, inspired by Chaumian DigiCash, which on the one hand enables users to utilize any number of Bitcoin-like pseudonyms with the default level of privacy in Bitcoin, and on the other hand, ensures the presence of these (parameterizable) "encumbrances", which mimic the physical properties of cash.

Chapter 1

Introduction

After the preliminary research and experimental evaluation ramping up in recent years, it is expected that a many central banks around the world will issue Central Bank Digital Currencies (CBDC) and make them available for pubolic use. It is likely that identification and authorization funtions related to CBDC will be ralized with Self-Sovereign Iden (SSI), which in themselves represent a major, ongoing wave of innovation in Information Technology. Meanwhile, innovation based on CBDC managing smart contracts and workflows is seriously hampered by the fact that freely accessible CBDC prototypes are scarce and often only exist as proof-of-concept software. An important development at the beginning of 2022 was that MIT (in collaboration with the Boston FED) made a preliminary CBDC system public, called openCBDC. Analysing the capabilities of, and design possibilities regarding such a system is crucial in pushing forward the possibilities of CBDCs.

Two key issues in CBDC design are performance and privacy, into which a marriage of SSI solutions and a publicly available CBDC prototype would surely grant insight.

Surveying existing research, there is yet to be any analysis of adding SSI-based management capabilities to openCBDC, and research regarding the use of SSI in closed-source CBDC solutions is scarce.

In this paper, as SSI has been identified as a point of interest in CBDC development, and numerous use cases have been identified where SSI may extend the capabilities of payment workflows, I put forward the following contributions with the goal of getting a clearer picture how SSI would help CBDC development.

I gave a structured argument for the benefits of using SSI technologies as the identity layer of future CBDC schemes. Such are surprisingly missing from the relevant literature, while SSI is being noted as an option to support.

I've proposed and prototyped an SSI-based fully auditable payment scheme for the openCBDC transaction processor prototype from MIT and The FED Boston.

I've proposed a private payment scheme over openCBDC which relies on Self-Sovereign Identities, draws on ideas from Chaumian Digicash, and enables considering Bitcoin equivalent privacy in a subset of openCBDC transactions by enforcing "encumbrances" on money flows, which imitate the behaviour of physical cash.

The paper is organised as follows: In Chaper 2 I introduce concepts and technologies on which the payment schemes will be based on.

First, I summarize existing research regarding a Central Bank's motivation for and possible design choices in the development of a Central Bank Digital Currency. Then I highlight the advantages of openCBDC, and enumerate the components of the system my schemes will interact with.

After, I introduce the key principles of Self-Sovereign identity, with focus on the ToIP protocol stack. Lastly, I present an existing implementation of an SSI stack, the Hyperledger Indy/Ursa/Aries stack.

In Chapter 3 I summarize the motivations a Central Bank may have for implementing an SSI based identity layer over an CBDC.

In Chapter 4 I explore and argue for a specific design model of payment schemes over a CBDC. I designate two possible design stances within this model. These will become my payment schemes.

In Chapter 5 I present my proposal and prototype for a fully auditable payment scheme utilising SSI technologies.

Then Chapter 6 I propose and define a privacy-preserving payment scheme, inspired by Digicash, enabling for the enforcement of "cash-like encumbrances".

Lastly, in Chapert 7 I discuss the results of my findings.

Chapter 2

Background

2.1 Central Bank Digital Currencies

Central Bank Digital Currencies are "a digital form of central bank money that is different from balances in traditional reserve or settlement accounts." [1] They are a digital form of fiat currency, denominated in the national unit of account, that is a direct liability of the Central Bank [2] ,

Their development, along with other forms of digital money, has gone on for decades. Still, in recent years - partly due to the popularity of cryptocurrencies - the introduction of a general-purpose CBDC in the near future seems more and more like a possibility.

While conceptual research around CBDCs is abundant, the technology is still very much in the developing stage. Many nations have begun conceptual research, with proof-of-concept and pilot work ramping up in 2021. [3] While reports are widely available on a large subset of proof-of-concept and pilot systems - such as e-CNY [4] –, they are yet to launch in a production setting and mostly remain closed source, with some notable exceptions, including openCBDC[5]. It is difficult to reproduce their findings, confirm their claims on resilience, robustness, and performance - key points of interest in developing a CBDC -or integrate them into payment processing systems.

The following sections explore the motivations, possible models, and design choices of CBDC - as laid out by *Auer and Böhme, 2020* [6]. My protocol's design considerations are closely tied to ones made in the underlying CBDC, mainly in terms of architecture and access technology.

2.1.1 Motivations for CBDC development

While motivations for Central Banks to adopt CBDCs are numerous [2], from transaction speed to cross-border payments, I will focus on ones relevant to my paper.

- **Financial inclusion** CBDCs may provide lower cost and larger access alternatives to Retail Bank money and cash in under-banked and cash-lacking areas of a country, regardless of the user's identity.
- **Security and Resilience** With a well-developed, robust payment system that functions 24/7 may serve as a backup in the event of bank outages or cash shortages. If the CBDC supports offline payments, the system may function through sporadic electricity outages.
- **Auditability** The digital nature of CBDCs allows greater and finer control for policymakers to enforce anti-money laundering and anti-terrorism laws and detect other illegal activities. With multiple methods of payment, or even multiple CBDCs within a financial ecosystem, with different levels of traceability and privacy, individuals may choose if they want cash-like privacy or the convenience of retail bank transfer-like payments.
- **Integration into digital payment processing systems** CBDCs being digital assets, they offer the opportunity to be included in digital workflows. Compared to cash, they offer much greater speed and ease of transfer; and compared to digital retail bank money, they provide a government-backed alternative to the numerous already existing payment processor platforms, which are impeded by the diversity of used technological frameworks, and the policies of the underlying retail banks - for example, bank holidays [7] .
- **Privacy** A public need has been identified for private, untraceable, anonymous digital payment systems[2, 6, 1, 8], well illustrated by the success of cryptocurrencies like Monero [9] `https://coinmarketcap.com/hu/view/privacy/`. If a Central Bank would be to introduce such a payment system, it would be much easier to regulate than private solutions and may provide functionality that falls in line with the intent of policymakers

As CBDCs will be part of an existing financial ecosystem, they will take a peculiar place alongside existing financial instruments, a subset of which is illustrated by Fig. 2.1. Wholesale CBDCs are like current digital Central bank reserves, as they are only accessible to commercial banks, unlike retail CBDCs, which may be universally accessible to all types of consumers, regardless of their identity.

Figure 2.1: Types of money, Bech and Garratt 2017 [10] Graph 3

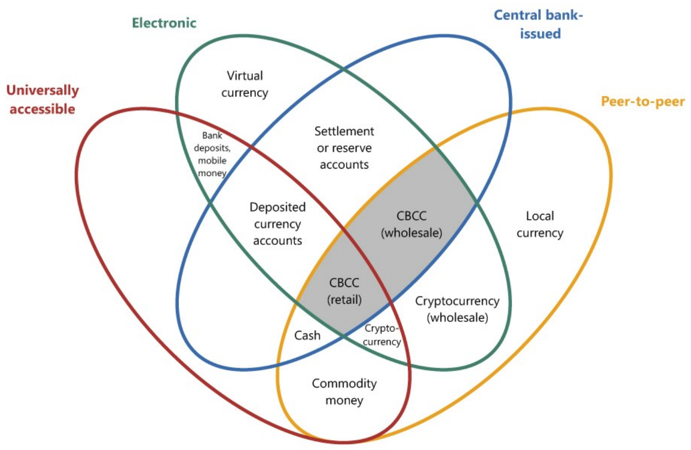
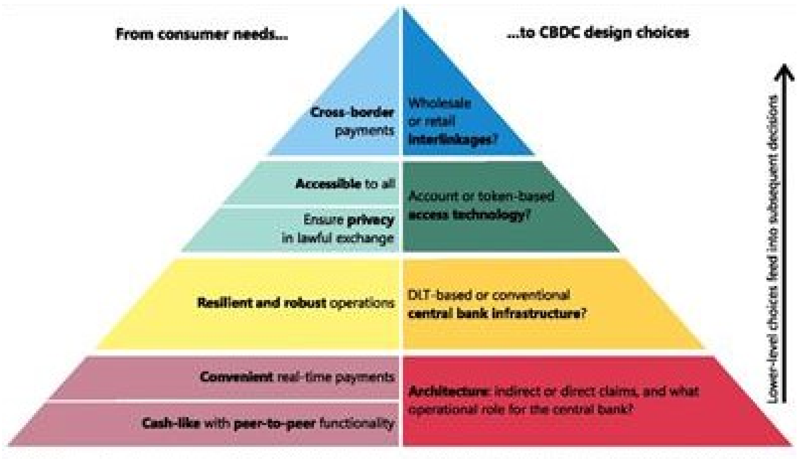


Figure 2.2: CBDC Pyramid, BIS 2020 [6] Graph 1


From consumer needs...
...to CBDC design choices
Cross-border payments
Wholesale or retail interlinkages?
Accessible to all
Ensure privacy in lawful exchange
Account or token-based access technology?
Resilient and robust operations
DLT-based or conventional central bank infrastructure?
Convenient real-time payments
Cash-like with peer-to-peer functionality
Architecture: indirect or direct claims, and what operational role for the central bank?
Lower-level choices feed into subsequent decisions

The CBDC pyramid maps consumer needs (left-hand side) onto the associated design choices for the central bank (right-hand side). The four layers of the right-hand side form a hierarchy in which the lower layers represent design choices that feed into subsequent, higher-level decisions.

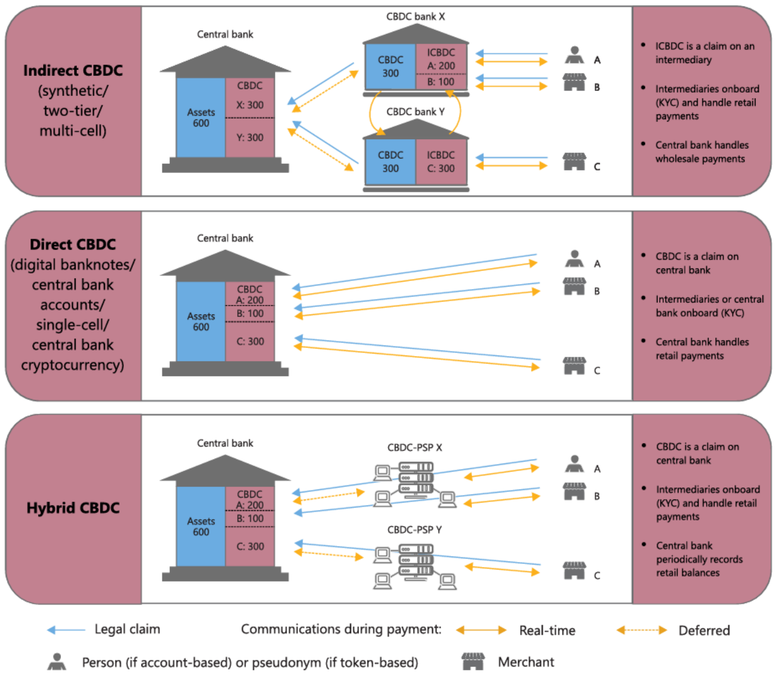
2.1.2 Design choices

To identify relevant design choices in the development of a retail CBDCs, BIS [6] has developed a framework that focuses on consumer needs, and identifies the dependencies across these design choices, the CBDC pyramid Fig. 2.2. The following few sections explore the relevant levels of the pyramid

2.1.3 Architecture

CBDCS, being a form of digital money, must keep track of the owner of the funds, the claim on the Central bank. The sum of these records is conceptualized as a ledger of transactions or a registry of account balances. While many CBDCs use these kinds of storage frameworks, an exception to be noted here is openCBDC, which stores only a representation - the hash - of the unspent funds. Still, the following architectures not only show how the claims may be technically organised, but how they may be legally organized. These, in order, as illustrated by 2.3, are:

Figure 2.3: CBDC models, BIS 2020 [6] Graph 2


Indirect CBDC
(synthetic/
two-tier/
multi-cell)
Central bank
Assets 600
CBDC
X: 300
Y: 300
CBDC bank X
CBDC 300
ICBDC
A: 200
B: 100
CBDC bank Y
CBDC 300
ICBDC
C: 300
A
B
C
ICBDC is a claim on an intermediary
Intermediaries onboard (KYC) and handle retail payments
Central bank handles wholesale payments
Direct CBDC
(digital banknotes/
central bank
accounts/
single-cell/
central bank
cryptocurrency)
Central bank
Assets 600
CBDC
A: 200
B: 100
C: 300
CBDC is a claim on central bank
Intermediaries or central bank onboard (KYC)
Central bank handles retail payments
Hybrid CBDC
Central bank
Assets 600
CBDC
A: 200
B: 100
C: 300
CBDC-PSP X
CBDC-PSP Y
CBDC is a claim on central bank
Intermediaries onboard (KYC) and handle retail payments
Central bank periodically records retail balances
Legal claim
Communications during payment:
Real-time
Deferred
Person (if account-based) or pseudonym (if token-based)
Merchant

- **Indirect** Consumers do not directly claim CBDC through the CB but rather through intermediaries, denoted CBDC Banks in the figure. These are also called "Two-tier" or "Indirect" CBDCs
- **Direct** They are no intermediaries involved, the consumers have a direct claim on the Central bank, it handles KYC
- **Hybrid** Consumers have a direct claim on the Central Bank, but transactions and KYC are handled through intermediaries.

2.1.4 Infrastructure

We may compare "conventional" and DLT solutions, as Auer and Böhme, 2020 [6] did, in ways that will be relevant when we look at the threat model of both openCBDC and our protocol. As for now, keep in mind that openCBDC uses a Raft-based storage solution, that is, it doesn't provide blockchain-like Byzantine fault tolerance, and achieves much larger performance than its DLT-based contemporaries.

This difference in performance is due to the way the two systems update their registries. While on a conventional ledger, you only need to interact with the leading node, the authoritative entity, in a DLT-based ledger, the consensus mechanism - through which updates have harmonized all nodes - represents a large overhead.

The use of Distributed Ledger Technology (DLT - Distributed Ledger Technology) can be a solution to increase **robustness**. Since the traditional infrastructure also stores data in several physical locations, the main issue is the distribution of authority for data modification. Because if the authoritative entity (e.g. leading node) is compromised in the case of a traditional infrastructure, it is much easier to achieve malicious effects than if it were one node among many in a DLT infrastructure.

In terms of **resilience**, both infrastructures are vulnerable, just in a different way. The most important vulnerability of the traditional architecture is the failure of the upstream node, for example, through a targeted hacking attack. The most important vulnerability of DLT is the consensus mechanism, which can be put under pressure by, for example, a denial-of-service (DoS) attack.

2.1.5 Access Technology

Identity-based access Similar to traditional banking, access to money is linked to some authentication (e.g. showing an identity card; username:password). This has various disadvantages, such as ensuring the robustness of authentication or making the chosen identifier available to everyone (for example: In the United States, many people do not have a generally accepted identity document [11]).

Token-based access The user authenticates themselves with some cryptographic mechanism. One example is the use of the private-public key pair. This solution is truly universal since everyone can generate such keys, and is inherently private. one of the downsides is that the keys are managed by the end user, who can easily lose them or deny law enforcement access to them. Two main parts of maintaining confidentiality with regard to CBDCs are how much data the parties share with each other during a transaction and how much data can fall into the hands of an attacker in the event of an attack on the users or the entire system.

Key Lesson

A reliable and widely usable retail CBDC must be secure and accessible, **offer cash-like convenience**, and **may** maintain privacy and confidentiality. Different technical solutions meet these criteria to varying degrees, and it is worth determining the trade-offs associated with them.

2.2 OpenCBDC

OpenCBDC is a token-based transaction processor prototype of a hypothetical direct, general-purpose, retail CBDC, which borrows many techniques from blockchains and Distributed Ledger Technologies, but itself not decentralized. It was created as part of Project Hamilton [5], a joint project of the MIT Media Lab and the Boston FED. It is open source, and available to everyone.

Their goal was to create a geographically scalable transaction manager capable of handling at least 100,000 transactions per second, most transactions within 5 seconds, and focusing on confidentiality and flexibility of token management.

In this regard, they were more than successful. It is an outstanding result among CBDC prototypes. The system can perform 1.7 Million transactions per second according to their own measurements. Although in its current state, it covers a small subset of potential functions of a CBDC [2], it is an interesting path for further exploration. Among the laid down CBDC developmental paradigms [6], it made decisions precisely in accordance with the laid out goals. The system achieves unique functionality and robust performance by using ideas from cryptography, distributed systems, and blockchain technologies. The next phase of Project Hamilton focuses on system-level auditability, programmability, privacy issues, the role of intermediaries, and resistance to DoS attacks.

2.2.1 Core design features

UHS Similar to Bitcoin [12], the system uses the UTXO currency model, as opposed to the Account/Balance model - used by Ethereum[13], for example. This means that the system records Unspent Transaction Outputs (further "spendable" through appropriate digital signatures), not the balance of individual wallets.

The system only stores the 32-byte hashes representing individual UTXOs. This hides transaction details such as sender and receiver addresses and the amount of currency sent. In addition, it reduces storage requirements and allows for higher performance. One consequence is that there is need for direct communication between payer and payee. This data structure is the **Unspent fund Hash Set (UHS).**

This data structure is not stored on a blockchain - due to the generally low throughput of Byzantine Fault Tolerant (BFT) consensus algorithms and other new consensus protocols [14], but within a Raft cluster. Raft is a high performance, but "only" *crash fault tolerant* group consensus protocol.

Transaction-local validation The transaction format created for the UHS can validate transactions in such a way that it has no information about the state of the entire system. This is achieved by making the outputs dependent on the local parameters of the transaction, rather than on the state of the system. This protects against double spend, replay and MIM attacks. Such a transaction is called a *compact transaction* within openCBDC.

System Architecture To facilitate the efficient commitment of atomic transactions into the UHS, two resilient, high-performance architectures have been created. The *Atomizer*, which creates the true order of transactions, and the *2-Phase Commit (2PC)*, which does not. The main difference between the two architectures is their performance, the Atomizer has a throughput of 170,000 Tx/s, the 2PC has a throughput of 1.7 M Tx/s with the same resources. Also, the horizontal scalability of the 2PC has been shown to be linear, while the Atomizer reaches a plateau.

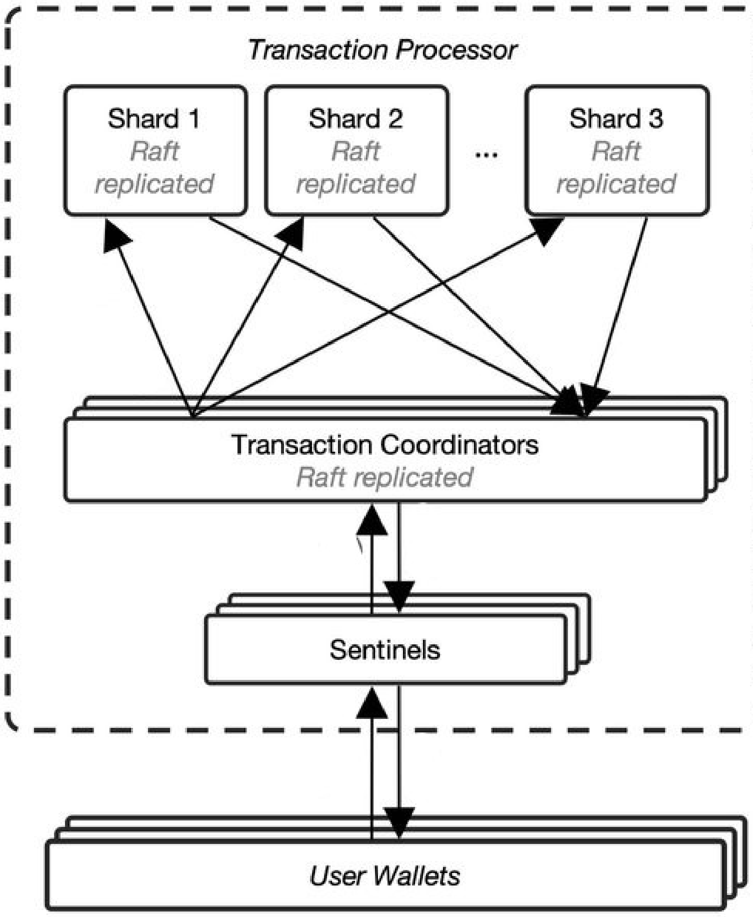


Figure 2.4: System diagram and inter-component data flow for the 2PC architecture of OpenCBDC [5]

The components of the 2PC architecture are:

Clients Client software running on the end user's machine, includes the wallets, which store the UTXOs and the cryptographic keys required to manage them. They build the full transactions to be sent to the transaction processor.

Sentinels The system's interfaces with the outside world. They convert the transactions into a compact transaction, that is, they perform the transaction-local validation. They also forward the status of the transaction to the client. Sentinels are stateless. The consequence of this is, for example, that if a user did not receive the packet confirming the successful completion of the transaction, he has to query them again.

Transaction Coordinators Executors of the transactions, they keep track of individual compact transactions, organize them into batches, and send them to the shards.

Shards They are the implementation of the UHS, where each corresponds to a disjoint domain of hashes within it. They perform the swap operations on the hashes representing individual records, i.e. UTXOs.

2.2.2 Threat model and security properties

OpenCBDC assumes that *"the transaction processor is faithfully executing our design, that users' wallets are able to maintain secret keys, and that the users are able to use a secure channels to communicate with the transaction processor."... "[They] aim to protect against an adversary who can freely interact with the system as a regular user, and as such make no additional assumptions about an adversary's capabilities or behaviour. For example, the adversary is free to create arbitrarily many identities and wallets, receive funds from other users, and engage in elaborate transaction patterns. Some of [their] designs are multi-server systems, and the adversary is free to attempt concurrent attacks against all externally-exposed parts of the system."*[5]. In this paper, I inherit these assumptions, and they will be crucial when I analyse attack vectors.

2.2.3 Client

The currently implemented client is a command-line application compiled from C++, running in docker, along with the other parts of the system.

2.3 Self-Sovereign Identity

Self-Sovereign Identity is an emerging model, which according to Brian Behlendorf, GM for Blockchain, Healthcare, and Identity at the Linux Foundation, is "the most crucial fix for today's broken Internet." It is a collection of ideologies, design paradigms, protocol frameworks, cryptographic principles, and, recently, software solutions that aim to provide an alternative to centralized and federated identity management. It is a decentralized, peer-to-peer identity model where each peer has almost total control over how their identity is stored and used. Among its many goals is to replace a physical, pre-internet system that has proven to be very effective and highly resilient. While some CBDCs aim to digitalize cash in our wallets, one of the central aims of SSI is to digitalize our plastic identity cards, and paper records associated with institutions such as the government, banks, utility companies, etc. In this chapter, I will briefly summarize the conceptual building blocks of SSI I will interact with within my protocol, as laid out by Drummond Reed in the book Self-Sovereign Identity [15].

2.3.1 ToIP protocol stack

One interesting protocol stack, which shall serve as my entry point into the introduction of SSI, is the Trust Over IP (ToIP) protocol stack. It aims to bring together and standardize the main ideas in SSI as a trust layer over the internet. I will mainly focus on the technology side. Although Governance frameworks may offer solutions for a CBDCs technical and legal challenges, such as how we delegate the trust in and claim on the Central Bank to Intermediaries in an indirect or hybrid CBDC, these solutions are beyond the scope of this paper. Luckily, the ToIP foundation offers an interactive tool to quickly understand their protocol stack at `https://trustoverip.org/wp-content/toip-model/`

One thing to note is the lower two levels focus on meeting the technical requirements of digital trust, with blockchains, and secure and private transaction protocols. In contrast, the top two layers focus on meeting human requirements, with cryptographic and generated credential protocols. If I were to classify my protocols, they would fit neatly on the top layer, as they are workflows for facilitating transactions with verified credentials from trusted sources.

In summary, ToIP can establish trusted, secure, and private peer-to-peer connections, issue, exchange, and verify digital credentials, and store public credential data on Verifiable Data registries, which may use decentralized or distributed record-keeping technologies.

2.3.1.1 Verifiable Credentials

"A verifiable credential is a tamper-evident credential that has authorship that can be cryptographically verified."[16]

Credentials contain claims about their subject made by their issuers, although they may not always be controlled by their subjects.

These claims can be verified by presenting a Verifiable Presentation to the party requesting verification.

Digital credentials are held within digital wallets, which are digital structures designed to store Verifiable Credentials securely.

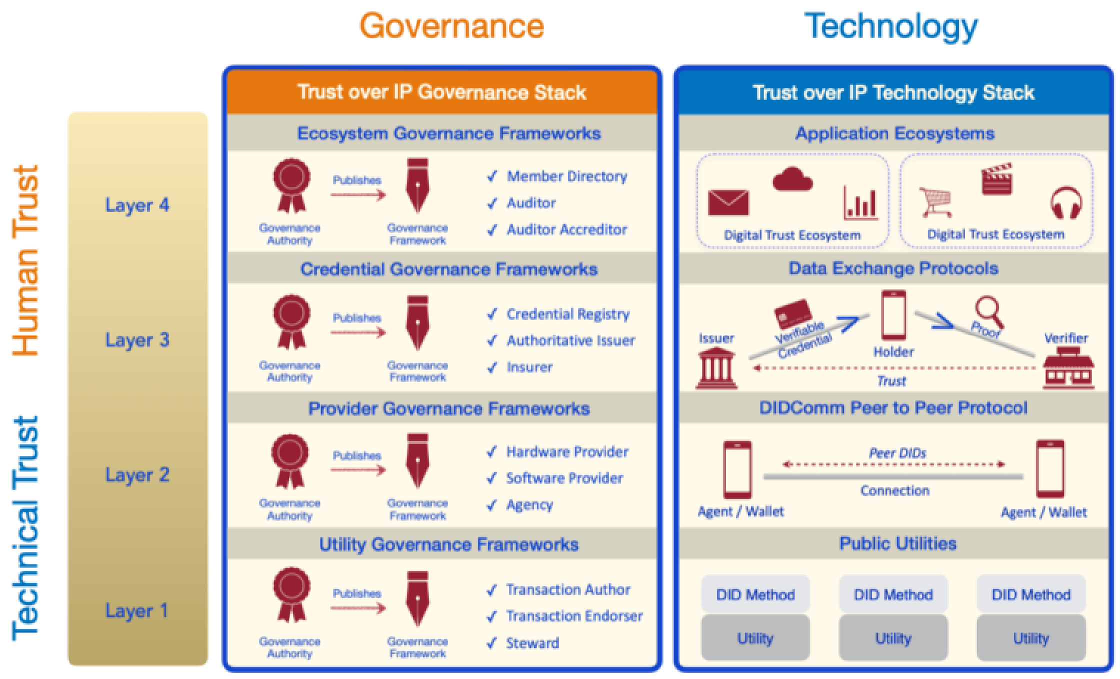


Figure 2.5: Trust Over IP Stack source: `https://trustoverip.org/toip-model/`

A *Credential Schema* is the definition of what data and claims a Verifiable Credential issued over that schema contains. They are published over the web or queriable from Verifiable Data registries.

A *Credential Definition* is a declaration from an issuer what exact schema the issued credential uses, what claims are in it, the public key used later to identify the issuer, and other metadata. It is used by a specific issuer to issue a specific version of Verifiable Credentials.

Anonymous Credentials [17] allow their controllers to prove some statement about their attributes without revealing their identity or even the underlying attribute. An often-used example is that at the bar, to prove that we are past the legal drinking age, show our photo ID, revealing our name, date of birth, and other sensitive attributes, even though they are not in contention. Anonymous Credentials use Zero-Knowledge Proofs that enable verifiably answering the question, "Are you above the legal drinking age?" without revealing any sensitive data.

Another important concept is the link secret [18], which allows users to prove that a credential is bound to them and that if they present multiple credentials within a single Verifiable Presentation, the credentials all belong to the same holder.

2.3.1.2 Digital Agents

Digital Agents are the software interface of an SSI system. They manage verifiable credentials in digital wallets and allow the users to communicate with other SSI applications, such as other agents and Verifiable Data Registries. They issue and receive VCs, and request and create Verifiable Presentations. They can be run on various hardware, from mobile devices to cloud platforms. To use a simile, they represent actors within an SSI system, like attorneys represent legal entities within the judicial system. One actor may have have multiple agents, with custom rules about what each agent can reveal about them, what kind of connections they are allowed to establish, and what cryptographic commitments they are allowed to update with Verifiable Data Registries.

2.3.1.3 Decentralized Identifiers - DIDs

Decentralized IDentifiers(DIDs) [19] - like URIs - identify subjects in a verifiable and decentralized way. They associate this subject with a DID document (DIDDoc). A DID-Document contains cryptographic information with which the controller of the DID may prove ownership of the DID. They may also contain service endpoints, which are URLs through which the controller of the DID may be reached.

The DID scheme may be implemented over various types of verifiable data registries. Over general-purpose blockchains[19, 20], purpose-built blockchains [21, 22], or even using a peer-to-peer model (pairwise and n-wise DIDs)[23].

2.3.2 Axes of identity

One interesting perspective, in which the difference between my two protocols can be clearly formulated, is put forward by Daniel Hardman and Jason Law [24]. They outline a multidimensional model of identities to which SSI components map cleanly onto. While an SSI-based payment processor system may have considerations on all three axes and planes of this model, my two proposed protocols in their current state of development show a notable difference on the Relationship-Attributes axis, the "Who knows what about me?" plane, in that we have different relationships with the Central Bank, in one we are audited, in the other we are not, and also share different parts of our identity.

In this context, my protocols share space with Gross et al.'s CBDC proposal [25], in the sense that in the high-level architecture model they also propose a private and transparent payment workflow.

2.4 The Hyperledger Aries/Indy/Ursa stack

The Hyperledger Identity Stack contains Indy, Ursa, and Aries. They are an implementation of the previously established SSI building blocks. The stack started with Indy, Hyperledger's first identity-focused blockchain framework. As the codebase evolved, parts of it separated into their own project. First, Ursa in 2018 from indy-crypto as a package of cryptographic primitives, then Aries as an extension of the indy-sdk to help others develop applications that use the building blocks of SSI.

2.4.1 Hyperledger Ursa

Hyperledger Ursa [26] produces cryptographic packages and develops primitives for other Hyperledger projects, focusing on peer-to-peer interactions. Its cryptographic primitives are packaged and embedded in other Hyperledger projects, such as Indy and Aries. It enables the generation of public/private keys, encryption and decryption, signing and verifying, hash functions, and Zero-Knowledge Proof(ZKP) technology, allowing the verification to of Zero-Knowledge Proofs built from Verifiable Credentials.

2.4.2 Hyperledger Indy

Indy [21] is the core component of Hyperledger's identity system. It implements a permissioned public distributed ledger, with indy-node serving as the blockchain's backbone, verifying ledger transactions and indy-sdk, indy-vdr, and indy-shared-rs, enabling clients to interact with the ledger.

As a permissioned blockchain, indy supports different roles for clients interacting with the ledger, such as trustees, stewards, and endorsers [27]. These roles govern how the clients can transact on the blockchain. For example, one needs to be at least an endorser to add schema definitions to the blockchain. Everyone can read the unencrypted transactions on the ledger, as the ledger contains no private data. It does not natively support querying, so one cannot simply ask for, for example, all defined DIDs. The agents must know about a transaction of interest to easily find it, either because they wrote it in the first place, or because they are configured to find it.

2.4.3 Hyperledger Aries

"Hyperledger Aries is infrastructure for blockchain-rooted, peer-to-peer interactions as defined in the Trust over IP Technical Stack, Layers 2 (secure peer to peer communications) and 3 (data exchange protocols). It defines messaging protocols and implements those protocols in shared, reusable, interoperable tool kits designed for initiatives and solutions focused on creating, transmitting and storing verifiable digital credentials" [28]. Its core functions include interfacing with the Identity Provider blockchain, securely storing cryptographic secrets and verifiable credentials encrypted peer-to-peer communication [29], and issuing and proving verifiable credentials.

2.4.4 Aries Cloudagent Python

ACA-Py [30] is a server application built on Aries features and concepts. It provides an openAPI interface to facilitate Aries functions. A controller application may send HTTP requests and receive webhook notifications to create DID-based connections, issue and verify Verifiable Credentials etc. It is a complete Aries Agent with which one can build applications and workflows utilising SSI services. It is easy to use and has language-specific APIs, e.g. for Java.

Chapter 3

Motivation

3.1 Self-Sovereign Identity and Central Bank Digital Currency

Self-Sovereign Identity has received significant attention within CBDC research [31, 25, 32]. Still, beyond identifying it as a possible solution for identity management within or between CBDC frameworks, no one has developed a framework for using it over an existing CBDC transaction processor. The technology has been dismissed as immature.

3.1.1 A secure and long-term identity layer of a CBDC

The goals of SSI align with cash-like retail CBDCs. Both fundamentally require long-term cryptographic privacy and anonymity, peer-to-peer interactions, and flexibility of governance frameworks and policies. Long-term cryptographic privacy and anonymity mean that no actor - external users or even the Central Bank - should be able to access or derive any identifiable data on the atomic transactions, set of transactions, or the Identity of the transacting parties.

By long-term, I mean that a cash-like retail CBDC shall not be vulnerable to the deprecation of cryptographic primitives, the way, for example, many cryptocurrencies currently are [33]. Suppose we store encrypted private data on the blockchain. In that case, the encryption algorithm will always be a vulnerability because if it deprecates and becomes breakable, by the immutable nature of a ledger, a malicious actor can get access to this data. Even if only cryptographic keys are encrypted, in many frameworks, they still can be or are explicitly associated with the Identity of the transacting party. For this reason, current SSI implementations, as shown in Sect. 2.4, already provide this long-term solution because they don't store any private data on their ledgers.

This long-term resilience of privacy is important because, as stated in Sect. 2.1.1, privacy has been identified as a key question in CBDC development, being second only to performance [2].

3.1.2 Aiding policy decisions

Existing Self-Sovereign Identity solutions also provide answers to issues raised in terms of identity management. As a separate identity layer, creating applications across which actors share identities is easier. And because identities are managed by their holders, while identifiers are managed by their issuers, it resolves the hierarchical tension current Identity Management solutions on the internet have, as explored in Sect. 2.3. Policymakers may introduce artificial encumbrances on user identities, but the control over a user's Identity is inherently shared between the authorities and the subject.

Verifiable Credentials are tools that allow Central Banks and Governments to create and enforce flexible policies. Because Credential Schemas inherit from each other, can have multiple definitions, and a Credential's issuance may be tied to the existence of another credential, Governments may create large, diverse, and flexible credential ecosystems.

They support no-contact revocation, which eases the technical load on the Central Bank's systems. Also, compared to physical paper-based credentials, digital credential revocation aids enforcement, e.g., a citizen doesn't have to give up their invalidated driver's license to the DMV in case of pre-term revocation. The revocation is initiated by the authority and realized by the identity layer, the citizen does not have to be compelled. VCs may also be incorporated into complex payment and exchange workflows that the responsible authority may define.

For example, suppose a citizen were to buy any controlled article or substance. In that case, the check for the relevant permits, and notification of the authorities about the transfer may be built into the payment workflow. A more specific example would be if a citizen would like to buy a gun, the digital title transfer workflow would check if he has a gun permit VC, notify the ATF about the transfer, conduct the payment with a cash-like CBDC, revoke the previous owner's digital title, and issue a new one the new owner. In this scenario, SSI helps policymakers create a cheap and easy technical solution, which with the accompanying policies, would facilitate the safe transfer of controlled articles. Currently, in the United States, a similar policy question is often on agenda, dubbed "The Gun Show Loophole" [34], to which Self-Sovereign Identity may offer a technical solution. Many similar use cases are explored in [15], many of which a Central Bank and the government would have some stake in.

If a Government were to create a VC ecosystem, each entity might issue VCs within its jurisdiction, delegating responsibility down the chain of command. Such a VC ecosystem may facilitate easy digital interactions between different government entities and promote standardization. In particular, in a Central Bank, a VC ecosystem may be created to better manage liabilities, with VCs inheriting from both public and government defined schemas. Also, if the CB were to diversify its liabilities, for example, with the introduction of a CBDC or the introduction of a CV-based bond, the existence of an international VC ecosystem would certainly help with meeting policy goals, and grant better access to publicly available liabilities to both domestic and international customers.

Verified Credentials help interoperability between institutions because they offer public, standardized schemas and provide the trust framework needed to facilitate these kinds of transactions. This interoperability is relevant to CBDC development because one of its identified goals is the effortless facilitation of cross-border payments [35].

In the context of a general-purpose retail CBDC, VCs may aid Central Banks with meeting KYC requirements, if they take it upon themselves to facilitate direct contact with the users, issuing a Direct CBDC. It is easy to envision a digital onboarding workflow where KYC requirements are met by checking relevant Verified Credentials without human interaction.

Also, it could prove beneficial for an entity to have multiple kinds of relationships with a government authority, to have multiple connections to the same institution on the relationships axis [24], to differentiate what context their interaction is in, what kind of role they each serve through the workflow of their interaction, and to preserve anonymity in use cases where it would be relevant.

3.1.3 Emerging technologies around SSI

As explored in Sect. 2.3 and Sect. 2.4, in recent years, SSI technology has gone through a maturation phase. With purpose-built blockchains like Indy [21] and Veress One [22], general purpose blockchain solutions [20], and complete development tools [28], I think it's time to take a look how the implementation of an SSI-based identity layer over a proposed CBDC would look like. The implementation of my auditable payment protocol in Chapter 5 is such a prototype.

3.2 Open ecosystem

If we separate the Identity layer of a CBDC, we allow for a different kind of interoperability. Many current CBDC proposals [6, 25] conceptualise identity management as a walled garden, where the payment processor controls identities directly or indirectly. However, SSI is building the substrate for open, widely available, secure cross-context identification, which not only alleviates the need to anchor the CBDC Identity layer in existing solutions with their known general drawbacks, but also facilitates creating such open ecosystems where CBDC acts as a controlled, but open "monetary layer" the same way that SSI is intended to serve with respect to identities.

3.2.1 Benefits of a CBDC ecosystem

With this layered approach, we may support different transaction processors, transaction protocols, or payment workflows within a single financial ecosystem. Provide diverse claims on the central banks based on identity, diversify the types of digital assets they offer, or allow for more granular control for users on how they would like to technically interact with their legal claims.

From a technical perspective, a separate identity layer may help CBDCs in development. A Central Bank may issue different CBDCs in parallel, and test key indicators such as performance, adoption, customer satisfaction, etc.

The identity layer connects the CBDCs within such an ecosystem, which negates some of the hurdles of CBDC issuance, mainly KYC. They could also share assets, with techniques such as "bridging" [36, 37].

There may be also benefits to using different identity layers over a single CBDC transaction. To provide users with multiple paths they can authenticate themselves through. An important decision in this regard would be to provide granular privacy for the end user. Such an approach may provide a resolution between the public's need for privacy and the Central Bank's burden for policy compliance, KYC, AML, CTF, travel rule etc. I will discuss these considerations in the next chapter as I define my two protocols, which mimic such an ecosystem.

3.2.2 Generalisability of a private cash-like CBDC

Using a private, cash-like CBDC transaction processor is a requirement if I would like to test out the full spectrum of privacy in an approach, where the transaction processor and identity layers are separated. openCBDC, being the first open-source CBDC transaction processor providing performance indicators, is a great candidate for designing an SSI-based CBDC payment protocol. Because it relies on cryptographic encumbrances to spend funds and requires inter-user communication; an identity layer which provides a developer-friendly wrapper for these functions. Also, being a simple transaction processor, in later stages of development, it may be easily interchanged with a different CBDC transaction processor in SSI-based CBDC payment workflows.

Surveying existing research, there is yet to be a payment solution implemented over openCBDC. In summary, my goal is to provide payment protocols using SSI technology over openCBDC, which explore the full spectrum of privacy in the context of a retail CBDC.

Chapter 4

Maximum and minimum CBDC privacy

4.1 Spectrum of privacy

I define a spectrum on which privacy may be understood regarding a retail CBDC, then designate two positions around the extremes of this spectrum. Each protocol will implement requirements associated with these selected privacy "classes". I discuss the advantage of this approach later in Section 4.4.

The spectrum of privacy regarding cash-like CBDCs comes from the tension between the public's need to remain anonymous and untraceable and the Central Bank's responsibility to serve the needs of its whole financial ecosystem and comply with internal, national and international policies (KYC, AML, CTF, travel rule, etc.) [6]. In this regard, the inflexibility of a CBDC is counterproductive.

To get a clearer picture, I will look at the forces defining this spectrum.

4.1.1 The needs of the public

As stated by [6], there is a trade-off between privacy and ease of law enforcement. The needs of the public in this regard are often articulated in the term that a retail CBDC needs to be "cash-like". This means that a proposed CBDC solution must be universally accessible to all citizens, have a low barrier to entry (e.g., the software needed to facilitate transactions should be able to run on low-end mobile devices effectively), be an easily enforceable claim on the central bank, and citizens should be able to transact without any party being able to correlate identities easily, or uncover money flow graphs.

The cryptocurrency field luckily has technical methods to achieve better privacy in pseudonymous systems , for example, using mixers/tumblers to obfuscate the transaction graph. By building a system that can facilitate such methods, we may get closer to reaching the public's need for privacy.

4.1.2 The responsibilities of the Central Bank

For a Central Bank Digital Currency to be worthwhile, it should be able to comply with existing digital transaction policies such as KYC, AML, CTF, travel rule(note on travel rule: in the context of a retail CBDC, a much larger proportion of transactions may occur directly between private parties, simply because the payment doesn't require intermediaries, such as retail bank today), etc., and provide functions that are beyond the capabilities of existing digital assets. To enforce such policies, the central bank should be able to conduct an audit of past transactions in a way that they can correlate identities between accounts or addresses and the transacting entities. While Zero-Knowledge based audit protocols are currently being explored [37], a straightforward audit model, where the central bank controls the transaction graph and can verify the identity of each transacting party, would still satisfy the Central Bank's need for auditability. It would also be easier to implement, allow for finer enforcement of policies, and allow the use of efficient audit algorithms already used by existing payment processor systems.

4.2 Privacy Classes

According to these needs, I define two privacy classes, which, when both implemented in a financial ecosystem, may strike a balance between the public's need for privacy and the Central Bank's need for regulatory compliance. These classes resemble the transaction types used in Zcash [38].

Auditable transactions or "unshielded transactions" to use the Zcash terminology, in which the sender's and the recipient's identity can be uncovered, and a transaction graph can be constructed.

Private transactions or "shielded transactions" where the transactions satisfy the requirements of being cash-like. Namely, no party can correlate identities easily, and the transaction graph can be obscured.

In Zcash, there are transaction types between privacy classes, between private and transparent addresses, called "shielding" and "unshielding" transactions. I do not define such protocols, as their exact specification is beyond the scope of this paper, but if one were to create a complete payment system based on my protocols, they would need to be implemented.

An important thing to note is that in such a CBDC ecosystem, the CB may limit the public's ability to utilize private, cash-like transactions because they introduce a high risk of facilitating illegal activities. To combat this risk, a proposal put forward by [36] is to endow private transactions with limitations, dubbed "cash-like encumbrances". I will explore these encumbrances when I define the protocols' privacy goals in Section 4.4.

4.2.1 Better privacy with SSI

As previously mentioned, SSI has been identified as a possible solution for identity management in a CBDC system. [36] explores a number of possible use cases. They clearly demonstrate how Verified Credentials could help the KYC process associated with opening a direct CBDC account and provide a possible model for how transactions could be facilitated on a CBDC using Decentralized Identifiers as addresses.

4.2.1.1 Roles of the Central Bank

I will now explore what responsibilities may the Central Bank take on if an SSI-based identity layer were to be used over a direct CBDC.

KYC For citizens to be able to open balances, the Central Bank shall be able to KYC them. For this purpose, the Central Bank should be able to accept KYC credentials issued by retail banks, and issue KYC credentials based on relevant VCs. Zero-Knowledge proofs and Selective Disclosure provided by SSI solutions may certainly aid this process. As laid out by [36], if a citizen were to prove their permanent residency in a country during KYC, they only need to prove that specific fact, but not disclose sensitive information, their whole address.

Tracking Identites If a Central Bank were to implement a retail CBDC facilitating auditable transactions, tracking the identities of transacting parties could be solved by simply storing them in an internal database, using their DIDs as CBDC addresses [36], or issuing credentials which contain some information about their relationship to the Central Bank, or by using credentials issued by retail banks. In my model, the Central Bank issues credentials which contain the CBDC addresses of the recipient. This "digital central bank credit card" opens up the question of who should designate these addresses; the Central Bank, the citizen, or an intermediary retail bank. This "credit card" model allows separating the identity layer because we can store arbitrary data structures within VCs.

Facilitating private transactions In my private transaction protocol, the Central bank needs to store a data structure associated with each citizen to facilitate the Zero-Knowledge proofs used to meet the "cash-like encumbrances".

4.3 Payment over openCBDC with Hyperledger Aries

My goal is to specify two protocols which implement the layered identity model over openCBDC using Aries Cloudagent Python [30] as the SSI implementation, connecting to an existing developmental NYM ledger. One protocol models the auditable privacy class; the other the private privacy class using "cahs-like encumbrances". OpenCBDC, as explored in the previous chapter, is a great candidate for my layered identity model, and its use may demonstrate the generalizability of my protocols. Aries Cloudagent[30] Python is an open-source implementation of an SSI agent, offering openAPI endpoints through which we can facilitate the interaction with the building blocks of SSI. It offers automatism for standard functions of the building blocks, like peer-to-peer connection, credential issuance, selective disclosure, zero-knowledge proofs etc. By using these building blocks, I will demonstrate how the layered in the development of a CBDC payment protocol.

4.4 Privacy goals

Some previously laid out privacy considerations are already met by using Hyperledger Indy and Aries. As stated before, they provide a long-term cryptographic privacy solution for storing and managing verifiable credentials; and methods for safe and private peer-to-peer communication based on DIDs. I highlight the fact that Indy doesn't store encrypted private data on its blockchain.

The privacy goals specific to the protocols are in order:

Auditable transaction protocol If the citizen maintains basic privacy practices in their SSI agents, no outside actor should be able to find out any details about his transactions, e.g. how much he paid, who he paid it to, or even what his public he uses to create his CBDC address.

Private transaction protocol Clients shall have the ability to send to and receive from Bitcoin-like addresses in a UTXO setting, without no party being practically able to correlate identities and money flow easier than it is for Bitcoin. I establish this metric to make my protocol somewhat comparable to other privacy-preserving techniques.

Limiting the public's ability to utilise this transaction protocol, making it as or more inconvenient than current existing forms of electronic money, translates to the following properties, as laid out by both [36] and [25]:

- No person can hold more than a policy-determined amount at any time.
- No person can send or receive more money in a time frame than allowed. (Amount and time policy determined.)
- No "handover" may occur without physical closeness.

My model focuses on preventing smurfing - a person or group circumventing these limitations by utilising multiple accounts - across the addresses of a single person.

Chapter 5

Auditable payment protocol

A currently in development implementation of this protocol is available on GitHub at `https://github.com/BlackLight54/TDK_openCBDC_SSI`

At the minimum of privacy, belonging into the privacy class of auditable transactions is my Auditable Payment Protocol. It enables the Central Bank to store the details of each facilitated transaction safely, authenticate users using Verifiable Credentials, and correlate their identities with their payments.

5.1 Actors and Components

Central Bank (CB) issues and manages Verified Credentials. KYCs the customer during onboarding. Publishes Credential Schemas and Definitions. In particular, it issues a credential to eligible clients with which they can facilitate transactions.

Sentinel Guard controls the outward-facing interface of the transaction processor, the Sentinel. Receives transaction requests, based on which request verifiable presentations to authenticate transaction parties. Maintains a log of received transaction proposals. Functions as a gatekeeper to the transaction processing system.

Alice and Bob are citizens who open CBDC accounts and make a transaction through the CBDC system.

All actors above have corresponding cloud agents

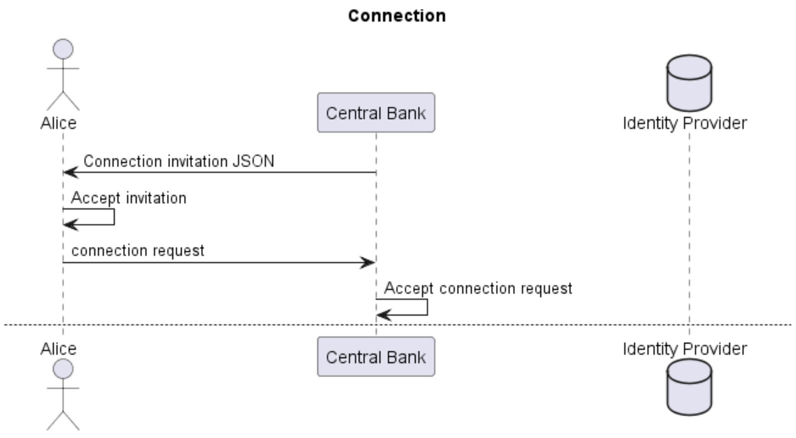
Identity Provider maintains registries of decentralized identifiers, credential schemas, and published credential definitions. In practice, this is an Indy network.

5.2 Preparation to issue Verifiable Credentials

Credential issuance is based on the Alice/Faber ACA-py demo [30]. To issue a credential, the Central bank must:

- Register a public DID and store it on the ledger;
- Create a schema and register it on the ledger;
- Create a credential definition and register it with the Identity Provider

Figure 5.1: Technical steps of an Aries peer connection



Note that the central bank must store the issued credential definition in its cloud agent wallet to use it to later issue credentials on it.

The credential schema contains an openCBDC address and its corresponding public key. This schema defines our "digital credit card".

5.3 Opening openCBDC accounts

Alice would like to open a CBDC account. She establishes a peer SSI connection to the Central Bank (Fig. 5.1). First, the CB conducts the required KYC checks (Fig. 5.2). In the implementation, Alice must present a VP where she proves that she is a national. If Alice passes all checks, the CB requests a public key from her, based on which a unique CBDC address can be generated for her. Then the CB issues a Verifiable Credential (Fig. 5.3) containing this address and her public key. It also stores her public key, along with the KYC data needed to associate Alice with this public key. Now Alice has a CBDC address only she can use.

5.4 Facilitating transactions

Alice and Bob would like to conduct a transaction over the CBDC system. They both went through the onboarding process and have credentials issued. Alice sends a transaction request to a Sentinel Guard, which contains her and Bob's public DID and the details of their transaction. The Sentinel uses this DID to look up the public DIDDoc of both Alice and Bob, from which it can find their public keys and how it can reach them. It then sends both Alice and Bob a proof request, asking them to provide the VC issued by the CV. If they provide valid proofs, the Sentinel Guard sends the transaction to an openCBDC sentinel. It also stores the details of the transaction.

Figure 5.2: Onboarding sequence including KYC [36]

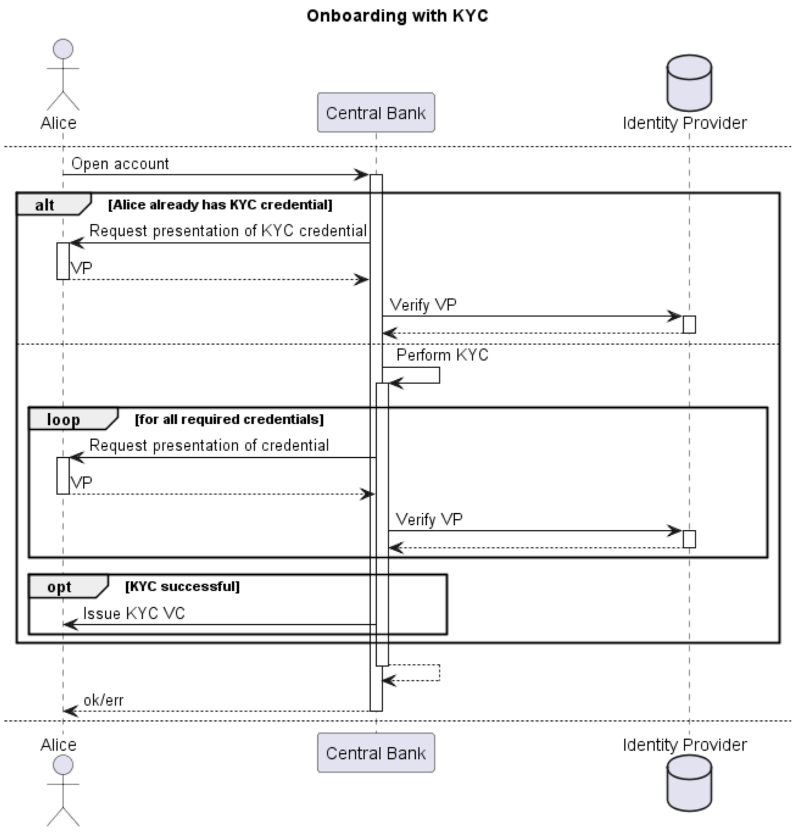


Figure 5.3: Technical steps of issuing a credential

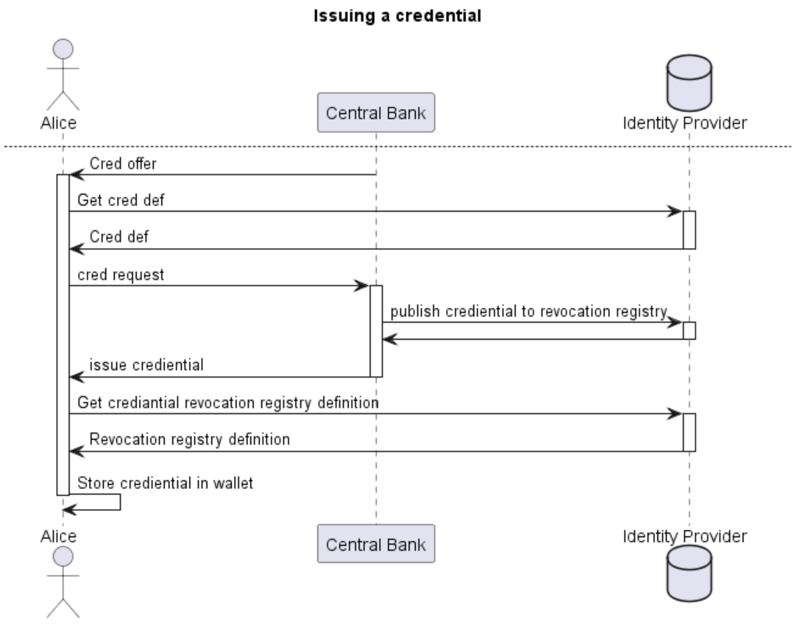


Figure 5.4: Sequence diagram of the Auditable transaction protocol

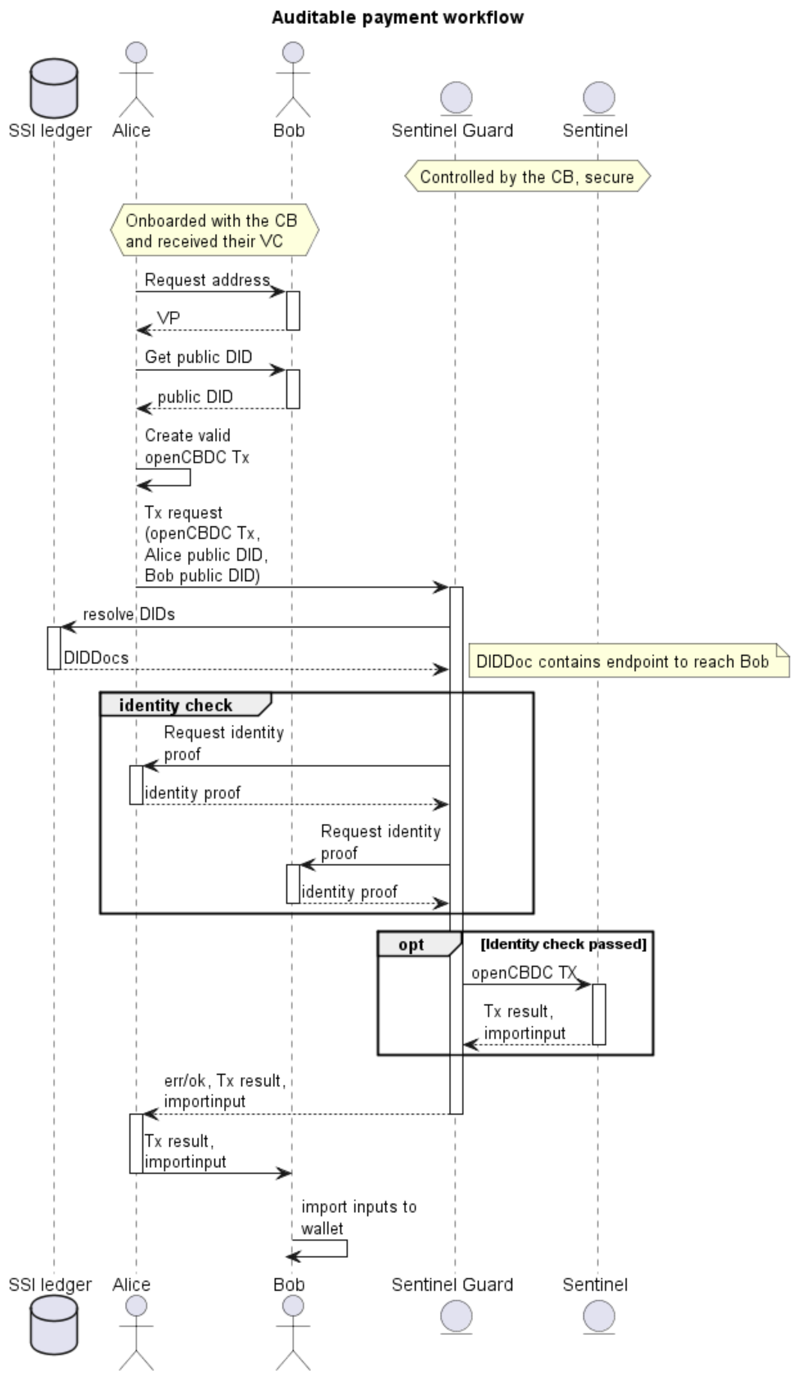
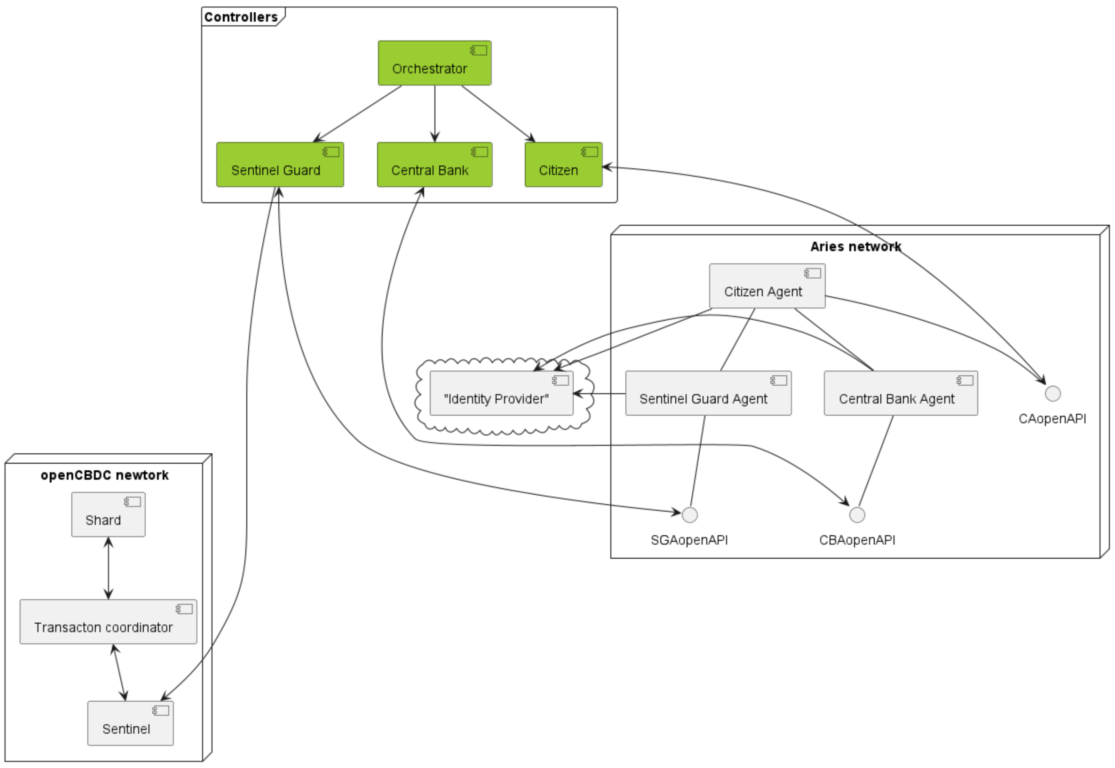


Figure 5.5: Component diagram for the entities present in the Auditable Payment Protocol. Green components are new contributions.



5.5 Implementation in development

I have begun the development of an implementation of the Auditable Payment Protocol. It is available on GitHub at `https://github.com/BlackLight54/bsc_thesis_openCBDC_SSI` The current infrastructure looks like this :

- openCBDC runs from a prebuilt image in Docker, set up with docker-compose for the 2-Phase Commit Architecture
- the Aries agents run in a docker-compose network, with endpoint ports and administration ports(through which the API is reachable) exposed
- A Spring application implements the controllers, realizing the specified workflow. Each controller is a Spring component, with an overseer orchestrating the protocol.

Currently, the application is able to verify the identity of the citizens and conduct openCBDC transactions, although not with this exact control and information flow. Right now, the Sentinel Guard manages the keys used in openCBDC by the clients. After identification, it simply calls the existing openCBDC CLI client. Because no transparent API is yet to be available for open CBDC, it may take a while to solve these issues. The schemas and credential definitions are programmatically defined, with a java API of aca-py, but I provide example schema definitions that travel over Aries in the ./VC examples/ directory.

Chapter 6

Private Payment protocol

The following protocol is one instantiation of the Private Payment privacy class. It aims to make the identities of the transacting parties and the atomic transactions unlinkable. It is inspired by the Chaumian Digicash, which was a type of private electronic money based on cryptographic protocols designed by its founder, David Chaum [39]. The company, unfortunately, went bankrupt, but a modern implementation of their protocol is provided by Taler Systems in the GNU Taler project [40].

6.1 Notation, and terminology

$x := y$: let x be equal to y, assignment

$hash(x)$: collision-resistant hash function

$x|y$: concatenation, bits of y are concatenated directly after bits of x

$salt$: essentially random string of bits used to avoid correlation of hashes

$sign(x, private\_key)$: the result of signing bits of x with the $private\_key$ with a Digital Signature Algorithm (DSA)

$encrypt(x, private\_key)$: the result of encrypting x with the $private\_key$

Merkle Tree : binary tree with n leaves, where each $leaf := hash(x_i), i \in 1...n$; each $node := hash(left\_subtree|right\_subree)$, can be used to Zero-Knowledge Prove $x \in x_1...x_n$ [41]

Blind signature : a form of digital signature in which the content of a message is disguised (blinded, here encrypted) before it is signed. The resulting blind signature can be publicly verified against the original, unblinded message in the manner of a regular digital signature [39].

6.2 Actors and Components

Central Bank issues time- and amount-limited spend- and receive authorizations to citizens.

Extended openCBDC Sentinels consume these authorizations.

Alice and Bob citizens who would like to transact.

6.3 Data structures

6.3.1 Data kept by the citizen

Citizen Verifiable Credential used to identify the citizen when they ask the Central Bank for a send- or receive token. Verifiable Credential, issued by the government.

citizen_private_key-s The cryptographic keys used by the citizen as the basis for Bitcoin-style openCBDC addresses and digital signatures. Kept private.

6.3.2 Data maintained at the CB

6.3.2.1 Data structure associated with each citizen VC

hash(current_balance|salt) commitment on the combined balance of addresses associated with the citizen, used to ensure holding limits.

addresses_merkle_root the root of a Merkle tree of the Bitcoin-like openCBDC addresses of the citizen, used to account for the source and target of each transfer. Algorithms for proving Zero-Knowledge Set Membership (ZKSM) within Merkle trees are known and widely used.

last_spend_request_tstp timestamp of the last approved spend request.

last_receive_request_tstp timestamp of the last approved receive request.

6.3.2.2 Data kept by only the CB

CB_private_key the key the Central Bank uses to blindly sign send- or receive tokens. It rotates periodically to enforce cash-like encumbrances.

policy_holding_limit The Central Banks defines, in policy, a holding limit. This is the maximum amount of currency a person may hold within addresses contained within their associated Merkle tree.

6.3.2.3 Data shared between the sentinels and the CB:

CB_public_key the key to check the signature on the access token. To enforce cash-like encumbrances, it is associated with a fixed amount and a timeframe (e.g., one specific within-day 5 minute, 10 minute or 1 hour window):

> **max_amount** the maximum spendable and receivable money within the timeframe by a spend- or receive-token authorized citizen.
>
> **usable_until** Time frame within which a spend- or receive-token can be used.

sentinelId Numeric identifier, uniquely identifies a Sentinel.

6.3.3 Data used during transactions

from_address the (Bitcoin-like openCBDC) address from which the spender spends money

to_address the (Bitcoin-like openCBDC) the address to which the receiver receives the money

transaction_key a unique transaction identifier agreed on by the payer and payee. They use it at the sentinel as an anonymous session token like ID for both proving spend and receive authorization for the same transaction.

transaction_amount the amount of money the spender wishes to send to the receiver

new_balance the citizen's balance after the transaction

Spend token access token in the format of $signed(from\_address|sentinelID)$ blindly signed by $CB\_private\_key$

Receive token access token in the format of $signed(to\_address|sentinelID)$ blindly signed by $CB\_private\_key$

6.4 Prerequisites for a receiving spend- and receive authorizations

Citizens must register a commitment on their addresses ($addresses\_merkle\_root$) and current balance ($hash(current\_balance|salt)$) with the Central Bank.

Citizens may update their associated $addresses\_merkle\_root$ later if they want to acquire new addresses. One issue to note is that currently, there is no defence against multiple citizens registering the same address, but as that would only limit the "spending power" or "receiving power" of an address, there does not seem to be any value in this kind of cooperation. Maliciously registering the address of another citizen is an attack vector; however, I do not detail defences against that here. Options may include, for example, limiting the size of the Merkle tree for every citizen through a ZKP obligation on $addresses\_merkle\_root$ change and limiting the frequency of updating the $addresses\_merkle\_root$.

6.5 Preparation for a transaction

The steps of requesting an access token are shown in Fig. 6.1.

When a citizen wishes to spend money from their address, first they must acquire a spend token.

They give **encrypted(from_address | sentinelId) := encrypt(from_address | sentinelID, citizen_private_key)** to the Central Bank and gets it blindly signed (**signed(encrypted(from_address|sentinelId)):= sign(encrypted(from_address|sentinelId), CB_private_key)**).

The blind signature key, $CB\_private\_key$ corresponds to a max_amount (quasi-note) and a $usable\_until$ timestamp, the end of an intra-day interval (hour, 5 minutes, etc.). When a Sentinel consumes a token, it checks that $amount < max\_amount$, and the current time is before $usable\_until$ timestamp, and enforces the "speed-of-money" encumbrance, $v_{money} = currency/t$, which means that within a specific time window, a citizen may only send and receive a policy-determined amount of currency.

Ideally, the citizen should also prove here that the from address is in its registered address set in a privacy-preserving way, but that is problematic with current ZKP technology (encryption proving). This will be checked in the protocol later, anyhow.

After decryption, the citizen has $signed(from\_address|sentinelID)$ spend-token.

Citizens are interested in preserving their privacy, which translates to making the issuance and usage of spend-tokens unlikable.

Unlinkability on the Sentinel side, assuming they may maliciously cooperate with the CB, requires that *usable_until* and *max_amount* cannot be used for correlation efficiently. This can be achieved by only periodically updating *usable_until*s and the same *max_amount* for every issued token.

If the *usable_until* window is long enough, enough clients use the system, and the clients all perform a random waiting phase between receiving the token and using it, timing-based linking of issued and used tokens should not be practically feasible.

The **Central Bank** is interested in enforcing maximum spend limits(*max_amount*) for a policy-defined time (*usable_until*) window for each citizen. Therefore, it will refuse to issue another spend token before a cooldown time has passed, e.g. not before the previously issued has reached its *usable_until* time. For receiving money, a receive token ($signed(to\_address|sentinelID)$) is created in a similar fashion. Here *to_address* is the address to which the recipient would like to receive the money.

Regarding the receive token, the Central Bank is interested in enforcing maximum holding limits. Therefore, it will issue the token only if the citizen can present proof that for the hash-committed *current_balance*, $current\_balance + max\_amount < policy\_holding\_limit$. This is simple in ZoKrates and can be interactive ZKP. (ZoKrates supports non-interactive zk-SNARKs.)

The receive authorization is similarly time managed.

6.6 Facilitaiting transactions

Alice and Bob agree that they want to transact for *transaction_amount* of money. Alice gets a spend-, and Bob gets a receive token to the same Sentinel. They agree on a random *transaction_key*.

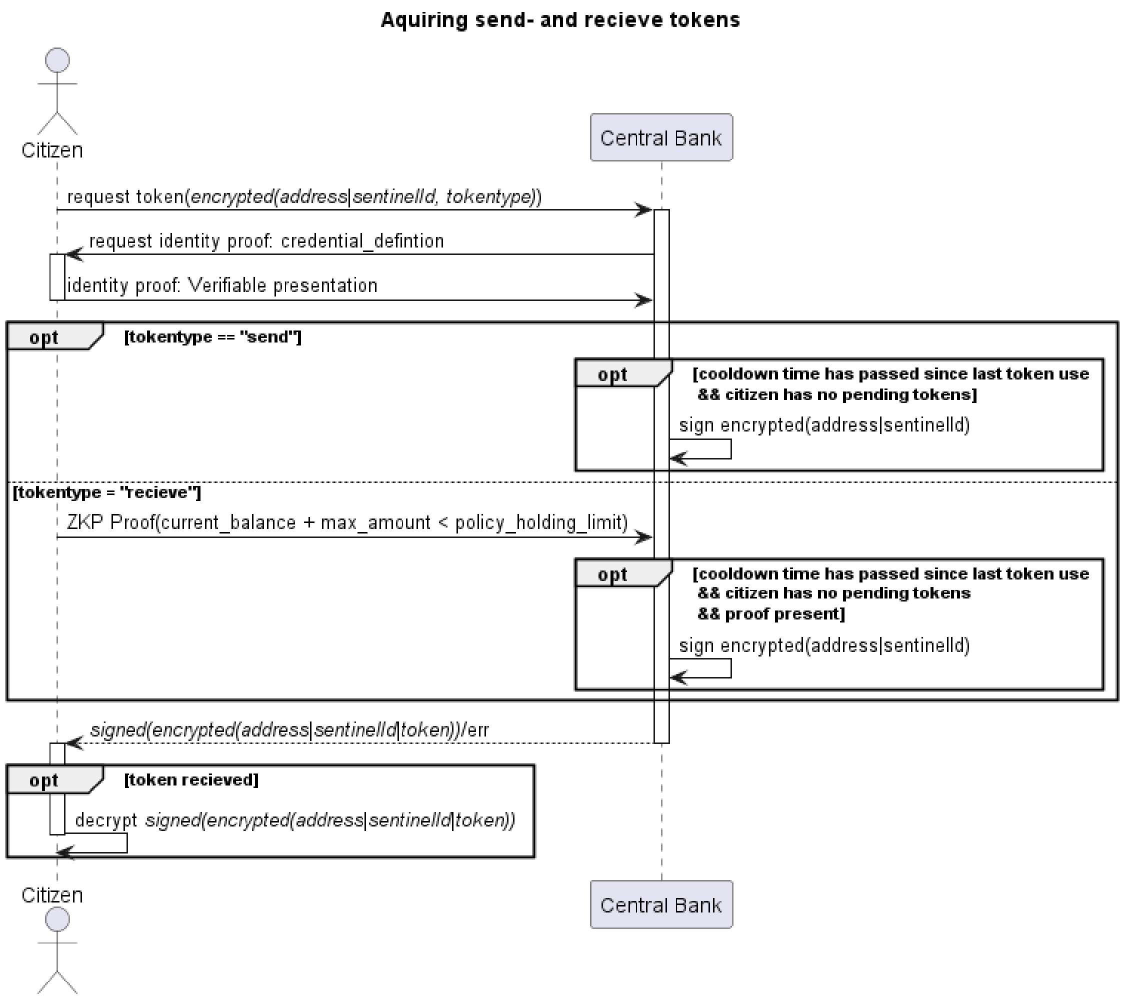
Alice, the payer anonymously turns to the Sentinel, and

- registers the *transaction_key*,
- presents the spend token, which the Sentinel records to avoid double use,
- declares the *transaction_amount*.
- signs the whole request with *citizen_private_key*, to prove that she owns the address within the spend token.

Bob does similarly for the recieve token.

At this point, if physical proximity is required for the transfers, infrastructure or hardware can be involved, which can provide timestamped geolocation attestation for senders and receivers, e.g. by cell towers for mobile devices.

Figure 6.1: Requesting and receiving a spend- and receive token from the Central Bank


Aquiring send- and recieve tokens
Citizen
Central Bank
request token(*encrypted(address|sentinelId, tokentype)*)
request identity proof: credential_defintion
identity proof: Verifiable presentation
opt
[tokentype == "send"]
opt
[cooldown time has passed since last token use && citizen has no pending tokens]
sign encrypted(address|sentinelId)
[tokentype = "recieve"]
ZKP Proof(current_balance + max_amount < policy_holding_limit)
opt
[cooldown time has passed since last token use && citizen has no pending tokens && proof present]
sign encrypted(address|sentinelId)
signed(encrypted(address|sentinelId|token))/err
opt
[token recieved]
decrypt *signed(encrypted(address|sentinelId|token))*
Citizen
Central Bank

Alice now creates a full openCBDC transaction based on the two presented tokens and sends it into the sentinel

Based on the assumption that the $from\_address$ was registered for Alice and she had a single spend token for that address, the following protocol can be followed.

6.7 Updating the commitments after a transaction

The Sentinel gives Alice a receipt: $signed(from\_address|transaction\_amount|timestamp)$. If the spend or receive tokens are not used after all for any reason, then the Sentinel can be requested, even after the validity of the token has passed, to issue a receipt with zero value, proving that a transaction did not take place.

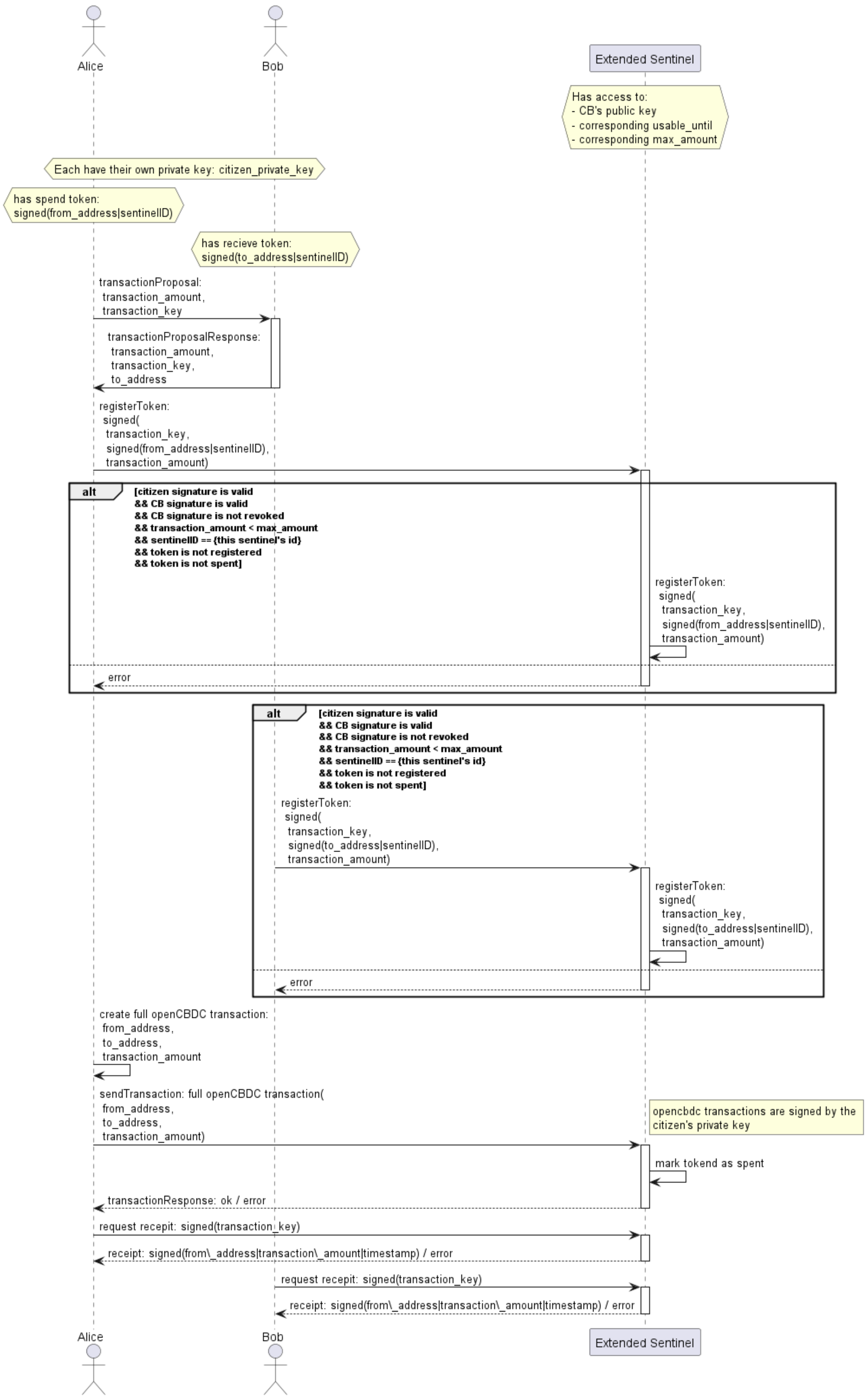
Alice turns to the Central Bank, who knows that she had a spend allowance for a time window, presents$hash(signed(from_address|transaction_amount|timestamp)|salt)$ and $hash(new\_balance|salt2)$ and ZKP proves the following.

- It knows $signed(from\_address|amount|timestamp)$ and salt and these lead to the presented hash.
- The signature is valid for the Sentinel on $signed(from\_address|amount|timestamp)$.
- Timestamp is valid for the issued allowance.
- $from\_address$ is in the Merkle tree of the addresses of the citizen.
- It knows $salt2$, $newbalance$ and $salt2$ hash to the presented value.
- $newbalance$ is equal to $current\_balance - transaction\_amount$.

The Central Bank accepts the receipt, updates Alice's balance hash commitment -$hash(new\_balance|salt2)$-, and allows her to get a new token. The same happens on the receiver side.

The citizen should be able to provide these proofs if the transaction really took place. Because we checked the compliance with the "cash-like" encumbrances before facilitating the transaction, he cannot force himself into an illegal state, i.e. he has too much money.

Figure 6.2: Using the spend- and receive tokens to facilitate a transaction


Alice
Bob
Extended Sentinel
Has access to:
- CB's public key
- corresponding usable_until
- corresponding max_amount
Each have their own private key: citizen_private_key
has spend token: signed(from_address|sentinelID)
has recieve token: signed(to_address|sentinelID)
transactionProposal: transaction_amount, transaction_key
transactionProposalResponse: transaction_amount, transaction_key, to_address
registerToken: signed(transaction_key, signed(from_address|sentinelID), transaction_amount)
alt
[citizen signature is valid && CB signature is valid && CB signature is not revoked && transaction_amount < max_amount && sentinelID == {this sentinel's id} && token is not registered && token is not spent]
registerToken: signed(transaction_key, signed(from_address|sentinelID), transaction_amount)
error
alt
[citizen signature is valid && CB signature is valid && CB signature is not revoked && transaction_amount < max_amount && sentinelID == {this sentinel's id} && token is not registered && token is not spent]
registerToken: signed(transaction_key, signed(to_address|sentinelID), transaction_amount)
registerToken: signed(transaction_key, signed(to_address|sentinelID), transaction_amount)
error
create full openCBDC transaction: from_address, to_address, transaction_amount
sendTransaction: full openCBDC transaction(from_address, to_address, transaction_amount)
opencbdc transactions are signed by the citizen's private key
mark tokend as spent
transactionResponse: ok / error
request recepit: signed(transaction_key)
receipt: signed(from_address|transaction_amount|timestamp) / error
request recepit: signed(transaction_key)
receipt: signed(from_address|transaction_amount|timestamp) / error
Alice
Bob
Extended Sentinel

Chapter 7

Results and Discussion

I have explored what motivations a Central Bank has for using SSI technology in the context of a CBDC; how different design principles help a CBDC fit within a broader financial ecosystem; identified key decisions within a design framework which aims to strike a balance between the needs of the main stakeholders of a CBDC; and specified two protocols which adhere to this framework.

Using similar, stakeholder-need-based design processes is not newfangled within existing CBDC research [6]. My approach focuses on the key issue of privacy. Such exploratory design may help us break free from base assumptions present in the field and identify novel solutions. In the future, as both the SSI and the CBDC fields develop, and more intersecting research is produced, researchers and developers will surely get a clearer picture of what other design considerations are to be made and how does system-design affect key issues in identity management and digital asset development.

Prototyping layered solutions allow developers to pinpoint one issue they would like to focus on without the need to maintain full focus on systemwide design considerations. This approach allowed me to focus on privacy as the main issue of this paper because openCBDC is already proven to be a performant and private transaction processor.

Putting an SSI-based payment solution over openCBDC was also beneficial because I was able to use existing software building blocks - due to the open-source nature of both openCBDC and Hyperledger Aries - which served as the foundations of the design process and enabled easy prototyping of my ideas.

The two protocols I specified are both SSI-based payment solutions over openCBDC and utilise algorithms for which open-source implementations are widely available. I would like to discuss the results regarding each one in detail.

The Auditable Payment workflow takes advantage of Verifiable Credentials for citizens to prove their identities. As stated in Ch. 4, using Verifiable Credentials as the basis of identification enables the Central Bank to create complex payment workflows utilising existing SSI infrastructure and freely layer a cash-like under this identity layer. Although there are other solutions to be explored [36], I have shown that a Verifiable Credential-based payment workflow is most certainly viable. My design follows the philosophy of [15] in the sense that Verifiable Credentials can be considered digital versions of existing physical credentials. In my model, as I stated before, the identifying VC can be considered a Digital Bank Card.

Aiding the Central Bank in compliance with existing international policies (KYC, AML, CTF) was a requirement. Using an SSI-based identity layer and storing atomic transactions, the Central Bank can easily uncover the transaction graph within registered addresses and associate identities with each address. This way, it can more than comply with these regulations.

The Private Payment Protocol, inspired by Chaumian Digicash, decouples identity from spending while enabling cash-like encumbrances by unlinking the permission to spend and the actual spending with the use of access tokens. Designing a cash-like identity layer while implementing the previously defined encumbrances is certainly feasible, although my protocol only serves as an entry point to designing such CBDCs. As mentioned before, an attempt has already been made by others [25], inspired by Zcash[38].

A brief note on the payment processor system by [25] is that by using the Zcash [38], they run into some issues present in the cryptocurrency, namely, they store encrypted non-public data, which makes them vulnerable to the deprecation of cryptographic primitives.

I assumed the CBDC system to be uncompromised and to execute the requested transactions faithfully. Certainly, attack vectors coming from within the Central Bank must be analyzed. This is beyond the scope of this paper, as I aimed to simply test the viability of a Verifiable Credential-based protocol and only considered attack vectors originating outside the defined actors. An attack vector to note in the Private Payment Protocol, discussed in Ch. 6 is if the CB maliciously chooses very short expiration windows for the spend- and receive tokens, in which case identities and transactions may be correlated based on their timing.

Regarding the exploration of the defined privacy spectrum, I think the two protocols provide proper representative samples from the two extremes of the spectrum. One unexplored issue is the interaction of these samples. In the previously mentioned Zcash [38] transaction model, there are transitions between private and transparent wallets. A clear next step would be the definition of the other two protocols, to enable the interaction of private and the auditable systems.

Another thing to note is that I cannot provide performance indicators for an identity layer over openCBDC because I do not have access to the required hardware capacity for testing the scalability and the peak performance of such a system. Because in both protocols, the endpoints modify a shared state -beyond the currency transfer-, perfectly linear horizontal scalability cannot be assumed. However, by utilising existing distributed system architectures and fine-tuning them for privacy and policy goals, and because the processing of the currency transfer is decoupled from the identity layer, the performance loss compared to the "naked" openCBDC may be manageable.

Chapter 8

Conclusion and Future Work

Given the scarcity of research around Self-Sovereign Identities related to openCBDCs, it is important to explore the possible design principles regarding systems utilising these technologies.

The benefits of using Self-Sovereign identities as the identity layer of a CBDC system are numerous. With clearly defined policy goals, a Central Bank can benefit from a rigorous design process. I've shown how design paradigms focusing on stakeholder needs aid the development of CBDC payment schemes and how separating the roles different technologies play within our system would result in better privacy while maintaining flexibility.

My auditable payment scheme is an important proof-of-concept, in the sense that integrating existing SSI technologies into CBDC payment workflows is not just viable, but can support a wide array of policy decisions.

Drawing from existing algorithms and workflows, but implementing them in concert with emerging technologies is widespread within CBDC research.

Proposing a viable fully private payment workflow inspired by Digicash is only the first step in creating a truly cash-like CBDC, but given its advantages over other approaches, it should be further analised.

In the future, the formal specification of the protocols must be discussed, and security analysis must be conducted, although i put it together from known blocks. SSI authenticated singleton access in digicash style only assume proof obligations, which piecewise are known to be supportable with ZKsnarks proving scheme e.g. Zokrates That being said, for evaluating practical feasibility, importantly to gauge the computational commitment necessary from the end users, a prototype implementation has to be created.

Bibliography

[1] Klaus Löbe and Aerdt Houben. Central bank digital currencies. Technical report, Committee on Payments and Market Infrastructures - Markets Committee, 2018.

[2] Multiple Central Banks. Central bank digital currencies: foundational principles and core features. techreport, Bank of International Settlements, 2020. URL `https://www.bis.org/publ/othp33.htm`.

[3] Anneke Kosse and Ilaria Mattei. Gaining momentum – results of the 2021 bis survey on central bank digital currencies. techreport, Bank of International Settlements, 2020. URL `https://www.bis.org/publ/bppdf/bispap125.htm`.

[4] Jingjing Sun. Reflections on the latest e-cny pilot test in china. *Journal of Asia-Pacific and European Business*, 1(01), 2021.

[5] James Lovejoy, Cory Fields, Madars Virza, Tyler Frederick, David Urness, Kevin Karwaski, Anders Brownworth, and Neha Narula. A high performance payment processing system designed for central bank digital currencies. *Cryptology ePrint Archive*, 2022. URL `https://dci.mit.edu/opencbdc`.

[6] Raphael Auer and Rainer Boehme. The technology of retail central bank digital currency. techreport, Bank of International Settlements, 2021. URL `https://www.bis.org/publ/qtrpdf/r_qt2003j.htm`.

[7] Antonios Koumbarakis and Guenther Dobrauz-Saldapenna. Central bank digital currency: Benefits and drawbacks. techreport, PWC, 2019. URL `https://ssrn.com/abstract=3429037`.

[8] Christian Grothoff David Chaum and Thomas Moser. How to issue a central bank digital currency. techreport, Swiss National Bank, 2021.03. working paper.

[9] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *arXiv preprint arXiv:1704.04299*, 2017.

[10] Morten Linnemann Bech and Rodney Garratt. Central bank cryptocurrencies. techreport, Bank of International Settlements, 2017. URL `https://ssrn.com/abstract=3041906`.

[11] Vanessa M Perez. Americans with photo id: a breakdown of demographic characteristics. *Project Vote*, 2015.

[12] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008. URL `https://bitcoin.org/bitcoin.pdf`.

[13] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014), 2014.

[14] Arshdeep Singh, Gulshan Kumar, Rahul Saha, Mauro Conti, Mamoun Alazab, and Reji Thomas. A survey and taxonomy of consensus protocols for blockchains. *Journal of Systems Architecture*, page 102503, 2022.

[15] Drummond Reed Alex Preukschat. *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials.* Manning, 2021. ISBN 9781617296598.

[16] David Chadwick Manu Sporny, Dave Longley. Verifiable credentials data model v2.0, . URL `https://www.w3.org/TR/vc-data-model-2.0/`.

[17] DK Michael Lodder and Dmitry Khovratovich. Anonymous credentials 2.0, 2019.

[18] Brent Zundel. How does a verifier know the credential is yours?, 2021. URL `https://www.evernym.com/blog/how-does-a-verifier-know-the-credential-is-yours/`.

[19] Decentralised identifiers, . URL `https://www.w3.org/TR/2022/REC-did-core-20220719/`.

[20] Blockcerts: The open standard for blockchain credentials, . URL `https://www.blockcerts.org/`.

[21] Hyperledger Foundation. Hyperledger indy wiki, . URL `https://wiki.hyperledger.org/display/indy`.

[22] Veress One Project. Veress one, . URL `https://veres.one/`.

[23] Oskar Deventer et al. Peer did method specification: Blockchain-independent decentralized identifiers, . URL `https://identity.foundation/peer-did-method-spec/`.

[24] Daniel Hardman and Jason Law. Three dimensions of identity a simplified look at a complex topic, 2021. URL `https://medium.com/evernym/three-dimensions-of-identity-bc06ae4aec1c`.

[25] Jonas Gross, Johannes Sedlmeir, Matthias Babel, Alexander Bechtel, and Benjamin Schellinger. Designing a central bank digital currency with support for cash-like privacy. *Available at SSRN 3891121*, 2021.

[26] Hyperledger Foundation. Hyperledger ursa, 2018. URL `https://wiki.hyperledger.org/display/ursa/Hyperledger+Ursa`.

[27] Hyperledger Indy. Default auth map rules, 2018. URL `https://hyperledger-indy.readthedocs.io/projects/node/en/latest/auth_rules.html`.

[28] Hyperledger Foundation. Hyperledger aries, 2019. URL `https://wiki.hyperledger.org/display/ARIES`.

[29] Daniel Hardman. Aries rfc 0005: Did communication, 2019. URL `https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0005-didcomm`.

[30] Hyperledger. Aries cloudagent python, . URL `https://github.com/hyperledger/aries-cloudagent-python`.

[31] Sarah Allen, Srđjan Čapkun, Ittay Eyal, Giulia Fanti, Bryan A Ford, James Grimmelmann, Ari Juels, Kari Kostiainen, Sarah Meiklejohn, Andrew Miller, et al. Design choices for central bank digital currency: Policy and technical considerations. Technical report, National Bureau of Economic Research, 2020.

[32] Satish Babu and KM Abraham. Central bank digital currencies: policy and operational perspectives for india. *CSI transactions on ICT*, 9(2):85–94, 2021.

[33] Joseph J. Kearney and Carlos A. Perez-Delgado. Vulnerability of blockchain technologies to quantum attacks. *Array*, 10:100065, 2021. ISSN 2590-0056. DOI: `https://doi.org/10.1016/j.array.2021.100065`. URL `https://www.sciencedirect.com/science/article/pii/S2590005621000138`.

[34] Andrew Goddard. A view through the gun show loophole. *Rich. JL & Pub. Int.*, 12: 357, 2008.

[35] Raphael Auer, Codruta Boar, Giulio Cornelli, Jon Frost, Henry Holden, Andreas Wehrli, et al. Cbdcs beyond borders: results from a survey of central banks. *BIS Papers*, 2021.

[36] Imre Kocsis, Bertalan Zoltán Péter, Attila Klenik, and László Gönczy. Self-sovereign identities on central bank digital currency ledgers. Working paper, 2022.

[37] Bertalan Zoltán Péter. Zkp-based audit for blockchain systems managing central bank digital currency. *29th MinisymposiumoftheDepartmentof Measurement and Information Systems*, pages 70–73, 2021.

[38] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, page 1, 2016.

[39] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[40] Christian Grothoff and Alex Pentland. Digital cash and privacy: What are the alternatives to libra? 2019.

[41] Ralph C. Merkle. Method of providing digital signatures, 1979. URL `https://patents.google.com/patent/US4309569`.