



Web3: A Decentralized Societal Infrastructure

for Identity, Trust, Money, and Data

**TU Delft**

J.W. Bambacht

Web3: A Decentralized Societal Infrastructure for Identity, Trust, Money, and Data

by

J.W. Bambacht

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday February 28, 2022 at 11:00 AM.

Delft University of Technology
Ministry of the Interior and Kingdom Relations



Author

J.W. Bambacht

Supervisors

Dr. ir. J.A. Pouwelse, TU Delft
A. de Kok, RvIG

Student number:	4025016
Project duration:	May 1, 2021 – February 28, 2022
Thesis committee:	Dr. ir. J.A. Pouwelse, TU Delft, supervisor Dr. C. Lofi, TU Delft

This thesis is confidential and cannot be made public until February 28, 2022.

An electronic version of this thesis is available at <https://repository.tudelft.nl/>.

Contents

Abstract	v
I Article	0
I.1 Introduction	1
I.2 Problem Description	2
I.3 Related Work	2
I.4 Infrastructure	4
I.5 Design	6
I.6 Implementation	10
I.7 Experimental Analysis and Evaluation	13
I.8 Time Management	17
I.9 Conclusion	17
II Supplementary Material	21
1 Mobile Application	25
1.1 Views	25
1.2 Settings	26
1.3 Dialogs	26
1.4 Identity Onboarding	26
1.5 Notifications	27
1.6 Phone Permissions	27
1.7 Screenshots	27
2 Implementation Details	35
2.1 Data Transfer Protocol	35
2.1.1 Protocol Settings	35
2.1.2 Transfer Sets	36
2.1.3 Callbacks	36
2.1.4 Scheduled Task	37
2.1.5 Main Methods	37
2.1.6 Packet/Payload types	38
2.1.7 Packet Listeners	38
2.1.8 Experimental Analysis	41
2.1.9 Performance Evaluation	41

Abstract

A movement for a more transparent and decentralized Internet is globally attracting more attention. People are becoming more privacy-aware of their online identities and data. The Internet is constantly evolving. Web2 focused on companies that provide services in exchange for personal user data. Web3 commits to user-centricity using decentralization and zero-server architectures. The current digital society demands a global change to empower citizens and take back control. Citizens are locked into big-tech for personal data storage and their for-profit digital identity. Protection of data has proven to be essential, especially due to increased home Internet traffic during the COVID pandemic. Citizens do not possess their own travel documents. The European Commission aims to transition this governmental property towards self-sovereign identity, introducing many new opportunities. Citizens are locked into banks with non-portable IBAN accounts and unsustainable legacy banking infrastructures. Migration to all-digital low-fraud infrastructures and healthier competitive ecosystems is essential.

The overall challenge is to return the power to citizens and users again. The transition to a more decentralized Internet is the first crucial step in the realization of user-centricity. This thesis presents the first exploratory study that integrates governmental-issued travel documents into a (decentralized) societal infrastructure. These self-sovereign identities form the authentic base to a private and secure transfer of money and data, and can effectively provide trust in authenticity that is currently missing in online conversations. A fully operational zero-server infrastructure that incorporates all our requirements has been developed for Android using the P2P network overlay IPv8 (Tribler, 2021), and a personalized blockchain called TrustChain (Otte et al., 2020). It contributes to a reformed tech and financial sector that is more efficient and effective in serving the wider economy, and more resistant to bad behavior of all kinds. Creating such an infrastructure that is decentralized and anti-fragile is deemed crucial for the future.

I

Article

Web3: A Decentralized Societal Infrastructure for Identity, Trust, Money, and Data

J.W. Bambacht and J.A. Pouwelse

Distributed Systems, Delft University of Technology
February 24, 2022

Abstract—A movement for a more transparent and decentralized Internet is globally attracting more attention. People are becoming more privacy-aware of their online identities and data. The Internet is constantly evolving. Web2 focused on companies that provide services in exchange for personal user data. Web3 commits to user-centricity using decentralization and zero-server architectures. The current digital society demands a global change to empower citizens and take back control. Citizens are locked into big-tech for personal data storage and their for-profit digital identity. Protection of data has proven to be essential, especially due to increased home Internet traffic during the COVID pandemic. Citizens do not possess their own travel documents. The European Commission aims to transition this governmental property towards self-sovereign identity, introducing many new opportunities. Citizens are locked into banks with non-portable IBAN accounts and unsustainable legacy banking infrastructures. Migration to all-digital low-fraud infrastructures and healthier competitive ecosystems is essential. The overall challenge is to return the power to citizens and users again. The transition to a more decentralized Internet is the first crucial step in the realization of user-centricity. This thesis presents the first exploratory study that integrates governmental-issued travel documents into a (decentralized) societal infrastructure. These self-sovereign identities form the authentic base to a private and secure transfer of money and data, and can effectively provide trust in authenticity that is currently missing in online conversations. A fully operational zero-server infrastructure that incorporates all our requirements has been developed for Android using the P2P network overlay IPv8 [1], and a personalized blockchain called TrustChain [2]. It contributes to a reformed tech and financial sector that is more efficient and effective in serving the wider economy, and more resistant to bad behavior of all kinds. Creating such an infrastructure that is decentralized and anti-fragile is deemed crucial for the future.

I. INTRODUCTION

The online world is dominated by big-tech monopolies. These companies hold a relatively large amount of power in relation to citizens. As a result, citizens have difficulty protecting their data. The WhatsApp messaging platform is a motivating example of market failure as it violates terms of service over a long period [3]. An update of their terms of service [4], providing mother company Facebook access to more user data, initiated a migration to other platforms. Competitors focused on privacy and openness have barriers to market entry, no network effect, and compete against long existent closed protocols. Citizens and small(er) competitors [5] are powerless in this uncompetitive market.

Digitization generally weakens the privacy of citizens. The European Union started an ongoing effort into the General Data Protection Regulation (GDPR) [6] in 2016, targeting the misuse of privacy-sensitive data by companies. Many companies and platforms failed to comply to personal data protection,

resulting in over 900 filed cases of GDPR complaints, with a total value over 1.3 billion euros [7]. Due to such efforts, citizens have become more privacy-aware of their online data and identities [8]. Commercial entities have an incentive to minimize spending on cybersecurity [9]. The storage of personal data and weak security mechanisms of platforms are both at the expense of the user. These companies often gain revenue and value by selling user data for personalized advertisements. Users have no other option but to rely on the best intentions of the platform owner on their data.

In many situations, personally identifiable information of citizens is unnecessarily exposed. Government-issued documents are often required for institutions or organizations like banks, insurance companies, hotels, and employers, but lack secure handling and storage. Online governmental authentication mechanisms for digital identities are widely deployed by authorized institutions but have equal concerns. The owners of these identities are forced to accept and transfer all personal information. Both citizens and governments could benefit from the use of self-sovereign identities (SSI). As a result, citizens will be given the control over their own identities. Furthermore, external dependencies on authentication and storage can be eliminated. Offline identity authentication and identity attestations in a user-centric fashion can still offer the required identity authenticity. The possibilities of the SSI are additionally suited to a wide range of applications. We can profit from the self-sovereign identity within our societal infrastructure for the enforcement of authentic trust between identities in online conversations with the goal to reduce phishing or impersonating attacks. Other applications may include authenticated signing of digital documents, and validated storage of diplomas and COVID vaccination certificates.

Governments, banks, and tax offices have general insight into the bank accounts and transactions of citizens, often using big-tech cloud services. Not only is this a violation of citizens' privacy, but these banking services also have several deficits. The transaction costs are disproportional as debit card transactions range from €0,05 to €0,20 per transaction [10], and even bigger numbers for external payment services like iDEAL [11]. Cross-border payments even have increased costs and settlements. Cash is only applicable in offline payments and still serves as a store of value for some. It offers respectable privacy but is (slowly) fading away [12]. The current financial system can benefit from the adoption of blockchain technology. Central Bank Digital Currencies (CBDC) aims to provide a faster, more efficient, and cheaper alternative to electronic payments. With characteristics as privacy-awareness, pseudo-anonymity, and

unrestricted cross-border payments, CBDC can transform the financial system into a sustainable and frictionless system.

This research contributes the following: (I) design of a novel decentralized infrastructure that incorporates a self-sovereign identity, (II) applicance of the self-sovereign identity for identity attestations and authentic trust enforcement between participants of communication, (III) generic transfer of money, (IV) generic transfer of data using a custom-designed P2P data transfer protocol.

II. PROBLEM DESCRIPTION

The goal of this study is to design a novel decentralized societal infrastructure that incorporates a self-sovereign identity as an authentic base. We can additionally apply this identity in a useful way to facilitate authentic trust between users, and to privately transfer money and data with other identities. In a centralized infrastructure, platform owners are still able to collect metadata belonging to encrypted data. Removing these single points of failure reduces the violation of privacy and security of users.

The key aspect of our research is the application of citizens' self-sovereign identities. Self-sovereign identity is characterized by the ten principles of Allen [13, 14], that try to assure the users' control within its own SSI, with a balance between transparency, fairness, and protection. Governments currently have ownership and control of these identities. Various opportunities arise by moving the power back to the user. Firstly, the user is the owner of their own identity and can view and decide what information to share. Secondly, as governments don't have control anymore, less personal data management is required, less bureaucracy, and a cost reduction for facilitating the heavily secured infrastructure and authentication mechanisms. Thirdly, the decrease of personal data on central servers reduces the possibility of data breaches and theft. And finally an opportunity arises to replace visual identity document checks. Currently, identity documents are exposed upon request of some authority. Not only does the requested information become visible, but also the complete document. By using identity attestations, authorities are able to verify information of the identity without exposing the actual value, e.g. age validation of a bouncer in the pub. Communication channels lack trust in the authenticity of other participants' online identities. Since we have access to an official SSI, we can even apply this information to build trust. No social platform currently integrates government-issued identity information in such a fashion. Generally, online identities consist of a name, picture, phone number, or email, that altogether contribute to confidence in the authenticity of the other user's identity. However, these attributes are manually forgeable and can be impersonated. Malicious actors try to make their fake identities look as genuine as possible to mimic someone's identity. Without these editable components and with automatic acquiring of information from the SSI, we can enforce authentic trust to other participants.

The global financial infrastructure is failing to provide real-time cross-border payments and is dominated by monopoly

players with high profits and near-zero innovation. Financial privacy disappeared in the last decade(s). Governments, banks, and tax offices heavily supervise the bank accounts and transactions of users, albeit using automatic cloud services, removing the option to privately exchange money digitally. The transfer of money comes with disproportional costs, adverse cross-border payments in terms of speed and additional costs, and unwanted transparency. Many of these issues can be solved by the use of blockchain technology, and specifically Central Bank Digital Currencies (CBDC). With (almost) zero costs, transactions are executed between wallets anywhere in the world in a matter of seconds. Even internal use of blockchain for banks themselves will save about 10 billion dollars globally [15]. Despite the openness of blockchain transactions to practically anyone, pseudo-anonymity is maintained as the identity is not revealed.

Centralized platforms inherently lack the self-sovereignty of data of their users. As the use of central servers is profitable in terms of availability and synchronization, personal data and metadata of users are stored. Although the data itself is often encrypted, the metadata contains valuable information (sender, recipient, time, location). WhatsApp is the most used messaging app [16] and promises its users end-to-end encryption. That does not withhold them to store a vast amount of metadata. Even though these platforms make us think that they prioritise security and privacy, not giving up their centralized nature is the primary reason for the existence of cyber attacks [17].

Decentralization as part of a societal infrastructure with a self-sovereign identity introduces many new opportunities and advantages. The user obtains a more centric position, has more privacy, and remains in control and in possession of their own data and identity. A self-sovereign identity can even serve a wider range of applications such as building authentic trust with others in online conversations. Communication no longer includes the storage of personal metadata on servers. Governments require less highly-secured infrastructures, which reduces the overall costs for both governments and citizens. The transfer of digital money between identities has major advantages. A reformed financial system is more efficient, faster, and optimal for cross-border payments while preserving the privacy of citizens. In the following sections, we discuss the related work in III, the design and implementation of the first user- and identity-centric infrastructure is presented in sections IV, V, and VI. Our custom-designed P2P data transfer protocol is analyzed and evaluated in Section VII.

III. RELATED WORK

Self-sovereign identity provides citizens the control of their own identity. Citizens currently authenticate their identity to organizations and institutions, stored on the governments central server. DigiD¹ is the primary identity authenticator in the Netherlands and enables citizens to authenticate. Personal data is transferred from the government's server to the organization's server. This requires a perfectly secure connection and infrastructure on both sides. The citizen has

¹<https://www.digid.nl>

TABLE I: Comparison with related works

	P2P	open source	E2E Encryption	metadata	requirements	attributes used for trust enforcement	wallet	maturity ^a	note
Centralized	WhatsApp [18]	✗	✗	curve25519	✓	phone number	phone number, name, profile picture and status	✗	high
	Facebook Messenger [19]	✗	✗	curve25519	✓	Facebook profile	Facebook profile, name, profile picture	✗	high ^b
	WeChat (QQ) [20]	✗	✗	✗	✓	phone number	phone number, name, profile picture and ID	money	high
	Telegram [21]	✗	✗	MTPProto	✓	phone number	phone number, name, username, profile picture and status	✗	high ^b
	iMessage [22]	✗	✗	NIST P-256 curve	✓	Apple profile	phone number, name, email, profile picture	✗	high
	Signal Messenger [23]	✗	✓	curve25519, curve448	minimum ^c	phone number	phone number, name, profile picture and status	crypto	high
Decentralized	Session Messenger [24]	✗	✓	curve25519, curve448	minimum ^c	✗	name, profile picture	✗	medium ^d
	Status.im [25]	✓	✓	curve25519	minimum ^c	✗	username, profile picture	crypto	high
	Sylo [26]	✓	✗	curve25519	✓	✗	name, profile picture	crypto	high ^e
	Berty [27]	✓	✓	curve25519	minimum ^c	✗	name, profile picture	✗	medium
	Our design (Section V)	✓	✓	curve25519	minimum ^c	official Identity	identity name and verification status, profile picture	crypto	medium

^athe current state of development in terms of completeness and usefulness

^bE2E encryption not enabled by default

^cno storage of metadata, only required for routing

^dfork of Signal Messenger, onion routing for metadata anonymity, undelivered messages stored one of the distributed service nodes

^eeveryone can set up node and will be rewarded in crypto token SYLO

no control over what information is shared. Additionally, these authentication mechanisms are disproportionately expensive and credited €0,13 for all 545 million successful authentications in 2021 [28, 29], especially due to increase of COVID-related authentications. Self-sovereign identities can be applied to mobile applications that replace authentication services like DigiD at a fraction of the cost. The first example is IRMA², an operational mobile platform that fetches the identity from the governmental servers once and stores the SSI and other personal information locally on the phone. The authentication to organizations can instead be performed offline using the stored SSI. The user remains in control by seeing the requested and shared information. Sovrin Network³ uses the same methodologies but instead enables other developers to build their own SSI application on top of their blockchain-based ecosystem. Some situations require the option to revoke the self-sovereign identity, for example when the identity document is lost or stolen. Both IRMA and Sovrin apply (centralized) authorities to revoke identities. This is a violation of the principles of SSI as it should be an authority-free system. The work of Chotkan [30] incorporates distributed attestations for self-sovereign identities that includes offline revocation. Offline verification of attestations offers increased privacy and robustness.

As mentioned before, this paper presents a *novel* decentralized infrastructure and incorporates a government-issued identity within a messaging platform. Many other platforms exist,

both centralized and decentralized, that apply at least some of the key points of this paper. Table I portrays a (non-exhaustive) list of significant competitors in the market. The difference between the centralized and decentralized platforms shows a clear clustering. All centralized platforms require a privacy-sensitive asset and attributes that are shared with contacts to identify and enforce trust. The decentralized platforms are examples of Privacy by Design [31] as they try to minimize the (centralized) storage of data and leakage of privacy-sensitive information. Decentralized platforms have not explicit initial requirements and the trust attributes are limited to manually chosen names and profile pictures.

WhatsApp [18], Facebook Messenger [19], and WeChat [20], are fully centralized platforms that store metadata of their users. The industry-standard E2E encryption curves are used by all platforms but Telegram [21], that uses their self-designed protocol, and the unencrypted WeChat. WeChat, which is monitored by the Chinese government, incorporates strong censorship and interception protocols for data exchanged by its citizens. This degree of monitoring is not present in other (centralized) platforms. Governments even oblige these centralized platforms to share their stored metadata or apply censorship in some situations [32, 33]. Signal Messenger [23], which is centralized but specifically designed with privacy in mind, does not store any personal information. However, central servers are necessary for routing and functionalities like account recovery. Account activation is validated using the phone number or email address of the user. Characteristically, most of the centralized platforms don't provide full transparency and do not disclose the complete

²<https://irma.app>

³<https://sovrin.org>

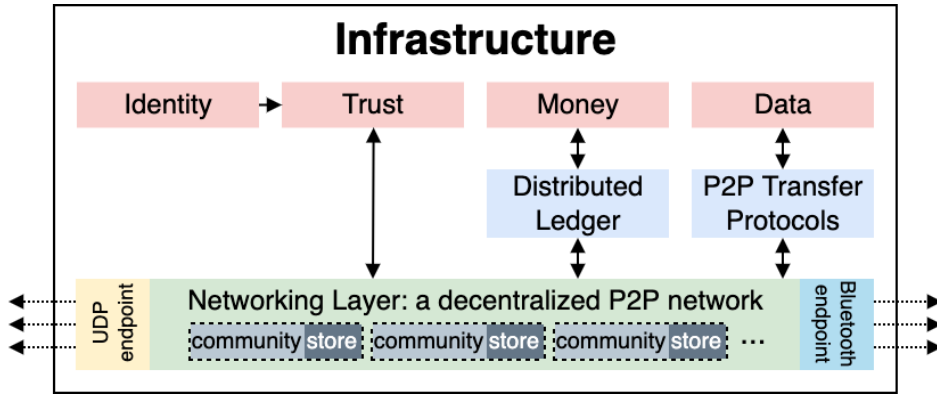


Fig. 1: General overview of the infrastructure

structure of their platform.

Decentralized infrastructures try to realize anonymity by minimizing the metadata in the network. Session Messenger [24], a *decentralized* fork of Signal Messenger, attempts to provide anonymity and preservation of privacy using onion routing [34]. It makes it nearly impossible for any intermediary (node) to derive both the sender and receiver of the message. Onion routing is not suitable in a (fully) P2P network as peers only know a limited number of other peers and do not (necessarily) communicate with nodes. Status [25], Sylo [26], and Berty [27] are decentralized, P2P, secure, minimize leakage of privacy-sensitive information, and have no initial requirements. Status is build on the Ethereum network and incorporates its own utility network token to provide paid features to users. Like Session Messenger, undelivered messages are stored on nodes that obtain your IP address to deliver it later. This characteristic contradicts the principles of privacy. Sylo is a fully operational platform that is not protected against possible leakage of metadata and does not provide full transparency to its users. Berty is secure and transparent, minimizes leakage of privacy-sensitive information in terms of metadata and requirements, and therefore suits our requirements best, but is currently still underdeveloped.

There are many implementations with the same design characteristics. The idiomatic platform is decentralized, does not require temporary storage of messages on nodes due to its P2P nature, incorporates trusted encryption, does not require and store metadata, and has no redundant identifiable requirements. Our design uniquely adds a self-sovereign identity, providing various functionalities. To achieve higher trustworthiness, trust enforcement attributes must not only consist of manually forge-able components. Furthermore, the platform must contain private and secure mechanisms for the transfer and storage of data and digital money.

IV. INFRASTRUCTURE

The dominant problem of market-leading societal platforms is their centralized nature. Decentralization targets many weak spots of centralization, such as providing private and secure storage of data in a distributed way. Even a decentralized network allows the storage of metadata by the nodes that are traversed on its path to the destination. Some nodes may even

sell metadata to third parties. A P2P network enables direct communication with peers without any intermediary. As no intermediary is able to act as a middle-man or adversary, it serves as an extra layer of protection against malicious or intentional behavior. The communication is only secure if the message, or data, is encrypted.

We also require a component to store and exchange data in a distributed manner. One of the requirements of distributed systems is synchronization across many independent nodes. This is difficult to realize in systems that include peers that are not connected at all times. To enable the transfer of (digital) money, the infrastructure requires a persistent and decentralized storage of data that does not require continuous synchronization for all peers. Blockchains store transactions between two wallets in a permanent and uneditable manner. Every transaction on the blockchain is entangled to its previous block, making it a reliable chain of tamper-proof assets. This form of storage is a fast, lightweight, and structured alternative to conventional storage, albeit visible to everyone. Every transaction can be back-traced to create a well-organized overview, which is well suited to serve as a wallet. The infrastructure additionally requires data transfer protocols to exchange data. Figure 1 portrays a low-level overview of our infrastructure.

Peers constantly observe other peers in the network. The infrastructure requires anonymous peer identification as it is not desirable to spread personal information to (unknown) peers in the network. To ensure a secure communication channel, the principles of the CIA Triad [35] must be in place. The objectives of a secure system include *Confidentiality*, *Integrity*, and *Availability*. Confidentiality ensures that information is only accessible to authorized parties. Digital signatures ensure the integrity of the information by providing proof that it originates from the sender and has not been altered by any third party. Availability ensures that information is available to authorized parties at any point in time. In P2P networks, only the last principle is difficult to realize due to its dependence on the connectivity of individual peers.

Confidentiality requires the encryption of information. Public-key cryptography [36] is the most commonly-used mechanism for secure communication. Confidential exchange of messages and data additionally enables the identification

of peers exposing private information. Each peer has a public-private key pair. The private key is cryptographically generated once and only be known to the owner. The private key decrypts the encrypted data. The public key is mathematically derived from the private key and may be public as it is computationally infeasible to derive the private key from the public key. The public key provides several applications in our infrastructure. Foremost, the public key of the intended recipient is used for the encryption of the data. Secondly, digital signatures prove the authenticity of data and can be verified using the public key of the signatory/sender. And lastly, peers can be identified and distinguished using their public key.

A. Networking Layer

Our infrastructures requires a networking layer to handle outgoing and incoming communication with peers. A P2P networking layer that is authenticated and privacy-aware is IPv8 [1]. IPv8 is developed as an academic successor of IPv4 and attempts to overcome IPv4's weak characteristics and growing number of problems. The objective of IPv8 is to provide communication in a zero-server infrastructure, equal status and power within the network for everyone, and perfect secrecy with E2E encryption. IPv8 can establish connections to peers, even for devices connected behind NAT or a (strong) firewall. The endpoints of the networking layer are independent of any central infrastructure.

The aim is to minimize the exposed metadata. For this reason, data packets can't be broadcasted on the network, in the hope that other peers can deliver the packet to the intended recipient. Some P2P systems employ distributed nodes to deliver the data in this situation. These nodes temporarily store data and metadata until delivered. Although this addresses the availability of information in the network, it exposes the storage and possible leakage of metadata. The metadata of a packet should (ideally) only contain information about delivery, that is, the receiver's public key or IP address. The metadata that is not relevant for delivery should be encrypted with the data. The risk of exposing privacy-sensitive information in our P2P network is minimized as peers directly communicate without serviceable nodes or peers. Peers change connectivity status or change their network address regularly. In these situations the peers announce their new address to all former peers to keep the previous connections alive. Peers also connect to random peers to broaden their network reach. While this may sound contradictory, it does not violate their privacy. No personal information, including the intentions and previous communication histories, can be deduced.

IPv8 applies the concept of network overlay or *community*. This enables developers to build applications on top of the base networking layer by creating their own community. We need several communities in our infrastructure to satisfy our requirements. The base community includes all functionalities related to peer connectivity, communication, data serialization, and encryption. The IPv8 networking layer combines responsibilities from several layers of the OSI model [37]. Communities that require the storage of data, can use an additional *store* that handles the interaction with a database. A

discovery community handles the discovery and connection of (new) peers present in the same community. Every community and peer may have a different list of connected peers. The communication between peers in the network and community is handled by endpoints/sockets. IPv8 provides support for both online and offline communication using UDP and Bluetooth endpoints. The support for offline communication increases the reliability and applicability of the platform, especially in areas with low network coverage.

B. Distributed Ledger

The infrastructure requires a distributed ledger that handles and stores the financial transactions. The permissionless and scalable distributed ledger TrustChain [2] is already integrated as a community in IPv8. TrustChain is capable of sending and receiving trusted transactions between peers. The blockchain-based data structure is a tamper-proof immutable chain of transactions. No central authority has control over the transactions. Every peer implements a personalized chain that contains only blocks related to that peer, i.e. either sent or received by the peer. TrustChain has three basic functionalities: transmit/receive blocks, broadcast blocks, and chain crawling. The transmit and receive process merges both parties in one transaction, see Figure 2. The initiator (p_0) creates, signs,

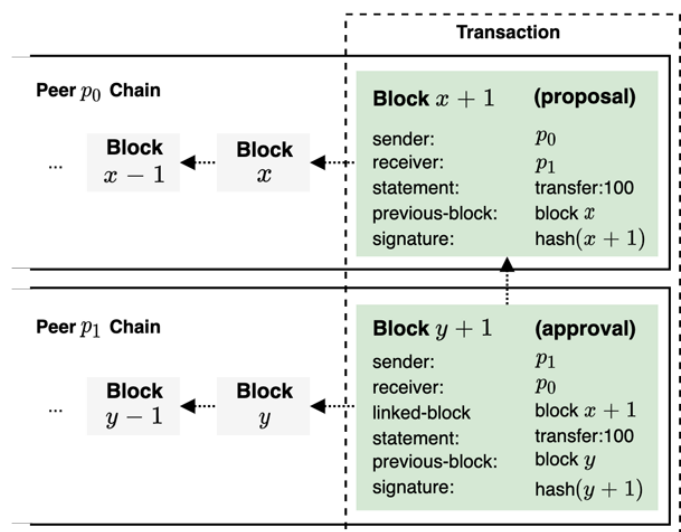


Fig. 2: Transaction in the ledger

and sends the *proposal block* to the counterparty (p_1). On receipt of the *proposal block*, (p_1) creates, signs, and sends the *agreement block* back to p_0 . Both blocks are linked by the public key of the counterparty and can be considered as half blocks that together form a transaction. During the process, the integrity of the received blocks is validated and both parties add the half blocks to their chains. The transaction is complete when both parties received and signed both half blocks, i.e. both acknowledged and accepted its contents. The broadcast functionality enables to transmit the block to all connected peers. Chain crawling is the retrieval of a peer's chain using its public key.

V. DESIGN

We can divide the design of the platform into four pillars: identity, trust, money, and data. These elements are integrated within the infrastructure of Section IV to create an ecosystem that combines these elements seamlessly. The elements, described in the following sections, must satisfy the requirements and functionalities that are deemed necessary for a self-sovereign, secure, and privacy-aware communication platform.

A. Identity

Identity is an integral part of citizens when it comes to ownership over their self-sovereign identity. Integration of government-issued travel documents in a self-sovereign manner introduces various new opportunities. One of the major applications is the authentication to online organizations. The citizen controls the exchange of its own identity information to organizations instead of the conventional online governmental authentication that blindly transfers all available information. Authentication can only serve its purpose if the information within the self-sovereign identity is authentic. IRMA, the application mentioned in Section III, achieves authenticity by fetching the identity information from the government's central server once. As we desperately want to eliminate external dependencies, this method is not suited to our system. Every (adult) citizen is obliged to legitimate himself with a passport or identity card upon request by the authorities. As it is mandatory to possess such a document we can apply these documents as an authentic base for our platform. The identity written in the machine-readable zone (MRZ) of the document, contains the same personal information as on the government's server.

The identity document onboarding process is executed in two consecutive steps as in Figure 3. In the first step the user must scan the text in the MRZ of the document using the camera of the device, as in fig. 3a. A combination of AI and check digits (embedded in the MRZ) ensure a syntactically valid, not necessarily correct, recognition of the identity information that is saved on the device. The second step is extremely important as it proves the authenticity of the document digitally. This validation step determines the correctness of the scanned attributes. Most current documents are equipped with a built-in biometric chip in which the information is embedded. The NFC chip of devices enables them to communicate with the document. The device must steadily be placed with its back to the document until all information has been transferred, see Figure 3b. The biometric chip contains multiple layers of protection against for example eavesdroppers and modification. The protection prevents the unauthorized reading of the document using for example NFC skimmers. The authentication requires the document number, date of birth, and date of expiry as passwords, as obtained in the initial step. After the connection has been established, the biometric chip transfers all requested attributes to the device. All stored attributes together form the self-sovereign identity of the user. The onboarding process is deemed authentic and secure because (I) a physical document is required and the

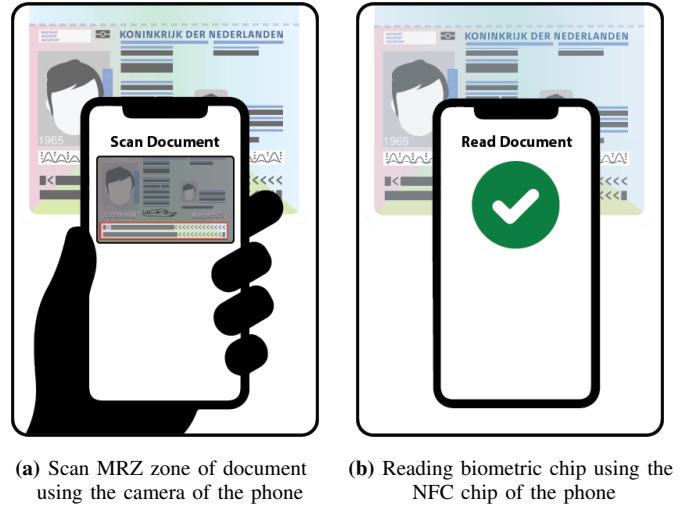


Fig. 3: Identity document onboarding process

attributes displayed on the card must match the content in the biometric chip of the card, and (II) the biometric document is widely applied for governmental purposes and international traveling, without excessive vulnerabilities [38]. One issue that remains is the possibility to revoke access to a self-sovereign identity. Millions of documents get stolen or lost yearly [39], and risk of being used by others. There is no way to recognize and revoke access without knowledge from a central server. The self-sovereign identity must be valid, just like physical documents, until the expiration date of the document.

A different situation arises when the phone has no support for the NFC chip, defectively or physically. There does not exist an (offline) method to obtain the identity authentically by only scanning the document. As the biometric chip performs the validation of the document, a malicious actor can forge the complete MRZ to its preference. No identity-related functionalities can be trusted to contain truthful and authentic information. There is no other option to either disable all these identity-related functionalities for these devices or to be dependent on the government's central server to obtain the identity.

In real-world situations, it is sometimes mandatory to show or even make a copy of your physical identity document to verify your identity or to serve as insurance. The authority is not only capable of unnecessarily viewing the requested attribute(s), but also other attributes on the document. This is a direct violation of citizens' privacy and can even lead to identity theft or misuse of a person's integrity. Citizens are forced to trust this authority to handle and store their identity secure and with care. Self-sovereign identities introduce the opportunity to use verifiable claims. Verifiable claims are claims about information that can be verified using attestations. Chotkan [30] designed a framework that incorporates revocable verifiable claims without revealing the actual requested piece of information using zero-knowledge proofs. To apply verifiable claims in a trustworthy manner, the information from the self-sovereign identity must, again, be authentic.

The focus should not only be on data in transit, but also on data at rest. In the latter, the data is stored somewhere without anyone currently accessing it. The data is often stored as a file or in a database. In our case, the identity must be secured significantly without risking identity theft. As mentioned before, our design applies public-private keys for encryption. The storage of the identity can easily be encrypted using the private key of the user, while only allowing the application to decrypt when absolutely required. That means that no one is able to access the contents outside the application environment as long as the private key is insusceptible. Biometric protection (face recognition or fingerprint) or the use of passcodes, can be applied as another layer of protection against unauthorized access to sensitive data. It can also serve as confirmation for irreversible actions like the transfer of sensitive information or money.

B. Trust

Platforms have to deal with several types of trust. The first natural form is trust in a system or platform. This is the case for nontransparent centralized platforms. As a user, you want to have faith that your personal data is handled and stored with care. This is often one of the primary problems with centralization. As all user data is stored on the platform's servers, you must have confidence that the data, including metadata, is protected with the highest security standards, exchanged in encrypted form, and not sold to any third parties. If no good alternative platform exists, or because no friends use other platforms, the user has to decide whether to continue to use the platform and neglect the privacy-related issues, or not use the platform anymore. Often the first choice is selected as people value the use of the service more than their own privacy. Decentralization (almost) completely eliminates this trust, or distrust, as there is no central component or authority that decides over you and your data. Also, a platform that is open source is generally more trusted as there are no hidden surprises due to the reviews of experts. We must take notice that in some networks, malicious actors actively crawl metadata in order to gain knowledge about confidential information. Communication in P2P networks, in the form of network packets, is directly transmitted to the network address of the recipient, making this problem almost redundant.

In messaging and societal applications another form of trust arises: the trust in the real identity of the other participant. The confidence in the authenticity of the contact is determined based on various factors like the (online) identity of the contact, the (dis)similarity in the way they communicate, and the discussed topics. The style of writing and difference in for example punctuation and the use of capital letters may also play a role in recognition. Unfortunately in most applications, personal information can easily be forged or stolen from people's (real) online identities. If we again look at Table I, most attributes for trust enforcement of centralized platforms are forgeable. Spear phishing [40] is a cyber attack in which individuals are targeted with the explicit use of personal information to gain knowledge or access to (more) sensitive information. In applications like WhatsApp, it is

possible to migrate from one phone to another. As this is a convenient feature, it also exposes the risk that hackers can take over your account on their phone and communicate with your contacts instead. This has the deficit that hackers are able to use these accounts to steal confidential information or even request money from trusted unsuspecting contacts. These hackers attempt to mimic as much confidential information of the hacked person to not arouse suspicion or to simply gain trust with their new victims. For this reason we did not include migration in our platform (yet). Another situation arises when the phone is stolen or lost. As we don't have control anymore, it is (similarly to other platforms) possible to impersonate someone on a lost or hacked phone.

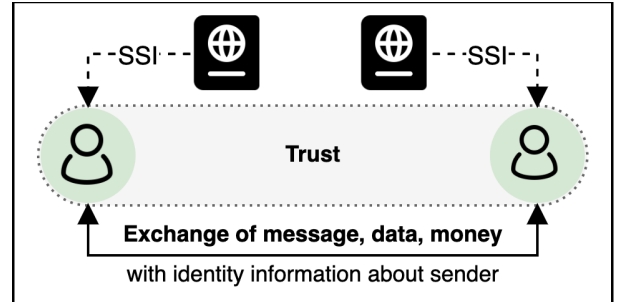


Fig. 4: Building trust using the self-sovereign identity

The challenge is to exchange the right amount of trust to the recipient of your message without excessively exposing private information. In the initial phase of the conversation, especially if the users connected in some online way, trust (or distrust) can play a major role. As valid self-sovereign identities are incorporated in our design, we can access and apply this authentic information. In a normal, physical first meeting, one would introduce themselves by their (first) name, and indirectly with facial expressions, the sound of their voice, and the overall atmosphere. These aspects are not available in the digital world without extra efforts. The only identifiable information that we can exchange is a person's name and photo as embedded in the self-sovereign identity. These attributes can provide the authenticity that our system requires by sending it to the other contact. To provide proof of its authenticity, we can include the identity verification status. The verification status formally denotes the trustworthiness of the name and photo, while it technically denotes the use of the biometric and NFC chip. A different situation arises when users can't use the NFC chip. These users can't be verified as a result, and are able to choose their own name and photo to allow them access and use of the platform. The contacts will in turn be notified of the unverified status, implicating they should act cautiously.

TABLE II: Trust enforcement combinations for identity name

Combination	Example
I {First Name}	Timothy John
II {Last Name}	Berners-Lee
III {First Name} {Surname[0]}	Timothy John B.L.
IV {First Name[0]} {Surname}	T.J. Berners-Lee
V {First Name} {Surname}	Timothy John Berners-Lee

Of all information in the self-sovereign identity only the name and picture, and possibly the age and gender, are suitable to enforce trust. The transfer of too much confidential information can lead to malicious misuse of them or their contacts. We should therefore limit the exchange of information. We can compose various combinations of the given and last name that aim to provide trust, see Table II. While combination I and II are too general and unidentifiable, the use of the full name in V, as identically embedded in the self-sovereign identity, is an easy source for malicious actors to take advantage of. The combinations III and IV include both the first and last name in a modified form and provide a more personal and identifiable view without exaggerating. The identity of a person is more decisive by its last name due to its uniqueness, and therefore combination IV fits our purpose best for the use as the trust enforcement attribute along with the picture. The exchange of the age and gender has been considered, but increases the risk of impersonating.

The name and photo attributes of the peer's identity are sent encrypted along with every message. Upon receipt, the system is able to detect differences with the information of the currently stored state. Initially, the state is empty. The recipient of the first message will be notified in a recognizable manner that the identity of the contact has been determined. For every other message, if at some point the state changes, the user will receive similar notifications stating that the information has been updated. This mechanism makes sure the user always has knowledge of the sender's *formal* identity. It is, however, impossible to notice difference in case a phone is stolen or accessed unauthorizedly without an alteration of the identity. On-device authorization is desirable using for example biometric protection. As long as biometric protection is in place, it should be difficult to impersonate. Also, a mechanism that requires the user to regularly verify its identity using their physical document could help to reduce misuse. Both options are currently not part of our initial design but serve as improvements on authentic communication. It is equally important to not only focus on building trust but to preserve the privacy of the receiver as well. The platform will *never* share identity information without having sent a message or transaction first. This reduces the risk of malicious actors purposely attempting to fetch names and photos linked to particular public keys.

C. Money

As the need for financial privacy grows, many Web3 applications integrate the transfer of some sort of value in the form of cryptocurrencies or NFT's. Governments of nations and unions are currently exploring the economic and technical feasibility of Central Bank Digital Currencies (CBDC) [41, 42, 43, 44]. Money in the form of digital currencies that reflect on their native currency, often also referred to as stable coins, could provide a fast, cheap, and private exchange between participants. Other cryptocurrencies are not suited for this purpose as they appear to be extremely volatile, therefore lacking the consistency for a safe store of value. CBDC has three main characteristics: it is a digital currency, it is issued

by a central bank, and it must be universally accessible. As these currencies will be legally recognized and backed by their governments and central banks, their introduction and effect on the financial system are heavily tested. If somewhere in the future, countries decide to become cashless, these currencies must be creditable to replace coins and banknotes. China is already in the advanced stages of the development of its CBDC and is testing the implementation of its digital Yuan wallet in a second pilot [45]. The Bank of China will still include regulation for larger transactions and seek only to collect personal information that is legally required. Compared to the principle of cryptocurrencies, which strives for pseudo-anonymous transactions, China doesn't make an effort to give up its (financial) regulation.

Governments, banks, and tax offices shouldn't have implicit insights into transactions of CBDC's. The privacy of the users will be preserved up to a certain level. As most ledgers and blockchains are transparent, transactions on the chain are visible to others, and can even view or attempt to trace back the wallet balance. Blockchain still provides pseudo-anonymity because participants of transactions are often only identified by a public key. No further personal information is attached to transactions apart from the sender and recipient and some unidentifiable transaction contents, statistics, and possibly some other (encrypted) data. Governments will not attempt to regulate and gain insight into these transactions because, similarly to cash, it is simply not feasible to do so. If we compare blockchain transactions with current digital payment solutions, it is definitely a step forward, while conventional cash remains the most private and anonymous form of payment and store of value. Not only would the use of CBDC's contribute to new innovations and direct accessibility of money without any external dependence, it may even serve some of the fundamental financial primitives (lending, borrowing, liquidity).

Currently, many external services or banks provide the functionality to create payment requests. Its creator shares a link to all participants that redirects to the payment portal of the service. This not only creates an additional dependency on the use of a centralized (paid) service but also opens abusive opportunities for malicious actors. Many fraud cases [46] using payment requests make the service vulnerable, especially for unsuspecting persons or the elderly. P2P digital payments can solve this problem by eliminating the dependence on the middle-man. The transaction or payment request is instead directly sent to the other peer, in an online or offline fashion. This not only makes it faster and cheaper but also offers a more private exchange of value.

Our design incorporates an existing implementation of an offline-capable euro CBDC called EuroToken [47]. It utilizes the distributed ledger TrustChain [2] that builds upon the technologies of the IPv8 network [1]. The EuroToken protocol tries to offer a scalable, privacy-aware, and cheating-resistant system for the exchange and storage of transactions. Every block stored on the ledger contains a single transaction that states the transfer of funds from one to another. A block is cryptographically linked to its predecessor and therefore preserving a chain of chronological and valid blocks. Transactions

are generally settled within seconds, independent of within- or cross-border payments, but require the connectivity of both parties to completely settle the transaction.

One of the aspects of a tokenized system is the acquisition of tokens. For the system to be useful, it requires at least one option to buy and sell these tokens. EuroToken incorporates a *central* exchange portal that allows users to exchange money on their bank account with EuroTokens, in both directions. The user and portal included in the exchange have to announce themselves to each other using their public key and additionally the transfer amount. The portal creates a request to the user that is automatically handled and stored by the protocol. Although it is not desirable to integrate centralized components in the system, it is considered an exception for the system to be functional. Improvements on the protocol could include distributed portals that are managed by random peers instead of a single entity.

After the initialization of the application, the wallet is ready to send and receive tokens. A balance is obviously required to send tokens. Tokens are either received by other peers or acquired using the exchange portal. The balance of a wallet is determined and validated using the blocks on the ledger. Our design fully integrates and stimulates the transfer of money between peers as it can be transferred directly from the wallet or indirectly from within a conversation with another peer. The integration of internal payment requests conveniently enables peers to request tokens from other peers. A transfer request differs from a transaction as is not formal and binding, and only contains the amount and the public key of the requestor.

D. Data

One of the key applications of secure and private communication is the decentralized transfer of data. Data is the collective name for everything that can be expressed in the form of human-unreadable blobs, a Binary Large Object. These blobs can in turn be deserialized into a format that is readable for humans, e.g. images or text documents. Messages, and even transactions, can thus be classified as data as well. The current implementation of IPv8 contains a basic data transfer protocol. This protocol is able to send blobs to other peers, containing metadata and data. The transfer of data in form of messages and small blobs is fast but unreliable. However, the protocol has proven to be limited in terms of performance for larger-sized blobs. For proper use of our platform, it was deemed necessary to design a custom data transfer protocol that provides performance and reliable exchange of data, in a secure and private fashion.

The designed data transfer protocol aims to provide reliable and optimal performance for everybody in a *progressive* and *adaptive* fashion. The protocol is fully integrated into IPv8 and available to every community if necessary. Due to limitations, IPv8 (currently) only allows one concurrent transfer between two peers. The protocol is based on the principles of TFTP [48], the Trivial File Transfer Protocol. TFTP is a simple and connection-less data transfer protocol, with the consequence that it is unauthenticated and no security mechanisms are provided. The transfer of confidential data in (external) networks is unsafe, and therefore not recommended. Dedicated

connections with other peers cannot be established in P2P networks due to the lack of end-to-end connectivity. The long-existent TCP protocol is connection-oriented and much more reliable, but is not suitable for that reason. For a connection-less protocol, the application of the User Datagram Protocol (UDP) [49] is an obvious choice. The question is how to effectively integrate the unreliable UDP in the design of a reliable data transfer protocol. For some purposes, for instance live video streaming, the loss of single packets does not impact the result as single video frames or pixels are simply skipped. In the case of our platform, the loss of packets would have the deficit of distorted and unusable data. Our protocol must therefore keep track of unreceived packets and request retransmission. The operation of the protocol is very basic and only consists of four different packet types. All packets are encrypted and in principle only decryptable to the intended receiver. The normal operation of the protocol is portrayed in Figure 5. The sender of the transfer first has to request to write

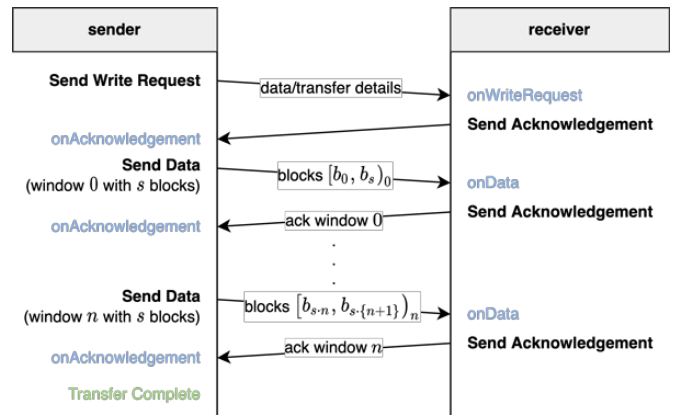


Fig. 5: Normal operation of the data transfer protocol

data by sending a **WriteRequest** payload/packet to the receiver. With this packet, the sender additionally announces all transfer- and data-specific details to make sure both parties have the same understanding. The receiver confirms this request by returning an **Acknowledgement** packet. This acknowledgment triggers the sender to start the transmission of data with **Data** packets. The data cannot be sent in one piece for multiple reasons. Firstly, the maximum UDP packet size is strictly limited to 1500 bytes due to the Maximum Transmission Unit (MTU) of the Ethernet [50]. The IPv8 protocol additionally requires a header of approximately 177 bytes to each block for routing, identification, and security purposes. Our **Data** payload header also requires identifiable information in the form of a block number, nonce, and some other attributes. This means that the data inside the packet can be roughly somewhere between 1200 and 1250 bytes. Secondly, since UDP is unreliable and packets are not guaranteed to be delivered, the transmission of the data at once (if technically feasible) would be too much of a gamble to arrive, especially for large blobs. The protocol is required to split the data into small(er) pieces to fit the packets, creating *blocks*. Each of the blocks has a particular size in bytes and all blocks concatenated in the correct order represents the data. To reliably transfer data from one to another, we have

to confirm the receipt of the data packets by again sending an **Acknowledgement** packet. To not unnecessarily wait for confirmation and delay the transfer, the acknowledgment (and any other packet) must be received within a certain interval before the previously sent packet is retransmitted. The principle of windowing allows multiple packets to be sent at once without requiring an acknowledgment for every single packet. This increases the performance majorly as most of the idle time of the sender is spent waiting for confirmation. The window size of the transfer defines the hard limit on how many bytes or, equivalently, the number of blocks if we take the block size out of the equation, can be sent within every window without intermediate acknowledging. After the last block of the window has been received, the protocol sends an acknowledgment to the sender. It does not necessarily mean that all blocks within that window have been received, as some arrive later and some will not be delivered at all. In that acknowledgment, the receiver includes the block numbers that have not been received (yet). The protocol could decide to only transmit individual packets and wait for the confirmation of receipt. This, however, has several disadvantages. Firstly, the transfer speed is significantly decreased as additional transmit and acknowledge stages are added, including waiting time. Regularly UDP packets arrive late or not at all, meaning that for a good part of the windows the unreceived blocks have to be retransmitted, even for single unreceived blocks. Secondly, by staying at the current window, it may occur that some of these packets have trouble being delivered, and the transfer may not progress further for a longer period (in terms of transfer time). To account for these drawbacks, the protocol will always try to *progressively continue* its normal operation. This means that these unreceived blocks will piggyback with the next transfer window in the hope to be delivered without causing an overall delay. For every next window that passes, the confidence in these blocks being delivered increases. If all windows are sent, it may occur that there are some unreceived blocks left. In this case, the protocol will remain in the transmit phase of the last window until all blocks have been confirmed. The transfer is considered complete, for both the sender and receiver, after receipt of the acknowledgment (sender) and last block (receiver).

The receiver may not be able to adhere to the write request because either the data does not comply with the allowed size between zero and a predefined limit, or both peers try to start a transfer at the exact same time. In these situations an **Error** packet is returned that contains the reason of refusal. Both the sender and receiver have no other option but to terminate the transfer. During the transfer, it regularly happens that no response is received for sent packets. Both the sender and receiver therefore use their own retransmit interval and retransmit attempt count. A retransmit is scheduled if the time between the last transmit and the allowed interval is exceeded. We don't want the protocol to retransmit infinitely during disconnectivity. This is prevented using the attempt count that only allows a maximum number of *consecutive* retransmits. If after retransmission the other peer suddenly responds, the count is reset and the protocol continues normal operation. If the peer appears to be unresponsive, and the number of

consecutive retransmits exceeds the attempt count, the transfer is considered timed out and will be terminated. In worst-case scenarios, often where the connection is unreliable or slow, the transfer of packets and confirmations may take too much time and can timeout the transfer. These connections could probably benefit from a lower window size as fewer blocks have to be transmitted and confirmed within the same time. The protocol *adaptively* downscales the window size of a transfer after every timeout to give slower connections a better chance of success. Although it is important that the protocol is suitable for everyone in every situation, it is undesirable to lower the performance for everyone, due to a minor number of failures. We, therefore, chose to initially apply the optimal transfer settings for every transfer and lower the performance when required. The default parameters of the protocol that provide an optimal performance are analyzed in Section VII. An optimization to the protocol can include the application of sliding windows for a more dynamic and optimal performance.

VI. IMPLEMENTATION

In Sections IV and V we've discussed the infrastructure and design of the main components that form the basis of our novel platform. The use of both IPv8 and TrustChain has proven to be a valuable fit for our infrastructure. The already existing implementations are applicable to a certain level as they have primarily been developed to serve as separate proofs-of-concept. These implementations additionally lack refinement and general cooperability and applicability with the other functionalities. In this section, the contributions to our complete infrastructure are discussed, as well as the changes to the existing implementations, and how they are integrated into our platform to provide a well-designed and well-functioning platform. The implementation is additionally concerned with the UX and UI design of the platform. The platform has been integrated within the TrustChain superapp⁴, an Android mobile application developed in Kotlin that contains many different small applications built on IPv8 and TrustChain. The data transfer protocol is integrated into the IPv8 stack⁵ and is available to every community.

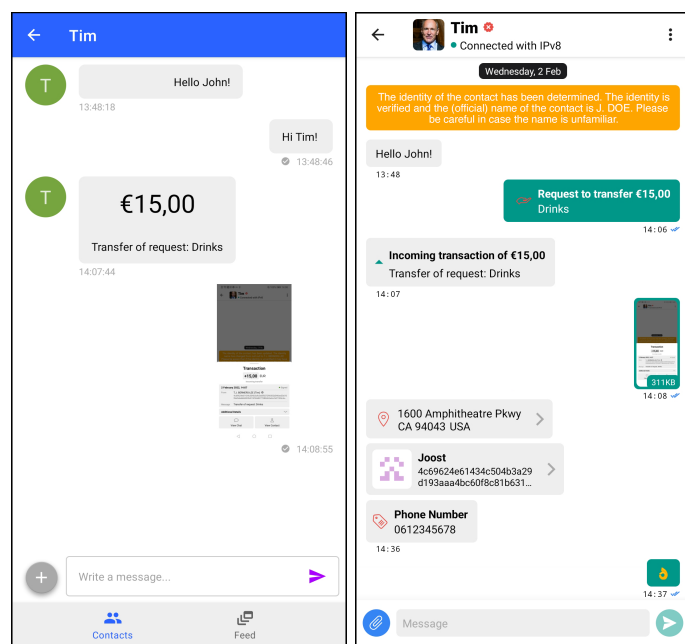
The platform consists of five general views that are interconnected using a navigation bar and direct links. The initial view, the wallet overview, as in Figure 8b, provides a widget-like overview of the identity, exchange, and chats components. For each widget, there is a separate view that includes all related aspects. The information in these widgets is carefully selected to not overflow the user with information.

The base functionality of the designed platform is obviously the societal component. The implementation of a simple chat functionality is present in the form of PeerChat [51], implemented as a community of IPv8. Its core functionality is the exchange of text messages and photos over the P2P network. Many more functionalities had to be implemented to be able to match the market-leading big-tech platforms. The ability to exchange files, locations, contacts, identity attributes, money, and payment requests has been integrated. The chat

⁴<https://github.com/Tribler/trustchain-superapp>

⁵<https://github.com/Tribler/kotlin-ipv8>

is an ordered list of the communication between two peers, and every type of message or attachment has its own view. It is again important to not display too much information for the attachments. Detail views display all available information about the attachment. Users have additional access to various convenient chat-related features to improve usability like searching and filtering, muting, archiving, or blocking. To reduce the receipt of unwanted spam the platform discards communication with blocked peers. Large attachments, in particular photos and files, are exchanged using our designed data protocol of Section V-D. All other attachments are exchanged as single UDP packets and will be delivered using IPv8's default method. If the recipient of a message (or data) is currently not connected, the message cannot be delivered. To track the delivery status of messages, peers are required to acknowledge its receipt. That enables the community to periodically check if the peer is connected and retransmit the unacknowledged messages. In a P2P network it regularly happens that messages can't be delivered as there is no central or distributed component temporarily storing the message. As a consequence, it may happen that two peers are unable to communicate when they are never connected at the same time.

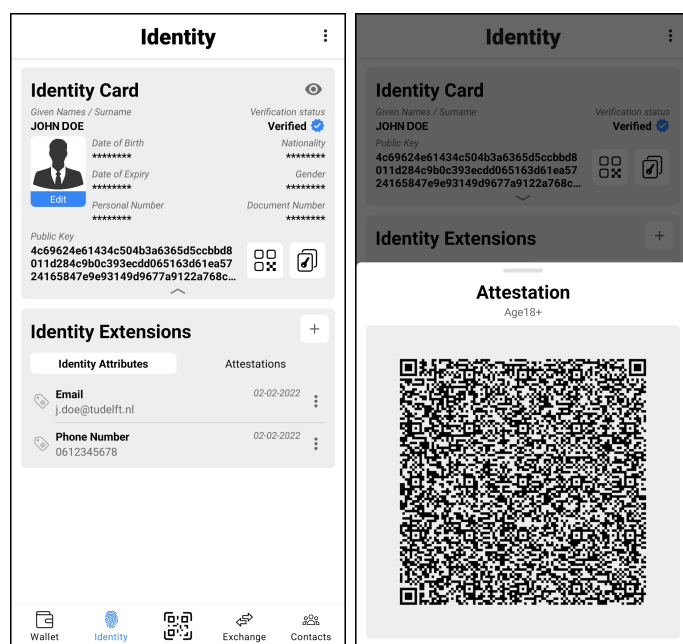


(a) Original PeerChat implementation (b) Our chat implementation [52]

Fig. 6: Difference between old and new implementation

In Figure 6 both the existing PeerChat implementation and our implementation are displayed. Not only is our implementation equipped with much more functionalities, but the design is also concerned about the user experience and ease of use. We've placed options and functionalities behind additional buttons, carefully selected required information to display and provided our platform with full integration and support for QR-codes. The use of QR-codes has many advantages. A lot of information can be embedded in these codes without having to worry about human errors. Inter-platform and offline communication still enable users to exchange the embedded

information, even when disconnected from the Internet. These codes can additionally provide direct navigation within the platform, based on the contents of the scanned code. This all combined improves the user experience as less time and effort is required from the user. Example applications of the QR-codes include offline scanning and adding public keys, unspecified payments requests, transfer announcements to and from the exchange portal, and creating and validating identity attestations. Currently, all QR-codes contain unencrypted data as it is not deemed necessary for offline communication. However, future functionalities may for instance include the exchange of information of the self-sovereign identity and would require additional security. QR-codes are perfectly suited to embed encrypted data, but requires the recipient's public key to be known in advance.



(a) Identity implementation (b) Attestation integration

Fig. 7: Implementation of self-sovereign identity

The integration of the identity component had to be designed from scratch. The identity community handles anything related to the functionalities of the self-sovereign identity, including its storage. Compared to other communities, the identity community does not support communication over the IPv8 network. There is currently no need to additionally share information from the self-sovereign identity, as it may only form a privacy vulnerability. To use the platform, the users are required to onboard their identity as explained in Section V-A. Any device with NFC support is required to verify its identity document, as it reduces misuse and provides authenticity. An extra layer of protection makes sure that the identity details are not visible for eavesdroppers, see Figure 7a. Apart from the self-sovereign identity itself, we've also integrated *identity attributes* and *identity attestations*. Identity attributes contain convenient information that serve as an unauthentic extension to the identity. These attributes are for example a phone number, email address, or home address. These attributes are

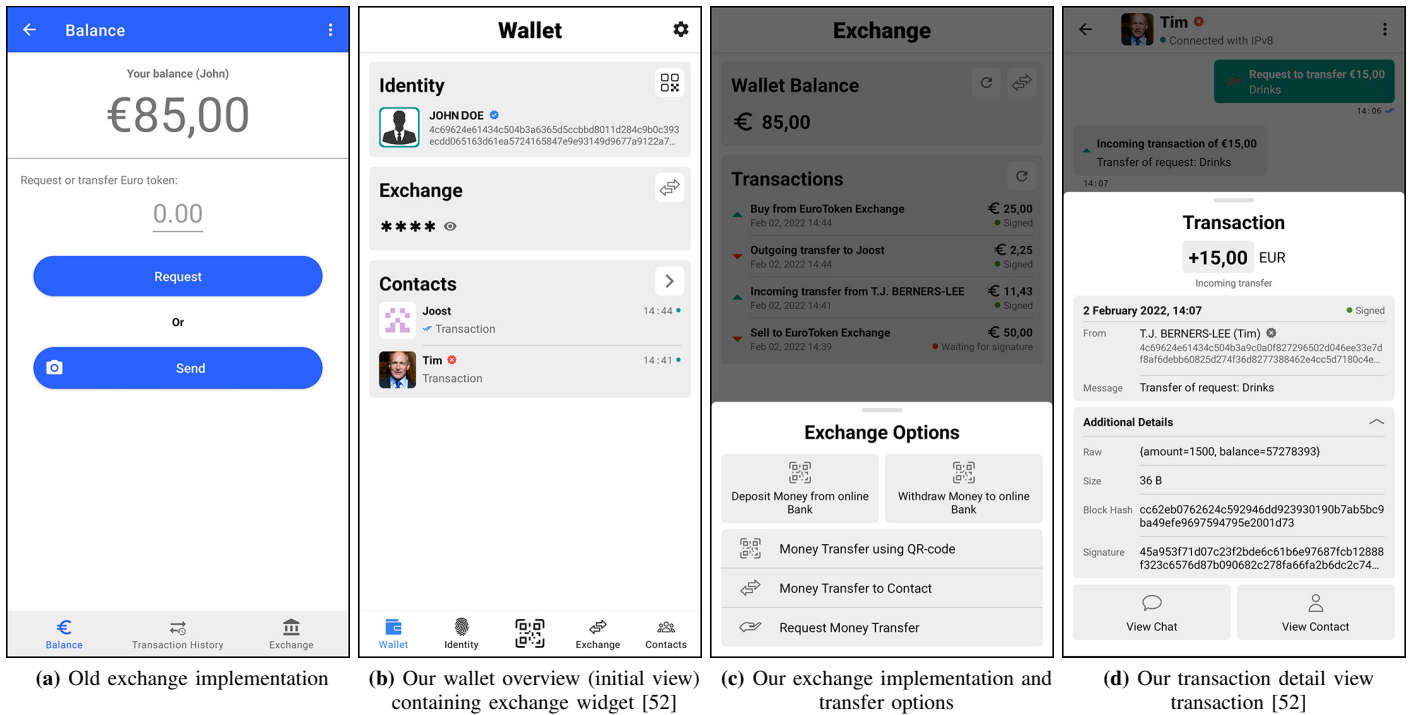


Fig. 8: Screenshots of implementation

shareable with other peers and solely serve to extend the use of the self-sovereign identity. Validation mechanisms to verify the identity attributes' authenticity are currently not included. The identity attestations are incorporated in unmodified form using the attestation community, as a result of the work of Chotkan [30]. An example QR-code of an 18+ attestation is portrayed in Figure 7b. Currently, only the age-related attestation types obtain the age directly from the self-sovereign identity. Future attestation types could embed more confidential information from the identity as it can serve additional purposes.

The identity of this contact has been determined. The identity is verified and the (official) name of the contact is [redacted]. Please be careful in case the name is unfamiliar.

(a) initial state

The identity of the contact has been updated. The identity name has changed from [redacted] to [redacted]. Please be careful in case the name is unfamiliar.

(b) updated state

Fig. 9: Detection in alteration of received trust attributes

The trust attributes, as explained in Section V-B, are directly deduced from the self-sovereign identity. These attributes are automatically added to every form of communication with other peers. The user is unable to choose to communicate without sending the trust attributes along, as this damages the integrity of our platform. Changes to the state as a result of the received trust attributes are notified in the chat as in Figure 9. The verification status of the contact is explicitly displayed at several different locations with the sole purpose to draw attention and recognition, especially when unverified. A

recognizable blue check star indicates successful verification while a red cross star, that is missing in existing platforms, indicates an unverified peer. Both verification statuses can be seen in Figure 8b.

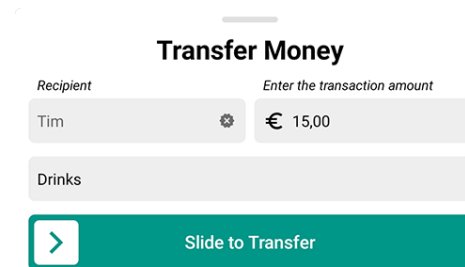


Fig. 10: 'Slide-to-Transfer' protection

An implementation to transfer digital money was already integrated in the superapp, see Figure 8a. The eurotoken community handles the wallet and transfer of the CBDC Eurotoken [47] with other peers, while the trustchain community handles the functionalities of the transactions in the distributed ledger. In our platform, the wallet balance is displayed and protected from eavesdroppers by initially hiding the balance, see for example Figure 8b. Various options to exchange money have been integrated as in Figure 8c. Firstly, QR-codes are used to deposit or withdraw tokens from and to the exchange portal. Secondly, users have the option to scan a QR-code payment request to transfer tokens, create a direct transfer to another contact, create and send payment requests to contacts over the network or create an unspecified payment request using a QR-code. The latter three are accessible from the

chat with the corresponding contact as well. An extra layer of protection, in the form of a 'slide-to-transfer' element as seen in Figure 10, withholds the accidental exchange of tokens. For convenience, transactions are not only visible in the list of transactions in the exchange view, but also in chats. In Figure 8d a detailed view of a transaction and its contents is displayed. As payment requests are attachments and not formal transactions, they are only included and visible in the chats.

As explained in Section V-D, our custom data transfer protocol handles the exchange of (large) data blobs. The protocol technically contains an entry point for peers to directly transmit data to another peer, or in some situations to be scheduled. Scheduled transfers are periodically checked and started if the following requirements are satisfied: (I) the peer is connected, (II) there's no other current transfer with the peer, and (III) the size of the transfer does not exceed the maximum allowed size. The protocol exploits packet listeners to be able to directly respond to each of the received packets. The transfer is initiated by announcing the transfer and transfer-specific settings. As (many) other communities may employ the protocol as well, it is convenient to annotate the destined community. As there may be various concurrent transfers with different peers, every transfer must contain a unique identifier to distinguish the transfers from one another. To make sure that both the sender and receiver are in agreement with the transfer parameters, and additionally allow other communities to use different parameters, these parameters are included within the transfer announcement. The transfer announcement and the last data packet of its window must be acknowledged. This acknowledgment either confirms the start of the transfer or receipt of a window of data blocks. In addition to the latter, the current window number and lost packets must be reported. This confirmation consecutively results in the transmission of the next window of blocks until all blocks have been transmitted and received. To be able to stitch all blocks together, the protocol additionally transmits the block number with the packet. The transfer is completed when all blocks have been received.

To realize a quick response to the transmitted packets, the recurrent packets must contain the minimal required information. The transfer announcement is only transmitted once and is therefore allowed to contain more information. As the data and acknowledgment packets are transmitted numerous times, we've made sure that no redundant information is transmitted.

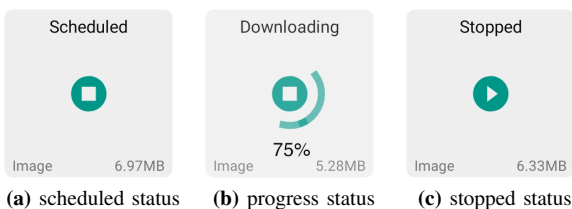


Fig. 11: Download progress indicators in chat

To accommodate communities and users with convenient information during and after the transfer, the protocol has built-in support for callbacks. These callbacks enable the execution of specific tasks after the transfer progressed to

another state. These tasks differ for each application and are therefore outside the scope of the protocol. The sender is able to execute specific code after the data has been successfully sent or upon receipt of an error. The receiver has access to the transfer progress, transfer completion, and erroneous updates. Specifically, the former two are important to our platform. Transfer progress updates enable the application to display the current download status to the user, see Figure 11. For convenience in certain situations, the user is able to stop and restart the transfer at a later time. The transfer complete update triggers the conversion of the raw binary data to the correct format, creates an external file on the phone storage, and visually embeds it in the chat. The protocol is dependent on many parameters that may have an impact on the performance. In Section VII these parameters are analyzed to obtain the optimal performance during *normal* operation.

✔ Tests passed: 85 of 85 tests – 9 sec 226 ms

(a) Unit tests

1.111 of 4.994 77%

(b) Code coverage

Fig. 12: Code quality

We've additionally analyzed the code quality of the protocol. A high code coverage is required to reduce bugs and ensure correctness. The protocol, including all associated methods and classes, is tested using 85 unit-tests that cover about 77% of the code (Figure 12).

VII. EXPERIMENTAL ANALYSIS AND EVALUATION

In the previous sections, we have discussed the design and implementation of our data transfer protocol. In this section, an experimental analysis is performed to derive the optimal protocol settings under normal operation. We've only considered normal cases in this analysis due to missing theoretical and applied expertise. We also performed an evaluation of a large-sized transfer with these optimal settings to prove its contribution and applicability to our platform.

A. Experimental Analysis

To exploit the best possible performance, the designed binary data transfer protocol requires its settings to be optimal. We define the protocol to be optimal if (I) the transfer speed is as high as possible, (II) the number of lost packets/blocks (as explained in Section V-D) is as low as possible, and (III) the number of retransmitted windows of blocks (by the sender) and acknowledgments (by the receiver) is as low as possible. The second constraint does not necessarily contribute to higher transfer speeds as lost packets are embedded in the next window of blocks. The last constraint should contribute to higher transfer speeds because no windows of blocks have to be retransmitted and there's no additional idle time waiting for an acknowledgment. The analysis is performed in ideal situations and focuses on the general picture, i.e. aspects as delays due to latency and packet loss are not included.

As mentioned before, the UDP packet size is limited due to Ethernet constraints. As two packets with a payload of 500 bytes carry twice as much redundant information as one packet of 1000 bytes, a transfer using a greater block size B is preferred and should theoretically have a positive impact on the runtime. The maximum *data* size of UDP packets for IPv8 has been determined (through trial-and-error) to be around 1241 bytes. The exact size may depend on each peer, the chosen packet header options (encryption, signature, public key, etc.), and the data packet metadata. To keep a safe margin we've decided to allow data of at most 1200 bytes in each packet. The window size W is defined as the number of bytes ($n_{\text{blocks}} \times \text{block size}$) the sender can transmit without having to wait for an acknowledgment of receipt from the receiver. Theoretically, a greater window size would directly contribute to larger *transfer speeds* as there are less windows of blocks to be transmitted. Also, a smaller number of acknowledgments has to be sent and received, reducing the overall idle time of both participants. Greater window sizes also increase the existence of late or lost blocks, specifically in imperfect or congested networks, with an increased number of retransmissions of windows of blocks as a result. The importance of this analysis is to find the best trade-off between a great window size and low delay due to lost packets.

The other parameters do not directly impact the performance, apart from the block and window size. The retransmit interval may affect the performance when it is either too tight or loose, but it will only play a role in a small part of the cases. A tight interval can force windows of blocks or acknowledgments to be retransmitted while they are still in transit and may arrive shortly after. For a loosely set interval, the protocol may unnecessarily have to wait for a window or acknowledgment. The transfer timeout interval is less critical and will only affect the performance when a window or acknowledgment has abused all retransmit attempts. The retransmit attempt count likewise has little influence on the performance.

Experimental Setup: The experimental setup is equipped with two phones, a Xiaomi Redmi 9T with Android 10 and a Huawei P20 Lite with Android 9, both 4GB RAM. The phones have installed the same version of the app and are connected to the same WiFi-6 mesh network (NETGEAR Orbi RBK753). To obtain more accurate results, each experiment is executed five times. Also, to verify the independence of the file size on the transfer, the experiment is executed for multiple file sizes. Each important step of the protocol is captured in a log to be processed in Python. An automatic Kotlin script makes sure that every combination of parameters and the five iterations are executed consecutively. Table III gives an overview of the analyzed parameter values. The maximum parameter settings are windows of 128 blocks of 1200 bytes each, equivalent to 150kB of unacknowledged data that is the fundamental limit to the data transfer performance. With a certain unknown latency this results in an exact transfer speed limit. As no a packet loss and latency emulation has been performed, we cannot determine this theoretical limit.

TABLE III: Parameters that are being tested for optimal execution. The number of iterations have only been used for consistency. In total 56 combinations of parameters have been executed 5 times.

Parameter	Values	
Block size (B)	600, 700, 800, 900, 1000, 1100, 1200	[bytes]
Window size (W)	16, 32, 48, 64, 80, 96, 112, 128	[blocks]
Iteration	0, 1, 2, 3, 4	[-]

Experimental Results: The optimality of the performance of the protocol can be determined in combination with the before mentioned requirements.

The results of the first requirement, the transfer speed of the protocol, is displayed in Figure 13a. We can clearly see the effect of the variation of the block and window size. Higher block sizes increases the transfer speed. The window size follows a parabolic curve and the transfer speed is optimal for a window size $W = 80$ and $W = 96$ blocks. We cannot yet determine the optimal value for the window size as these values are very similar and are less pronounced than the block size. To decide on the optimal sizes, we have to include the requirements as well.

The results for the second requirement, the number of lost packets/blocks during a transfer, is visualized in the plots of Figure 13b. The block size shows a slightly decreasing pattern and overall contains the lowest number of lost packets for greater block sizes. There is no consensus on the block size *within* each window size, as there are small deviations and not consistently decreasing or increasing. The trendline, a combined average of all block sizes within each window, shows a minimum for the same window sizes as of the first requirement, but slightly favors $W = 96$ blocks. For window sizes greater than $W = 96$ blocks the number of lost packets again increases. The protocol is behaving more unreliable as more lost blocks have to be added to the next window. Every lost block will cause a marginal decrease of the transfer speed as more blocks have to be delivered in the next window. The aim remains to reduce these lost blocks as much as possible. Especially for more unreliable connections, the impact of lost blocks may become more pronounced as the transfer progresses further.

The last requirement, the number of retransmitted windows of blocks and the number of retransmitted acknowledgments, are entangled as it takes both the sender and receiver in the equation for the same transfers. In Figure 13c and 13d the results for the sender and receiver are visualized, respectively. Both diagrams show the same pattern. The optimal block size again shows no notable preference within each window size. The results for the window sizes are a strong indicator that we don't want them to be oversized. The number of retransmitted windows and acknowledgments is of neglectable proportion for a window size less than $W = 112$ blocks, or even $W = 96$ blocks if we would be really strict. As the protocol has to wait for a full interval for every retransmission, it has a big impact on the overall performance. It is crucial to reduce the number of retransmissions to an absolute minimum. For the largest three window sizes, a small experiment was executed

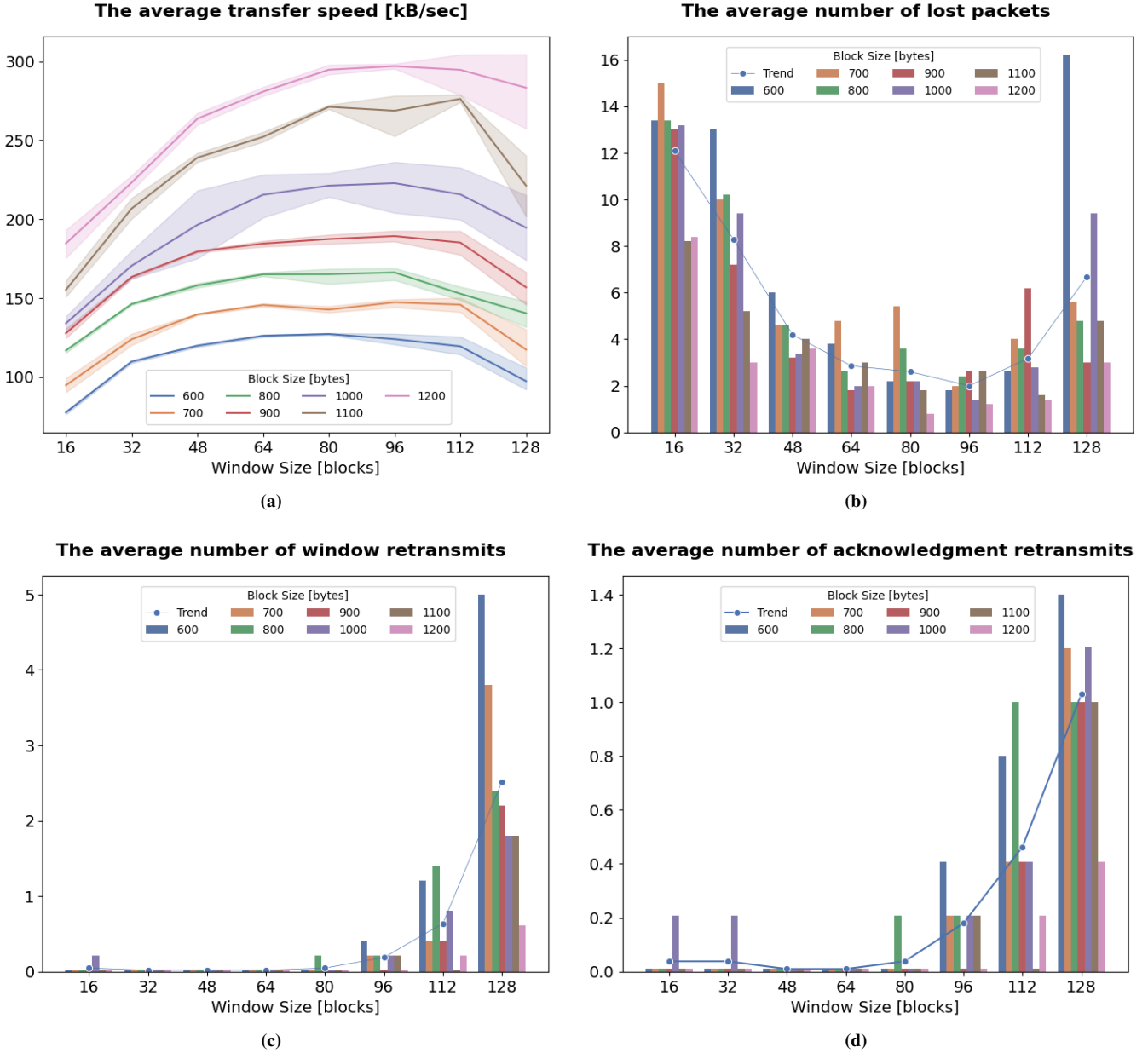


Fig. 13: The results of all executed tests for each variable window and block size, averaged over five iterations.

that verified if the large increase of retransmissions were the result of a too tightly set retransmit interval. The interval was increased majorly, just for verification purposes, and solely served for additional analysis of the third requirement, without including the results of the other requirements. The results showed that the number of retransmissions slightly improved, but the pattern was equally in place. It was deemed unnecessary to investigate it further.

We must take into account that the experiments have been performed under somewhat optimal circumstances: phones only running system services and the platform itself, and both connected to the same *local* WiFi network. From this, we could argue that for worsened conditions, the transfer speed and the number of lost packets and retransmissions would logically

increase. We can combine and summarize our findings based on the results of the experiments and analysis. The performance of the transfer speed (continuously) increases with greater block sizes, with the optimal block size of $B = 1200$ bytes. For the other requirements, the difference in block size was less definite. The transfer speed found an inconclusive optimal for window sizes of $W = \{80, 96\}$ blocks. The second requirement showed similar results, although slightly in favor of the latter. Both types of retransmissions for the third requirement indicated an increase of retransmissions for greater window sizes, specifically above $W = 80$ blocks. To obtain the optimal performance, in combination with the minimal number of lost blocks and retransmissions, we can conclude that a window size of $W = 80$ blocks is optimal

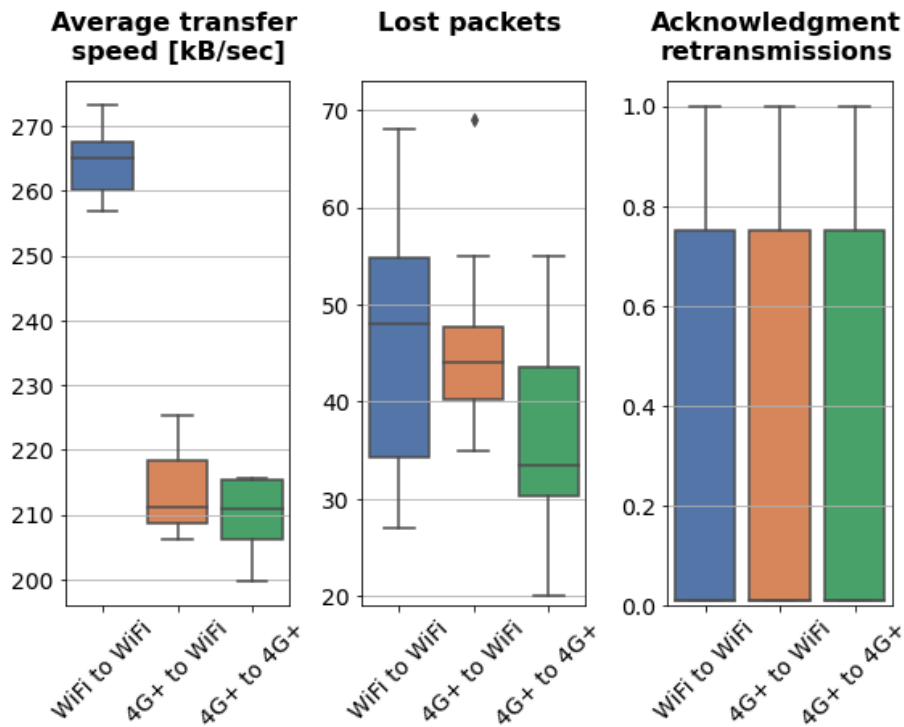


Fig. 14: Evaluation of the performance of a transfer of 250MB using the optimal window size of 96kB, executed 10 times.

under normal operation. The optimal window size based on our findings is thus $W = 96000$ bytes or 96kB. We've deduced these optimal parameters for ideal situations, and therefore does not include lateness and loss of blocks.

We've additionally executed the same experiments using multiple file sizes to verify its independence on the performance. All results are similar and seem to indicate that the file size is independent on the performance. We've also concluded that an increase of the retransmit interval does not necessarily give a significant overall reduction of the number of retransmissions, and therefore not contributes to better performance.

B. Performance Evaluation

Now that we've determined the optimal parameters for the protocol, we want to see how it performs in the wild for a large-sized transfer of 250MB. This enables us to evaluate the performance more consistently over a longer period of (transfer) time. In most cases, phones are not connected to the same local WiFi network. We have to consider three commonly-used situations (I) WiFi to WiFi, (II) WiFi to 4G+, and (III) 4G+ to 4G+. For the connections of 4G+, we use the telecom providers Vodafone and KPN on the same phones as before. Our evaluation includes the same aspects as the experimental analysis. Instead of finding the optimal parameters, we this time evaluate the applicability and the difference between the three situations. Each experiment is executed 10 times to obtain more consistent results. We expect the first situation to offer the highest transfer speed and least number of lost packets and retransmissions as the packets are only exchanged within the local network.

The performance results for each of the connection types are portrayed in Figure 14. The transfer speed shows a clear division in performance, in favor of the inter-WiFi transfer of about 260kB/s. The transfer speeds for the two situations using a mobile connection have very similar speeds of about 213kB/s and 210kB/s. It's a good indication that there is no extreme performance difference between an exchange using one WiFi-connected device and a complete mobile network exchange. The transfer speed of the inter-WiFi transfer is executed on a local network and therefore sketches a slightly biased image. The transfer speed of an exchange between two non-local WiFi networks would theoretically be lower, and possibly be more similar to the other situations. However, we do notice a bigger spread in terms of lower speeds for the complete mobile network exchange. If we look at the absolute speed, we must conclude that our protocol is nowhere near the download speeds of current network infrastructures. The exact reason is unknown, but it is expected to be some limitation in IPv8 and possibly sub-optimal UDP socket buffering.

The number of lost packets contradicts our expectation. The number of lost packets is on average larger for an inter-WiFi transfer in comparison with the other two situations. In absolute numbers though, we can conclude that the total loss of about 50 packets in a transfer of over 200.000 packets, equivalent to one lost packet within every fifty windows, is neglectable. This number would be much higher for sub-optimal connections. No retransmission of windows or blocks were encountered, and therefore left out of the figure. The number of retransmissions of acknowledgments are very equivalent for all three situations, and only allows one acknowledgment retransmission per transfer on average as result of unresponsiveness.

From these results we can conclude that our protocol has been performing at the top of its *abilities* in normal operation. The absolute transfer speed is disappointing. During the experiments, it was noticed that larger-sized transfers experienced memory allocation issues on the phones. Currently, the protocol stores the sent and received data in memory for reconstruction purposes. The current maximum file size has therefore been limited to 250MB, but can differ from phone to phone.

VIII. TIME MANAGEMENT

The research, design, implementation, analysis, and documentation have been an effort of one person in roughly nine full-time months. The research phase required about two months to study the literature and existing platforms, including the basics and characteristics of IPv8, TrustChain, and other components. The design and implementation phase were entangled as the scope of the research widened several times along the way. You could say that it was a repeating process of invent-design-implement for most of the features. An initial layout of the application was designed and implemented to provide a basic platform that slowly evolved to its final state. This included familiarizing the style and coding within the *Trustchain* superapp and the *Kotlin* language. The design and implementation of features were not only concerned with the features themselves, but also the UX and UI design of the platform. In total, about six months have been allocated to the design, implementation, and analysis of the platform. The data transfer protocol is the only component that has additionally been analyzed as part of experimental analysis and evaluation. Of these six months, about one month of work was required to optimize and analyze the data transfer protocol. The last month, and also some time earlier in the process, was dedicated to documenting and finalizing this paper.

IX. CONCLUSION

Ownership and exchange of sensitive or private information has never been a more relevant topic, also due to the COVID pandemic. This paper has presented a novel Web3 platform for identity, trust, money, and data. Decentralization partially solves the lack of self-sovereignty for identity, money, and data in the current online world. We performed the first exploratory study that shows the viability of the integration of a self-sovereign identity in a social platform in a useful, secure, and private fashion. The application of self-sovereign identities has additionally shown to be effective in providing trust to peers in the P2P network. Governments may change their mind about their identity management systems in the near future. Many centralized tasks can be replaced, providing their citizens more power and ownership over their data, while retaining authenticity and majorly reducing costs. Central banks have started designing Central Bank Digital Currencies globally. Our platform incorporates the exchange of digital money in a private and informal way, similar to what cash once was. Data and personal information has been owned and managed by big-tech companies for far too long. The centralized structure is

the core of the problem in online communication. Our P2P data transfer protocol enables peers to securely and privately exchange messages and data while reducing the leakage of metadata to a minimum.

Unfortunately, we cannot neglect some major disadvantages in our platform as well. Firstly, the availability and connectivity of peers remain an open issue as it introduces lateness in the delivery of messages and data. People are currently not used to lateness in existing platforms which use central servers or nodes. Secondly, peers in a fully P2P network must keep themselves online by constantly discovering and connecting to peers with the consequence of draining the phone's battery. Improvements could be made, but it will always remain a weak spot of P2P systems. Finally, the implementation of the CBDC EuroToken is dependent on central components and not yet reliable enough.

Overall, we conclude that we've designed a well-functioning platform that incorporates all aspects of our research in a valuable, private, and secure manner. There is still a lot to discover and we're curious to see what direction big-tech companies, governments, and banks will pursue in the near future.

REFERENCES

- [1] Tribler. Ipv8 documentation. 2021. URL https://py-ipv8.readthedocs.io/_/downloads/en/latest/pdf/.
- [2] Pim Otte, Martijn de Vos, and Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems*, 107: 770–780, 2020. ISSN 0167-739X. doi: <https://doi.org/10.1016/j.future.2017.08.048>. URL <https://www.sciencedirect.com/science/article/pii/S0167739X17318988>.
- [3] Nicolo Zingales. Between a rock and two hard places: Whatsapp at the crossroad of competition, data protection and consumer law. *Computer Law & Security Review*, 33(4):553–558, 2017.
- [4] The Guardian. Whatsapp loses millions of users after terms update. [Online] Available: <https://www.theguardian.com/technology/2021/jan/24/whatsapp-loses-millions-of-users-after-terms-update>, 2021.
- [5] "Wall Street Journal". Big tech braces for a wave of regulation. [Online] Available: <https://www.wsj.com/articles/big-tech-braces-for-wave-of-regulation-11642131732>, 2022.
- [6] Official Journal of the European Union. Regulation (eu) 2016/679 of the european parliament and of the council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). [Online] Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN>, 2016.
- [7] Gdpr enforcement tracker. [Online] Available: <https://www.enforcementtracker.com>.
- [8] Aikaterini Soumelidou and Aggeliki Tsohou. Towards the creation of a profile of the information privacy aware

- user through a systematic literature review of information privacy awareness. *Telematics and Informatics*, 61: 101592, 2021.
- [9] ZDNet. The biggest data breaches, hacks of 2021. [Online] Available: <https://www.zdnet.com/article/the-biggest-data-breaches-of-2021/>, Dec 2021.
- [10] PinDirect. Wat kost een pintransactie? [Online] Available: <https://pindirect.nl/kennisbank/uw-eigen-pinautomaat/wat-kost-een-pintransactie/>, 2020.
- [11] ZakelijkBankieren.nl. Vergelijk ideal kosten per aanbieder in nl. [Online] Available: <https://www.zakelijkbankieren.nl/kosten-ideaal/>, 2021.
- [12] Dutch Payments Association. Facts and figures on the dutch payment system in 2020. [Online] Available: <https://factsheet.betalvereniging.nl/en/>, 2020.
- [13] Christopher Allen. The path to self-sovereign identity. [Online] Available: <http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html>, 2016.
- [14] Quinten Stokkink and Johan Pouwelse. Deployment of a blockchain-based self-sovereign identity. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1336–1342, 2018.
- [15] "Computer Weekly". "blockchain technology will help banks will cut cross-border payment costs by \$10bn in 2030". [Online] Available: <https://www.computerweekly.com/news/252509262/Blockchain-technology-will-help-banks-will-cut-cross-border-payment-costs-by-10bn-in-2030>, 2021.
- [16] Statista. Most popular global mobile messenger apps as of october 2021, based on number of monthly active users. [Online] Available: <https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/>, 2021.
- [17] Roman Zaikin and Oded Vanunu. Reverse engineering whatsapp encryption for chat manipulation and more. [Online] Available: , August 3-8, 2019.
- [18] WhatsApp. Whatsapp encryption overview. technical white paper. Nov 2021. URL <http://www.cdn.whatsapp.net/security/WhatsApp-Security-Whitepaper.pdf>.
- [19] FaceBook Inc. Messenger secret conversations. technical whitepaper. May 2017. URL <https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf>.
- [20] WeChat. Wechat - free messaging and calling app. "[Online] Available: <https://weixin.qq.com>", Jan 2022.
- [21] Telegram. Telegram messenger. "[Online] Available: <https://telegram.org/>", Jan 2022.
- [22] Apple. imessage security overview. "[Online] Available: <https://support.apple.com/en-au/guide/security/sec9764312f/web>", May 2021.
- [23] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 451–466, 2017. doi: 10.1109/EuroSP.2017.27.
- [24] Kee Jefferys, Maxim Shishmarev, and Simon Harman. Session: A model for end-to-end encrypted conversations with minimal metadata leakage. *CoRR*, abs/2002.04609, 2020. URL <https://arxiv.org/abs/2002.04609>.
- [25] Status. The status network. a strategy towards mass adoption of ethereum. June 2017. URL <https://status.im/whitepaper.pdf>.
- [26] Felix Schlitter, John Carlo San Pedro, Paul Freeman, and Callum Lowcay. Sylo protocol: Secure group messaging. 2020. URL <https://www.sylo.io/whitepaper/sylo-protocol.pdf>.
- [27] Berty. Berty protocol. "[Online] Available: <https://berly.tech/docs/protocol/>", Oct 2020.
- [28] Logius. Tarieven 2022 voor digid, digid machtigen en mijnoverheid. [Online] Available: <https://logius.nl/onze-organisatie/zakendoen-met-logius/doorbelasting>, 2021.
- [29] Logius. Lack. [Online] Available: <https://logius.nl/actueel/gebruik-digid-de-lift>, 2022.
- [30] R. Chotkan. Industry-grade self-sovereign identity, on the realisation of a fully distributed self-sovereign identity architecture. Master's thesis, Delft University of Technology, 2021.
- [31] Ann Cavoukian. Privacy by design: The 7 foundational principles. May 2010.
- [32] Tweakers. Belgische overheid verplicht chatapps zoals whatsapp metadata op te slaan. [Online] Available: <https://tweakers.net/nieuws/183702/belgische-overheid-verplicht-chatapps-zoals-whatsapp-metadata-op-te-slaan.html>, 2021.
- [33] RollingStone. Fbi document says the feds can get your whatsapp data — in real time. [Online] Available: <https://www.rollingstone.com/politics/politics-features/whatsapp-imessage-facebook-apple-fbi-privacy-1261816/>, 2021.
- [34] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998. doi: 10.1109/49.668972.
- [35] Wikipedia. Information security. "[Online] Available: https://en.wikipedia.org/wiki/Information_security", 2022.
- [36] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. Nov 1976. URL <https://ee.stanford.edu/~hellman/publications/24.pdf>.
- [37] International Telecommunication Union (ITU). Data networks and open system communications. open systems interconnection - model and notation. July 1994. URL https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-1!!PDF-E&type=items.
- [38] Keesing Technologies. Security vulnerabilities associated with e-passports. [Online] Available: <https://platform.keesingtechnologies.com/e-passport-security/>, Feb 2020.
- [39] Fortunly. 20 worrying identity theft statistics for 2022. [Online] Available: <https://fortunly.com/statistics/identity-theft-statistics/>, Feb 2022.

- [40] Europol EC3. Spear phishing, a law enforcement and cross-industry perspective. Nov 2019. URL https://www.europol.europa.eu/sites/default/files/documents/report_on_phishing_-_a_law_enforcement_perspective.pdf.
- [41] BOARD OF GOVERNORS OF THE FEDERAL RESERVE SYSTEM. Money and payments: The u.s. dollar in the age of digital transformation. Jan 2022. URL <https://www.federalreserve.gov/publications/files/money-and-payments-20220120.pdf>.
- [42] European Central Bank (ECB). Central bank digital currency: functional scope, pricing and controls. 286, Dec 2021. URL <https://www.ecb.europa.eu/pub/pdf/scpops/ecb.op286~9d472374ea.en.pdf>.
- [43] Bank of England. Central bank digital currency, opportunities, challenges and design. Mar 2020. URL <https://www.bankofengland.co.uk/-/media/boe/files/paper/2020/central-bank-digital-currency-opportunities-challenges-and-design.pdf>.
- [44] Working Group on E-CNY Research and Development of the People's Bank of China. Progress of research & development of e-cny in china. July 2021. URL <http://www.pbc.gov.cn/en/3688110/3688172/4157443/4293696/2021071614584691871.pdf>.
- [45] Reuters. China central bank launches digital yuan wallet apps for android, ios. [Online] Available: <https://www.reuters.com/markets/currencies/china-cbank-launches-digital-yuan-wallet-apps-android-ios-2022-01-04/>, Jan 2022.
- [46] Het Parool. Vier amsterdammers vast voor 'tikkiefraude' honderden mensen. "[Online] Available: <https://www.parool.nl/amsterdam/vier-amsterdammers-vast-voor-tikkiefraude-honderden-mensen~b33baf1a>", Nov 2020.
- [47] R.W. Blokzijl. A central bank digital currency (cbdc) with offline transfers. Master's thesis, Delft University of Technology, 2021.
- [48] Network Working Group. The tftp protocol (revision 2). June 1981. URL <https://www.rfc-editor.org/rfc/pdf/rfc/rfc783.txt.pdf>.
- [49] J. Postel. User datagram protocol. Aug 1980. URL <https://www.rfc-editor.org/rfc/pdf/rfc/rfc768.txt.pdf>.
- [50] Andrew S. Tanenbaum and David Wetherall. *Computer networks, 5th Edition*. 2011.
- [51] M. Skála. Technology stack for decentralized mobile services. Master's thesis, Delft University of Technology, 2020.
- [52] Paul Clarke. Wikimedia commons, the free media repository. [Online] Available: [https://commons.wikimedia.org/wiki/File:Sir_Tim_Berners-Lee_\(cropped\).jpg](https://commons.wikimedia.org/wiki/File:Sir_Tim_Berners-Lee_(cropped).jpg), Sep 2014.

II

Supplementary Material

**Supplementary Material to "Web3: A
Decentralized Societal Infrastructure for
Identity, Trust, Money, and Data"**

1

Mobile Application

This chapter serves as a more detailed view of our mobile platform. The platform has been implemented as an Android application developed in Kotlin in a user-centered design. We've aimed to keep the platform as simple and consistent as possible. This includes a reduced amount of displayed information, consistent design, similar usage of buttons and functionalities throughout the complete platform, and giving the user (some) control over the behavior of the platform. The platform makes use of specific views that each contain different information. The use of a bottom navigation bar enables easy navigation between the views.

1.1. Views

The following views have been implemented in the platform (Figures 1.1 and 1.2).

Wallet overview contains widgets for identity, exchange, and conversations, as in Figure 1.1a. As it serves as a general overview, we only want to include the absolute necessary information. The identity widget contains the users' official name, profile picture, public key abbreviation, connection status, and a button to show the public key in a QR code. The exchange widget contains the wallet balance and a button that displays the options to transfer money. The balance is initially protected against eavesdroppers and can be toggled by clicking it. The contacts widget contains the three most recent chats including the last message and connection and verification status of the contact and can be opened by clicking it.

Identity view contains everything related to the self-sovereign identity, see Figures 1.1b and 1.1c. Sensitive identity information is initially hidden and protected with placeholder asterisks. By expanding and clicking the eye icon all information is displayed. Future upgrades could require biometric protection to toggle the details. The identity attributes and identity attestations are placed below the identity card, and only show either one of them at the same time. Additional options are accessible by clicking the required icons or buttons.

QR-code controller opens the camera and enables QR-codes scanning, see Figure 1.1d. Upon a successful scan, the platform will automatically navigate or display the destined view. This enables the user to efficiently execute certain tasks while minimizing efforts and human errors.

Exchange view contains the wallet balance and the list of transactions as in Figure 1.1e. Only basic information about each transaction is displayed. The user can click the transaction to show a detailed overview of the complete transaction, as in Figure 1.3f. The wallet balance is initially protected against eavesdroppers. The view also contains buttons for validating the balance, updating the list of transactions, and displaying the various options to transfer money.

Contacts view contains the address book of contacts and list of recent chats as in Figure 1.1f. The user has the ability to search a contact in the address book or chats, add a contact, or view the archived or blocked chats (Figures 1.2b and 1.2c). By clicking the contact or chat-preview the platform navigates to the corresponding chat.

Chat view contains the chat with a contact or peer, see Figure 1.2d. This view contains a list of all communication between both parties, ordered with the most recent at the bottom. Every message or attachment type has a preview with some information. If available, the user can click on the attachment to display the attachment in more detail. Apart from the message typing field, the view also has buttons for showing the chat/contact options (Figure 1.5e), showing the list of attachments (Figure 1.5f), and sending the message or attachment.

1.2. Settings

To give the user a bit more control over the platform we've also included a settings view, as in Figure 1.2e. The user can adjust the theme of the layout to its preference or the time of the day. The user can decide between the day, night, and system default theme. The user has the option to change the use of notifications by clicking the corresponding button that navigates to the system settings for adjustment. The user is also able to replace or remove its identity.

1.3. Dialogs

In the application, we extensively use dialogs to display additional information in an efficient manner. It is in most cases unnecessary to directly display all available information for a particular item. These dialogs are small views that slide in from the bottom and float over the parent view using an opaque grey background. The dialogs can easily be dismissed by either swiping down or clicking the opaque grey background. All dialogs that are implemented in the platform are visualized in Figures 1.3 to 1.6.

1.4. Identity Onboarding

The identity onboarding process consists of three consecutive steps, as displayed in Figure 1.7. In the initial view, for each of the next steps, a small description is provided. In addition to the description of the second step, the reading of the document, the device support status for the NFC chip is displayed. If the device doesn't support the NFC chip, this step will be skipped with the downside of an unverified identity. If the device has support but hasn't currently enabled the NFC chip, a button is displayed to navigate to the settings of the device to enable it. In that case, the importing process

cannot start until the NFC chip is enabled. In the document type view the user must decide which type of identity document he wants to use. The scan document view uses the device camera to scan the MRZ-zone of the document. After all details are scanned and appear to be technically valid, the process continues to the reading view. In the reading view, the user has to place its device on top of the document and hold it steady as long as the device is reading the document. After the document has been successfully read, the user can proceed to the last view. The confirm view displays a summary of the imported identity. The user must complete the onboarding process by confirming the identity. The application can only be used after an identity is imported.

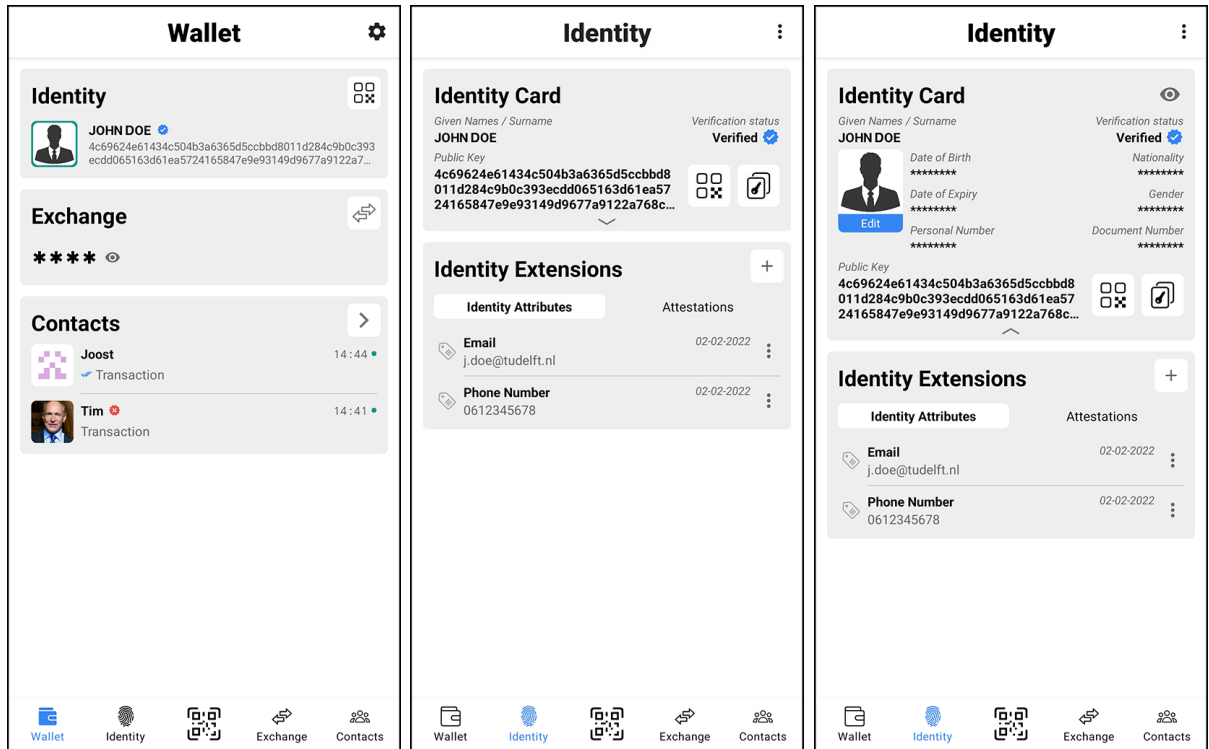
1.5. Notifications

The platform supports notifications for displaying relevant information about a received message, attachment, or transaction. The notifications contain information about the sender's name, profile picture, and message or transaction contents. By clicking the notification the app will open and/or navigate to the corresponding view.

1.6. Phone Permissions

The application requires the user to authorize permissions for the use of the camera for the identity onboarding process and Bluetooth for offline communication using IPv8. The location and write/read storage permissions are not required but advised to utilize all available functionalities.

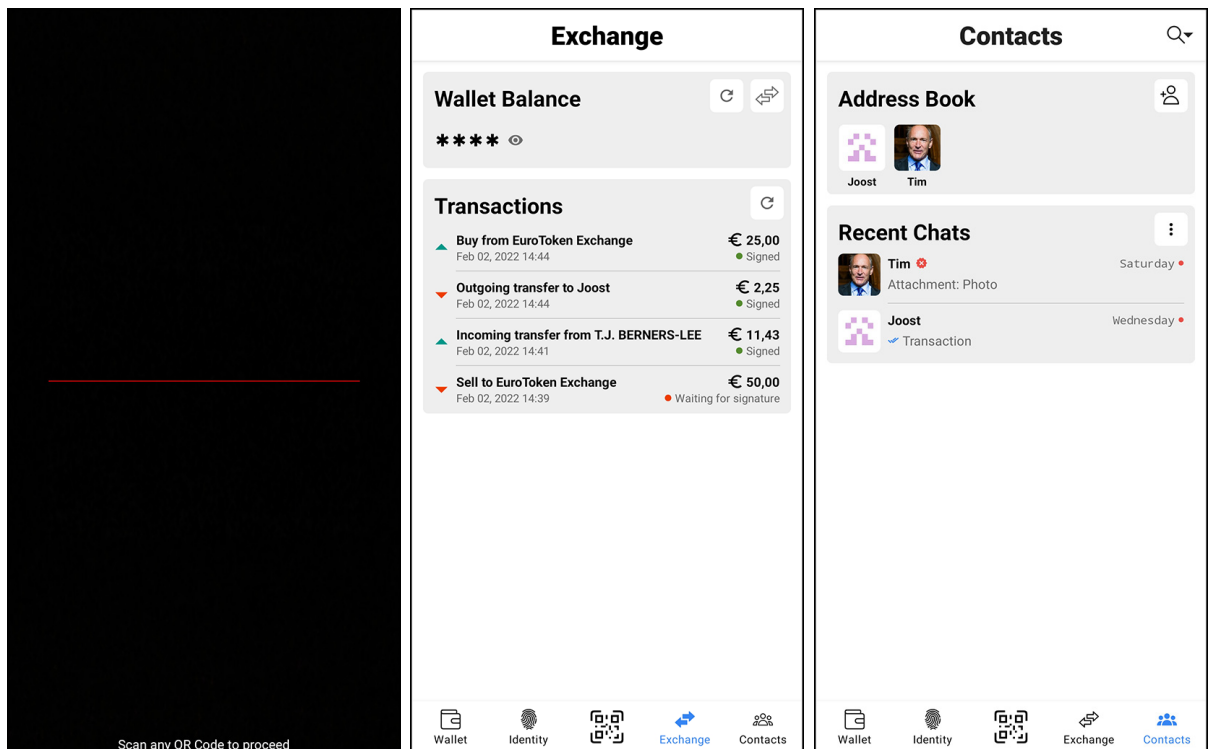
1.7. Screenshots



(a) Wallet Overview

(b) Identity View Collapsed

(c) Identity View Expanded

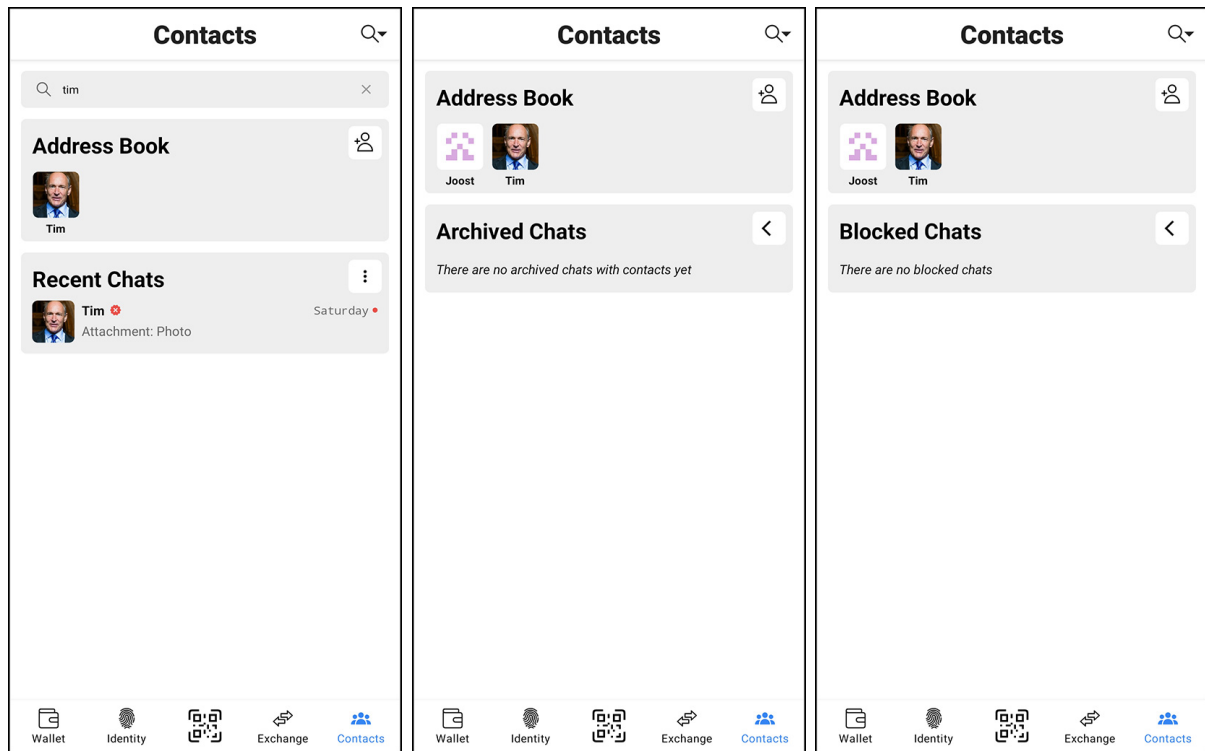


(d) QR-code Scanner View

(e) Exchange View

(f) Contacts View

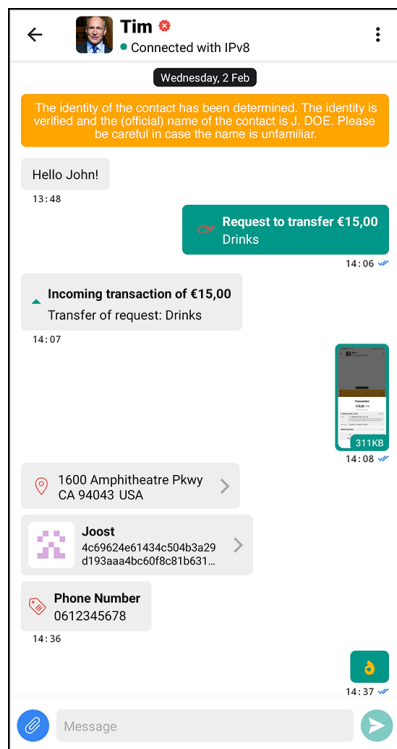
Figure 1.1: Main views of the platform (1)



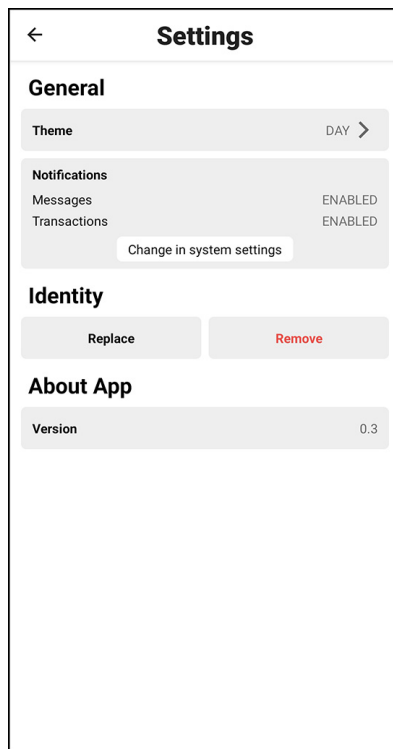
(a) Contacts View Search

(b) Contacts Archived View

(c) Contacts Blocked View

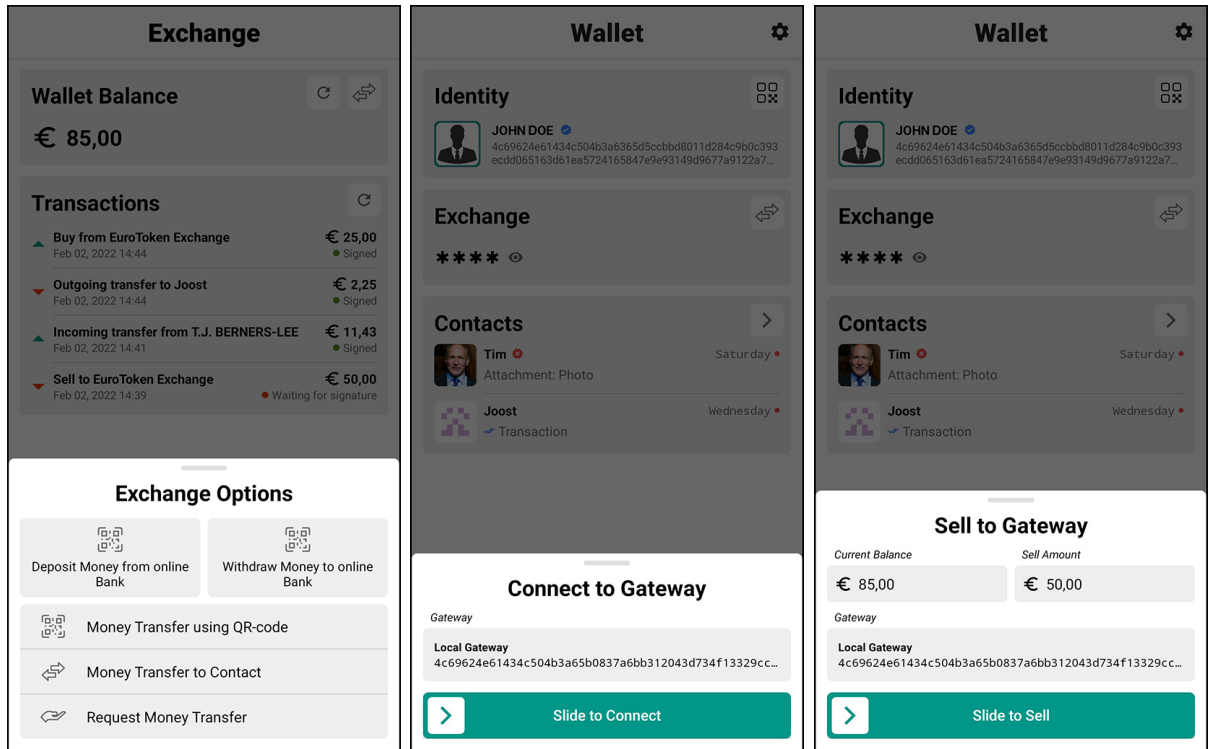


(d) Chat View



(e) Settings View

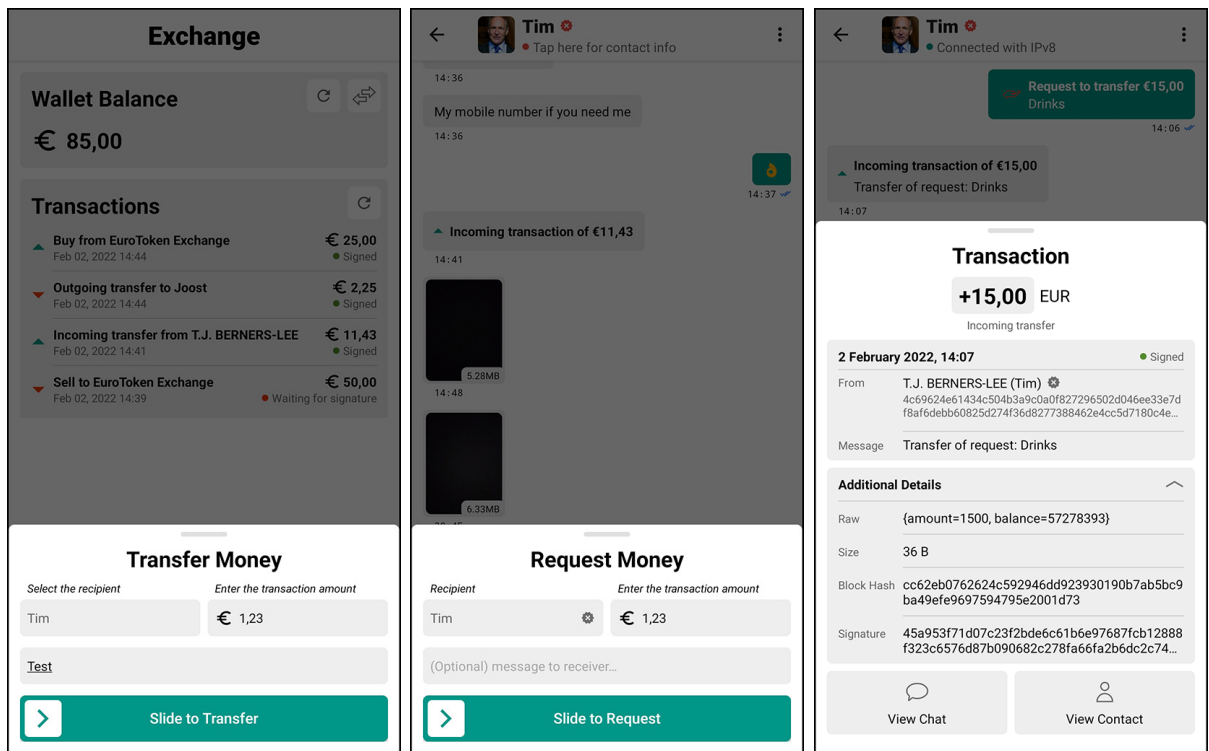
Figure 1.2: Main views of the platform (2)



(a) Dialog Exchange Options

(b) Dialog Exchange Portal Buy

(c) Dialog Exchange Portal Sell

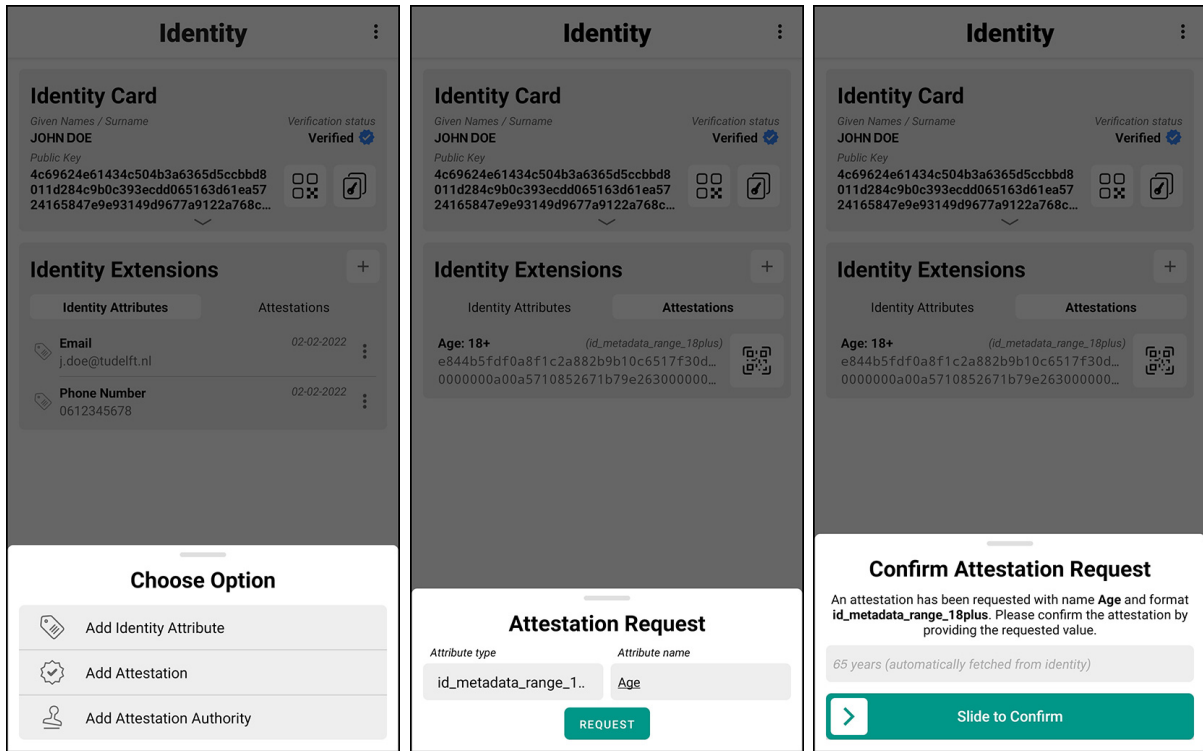


(d) Dialog Exchange Transfer

(e) Dialog Exchange Request

(f) Dialog Exchange Transaction

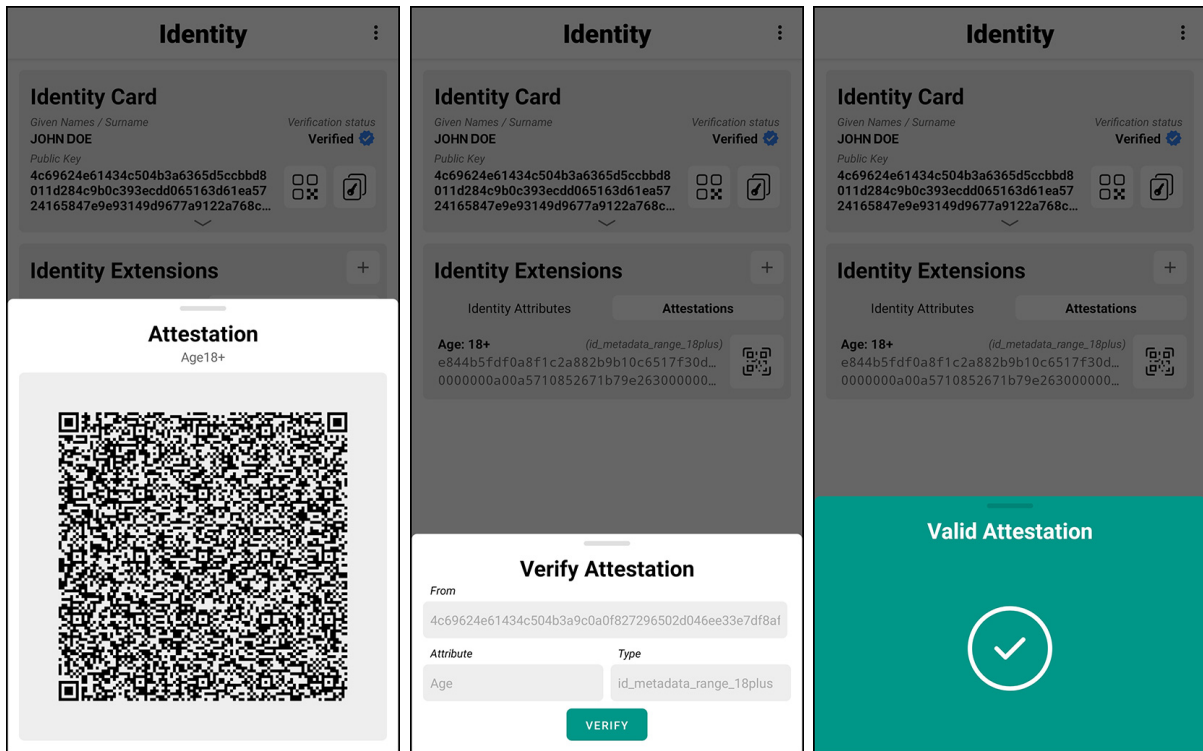
Figure 1.3: Dialogs used in platform (1)



(a) Dialog Identity Options

(b) Dialog Attestation Request

(c) Dialog Attestation Confirm Request

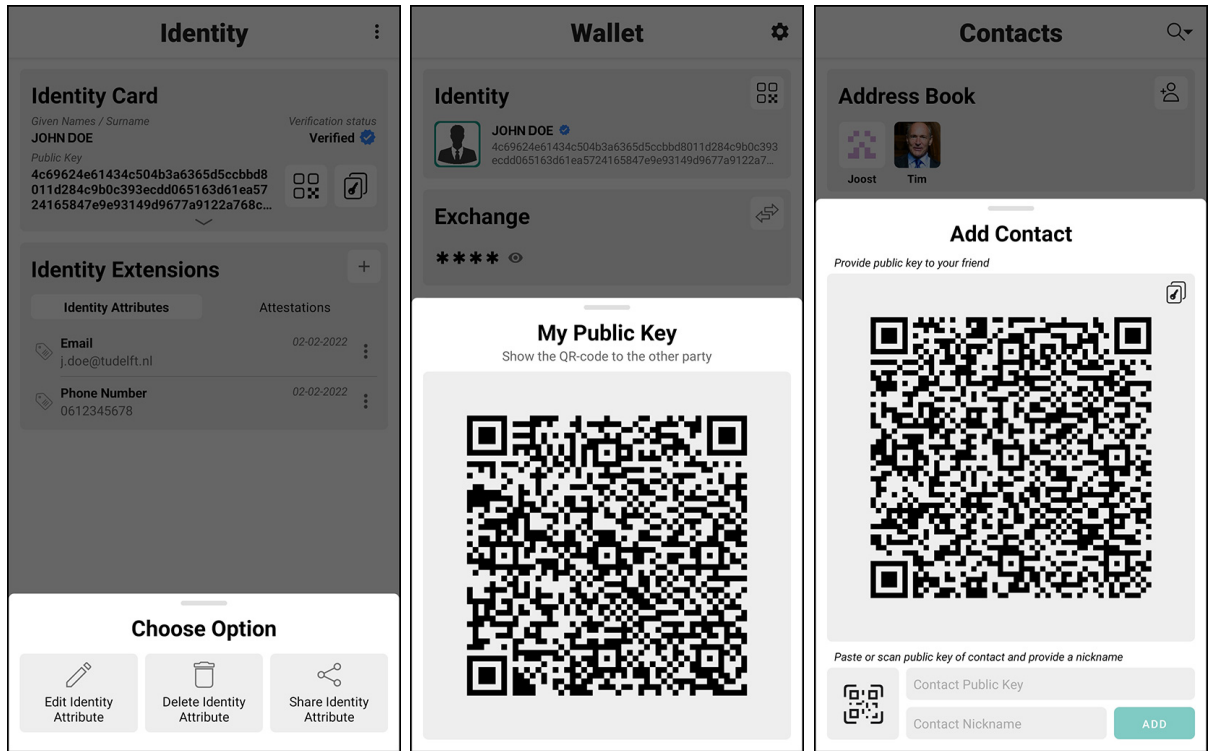


(d) Dialog Attestation QR-code

(e) Dialog Attestation Verify

(f) Dialog Attestation Verify Succeed

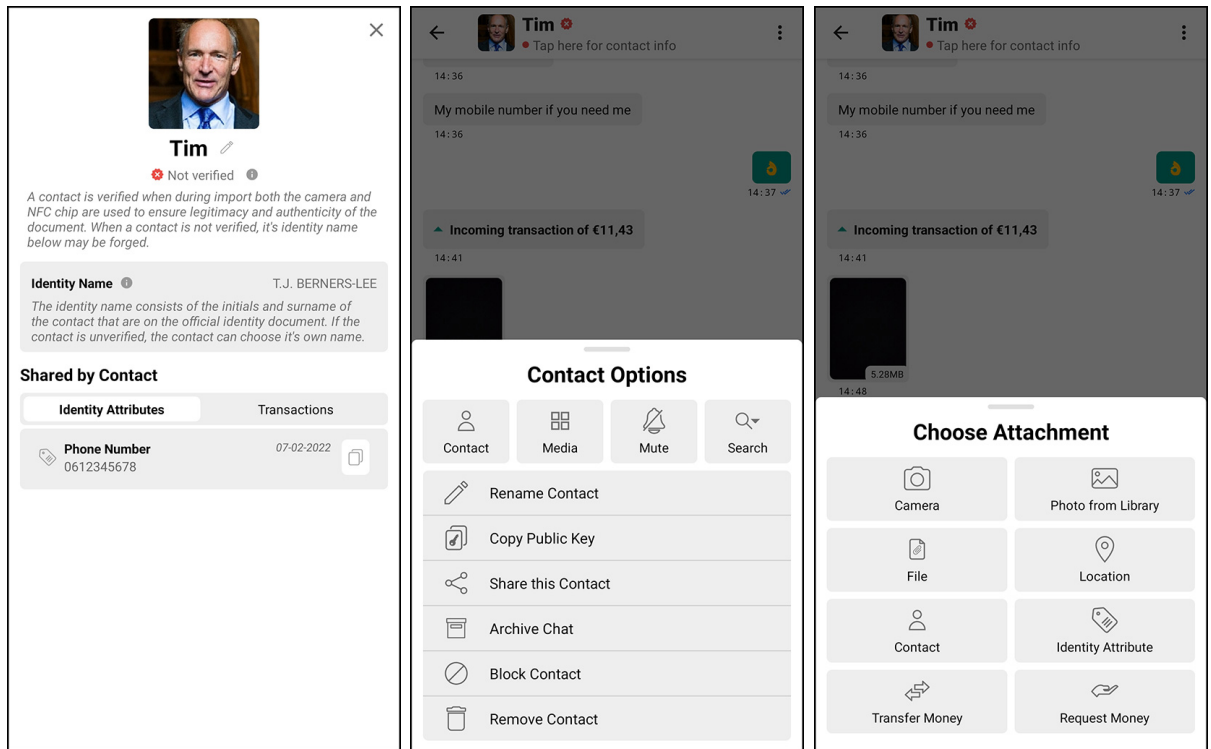
Figure 1.4: Dialogs used in platform (2)



(a) Dialog Identity Attribute Options

(b) Dialog My Public Key

(c) Dialog Add Contact

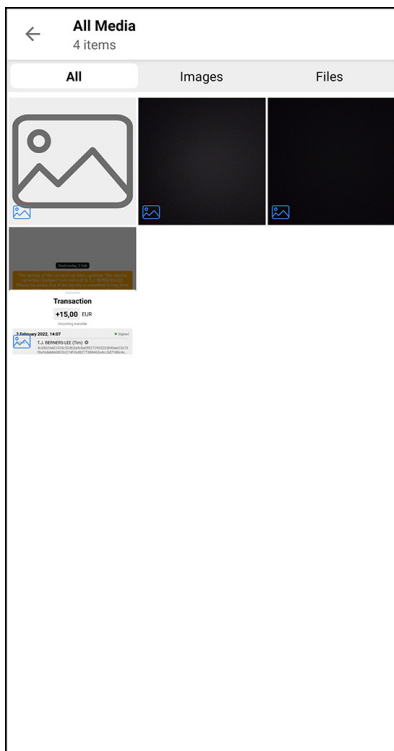


(d) Dialog View Contact

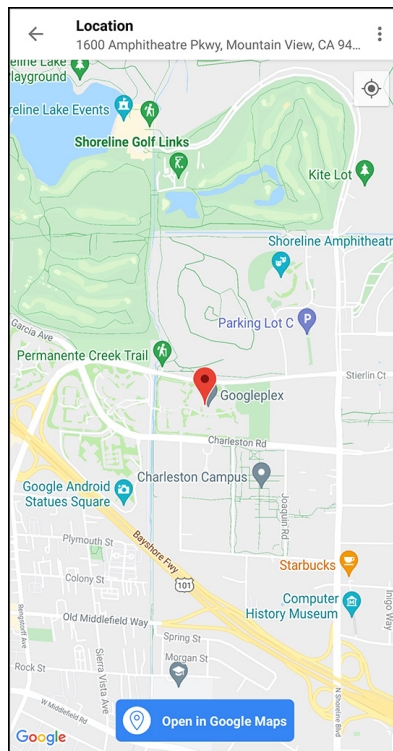
(e) Dialog Chat Options

(f) Dialog Chat Attachments

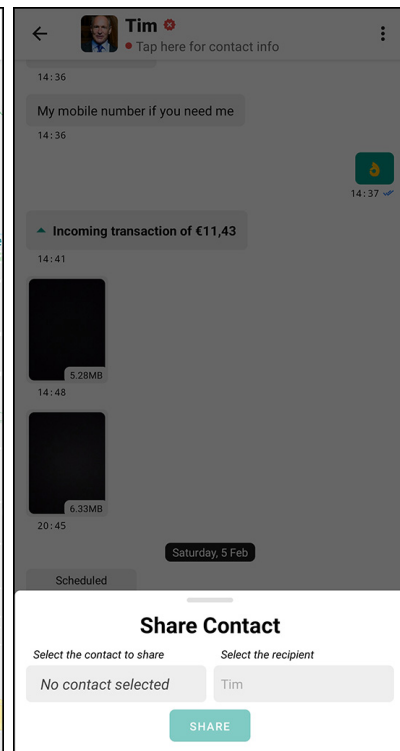
Figure 1.5: Dialogs used in platform (3)



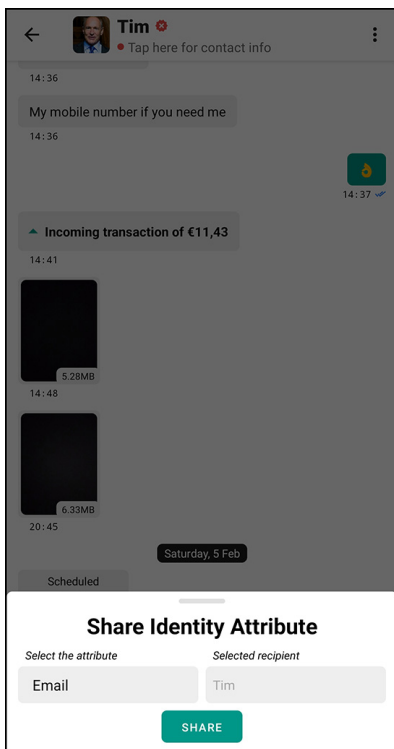
(a) Dialog Chat Media



(b) Dialog Chat Location



(c) Dialog Share Contact



(d) Dialog Share Identity Attribute

Figure 1.6: Dialogs used in platform (4)

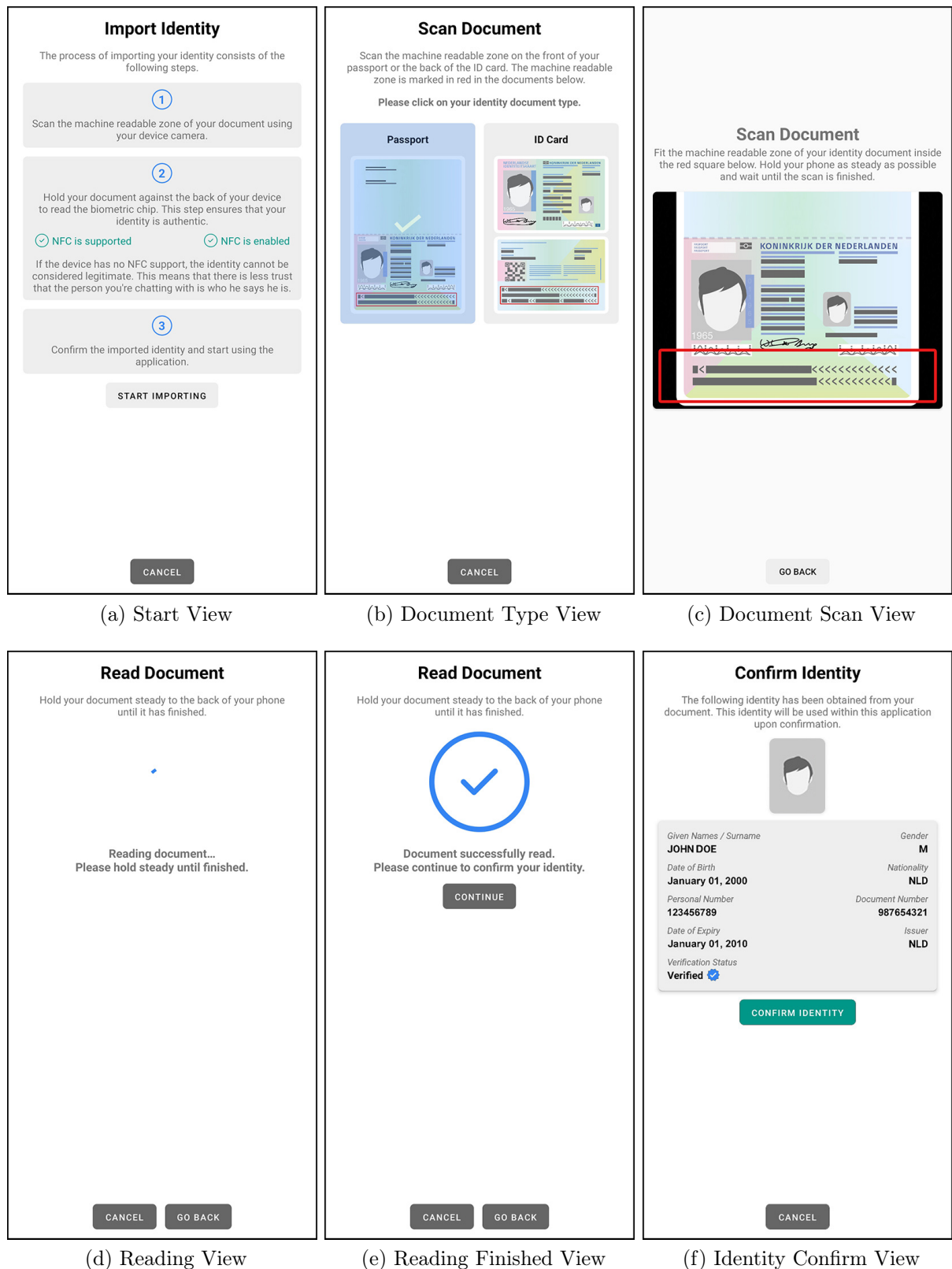


Figure 1.7: Identity onboarding views from start to end

2

Implementation Details

2.1. Data Transfer Protocol

This section mainly serves as a technical documentation of the protocol. The data transfer protocol is specifically designed for use in the P2P network IPv8¹. In this section, we briefly discuss the main components, as well as its integration in the protocol itself. The non-exhaustive flow diagram of the designed data transfer protocol is displayed in Figure 2.1. The settings of the protocol are set upon initialization, as defined in the community, or to default otherwise. The protocol requires the data to be in binary form. The process of converting the data to/from binary form, serialization, and deserialization, is the responsibility of the base community and not the protocol. The protocol can be found in the `kotlin-ipv8` repository².

2.1.1. Protocol Settings

Some of the settings are optimized to ensure maximum performance of the protocol. Other settings are only used to enable specific options. The list below contains all used parameters, including their default value either derived from the experimental analysis or through our findings and estimations.

blockSize (1200) the size (bytes) of the blocks

windowSize (80) the number of blocks in one window

binarySizeLimit (250MB) the total allowed data size (bytes)

retransmitEnabled (true) enables/disables retransmission of packets

retransmitInterval (3000) the allowed interval (msec) before retransmission

retransmitAttemptCount (3) the allowed numbers of retransmits before timing out

scheduledSendInterval (5000) the interval (msec) before checking scheduled transfers again

¹<https://github.com/Tribler/kotlin-ipv8/>

²<https://github.com/Tribler/kotlin-ipv8/tree/master/ipv8/src/main/java/nl/tudelft/ipv8/messaging/eva>

scheduledTasksCheckInterval (1000) the interval (msec) before checking scheduled tasks again

terminateByTimeoutEnabled (true) enables/disables the use of timeouts

timeoutInterval (12000) the allowed interval (msec) before the transfer timeouts

reduceWindowAfterTimeout (16) the number of blocks that reduce the window size after a timeout

loggingEnabled (false) enables/disables logging for debugging

2.1.2. Transfer Sets

To keep track of transfers and other functionalities, the protocol uses sets. Every set starts empty on initialization. During execution, some of these sets will be filled and/or emptied (again). The list below contains the sets that are deemed necessary.

Scheduled set contains the currently scheduled transfers, outgoing for the sender and incoming for the receiver. For each peer, a queue is maintained that orders the transfers by time

Incoming set contains the currently active incoming transfers per peer

Outgoing set contains the currently active outgoing transfers per peer

Transferred sets contain the finished transfers per peer, separated by outgoing and incoming

Stopped set contains the incoming transfers that have been manually stopped

Timed Out set contains the number of time outs for each transfer

Scheduled Tasks set contains the tasks that are scheduled to be executed at a specific time

2.1.3. Callbacks

Callbacks are pieces of executable code that are executed after the occurrence of specific events and enable the initiator of the protocol to execute code that is not part of the protocol itself. An example is the post-processing of data after a successful exchange. The protocol provides support for four callbacks:

onReceiveProgressCallback enables the recipient of the transfer to keep track of the download progress of the transfer. The transfer progress states are scheduled, initializing, downloading, stopped, finished, and unknown

onReceiveCompleteCallback enables the recipient of the transfer to perform actions after the download has completed

onSendCompleteCallback enables the sender of the transfer to perform actions after the upload has completed

onErrorCallback enables both the sender and recipient of the transfer to perform actions upon the occurrence of an error

2.1.4. Scheduled Task

During the transfer, the protocol is required to execute tasks at specific times. If enabled, the protocol is able to schedule transfer timeouts, and retransmission of windows and acknowledgments. These tasks are scheduled to be executed after the allowed `scheduledTasksCheckInterval` has expired. Scheduled tasks are added to the priority queue of scheduled tasks containing the corresponding executable code and the time of execution. The protocol checks every second to find tasks that require execution, based on the defined time.

2.1.5. Main Methods

Send Scheduled

The `sendScheduled` method checks if there are transfers able to start. It is executed periodically with the `scheduledSendInterval` parameter. Transfers in the `scheduled` set are grouped by recipient and maintains a queue of scheduled transfers for each recipient. The transfer can start if the following requirements are satisfied, as also can be seen in Figure 2.1. Firstly, the peer must be connected and idle, meaning there is no other transfer currently active. Secondly, the data size must not exceed the allowed maximum size of `binarySizeLimit`.

Send Binary

The `sendBinary` method acts as an entry point for communities to send or schedule a transfer to another peer. Some additional information is required to start the transfer. Apart from the recipient, knowledge about the community for which the transfer is destined, the data, and an identifier that uniquely defines the data are required. The transfer can be directly started if the same requirements as in the `Send Scheduled` methods are satisfied. If not, the transfers are scheduled to be started at a later time.

Transmit Data

During the transfer, the protocol determines the blocks for the next window. This is a combination of the blocks from the next window and possibly some lost blocks. The `transmitData` method simply creates a data packet for each block and transmits it over the community endpoint.

Send Acknowledgement

The recipient is required to confirm receipt of packets or windows of packets. If during data transmission, some blocks appear to be lost, the protocol embeds these block numbers in the acknowledgment. The `sendAcknowledgement` method creates an acknowledgment packet and transmits it back to the sender. It also schedules a task to terminate the transfer and to resend the acknowledgment if no response returns within the `timeoutInterval` and `retransmitInterval`, respectively.

Terminate

A transfer, outgoing or incoming, is terminated by removing the binary data from memory and removing it from the corresponding `outgoing` or `incoming` set. If the transfer is timed out, the protocol adaptively lowers the window size by `reduceWindowAfterTimeout` to provide the transfer confidence to succeed in the next attempt. This does have a decrease the transfer speed as fewer blocks are sent in one window, requiring more windows and more waiting time.

2.1.6. Packet/Payload types

Each payload serves a different purpose and contains different information.

WriteRequest packet serves as an announcement of a new transfer and announces the transfer-specific settings. The payload contains:

- `info`: community identifier, to which community the transfer belongs
- `id`: unique identifier of data
- `nonce`: unique identifier of transfer
- `dataSize`: size of the binary data
- `blockCount`: number of blocks in transfer
- `blockSize`: size of the blocks in bytes
- `windowSize`: number of blocks in one window

Acknowledgement packet serves as confirmation of a received `WriteRequest` or `Data` packet. The payload contains:

- `nonce`: unique identifier of transfer
- `ackWindow`: received window number
- `unAkedBlocks`: block numbers of unreceived blocks

Data packet serves as container of the binary data, splitted in blocks. The payload contains:

- `blockNumber`: receive block number
- `nonce`: unique identifier of transfer
- `data`: the data embedded in this block

Error packet serves as announcement for occurrence of an error. The payload contains:

- `info`: community identifier, to which community the transfer belongs
- `message`: explanation of the error

2.1.7. Packet Listeners

To be able to handle incoming packets efficiently, the protocol applies the concept of listeners. A listener is a function that waits for the occurrence of a specific event, in this case, the receipt of a packet. These listeners are called from the main IPv8 community as it handles all incoming communication with other peers. Since we've incorporated four packet types in our protocol, there are also four listeners attached to our protocol. The process of each listener is displayed in an understandable manner in Figure 2.1.

onWriteRequest

This listener belongs to the recipient of the transfer only and is triggered upon receipt of a `WriteRequest` packet. Upon receipt it first checks if (I) the transfer is not already incoming, meaning this announcement has been retransmitted, (II) the transfer has not already been transferred, and (III) the transfer is currently stopped. Additionally, the size of the transfer is validated to be within defined boundaries. If all those requirements are satisfied, the transfer is accepted by sending an `Acknowledgment` packet back to the sender of the transfer and placing the transfer in the set of incoming transfers, including its transfer-specific settings embedded in the packet. The receiver also schedules the retransmission of the acknowledge and schedules the termination. If a data packet arrives (within the allowed intervals), both scheduled tasks are discarded.

onAcknowledgement

This listener belongs to the sender of the transfer only and is triggered upon receipt of an Acknowledgement packet. The receipt of this packet confirms the receipt of either a WriteRequest or Data packet. First it has to be determined to which transfer this acknowledgment belongs. Using the outgoing set of the peer and the transfer nonce in the payload, the correct transfer is identified. If it is a confirmation of receipt on the previously sent WriteRequest, it starts by sending the first window of blocks and increasing the window number by one. If it is a confirmation on a window of blocks, the next window of blocks is determined, including the unreceived blocks that are announced in this acknowledgment. The window number is increased by one and transmitted to the peer. If it was a confirmation on the last window of blocks, and there are no unreceived blocks, indicating that all blocks have been received, the transfer is finished, terminated, and added to the transferred set. There is no further communication required as the recipient received all blocks and terminates as well. The protocol always tries to progressively continue to the next window, even if there are unreceived blocks. The protocol only remains in the same window when the window of blocks haven't been received or acknowledged.

onData

This listener belongs to the recipient of the transfer only and is triggered upon receipt of a Data packet. The transfer is located in the incoming set, with the corresponding peer and nonce. By means of the block number, the data in the block is inserted to the correct location/place in the total data file. Even blocks that arrive in a different order are able to insert their data to the total data file. After the last block has been received, an acknowledgment is sent, the incoming transfer is finished, added to the transferred, and terminated. The corresponding callback function is executed that converts the raw binary data to a file of the correct format. The data temporarily stored in memory is cleared. If there are still unreceived blocks left, and this block appears to be the last block of its window, the protocol only sends acknowledgment. If the block is not the last block of the window, the protocol does not send an acknowledgment.

onError

This listener belongs to both sender and recipient of the transfer and is triggered upon receipt of a Error packet. Upon receipt, the error (message) is notified to the protocol and the corresponding transfer is terminated.

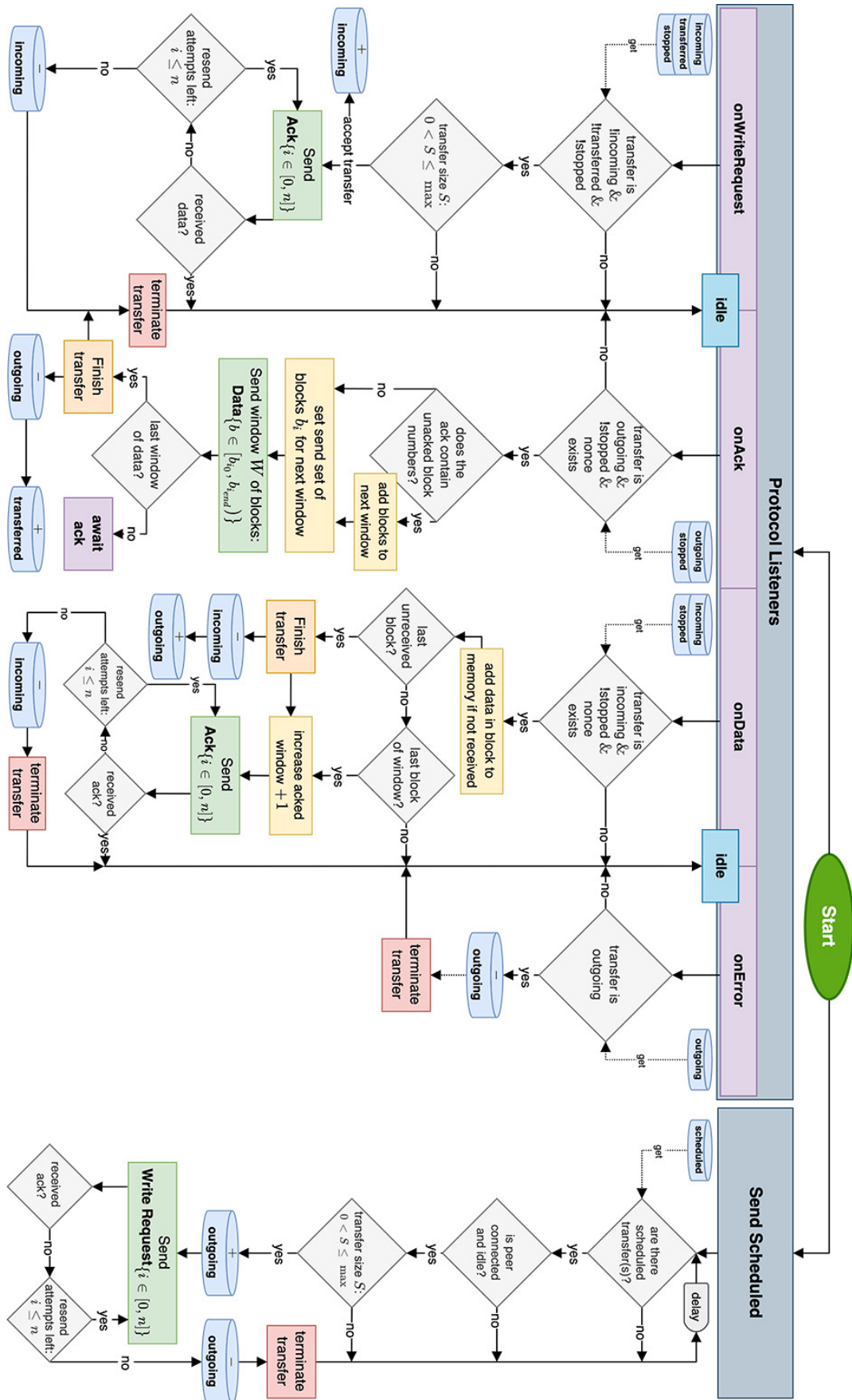


Figure 2.1: Flow diagram of data transfer protocol

2.1.8. Experimental Analysis

The purpose of the experimental analysis is to find the optimal settings under normal operation in terms of performance. We define the protocol to be optimal if (I) the transfer speed is as high as possible, (II) the number of lost packets/blocks is as low as possible, and (III) the number of retransmitted windows of blocks (by the sender) and acknowledgments (by the receiver) is as low as possible. By varying each parameter while keeping the other parameters constant, we can observe the effect of the parameter on the performance. The block size is varied in steps of 100 bytes, ranging from 600 to 1200 bytes. The window size is varied in steps of 16 blocks, ranging from 16 to 128 blocks. Every combination of parameters is executed precisely five times for two file sizes. Figure 2.2 contains the results of all executed tests. No latency and package loss emulation has been performed. The obtained parameters provide optimal performance under normal operation of the protocol. The optimal block size is 1200 bytes and the optimal window size is 80 blocks, equivalent to 96000 bytes or 96kB (decimal). The file size appears to be independent of the performance of the protocol. No latency or packets loss emulation is executed. The optimal parameters are analyzed under normal operation only.

2.1.9. Performance Evaluation

In the experimental analysis, we've only tested relatively small files. For the evaluation of the performance, using the optimal parameters, we transfer a large file of 250MB and evaluate the performance using the same indicators as in the experimental analysis. The exchange of the files in the experimental analysis was executed within the same (local) WiFi environment. As this is not the case in most situations, we, therefore, evaluate three different scenarios: WiFi to WiFi, 4G+ to WiFi, and 4G+ to 4G+. For the inter-WiFi exchange, the devices are connected to the same local network. The devices with 4G+ use the network of telecom providers Vodafone and KPN. The transfer of each file is executed 10 times for each situation. Figure 2.3 contains the results of the evaluation of the performance and delays. As there were no window retransmissions it was left out of the figure.

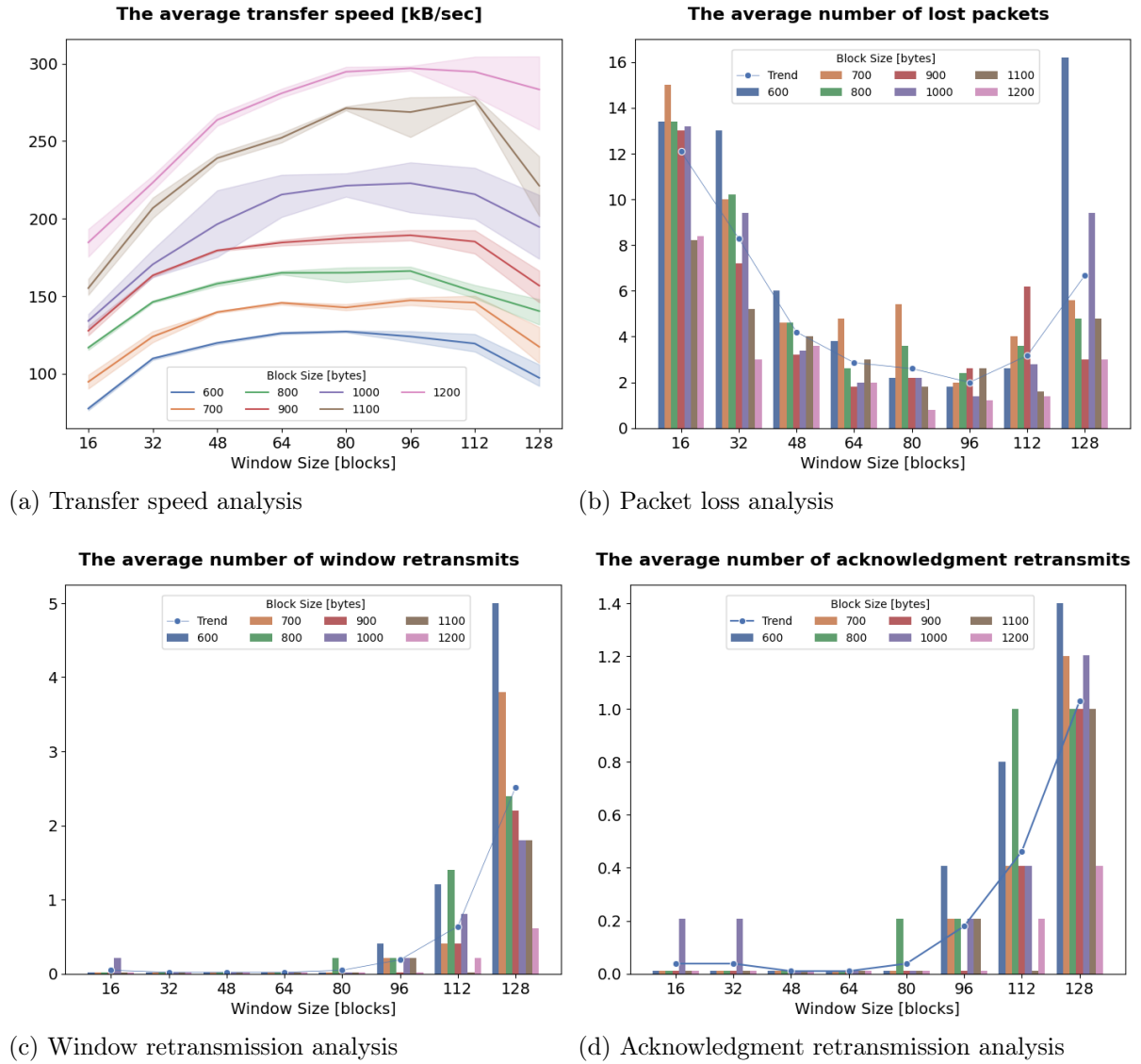


Figure 2.2: The results of all executed tests for each variable window and block size, averaged over five iterations.

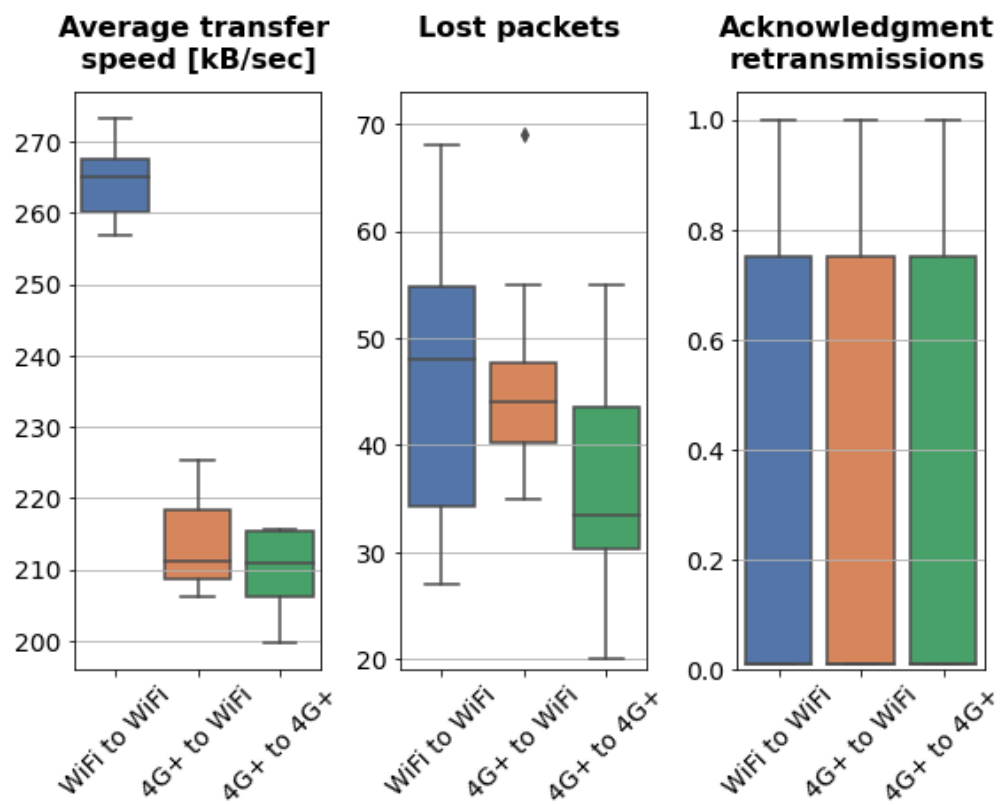


Figure 2.3: Evaluation of the performance of a transfer of 250MB using the optimal window size of 96kB, executed 10 times.

Bibliography

- Clarke, P. (2014). Wikimedia commons, the free media repository. [https://commons.wikimedia.org/wiki/File:Sir_Tim_Berners-Lee_\(cropped\).jpg](https://commons.wikimedia.org/wiki/File:Sir_Tim_Berners-Lee_(cropped).jpg)
- Otte, P., de Vos, M., & Pouwelse, J. (2020). Trustchain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems*, 107, 770–780. <https://doi.org/https://doi.org/10.1016/j.future.2017.08.048>
- Rost9. (2020). Depositphotos. <https://nl.depositphotos.com/serie/208667760.html>
- Tribler. (2021). Ipv8 documentation. https://py-ipv8.readthedocs.io/_/downloads/en/latest/pdf/