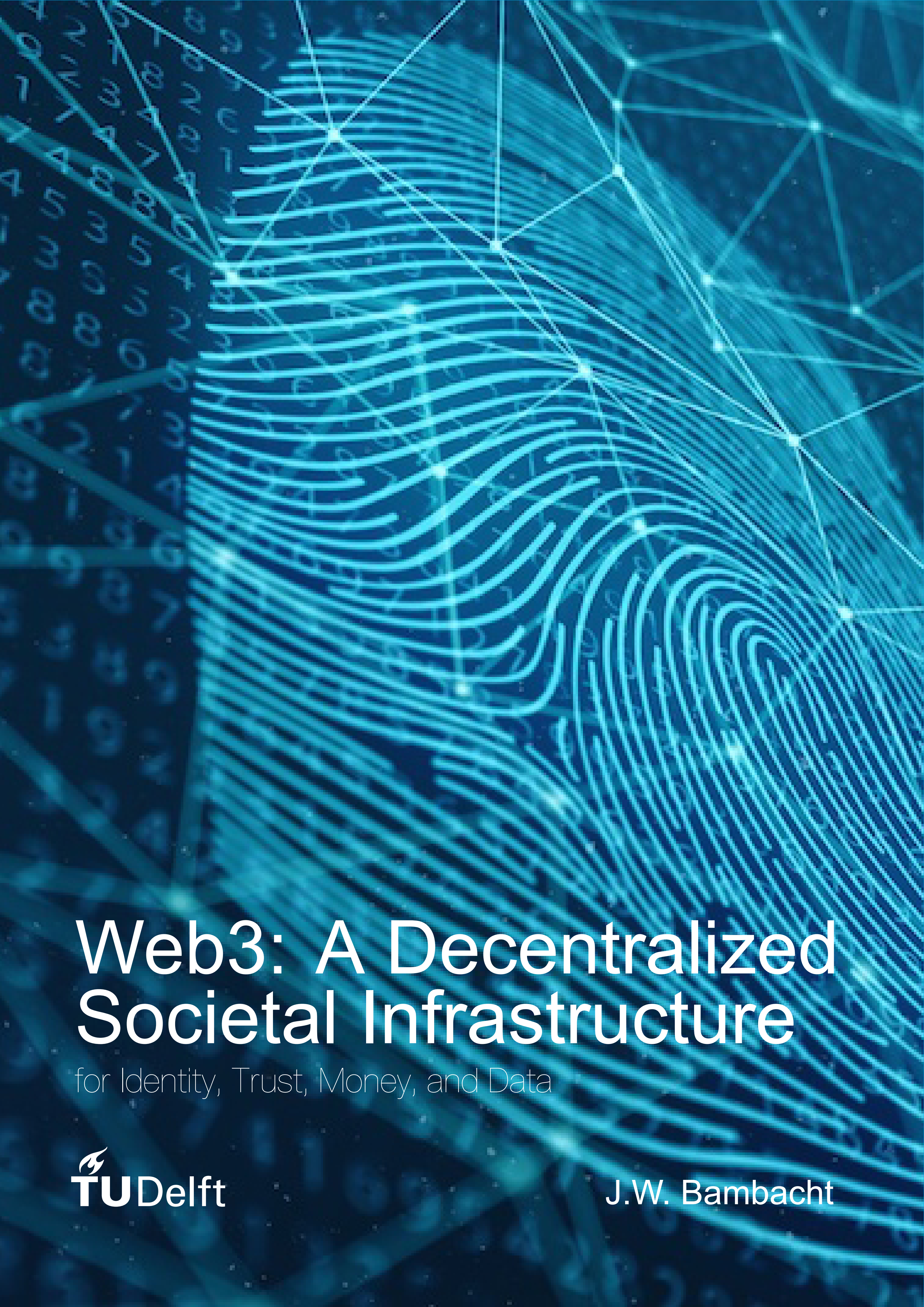


Web3: A Decentralized Societal Infrastructure

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 XXXX February XX, 2022 at XX:XX AM.

Delft University of Technology
Ministry of the Interior and Kingdom Relations

Rijksdienst voor Identiteitsgegevens
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Author
J.W. Bambacht

Supervisors
J.A. Pouwelse, TU Delft
A. de Kok, RvIG

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

Contents

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
J.W.Bambacht@student.tudelft.nl, J.A.Pouwelse@tudelft.nl
Distributed Systems, Delft University of Technology
February 17, 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. Governments similarly keep control over their citizens' personal identities. 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 they deserve. 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 SSI 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 digital signing of documents, validated storage of diplomas, and COVID vaccination certificates of the concerned citizen.

Governments, banks, and tax offices have general insight into the bank accounts and transactions of citizens, often exploited using big-tech cloud services. Not only is this a violation of citizens' privacy, but the application of these banking services also has 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]. The costs and speed are even worse for cross-border payments. 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 restriction-less cross-border payments, CBDC can transform the financial system into a sustainable and frictionless system.

This research makes the following contributions: (I) design of a novel decentralized infrastructure that incorporates a self-sovereign identity, (II) 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 and applies it in a useful manner to facilitate authentic trust between users, and private transfer of money and data between identities in a permission-less fashion. In a centralized infrastructure, platform owners are still able to collect metadata belonging to encrypted data. By removing these single points of failure, the violation of privacy and security of users is reduced.

One of the key aspects of our research is the application of citizens' self-sovereignty 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. Although 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 lastly, 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 a select few 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.

The self-sovereignty of data, in any form, is a fundamental issue of centralized platforms. 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, is more respected in terms of their 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 as citizens control and manage their own identity, 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

The application of self-sovereign identity enables citizens to be in control of their own identity. Governments enable citizens to authenticate organizations and institutions to their identities, stored on their central server. DigiD[1], the primary identity authenticator in the Netherlands, enables citizens to authenticate using their mobile phone. Personal data is transferred from the government's server to the organization's server. This requires a perfectly secure connection

[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]	✗	✓	MTProto	✓	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	

[a]the current state of development in terms of completeness and usefulness
[b]E2E encryption not enabled by default
[c]no storage of metadata, only required for routing
[d]fork of Signal Messenger, onion routing for metadata anonymity, undelivered messages stored one of the distributed service nodes
[e]everyone can set up node and will be rewarded in crypto token SYLO

and infrastructure on both sides. Unfortunately, the citizen has no control over what information is actually shared. These authentication mechanisms are extremely expensive, especially during the COVID crisis due to scheduling of vaccination and test appointments, as €$0,13$ is credited for every successful authentication [28]. Self-sovereign identities can be successfully applied to mobile applications that replace the necessity of authentication services like DigiD. The first example is IRMA[2], 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 and is able to see the required information and what is actually shared. Sovrin Network[3] shares the same methodologies and enables other developers to build their own SSI application on top, but instead uses a 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 the concept of (centralized) authorities that handle the revocation. This is a violation of the principles of SSI as it should be an authority-free system. The work of Chotkan [29] incorporates a distributed attestation revocation for self-sovereign identities. Offline verification is more privacy-aware and offers a more robust solution for digital attestations.

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. In Table I a (non-exhaustive) list of significant and related competitors in the market is portrayed. The difference in characteristics between the centralized and decentralized platforms shows a clear clustering. The centralized platforms all require a privacy-sensitive asset to be able to use it and many different attributes are shared with contacts for identification and trust enforcement purposes. The decentralized platforms are examples of Privacy by Design [30] implementations as they try to minimize the leakage of privacy-sensitive information. There are no explicit requirements for its use and the trust attributes are limited to manually chosen names and profile pictures.

WhatsApp [18], Facebook Messenger [19], and specifically WeChat [20], are all fully centralized platforms that store metadata of their users. All platforms but WeChat have integration for commonly-used E2E encryption curves, due to their performance in terms of speed and secrecy, and only Telegram [21] applies their self-designed protocol. WeChat, which is monitored by the Chinese government, incorporates strong censorship and interception protocols for data exchanged by its citizens. Luckily, this degree of violation is not present in any other (centralized) platform. Also, these centralized platforms are often obliged, also because of their infrastructure design, to provide information upon request in the form of metadata to governmental instances or apply censorship in some situations. Signal Messenger [23], which is centralized but specifically designed with privacy in mind, do not store any

[2]https://irma.app
[3]https://sovrin.org

personal information. Central servers, however, are deemed necessary for routing and functionalities like account recovery using the same phone number or email. 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 rather do not share the complete structure of their platform openly.

Decentralized infrastructures try to realize anonymity by reducing the metadata in the network as much as possible. Session Messenger [24], a *decentralized* fork of Signal Messenger, attempts to provide anonymity and preservation of privacy using a technique called onion routing [31]. It makes it nearly impossible for any intermediary (node) to derive both the sender and receiver of the message. It is not possible to apply this technique 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 provide the most preferable features, apart from the absence of self-sovereign identity integration. Status is developed on the Ethereum network and incorporates their own utility network token to fuel their network and provide paid features to users. In a similar fashion to Session Messenger, undelivered messages are stored on nodes that obtain your IP address to deliver at a later moment. This is not a desirable characteristic as this contradicts the principles of privacy. Sylo is a fully operational platform that cannot withstand the leakage of information in the metadata and does not provide full transparency to its users. Berty has all potential as it is secure and transparent, minimizes leakage of privacy-sensitive information in terms of metadata and requirements, but is currently underdeveloped.

As we've seen, there exist many different implementations that are designed on similar characteristics. The idiomatic platform is decentralized and P2P with no temporary storage of messages on nodes (as convenient as they are), incorporates trusted curves for E2E encryption, does not require and apply the use of metadata, and has no useless identifiable requirements. Our design additionally and uniquely incorporates a self-sovereign identity as a requirement, providing various functionalities. Trust enforcement attributes should be limited to not only manually forge-able components as it achieves higher trustworthiness. Furthermore, the platform must contain private and secure mechanisms for data transfer and the transfer and store of 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 data to traverse many other nodes on its path to its destination, still allowing them to store metadata. A P2P network makes it possible to communicate directly between 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. Nobody other than the participants is able to store or share metadata, or even sell metadata to third parties. The communication itself can only be sufficiently secure if the message, or data, is encrypted. We also require a component to store and exchange data in a distributed manner. In distributed systems, one of the requirements is full synchronization across many independent nodes. This is difficult to realize for 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 store of data that does not need continuous synchronization for all peers. The blockchain is often applied to store transactions between two individuals 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 very basic form of storage is a fast, lightweight, and structured alternative to conventional storage. Every transaction can be back-traced to create a well-organized overview, which is well suited to serve as a wallet. In Figure 1 a low-level overview of our infrastructure is portrayed.

Peers in the network are constantly observing (and updating connectivity status) other peers in the network as no central server or node monitors the online activity of participants. A certain degree of anonymity is required as it is not desirable to spread personal information to (unknown) peers in the network. The infrastructure desperately requires some sort of anonymous form of peer identification. To ensure a completely secure communication channel, the principles of the CIA Triad [32] must be in place. The objectives of a secure system include *Confidentiality*, *Integrity*, and *Availability*. Confidentiality is achieved by encryption as it ensures that data is only accessible to authorized parties. Digital signatures ensure the integrity of the data by providing proof that it is originated from the sender and has not been altered by any third party. The availability is slightly more difficult to ensure in decentralized systems, especially in P2P networks, due to the dependence on the connectivity of individual peers (or nodes).

Public-key cryptography [33] is the most commonly-used mechanism for secure communication. Not only does it provide a confidential exchange of messages and data, but it is also a way to identify peers without exposing any private information. Each peer is provisioned with a so-called public-private key pair. The private key is generated at once and should only be known to its owner. The private key performs the decryption of encrypted data. The public key is mathematically derived from the private key and may be publicly disclosed to others without damaging its security as it is computationally infeasible to derive the private key from the public key. The public key serves multiple purposes. Firstly, it provides a way to find and identify other peers. Secondly, encryption of data is performed using the public key of the receiver. Even the encrypter is unable to read the contents anymore. And lastly, digital signatures provide proof of authenticity of a piece of data and can be verified with the public key of the signatory.

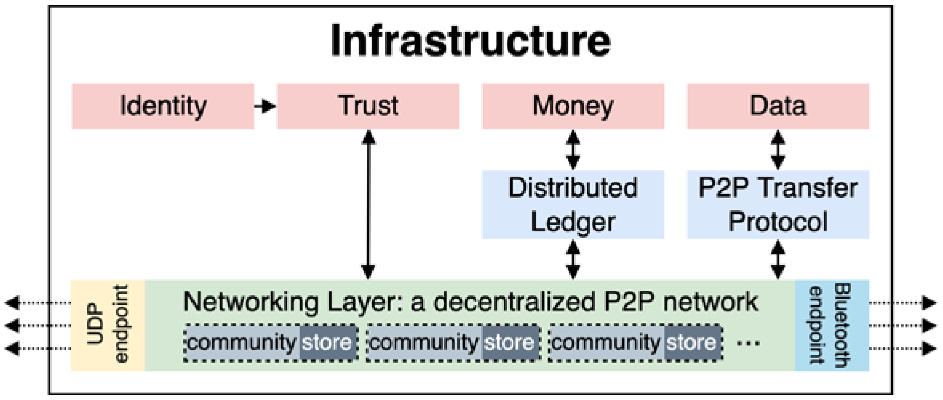


Fig. 1: General overview of the infrastructure

A. *Networking Layer*

To be able to communicate with other peers we need a networking layer that handles all outgoing and incoming communication. This can be achieved with IPv8 [1], a P2P networking layer that provides authenticated and privacy-aware communication between peers. IPv8 is developed as an academic successor of IPv4 and attempts to overcome IPv4's weak characteristics and increasing list 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 is capable to establish connections to other peers, even for devices that are connected behind NAT or a (strong) firewall. Customized NAT traversal techniques, e.g. UDP hole-punching, are effectively applied to provide connectivity with increased privacy. The endpoints of the networking layer are independent of any central infrastructure.

The aim is to reduce the exposed metadata to an absolute minimum. It requires that data packets are not widely broadcasted on the network, hoping other peers are able to deliver the packet to the intended recipient. Some P2P systems conveniently employ distributed nodes to deliver data to peers that are currently not connected to the network. These nodes temporarily store the data and metadata of the message until it has been successfully delivered. Although this conveniently addresses the availability property of the network, it also risks the storage and 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 any intermediary nodes or peers. As peers come and go, it may happen that peers change connectivity status or change their network address. In these situations the peers announce their new address to all previously connected peers. This, however, may sound contradicting in terms of privacy. Hence no personal information, including the intentions and communication histories, can be deduced as peers also connect to random peers to increase their network reach.

IPv8 applies the concept of the so-called network overlay or *community*. This enables developers to build applications on top of the base networking layer by creating their own community. In our case, we need several communities to satisfy our requirements. The base community includes all functionality in regards to peer connectivity, communication, data serialization, and encryption. One could argue that the IPv8 networking layer actually combines multiple layers together, as it handles various functionalities from the OSI model [34]. Every community may have a different list of connected peers, based on the peers that chose to join the community or haven't connected (yet). Every community that requires the storage of data is in need of a service that handles the interaction with a database, alternatively called a *store*. A specific discovery community handles the discovery and the connection with (new) peers that are present in the same community. 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.

B. *Distributed Ledger*

The ecosystem requires a distributed ledger that provides the transfer of money and storage of transactions. TrustChain [2], a permission-less scalable distributed ledger, is already integrated as a community on top of IPv8 and is therefore a proven candidate. TrustChain has the capability of sending and receiving trusted transactions between peers. The blockchain-based data structure is a tamper-proof immutable chain of transactions. There is no central control over the transactions. Every peer implements a personalized chain that only contains related blocks to the peer, i.e. either sent or received by the peer. TrustChain has three basic functionalities: sending/receiving of blocks, broadcasting of blocks, and crawling of chains. The send and receive process merges both parties in one transaction, see Figure 2. The initiator (p_0) signs and sends a *proposal block* (a half block) to the counterparty (p_1). On receipt of the *proposal block*, (p_1) creates, signs, and sends the *agreement block* back. During the process, the integrity of the received block is validated and both parties add the half blocks to their chains. The half blocks are linked by the public key of the counterparty. The transaction is considered complete

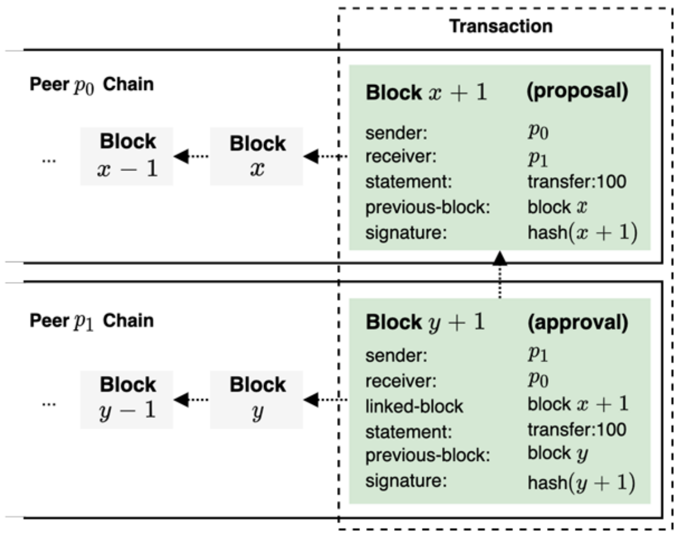


Fig. 2: Transaction in the ledger

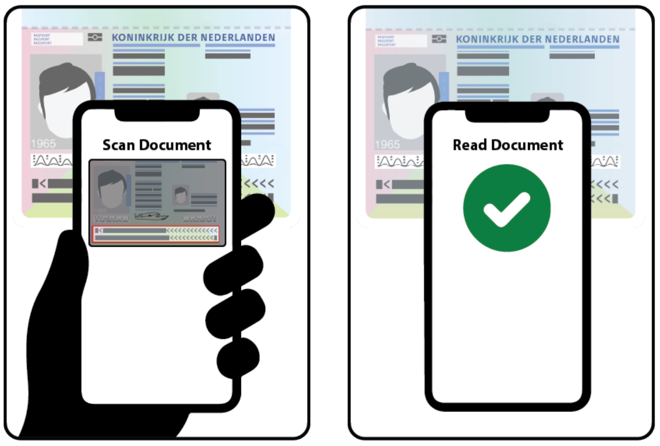
when both parties received and signed both half blocks. Not only is it possible to send a block to a specific peer, but the broadcast of a block will also send the block to all currently connected peers. The crawl functionality is nothing more than retrieval of a peer's chain using its public key.

V. Design

The design of the platform can roughly be divided into four main elements: 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 their legally valid governmental document in a self-sovereign manner introduces various new opportunities. One of these purposes is the authentication of online institutions. The owner controls the exchange of its own identity information to organizations instead of the conventional online governmental authentication that blindly transfers every available piece of 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 attributes from the government's central server once. This is not a suitable option in our system as we desperately want to eliminate the use of external central servers. Every (adultery) citizen is obliged to possess a physical government-issued travel document in the form of a passport or identity card. These documents contain (visually) the exact same identity information in the machine-readable zone (MRZ) as on the government's server, and are therefore perfectly suited to be used instead. The identity document onboarding process is executed in two consecutive steps. The first step consists of scanning the MRZ zone of the document using the camera of the phone, see Figure 3a. A combination of AI and check digits embedded in the MRZ zone ensure that the required attributes for the second step are valid. These attributes are stored on the phone after a successful scan. The



(a) Scan MRZ zone of document using the camera of the phone **(b)** Reading document using the NFC chip of the phone

Fig. 3: Step of identity document onboarding process

second step is executed in combination with the documents' built-in biometric chip and the NFC chip of the phone. This step is extremely important as it proves the authenticity of the document in digital form. Without this validation step, there is no way to determine the correctness of the scanned attributes. The phone has to be placed with its back to the document to make contact, see Figure 3b. The biometric chip communicates with the NFC chip to fetch all attributes digitally from the document. Before these attributes can be transferred to the phone, a connection has to be established. The authentication requires three valid attributes (document number, date of birth, and date of expiry) to successfully establish a connection and exchange the document attributes to the receiving phone. The application stores all document attributes on the phone and combines them to create the self-sovereign identity. This process is deemed authentic and secure because (I) a physical document is required and the 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 [35]. It is important to note that it's not possible, and possibly never will, to revoke access to a stolen or lost document without online knowledge from a central server. The self-sovereign identity will be valid, just like official 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 any detail in the MRZ zone 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 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 some details 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 directly violates your privacy and can even cause identity theft or misuse of a person's authenticity. People 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 pieces of information that can be verified using attestations. Chotkan [29] designed a framework that incorporates revocable verifiable claims without revealing the actual requested piece of information using zero-knowledge proofs. To apply variable 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 effectively be applied as another layer of protection for unauthorized access of 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 multiple types of trust. The first natural form is trust in a system or platform, especially for a centralized platform. 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 better 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 sent 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 authenticity and identity of the other participant. The confidence in the authenticity of the contact is determined based on 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 can 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 [36] is for example an attack in which individuals are targeted with the explicit use of personal information with the goal 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 in case you got a new phone, it also exposes risks as hackers are able to take over your account on their phone and communicate with your contacts instead. This has the deficit that hackers, within the context of (spear) phishing attacks in messaging platforms, are able to use these accounts to steal confidential information or money from trusted unsuspecting contacts. These hackers try to mimic as much confidential information of the hacked person to not arouse suspicion or to simply gain trust with their new victims. Fortunately, this type of attack does not exist in our design as migration is not possible.

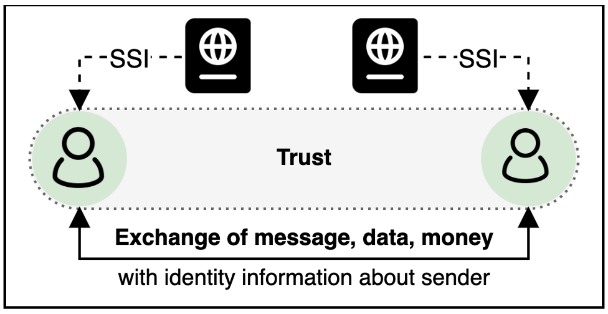

SSI
SSI
Trust
Exchange of message, data, money
with identity information about sender

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

The challenge is to exchange just enough trust to the recipient of your message, without exposing an unnecessary amount of private information. In the initial phase of the conversation, especially if the users are connected in some online way, trust (or distrust) plays a major role. As valid self-sovereign identities are incorporated in our design, we can access and conveniently 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. 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 have not been able to use the NFC chip. These users can't be verified 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

The next step is to decide on the contents of the exchanged name, as we basically have access to the full official name. The transfer of too much confidential information can lead to malicious misuse of them or their contacts. Various combinations of the given and last name exist that aim to provide trust, see Table II. While combination `I` is too general and unidentifiable, the use of the last name in `II` provides more specificity but may still be too general in most cases. 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.

The name and photo attributes of the peer's identity are sent along with every message, obviously in an encrypted manner. 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 alteration of the identity in case a phone is accessed unauthorizedly or stolen. To provide on-device authorization, it is desirable to utilize 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 that purposely attempt 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 countries and unions are currently exploring the economic and technical feasibility of Central Bank Digital Currencies (CBDC) [37, 38, 39, 40]. 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 offers 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, which is already in the advanced stages of the development of its CBDC, is currently testing the implementation of its digital Yuan wallet in a second pilot [41]. 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 make an attempt to trace back the wallet balance. Blockchain still provides pseudo-anonymity because participants of transactions are often identified by a public key only. 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 the 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, but it may even serve some of the fundamental financial primitives (lending, borrowing, liquidity, etc.).

Currently, many external services or banks provide the functionality to create payment requests. Its creator shares a link to all participants that redirects to a 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 [42] using payment requests make the service vulnerable, specifically 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 [43]. It utilizes the distributed ledger TrustChain [2] that builds upon the technologies of IPv8 [1]. The EuroToken protocol tries to offer a scalable, privacy-aware, and cheating-resistant system for the exchange and storage of transactions. TrustChain implements the ledger and communication over the IPv8 network. 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. TrustChain is not only limited to financial transactions and can basically serve any purpose. Transactions are generally settled within seconds, independent of within-border or cross-border payments, but require the availability of both parties to completely settle the transaction.

One of the aspects of a tokenized system is the acquiring of tokens. It requires at least one option to buy and sell these tokens for the system to be useful. 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 be able to send tokens. A balance is either acquired using the exchange portal or received by other peers. The balance of a wallet is determined and validated using the blocks on the chain. 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 aspects of secure and private communication is the transfer of data in a decentralized and P2P manner. 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 may be 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 and reliable. However, the protocol has proven to be limited in terms of performance and unreliability for larger-sized blobs. For proper use in our platform, it was deemed necessary to design a custom data transfer protocol that provides decent performance and reliable exchange of data, in a similarly 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 [44], the Trivial File Transfer Protocol. TFTP is a simple and connection-less data transfer protocol, with the consequence that no authentication and security mechanisms are provided. The transfer of confidential data in (external) networks is unsafe, and therefore not recommended. In P2P networks it is difficult to maintain established connections, and therefore does not suit the long-existent connection-oriented TCP protocol. For a connection-less protocol, the application of the User Datagram Protocol (UDP) [45] 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 handle and keep track of unreceived packets and request retransmission if necessary. 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 receiver of the packet. The normal operation of the protocol is showcased in Figure 5. The sender of the transfer first has to request

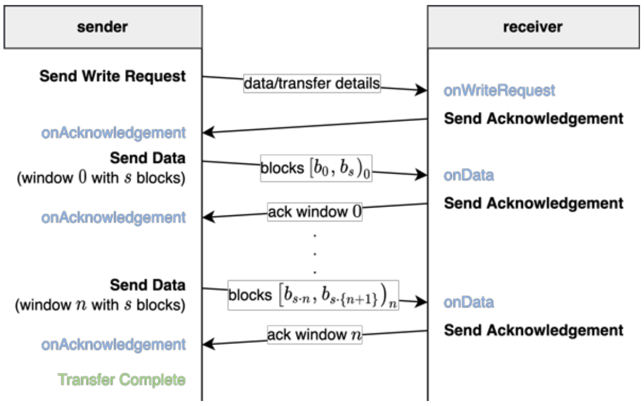


Fig. 5: Normal operation of the data transfer protocol

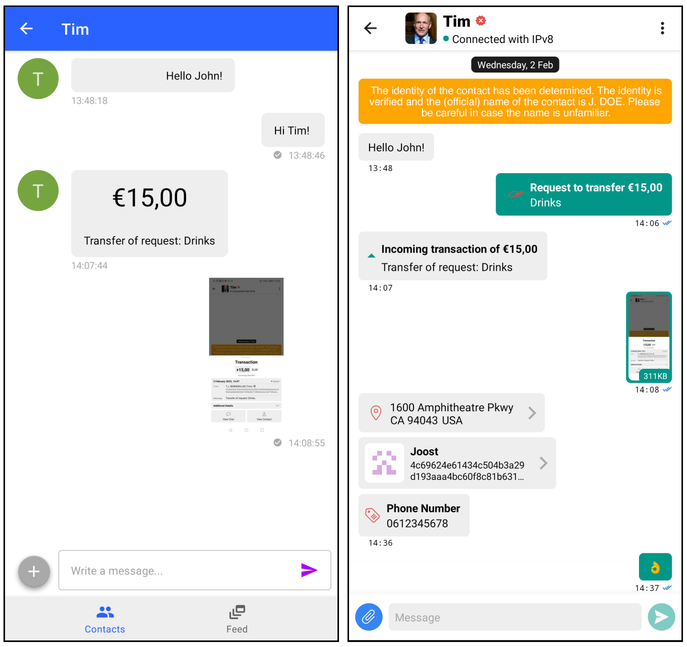
to write data by sending a **`WriteRequest`** payload/packet to the receiver. With this packet, the sender additionally announces all transfer- and data-specific details to make sure both parties have the same understanding. The receiver confirms this request by returning an **`Acknowledgement`** packet. 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 MTU (Maximum Transmission Unit) of the Ethernet [46]. 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 so-called *blocks*. Each of the blocks has a particular block size in bytes and all blocks combined contain the complete data blob. 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 send 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 unreceived 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 the predefined limit, or both peers try to start a transfer at the exact same time. In these situations an **`Error`** packet that contains the reason of refusal is returned. 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 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 while the other peer is for example disconnected. 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 timeouts 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* lowers 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 and every situation, it is undesirable to lower the performance of all normal operations, due to an (extremely) small 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.

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 of the platform is additionally concerned with the UX and UI design of the platform. The platform has been integrated within the TrustChain superapp[4], an Android mobile application developed in Kotlin that contains many different mini-applications built on IPv8 and TrustChain. The data transfer protocol is integrated into the IPv8 stack[5] 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 9b, provides a widget-like overview of the identity, exchange, and chats components. For each of the widgets, there is a separate view that includes all related aspects. The information in these widgets is carefully selected to not overload the user with information.


Tim
Hello John!
13:48:18
Hi Tim!
13:48:46
€15,00
Transfer of request: Drinks
14:07:44
14:08:55

Tim
Connected with IPv8
Wednesday, 2 Feb
The identity of the contact has been determined. The identity is verified and the (official) name of the contact is J. DOE. Please be careful in case the name is unfamiliar.
Hello John!
13:48
Request to transfer €15,00
Drinks
14:06
Incoming transaction of €15,00
Transfer of request: Drinks
14:07
311KB
14:08
1600 Amphitheatre Pkwy
CA 94043 USA
Joost
4c69624e61434c504b3a29
d193aaa4bc60f8c81b631…
Phone Number
0612345678
14:36
14:37

(a) Original PeerChat implementation **(b)** Our chat implementation

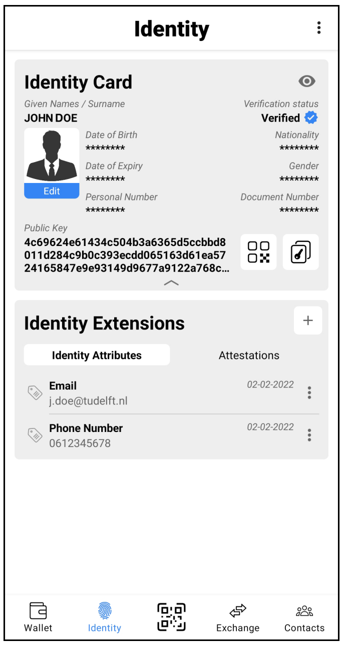
Fig. 6: Difference between old and new implementation

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 [47], 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 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 for the attachments can provide every piece of information about the attachment by clicking the attachment. Users have additional access to various convenient chat-related features like searching and filtering, muting, archiving, or blocking, that attempt to improve usability. 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 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 their receipt. That enables the community to periodically check if the peer is connected and retransmit the unacknowledged messages. In a P2P network (without intermediary nodes) it regularly happens that messages can't be delivered as there is no central component storing the message temporarily until delivered. As a consequence, it may happen that two peers are unable to communicate when they are never connected at the same time.

[4]https://github.com/Tribler/trustchain-superapp
[5]https://github.com/Tribler/kotlin-ipv8

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 introduces tons of possibilities and conveniences. 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 (online) network. 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. Future functionalities, however, 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 require to have knowledge of the public key of the recipient before the data can be encrypted.

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 many other communities, the identity community does not include support for 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. The identity is onboarded as explained in Section V-A and stored in the database. Any device with NFC support is required to verify its identity document, as it reduces misuse and provides authentic trust to other users. 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, *identity attributes* and *identity attestations* have been integrated as well. Identity attributes are convenient


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Date of Birth: ********
Nationality: ********
Date of Expiry: ********
Gender: ********
Personal Number: ********
Document Number: ********
Public Key: 4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393ecdd065163d61ea5724165847e9e93149d9677a9122a768c...

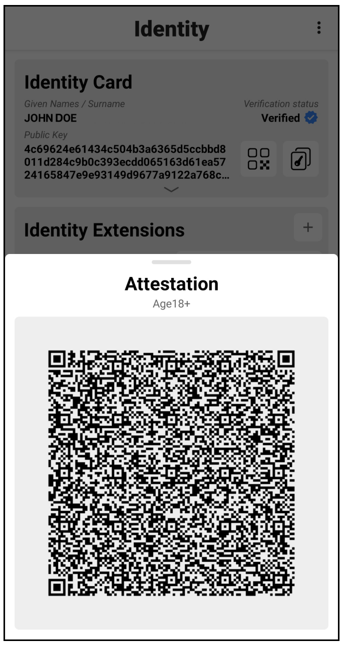
Identity Extensions

Identity Attributes | Attestations

Email 02-02-2022
j.doe@tudelft.nl

Phone Number 02-02-2022
0612345678

(a) Identity implementation


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393ecdd065163d61ea5724165847e9e93149d9677a9122a768c...

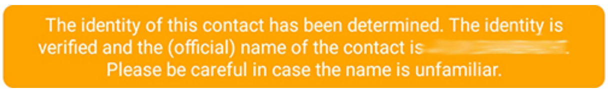
Identity Extensions

Attestation
Age18+

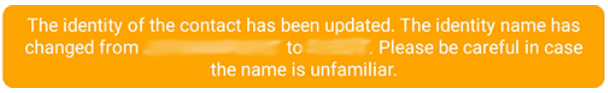
(b) Attestation integration

Fig. 7: Implementation of self-sovereign identity

pieces of information that serve as an unofficial extension to the identity. These attributes are for example a phone number, email address, or home address. These attributes are shareable with other peers and solely serve to extend the use of the SSI. The identity attestations are incorporated in unmodified form using the attestation community, as a result of the work of Chotkan [29], and a QR-code of an example 18+ attestation can be seen in Figure 7b. Currently, only the age-related attestation types obtain the value of the age directly from the SSI. Future attestation types could embed more confidential information from the SSI as it may 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.

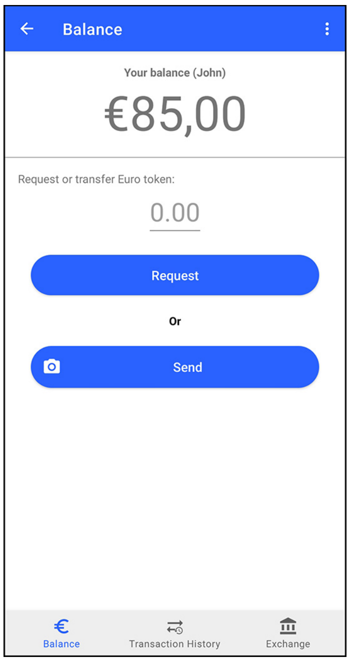
(b) updated state

Fig. 8: Detection of change in 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 exposes 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 8. The verification status of the contact is explicitly displayed at several different views with the sole purpose to draw attention and recognition, especially when unverified. A recognizable blue check star, similar to that of popular social media platforms, indicates successful verification while a red cross star indicates an unverified peer. Both verification statuses can be seen in Figure 9b.

An implementation to transfer digital money was already integrated in the `superapp`, see Figure 6a. The eurotoken community handles the transfer of the academic CBDC 'Eurotoken' [43] with other peers, while the trustchain community handles the storage-related 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 9b. Various options to exchange money have been integrated as in Figure 6b. 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 transfer to another contact, create and send a payment request to a contact 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 the chat with the corresponding contact. Figure 9c shows a detailed view of a transaction displaying all its contents. 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 transmit data to another peer directly, 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. Every packet type serves its own purpose and initiates certain actions. 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 with 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 missing blocks (if any) must be reported. This confirmation consecutively results in the transmission of the next window of blocks until all blocks have been sent and received. To be able to stitch all blocks together, the protocol additionally transmits the block number with the block. The transfer can be considered completed when



Balance

Your balance (John)

€85,00

Request or transfer Euro token:

0.00

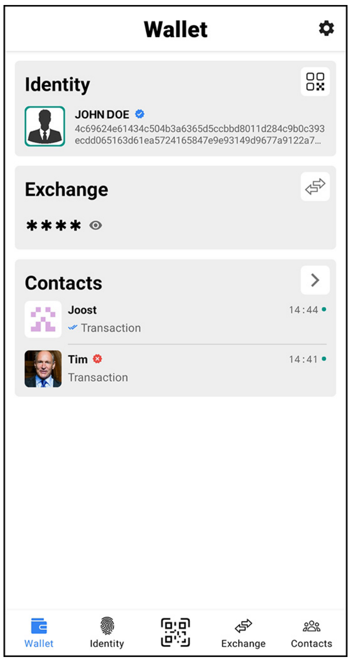
Request

Or

Send

Balance | Transaction History | Exchange

(a) Old exchange implementation



Wallet

Identity

JOHN DOE

4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393
ecdd065163d61ea5724165847e9e93149d9677a9122a7...

Exchange

****

Contacts

Joost 14:44

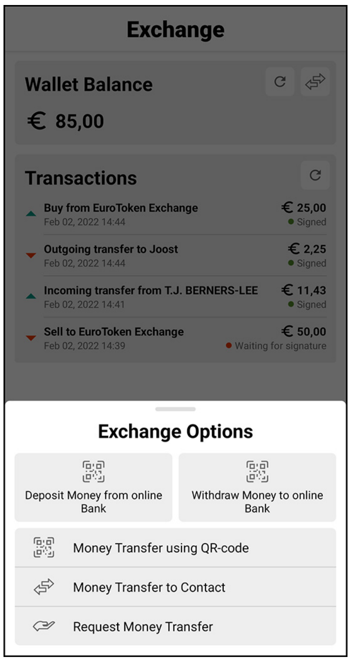
Transaction

Tim 14:41

Transaction

Wallet | Identity | Exchange | Contacts

(b) Our wallet overview (initial view) containing exchange widget



Exchange

Wallet Balance

€ 85,00

Transactions

Buy from EuroToken Exchange Feb 02, 2022 14:44	€ 25,00 Signed
Outgoing transfer to Joost Feb 02, 2022 14:44	€ 2,25 Signed
Incoming transfer from T.J. BERNERS-LEE Feb 02, 2022 14:41	€ 11,43 Signed
Sell to EuroToken Exchange Feb 02, 2022 14:39	€ 50,00 Waiting for signature

Exchange Options

Deposit Money from online Bank

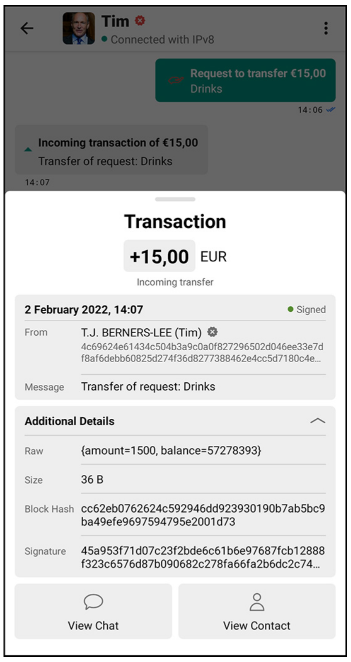
Withdraw Money to online Bank

Money Transfer using QR-code

Money Transfer to Contact

Request Money Transfer

(c) Our exchange implementation and transfer options



Tim

Connected with IPv8

Request to transfer €15,00
Drinks

14:06

Incoming transaction of €15,00
Transfer of request: Drinks

14:07

Transaction

+15,00 EUR

Incoming transfer

2 February 2022, 14:07 Signed

From	T.J. BERNERS-LEE (Tim) 4c69624e61434c504b3a9c0a0f827296502d046ee33e7d f8af6debb60825d274f36d8277388462e4cc5d7180c4e...
Message	Transfer of request: Drinks

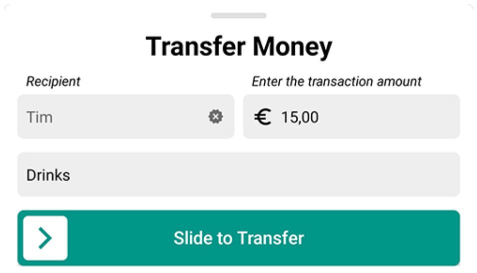
Additional Details

Raw	{amount=1500, balance=57278393}
Size	36 B
Block Hash	cc62eb0762624c592946dd923930190b7ab5bc9 ba49efe9697594795e2001d73
Signature	45a953f71d07c23f2bde6c61b6e97687fcb12888 f323c6576d87b090682c278fa66fa2b6dc2c74...

View Chat | View Contact

(d) Our transaction detail view transaction

Fig. 9: Screenshots of implementation



Transfer Money

Recipient: Tim

Enter the transaction amount: € 15,00

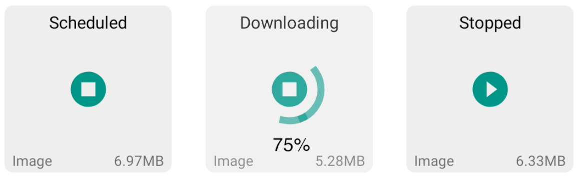
Drinks

Slide to Transfer

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

all blocks have been received.

To realize a quick response to the transmitted packets, the recurrent packets must contain the least possible 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.



(a) scheduled status (b) progress status (c) stopped status

Fig. 11: Download progress indicators in chat

To accommodate communities, and users, with useful 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 are outside the scope of the protocol itself. 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 (temporarily) 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.

VII. Experimental Analysis and Evaluation

In the previous sections, we have discussed the design and implementation of the designed data transfer protocol. In this section, an experimental analysis is performed to derive the optimal protocol settings. Additionally, an evaluation of a large-sized transfer, using the optimal settings, is executed 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 runtime,

TABLE III: Parameters that are being tested for optimal execution. The file size and number of executions have only been used for consistency. In total 80 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]
File size (F)	$5, 10$	[MB]
Iteration	0, 1, 2, 3, 4	[-]

the time between the start and end of the start, is as low as possible, (II) the number of unacknowledged blocks (as explained in Section V-D) is as low as possible, and (III) the number of retransmits of windows of blocks (by the sender) and acknowledgments (by the receiver) is as low as possible. The second constraint (II) does not necessarily contribute to a lower runtime as unacknowledged blocks are embedded in the next window of blocks. The latter constraint (III) should contribute to a lower runtime because no blocks have to be transmitted and there's no additional idle time waiting for an acknowledgment.

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 ($n_{\text{blocks}} \times$ 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*. A smaller number of acknowledgments has to be sent and received, reducing the 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 block retransmits as a result. The importance of this analysis is to find the trade-off between a great window size and low delay due to undelivered blocks.

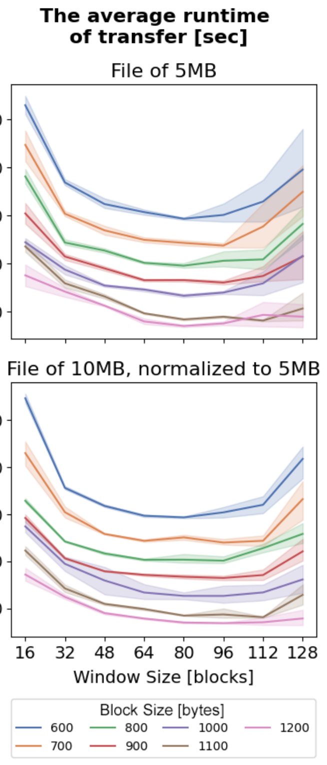
The other parameters do not directly impact the performance, apart from the block and window size. The retransmit interval may affect the performance when it is either too tight or loose, but it will only play a role in a small part of the cases. A tight interval can force blocks or acknowledgments to 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 a file size of 5MB and 10MB (normalized to 5MB for comparison). 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, the file sizes, and the five iterations are executed consecutively. Table III gives an overview of the analyzed parameter values.

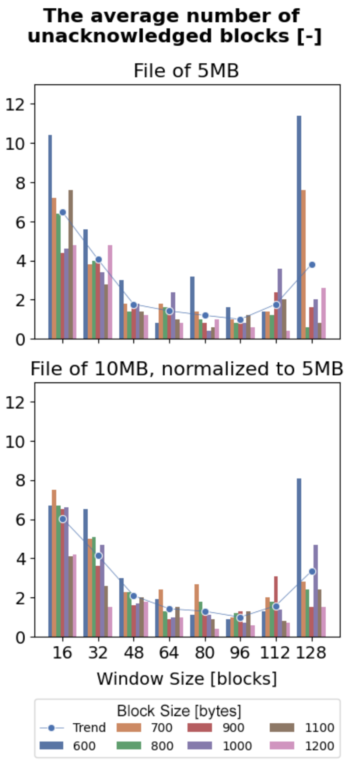
Experimental Results: The optimality of the performance of the protocol can be determined in combination with the before mentioned requirements.
The first requirement, the runtime of the protocol, is displayed in Figure 12a. We can clearly see the effect of the variation of the block and window size. An increasing block size positively affects the runtime. The difference in runtime becomes less significant for greater block sizes, indicating that if we could neglect the maximum packet size, the optimal block size would expectantly be not much greater than $B = 1200$ bytes. The window size follows a parabolic curve and the runtime is optimal for a window size $W = \{64, 80, 96\}$ 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 second requirement, the number of unacknowledged blocks during a transfer, is visualized in the plots of Figure 12b. The block size shows a decreasing pattern and overall contains the lowest number of unacknowledged blocks 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 three window sizes of the first requirement, but slightly favors $W = 96$ blocks. For window sizes greater than $W = 96$ blocks the number of unacknowledged blocks again increases. The protocol is behaving more unreliable as more unacknowledged blocks have to be added to the next window. Every unacknowledged block will cause the runtime to slightly increase as more blocks have to be delivered. The aim remains to reduce these undelivered blocks as much as possible. Especially for more unreliable connections, unacknowledged blocks may have a big impact on the transfer.

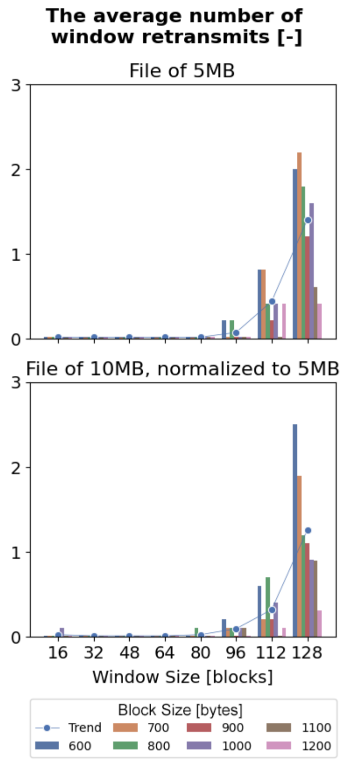
The last requirement, the number of retransmits of windows and the number of retransmits of acknowledgments, are combined as it takes both the sender and receiver in the equation for the same transfers. In Figure 12c and 12d the results for the sender and receiver are visualized, respectively. Both diagrams 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



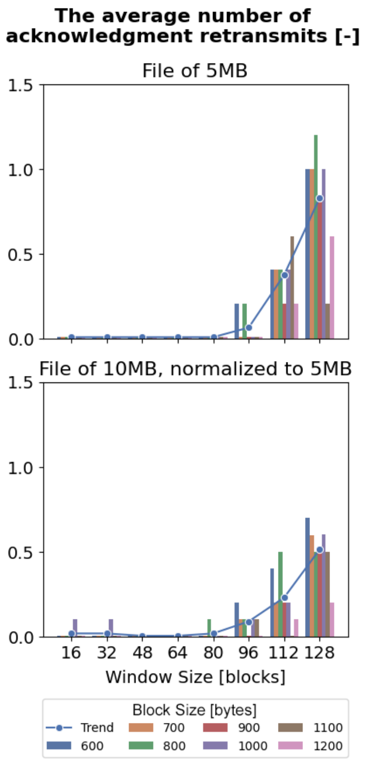
(a)



(b)



(c)



(d)

Fig. 12: The results of all executed tests for each variable window and block size. Each experiment is averaged over five iterations and executed for a file size of 5MB and 10MB, the latter is normalized to 5MB.

them to be too great of size. The number of window and acknowledgment retransmits 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 retransmit, it has a big impact on the overall performance. It is crucial to reduce the number of retransmits to an absolute minimum. For the largest three window sizes, a small experiment was executed that verified if the large increase of retransmits were the result of a too loosely set retransmit interval. The interval was increased majorly, just for verification purposes, and only served for additional analysis of the third requirement, without including the results of the other requirements. The results showed that the number of retransmits 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 if conditions worsen, the runtime and the number of retransmits of windows and acknowledgment would logically increase. We can summarize our findings based on the results of the experiments and analysis. The performance of the runtime (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 runtime found an inconclusive optimal for window sizes of $W = \{64, 80, 96\}$ blocks. The second requirement showed similar results, although slightly in favor of $W = 96$ blocks. Both types of retransmits for the third requirement indicated an increase of retransmits for greater window sizes, specifically above $W = 80$ blocks. To obtain the optimal performance, in combination with a reduction of retransmits and unacknowledged blocks, we can conclude that a window size of $W = 80$ blocks is optimal. The optimal window size based on our findings is $W = 96000$ bytes or 96kB.

We've executed the experiments using two file sizes to verify its independence on the file size. As all diagrams provide an almost equivalent view without any differences in trends, it is fair to say that the file size has no impact 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 retransmits, and therefore not contribute 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 time. In most cases, phones are not connected to the same local WiFi network. We have to consider three commonly used situations: WiFi to WiFi, WiFi to 4G+, and 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 requirements as in the experimental analysis, focused on the performance and delays. 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 obviously expect the first situation to offer the most performance and least delays, as the packets are only exchanged within the local network.

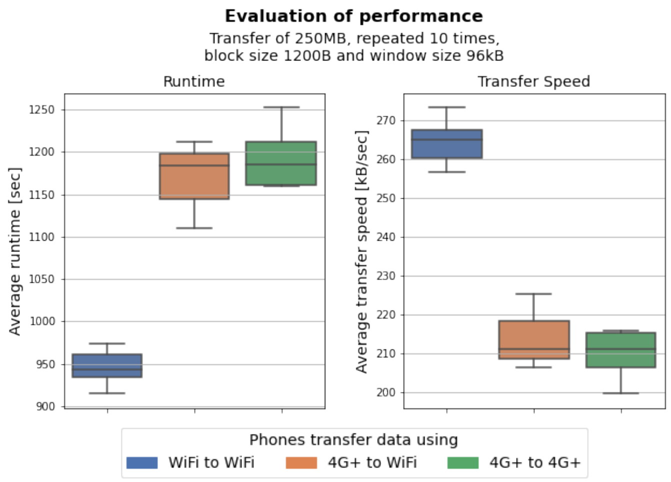


Fig. 13: Evaluation of the performance of a transfer of 250MB

The performance results for each of the connection types are portrayed in Figure 13. The diagrams show a clear division in performance. The runtime, and equivalently the transfer speed, is least and most for the inter-WiFi transfer, with an average transfer speed of about 260kB/s. The other two connection types offer almost equal speeds, of about 213kB/s and 210kB/s, respectively. 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. It's a good indication that there is no extreme performance difference between an exchange that uses one WiFi-connected device and a complete mobile network exchange. We do however notice that the complete mobile network exchange has a bigger spread in terms of lower speeds. If we look at the absolute speed, we must conclude that our protocol is nowhere near the download speeds we experience today. The exact reason for this is unknown, but it is expected to be some limitation in IPv8.

In Figure 14 the evaluation of the delays is portrayed. Unexpectedly, the number of unacknowledged blocks per transfer is on average larger for an inter-WiFi transfer in comparison with the other two situations. In absolute numbers, we can conclude that the total loss of about 50 blocks in a transfer of over 200.000 blocks, equivalent to one unacknowledged block within every fifty windows, is neglectable. Surprisingly, the retransmit diagrams conclude that no window retransmits and almost no acknowledgment retransmits were encountered as a result of unresponsiveness. This proves that our protocol has been performing at the absolute top of its abilities, leaving almost no room for errors.

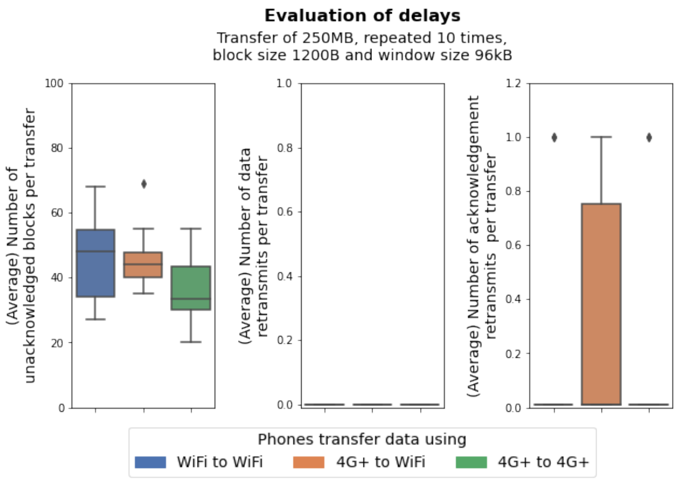


Fig. 14: Evaluation of the delays of a transfer of 250MB

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 convenient 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 the final state. This included familiarizing the style and coding within the `Trustchain superapp`. The design and implementation of features were not only concerned with the features themself, 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 a bit of time earlier in the process, was dedicated to documenting and finalizing this paper.

IX. Conclusion

Ownership and exchange of sensitive or private data and information has never been a more eloquent topic, also due to the COVID crisis. This paper has presented a novel Web3 platform for identity, trust, money, and data. 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 an effective mechanism in the provision of trust to other peers in the P2P network. Governments may change their mind in the near future about their identity management systems. Many of these 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 to pursue the integration of Central Bank Digital Currencies globally. Our designed platform likewise 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. Our designed P2P data transfer protocol enables peers to exchange messages and data securely and privately while reducing the leakage of metadata to an absolute 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, not something people are used to in existing platforms due to the use of central servers or nodes. Secondly, peers in a fully P2P network must keep themselves online by constantly connecting to other peers with the consequence of draining the phone's battery. This is something that probably can be improved, but will always be a weak spot of P2P-based systems. And lastly, the current EuroToken implementation is not reliable enough yet.

Overall, we can 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, but we are curious to see the directions big-tech companies, governments, and banks 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-ideal/, 2021.

[12] Dutch Payments Association. Facts and figures on the dutch payment system in 2020. [Online] Available: https://factsheet.betaalvereniging.nl/en/, 2020.

[13] Christopher Allen. The path to self-sovereign identity. [Online] Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-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/secd9764312f/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 (EuroS P)*, 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://berty.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] 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.

[30] Ann Cavoukian. Privacy by design: The 7 foundational principles. May 2010.

[31] 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.

[32] Wikipedia. Information security. "[Online] Available: https://en.wikipedia.org/wiki/Information_security", 2022.

[33] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. Nov 1976. URL https://ee.stanford.edu/~hellman/publications/24.pdf.

[34] 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-I!!PDF-E&type=items.

[35] Keesing Technologies. Security vulnerabilities associated with e-passports. [Online] Available: https://platform.keesingtechnologies.com/e-passport-security/, Feb 2020.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] R.W. Blokzijl. A central bank digital currency (cbdc) with offline transfers. Master's thesis, Delft University of Technology, 2021.

[44] Network Working Group. The tftp protocol (revision 2). June 1981. URL https://www.rfc-editor.org/rfc/pdfrfc/rfc783.txt.pdf.

[45] J. Postel. User datagram protocol. Aug 1980. URL https://www.rfc-editor.org/rfc/pdfrfc/rfc768.txt.pdf.

[46] Andrew S. Tanenbaum and David Wetherall. *Computer networks, 5th Edition*. 2011.

[47] M. Skála. Technology stack for decentralized mobile services. Master's thesis, Delft University of Technology, 2020.

II
Supplementary Material

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

1 Mobile Application

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 absolutely 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 on 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 on 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 include the use of 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. Options are conveniently accessible by clicking the required icons or buttons.

QR-code controller opens the camera and enables the scanning of QR-codes. Upon a successful scan, the platform will automatically navigate or display the destined view. This enables the user to efficiently execute certain tasks while reducing 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 again initially protected against eavesdroppers. The view also contains buttons for validating the balance, updating the list of transactions, and displaying the 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 the platform navigates to the destined 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 by the recent time at the bottom. Every message or attachment type has a different view and previews some information. If necessary and available, the user can click on the attachment to display the attachment in more detail. Apart from the message typing field, the view contains three other buttons: one for showing the chat/contact options (Figure 1.5e), one for showing the list of attachments (Figure 1.5f), and one for 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. To give the user the option to toggle the use of notifications, a button is included that navigates to the system settings for adjustment. The user is also able to replace or remove its identity.

1.3. Dialogs

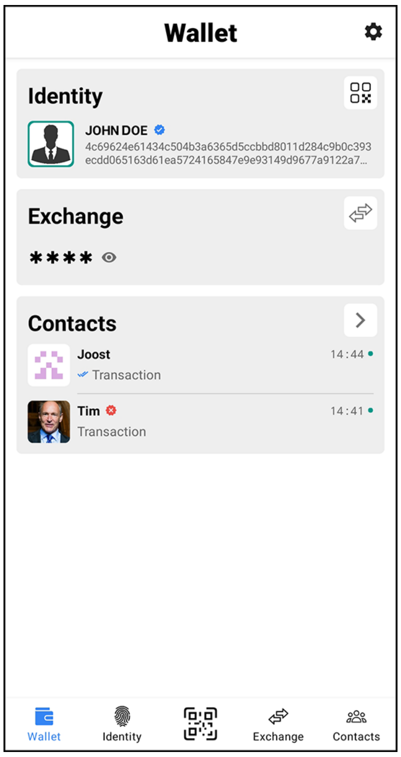
In the application, we extensively use dialogs to display additional information in an efficient manner. In most situations, it is unnecessary to directly display all available information for a particular item. These dialogs are small views that slide in from the bottom and float over the parent view 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 have support for 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 that directs 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 on which type of identity document he possesses. 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 correct, 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 final 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 the 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 on the sender's name, profile picture, and message or transaction contents.


Wallet

Identity

JOHN DOE
4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393
ecdd065163d61ea5724165847e9e93149d9677a9122a7...

Exchange

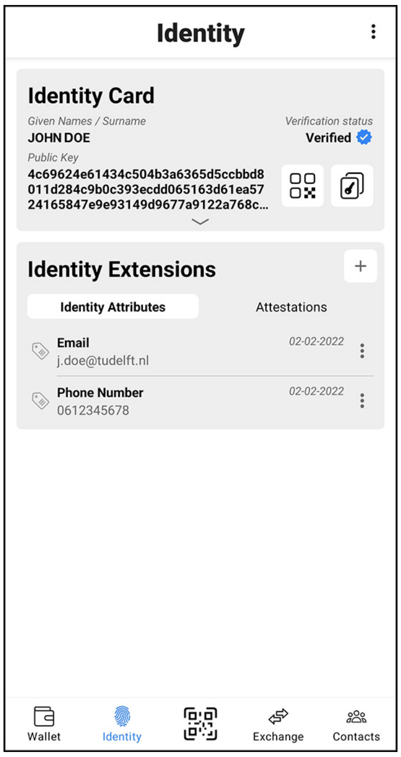
****

Contacts

Joost 14 : 44
Transaction

Tim 14 : 41
Transaction

(a) Wallet Overview


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key
4c69624e61434c504b3a6365d5ccbbd8
011d284c9b0c393ecdd065163d61ea57
24165847e9e93149d9677a9122a768c...

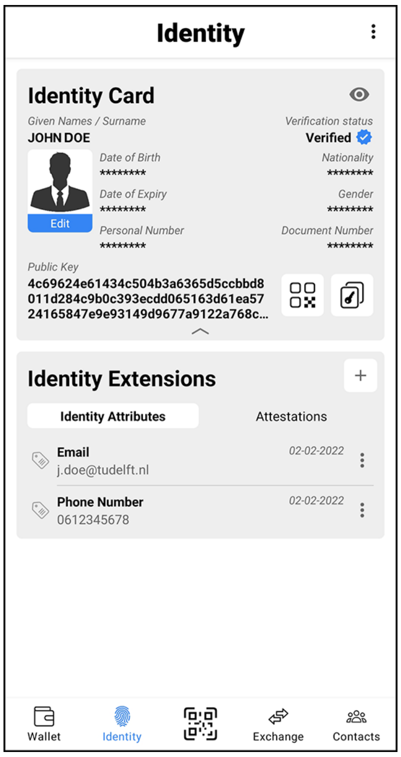
Identity Extensions

Identity Attributes | Attestations

Email 02-02-2022
j.doe@tudelft.nl

Phone Number 02-02-2022
0612345678

(b) Identity View Collapsed


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Date of Birth: ********
Nationality: ********
Date of Expiry: ********
Gender: ********
Personal Number: ********
Document Number: ********
Public Key
4c69624e61434c504b3a6365d5ccbbd8
011d284c9b0c393ecdd065163d61ea57
24165847e9e93149d9677a9122a768c...

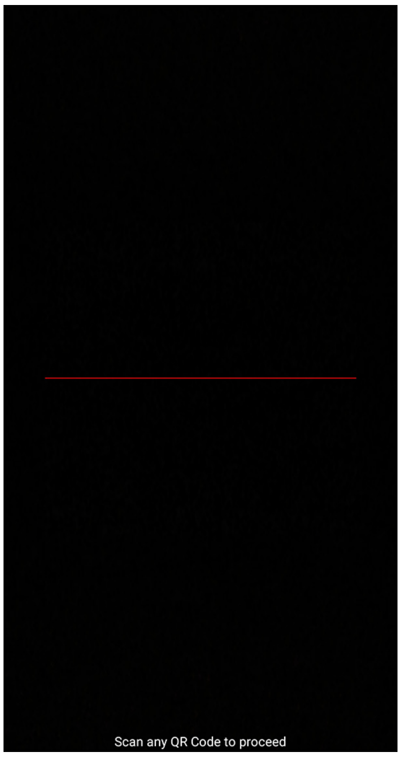
Identity Extensions

Identity Attributes | Attestations

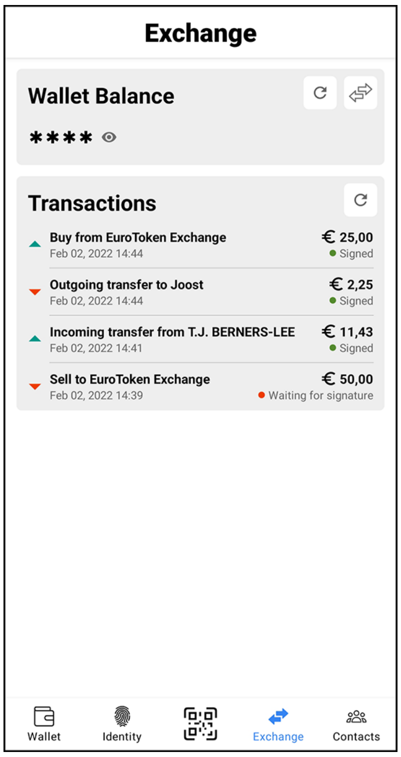
Email 02-02-2022
j.doe@tudelft.nl

Phone Number 02-02-2022
0612345678

(c) Identity View Expanded



(d) QR-code Scanner View


Exchange

Wallet Balance

****

Transactions

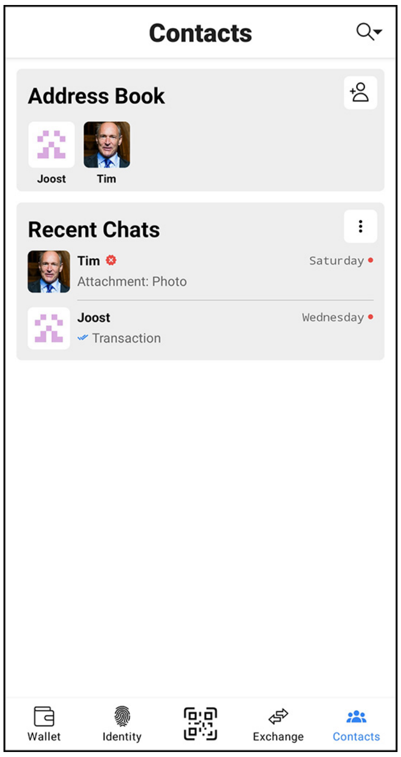
Buy from EuroToken Exchange — Feb 02, 2022 14:44 — € 25,00 — Signed

Outgoing transfer to Joost — Feb 02, 2022 14:44 — € 2,25 — Signed

Incoming transfer from T.J. BERNERS-LEE — Feb 02, 2022 14:41 — € 11,43 — Signed

Sell to EuroToken Exchange — Feb 02, 2022 14:39 — € 50,00 — Waiting for signature

(e) Exchange View


Contacts

Address Book

Joost
Tim

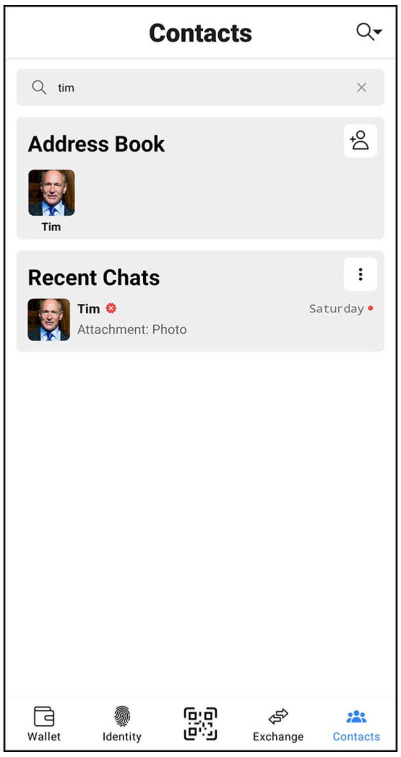
Recent Chats

Tim Saturday
Attachment: Photo

Joost Wednesday
Transaction

(f) Contacts View

Figure 1.1: Main views of the platform (1)


Contacts

tim

Address Book

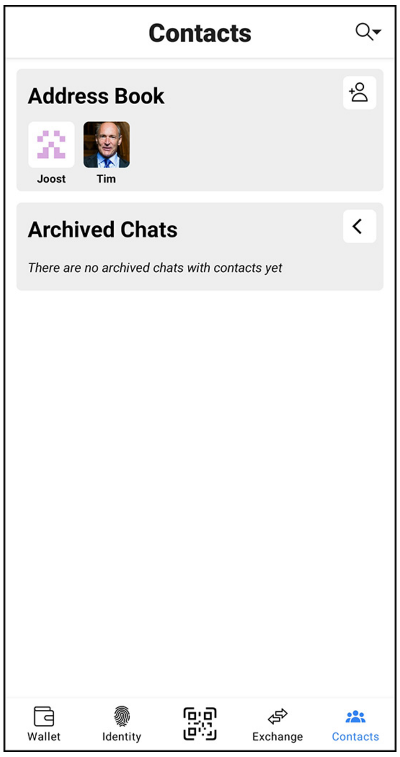
Tim

Recent Chats

Tim Saturday

Attachment: Photo

(a) Contacts View Search


Contacts

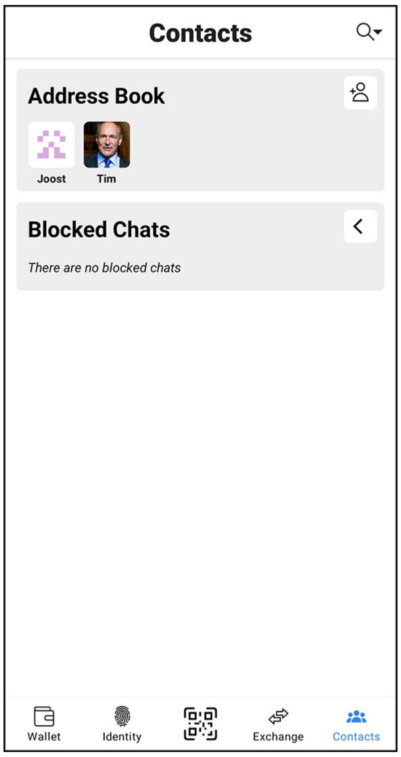
Address Book

Joost Tim

Archived Chats

There are no archived chats with contacts yet

(b) Contacts Archived View


Contacts

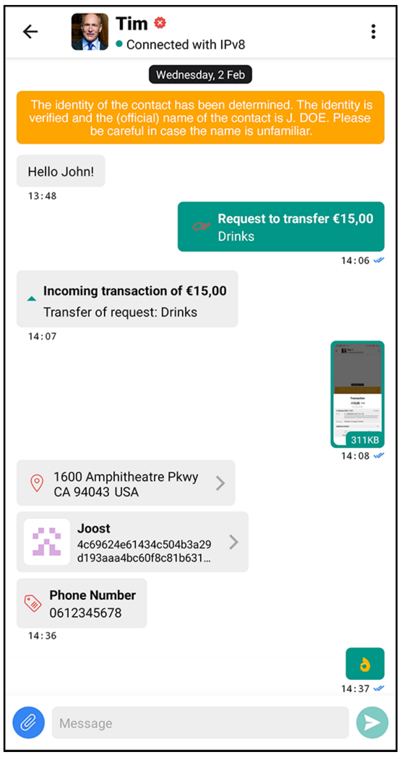
Address Book

Joost Tim

Blocked Chats

There are no blocked chats

(c) Contacts Blocked View


Tim

Connected with IPv8

Wednesday, 2 Feb

The identity of the contact has been determined. The identity is verified and the (official) name of the contact is J. DOE. Please be careful in case the name is unfamiliar.

Hello John!

13:48

Request to transfer €15,00

Drinks

14:06

Incoming transaction of €15,00

Transfer of request: Drinks

14:07

311KB

14:08

1600 Amphitheatre Pkwy

CA 94043 USA

Joost

4c69624e61434c504b3a29

d193aaa4bc60f8c81b631...

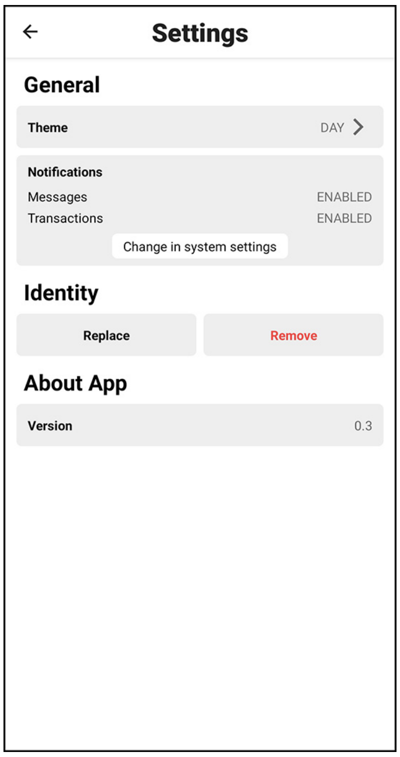
Phone Number

0612345678

14:36

14:37

(d) Chat View


Settings

General

Theme	DAY
Notifications	
Messages	ENABLED
Transactions	ENABLED

Change in system settings

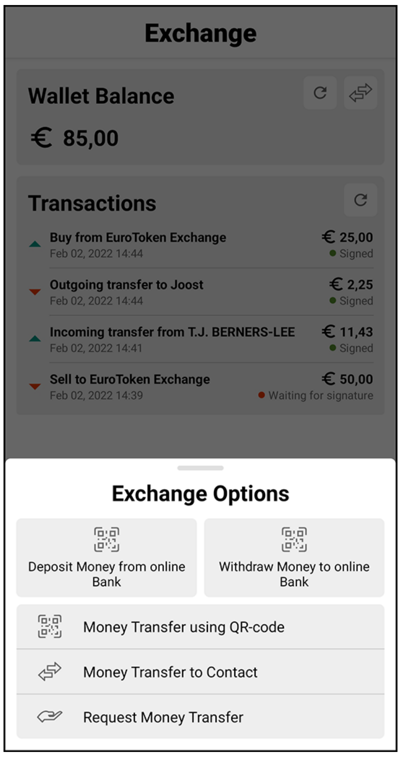
Identity

About App

Version	0.3

(e) Settings View

Figure 1.2: Main views of the platform (2)


Exchange

Wallet Balance

€ 85,00

Transactions

Buy from EuroToken Exchange Feb 02, 2022 14:44	€ 25,00 Signed
Outgoing transfer to Joost Feb 02, 2022 14:44	€ 2,25 Signed
Incoming transfer from T.J. BERNERS-LEE Feb 02, 2022 14:41	€ 11,43 Signed
Sell to EuroToken Exchange Feb 02, 2022 14:39	€ 50,00 Waiting for signature

Exchange Options

Deposit Money from online Bank

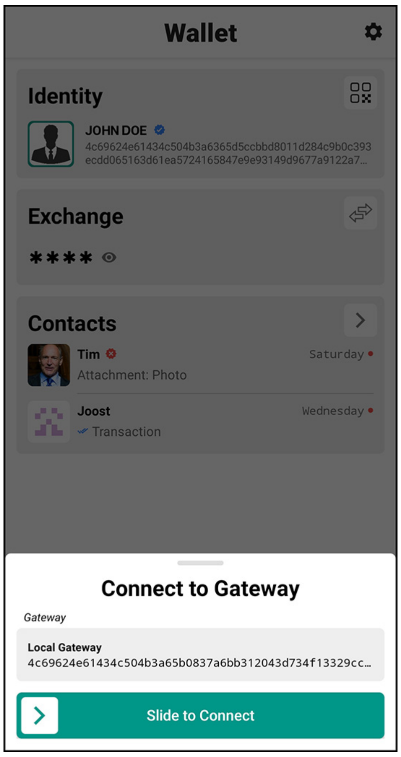
Withdraw Money to online Bank

Money Transfer using QR-code

Money Transfer to Contact

Request Money Transfer

(a) Dialog Exchange Options


Wallet

Identity

JOHN DOE
4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393
ecdd065163d61ea5724165847e9e93149d9677a9122a7…

Exchange

****

Contacts

Tim — Saturday
Attachment: Photo

Joost — Wednesday
Transaction

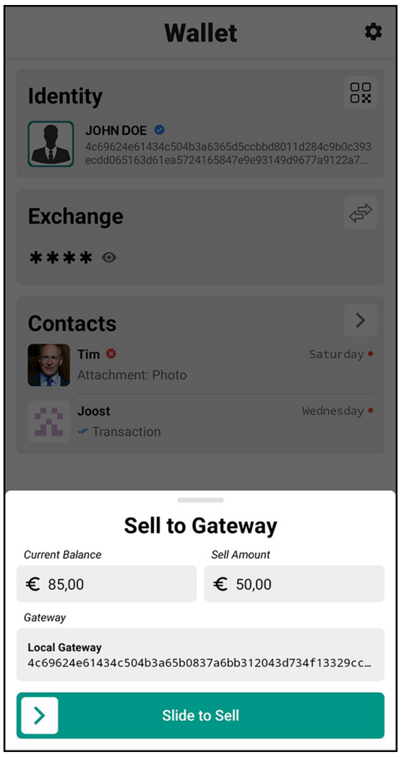
Connect to Gateway

Gateway

Local Gateway
4c69624e61434c504b3a65b0837a6bb312043d734f13329cc…

Slide to Connect

(b) Dialog Exchange Portal Buy


Wallet

Identity

JOHN DOE
4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393
ecdd065163d61ea5724165847e9e93149d9677a9122a7…

Exchange

****

Contacts

Tim — Saturday
Attachment: Photo

Joost — Wednesday
Transaction

Sell to Gateway

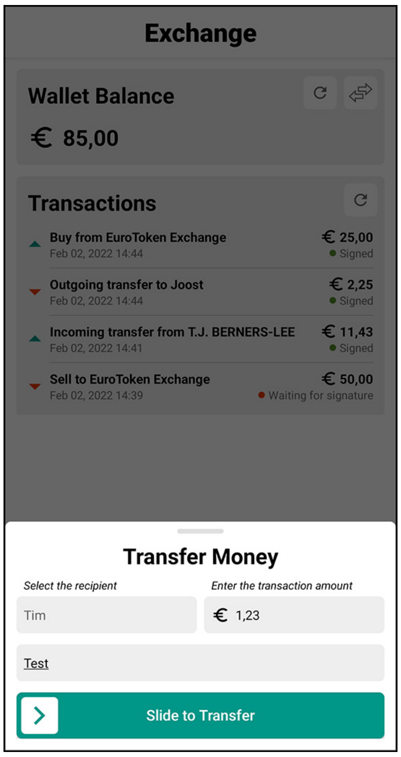
Current Balance	Sell Amount
€ 85,00	€ 50,00

Gateway

Local Gateway
4c69624e61434c504b3a65b0837a6bb312043d734f13329cc…

Slide to Sell

(c) Dialog Exchange Portal Sell


Exchange

Wallet Balance

€ 85,00

Transactions

Buy from EuroToken Exchange Feb 02, 2022 14:44	€ 25,00 Signed
Outgoing transfer to Joost Feb 02, 2022 14:44	€ 2,25 Signed
Incoming transfer from T.J. BERNERS-LEE Feb 02, 2022 14:41	€ 11,43 Signed
Sell to EuroToken Exchange Feb 02, 2022 14:39	€ 50,00 Waiting for signature

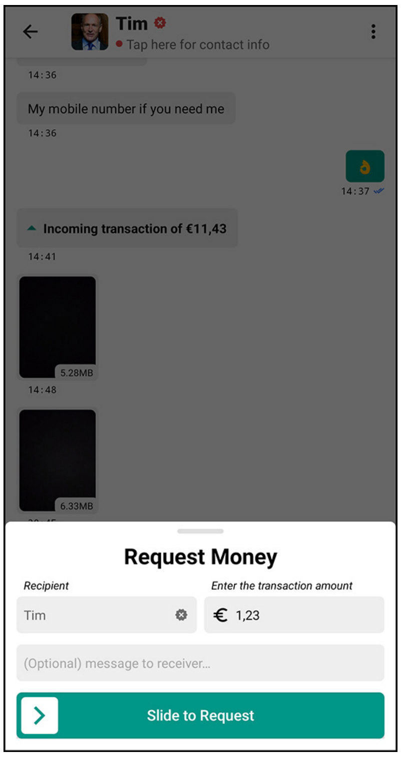
Transfer Money

Select the recipient	Enter the transaction amount
Tim	€ 1,23

Test

Slide to Transfer

(d) Dialog Exchange Transfer


Tim
Tap here for contact info

14:36

My mobile number if you need me

14:36

14:37

Incoming transaction of €11,43

14:41

5.28MB

14:48

6.33MB

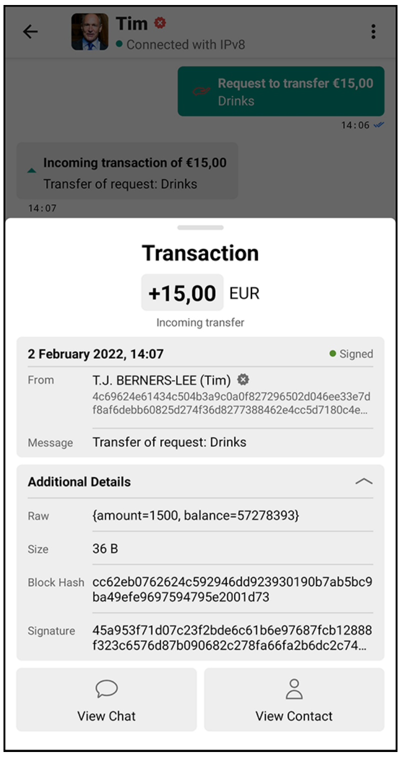
Request Money

Recipient	Enter the transaction amount
Tim	€ 1,23

(Optional) message to receiver...

Slide to Request

(e) Dialog Exchange Request


Tim
Connected with IPv8

Request to transfer €15,00
Drinks

14:06

Incoming transaction of €15,00
Transfer of request: Drinks

14:07

Transaction

+15,00 EUR

Incoming transfer

2 February 2022, 14:07 — Signed

From	T.J. BERNERS-LEE (Tim) 4c69624e61434c504b3a9c0a0f827296502d046ee33e7d f8af6debb60825d274f36d8277388462e4cc5d7180c4e…
Message	Transfer of request: Drinks

Additional Details

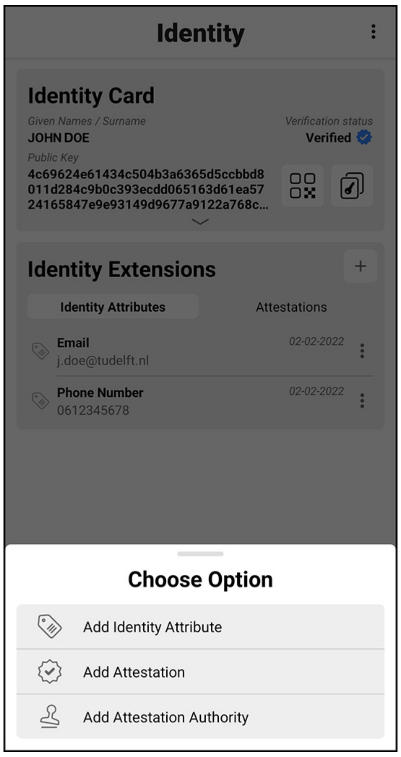
Raw	{amount=1500, balance=57278393}
Size	36 B
Block Hash	cc62eb0762624c592946dd923930190b7ab5bc9 ba49efe9697594795e2001d73
Signature	45a953f71d07c23f2bde6c61b6e97687fcb12888 f323c6576d87b090682c278fa66fa2b6dc2c74…

View Chat

View Contact

(f) Dialog Exchange Transaction

Figure 1.3: Dialogs used in platform (1)


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8 011d284c9b0c393ecdd065163d61ea57 24165847e9e93149d9677a9122a768c...

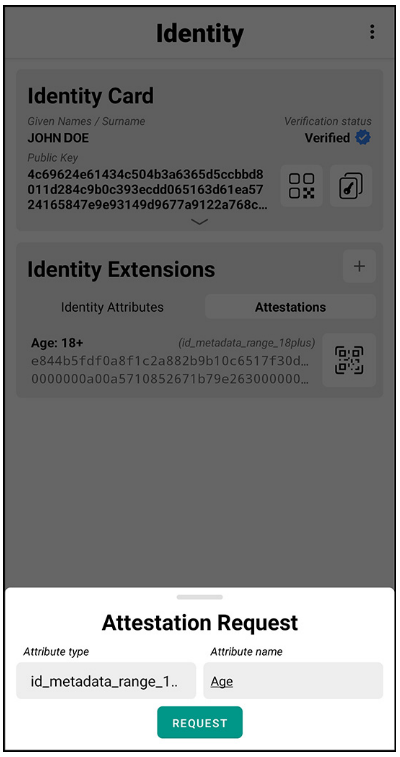
Identity Extensions

Email 02-02-2022
j.doe@tudelft.nl

Phone Number 02-02-2022
0612345678

Choose Option

(a) Dialog Identity Options


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8 011d284c9b0c393ecdd065163d61ea57 24165847e9e93149d9677a9122a768c...

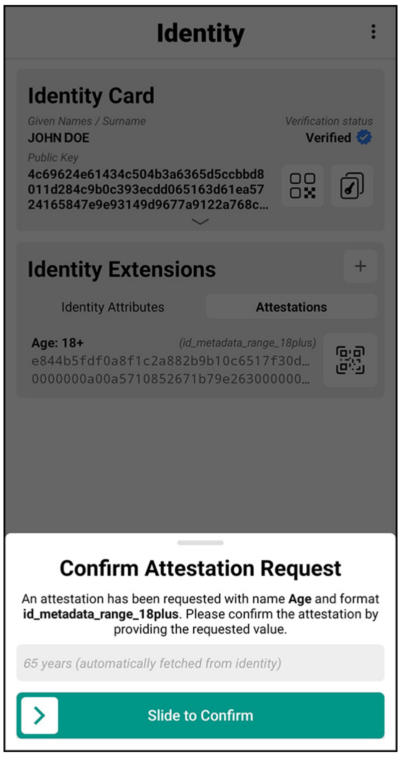
Identity Extensions

Age: 18+ (id_metadata_range_18plus)
e844b5fdf0a8f1c2a882b9b10c6517f30d...
0000000a00a5710852671b79e263000000...

Attestation Request

Attribute type: id_metadata_range_1..
Attribute name: Age

(b) Dialog Attestation Request


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8 011d284c9b0c393ecdd065163d61ea57 24165847e9e93149d9677a9122a768c...

Identity Extensions

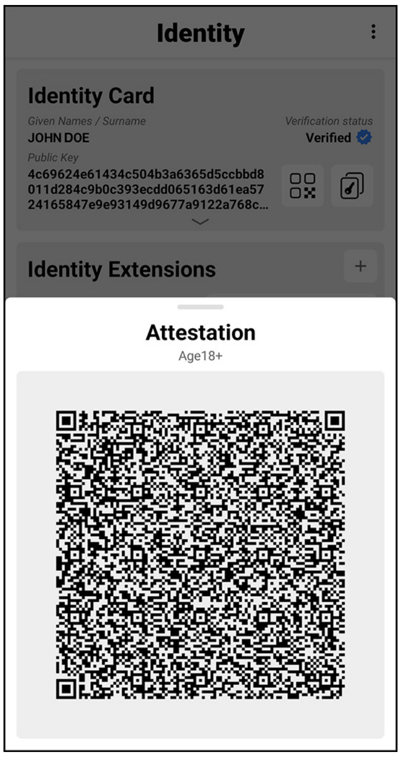
Age: 18+ (id_metadata_range_18plus)
e844b5fdf0a8f1c2a882b9b10c6517f30d...
0000000a00a5710852671b79e263000000...

Confirm Attestation Request

An attestation has been requested with name **Age** and format **id_metadata_range_18plus**. Please confirm the attestation by providing the requested value.

65 years (automatically fetched from identity)

(c) Dialog Attestation Confirm Request


Identity

Identity Card

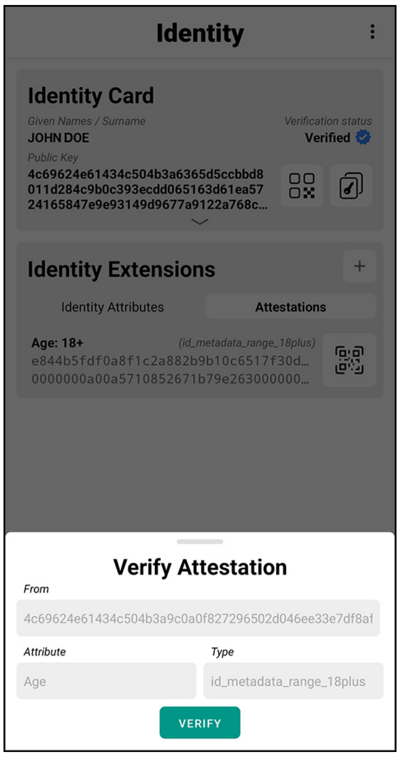
Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8 011d284c9b0c393ecdd065163d61ea57 24165847e9e93149d9677a9122a768c...

Identity Extensions

Attestation

Age18+

(d) Dialog Attestation QR-code


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8 011d284c9b0c393ecdd065163d61ea57 24165847e9e93149d9677a9122a768c...

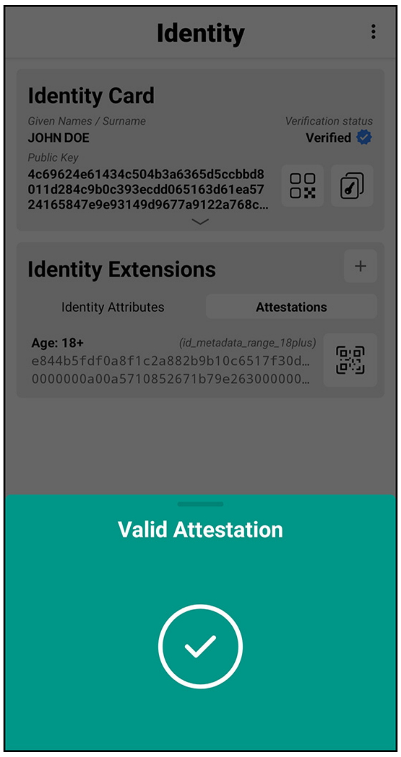
Identity Extensions

Age: 18+ (id_metadata_range_18plus)
e844b5fdf0a8f1c2a882b9b10c6517f30d...
0000000a00a5710852671b79e263000000...

Verify Attestation

From: 4c69624e61434c504b3a9c0a0f827296502d046ee33e7df8af
Attribute: Age
Type: id_metadata_range_18plus

(e) Dialog Attestation Verify


Identity

Identity Card

Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8 011d284c9b0c393ecdd065163d61ea57 24165847e9e93149d9677a9122a768c...

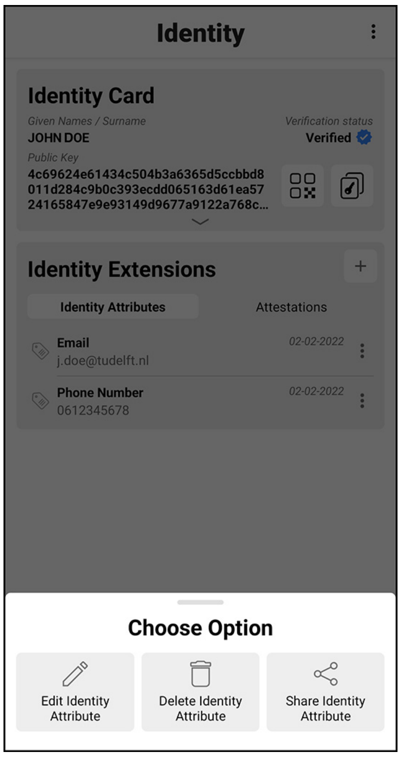
Identity Extensions

Age: 18+ (id_metadata_range_18plus)
e844b5fdf0a8f1c2a882b9b10c6517f30d...
0000000a00a5710852671b79e263000000...

Valid Attestation

(f) Dialog Attestation Verify Succeed

Figure 1.4: Dialogs used in platform (2)

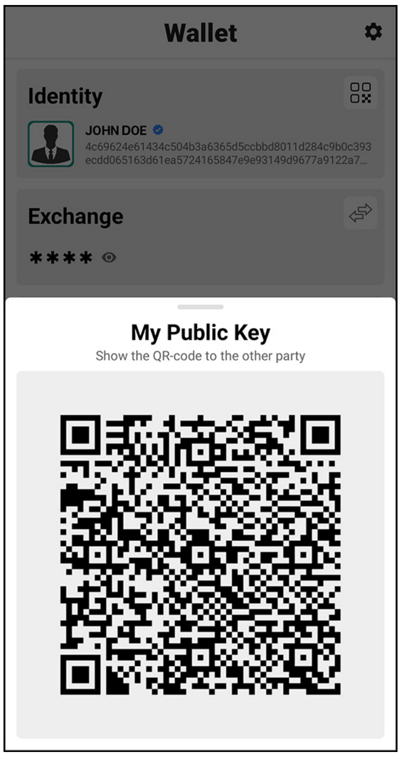


Given Names / Surname: JOHN DOE
Verification status: Verified
Public Key: 4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393ecdd065163d61ea5724165847e9e93149d9677a9122a768c...

Email 02-02-2022
j.doe@tudelft.nl

Phone Number 02-02-2022
0612345678

(a) Dialog Identity Attribute Options

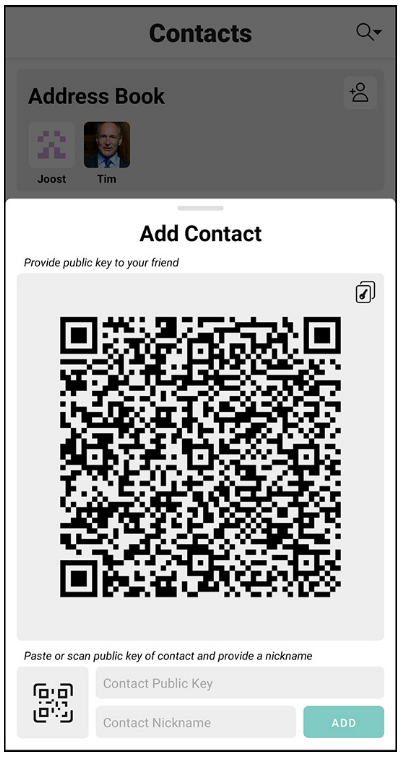


JOHN DOE
4c69624e61434c504b3a6365d5ccbbd8011d284c9b0c393ecdd065163d61ea5724165847e9e93149d9677a9122a7...

****

Show the QR-code to the other party

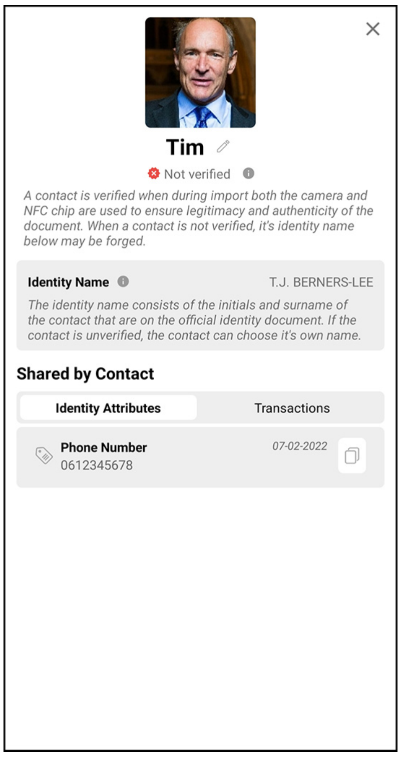
(b) Dialog My Public Key



Provide public key to your friend

Paste or scan public key of contact and provide a nickname

(c) Dialog Add Contact



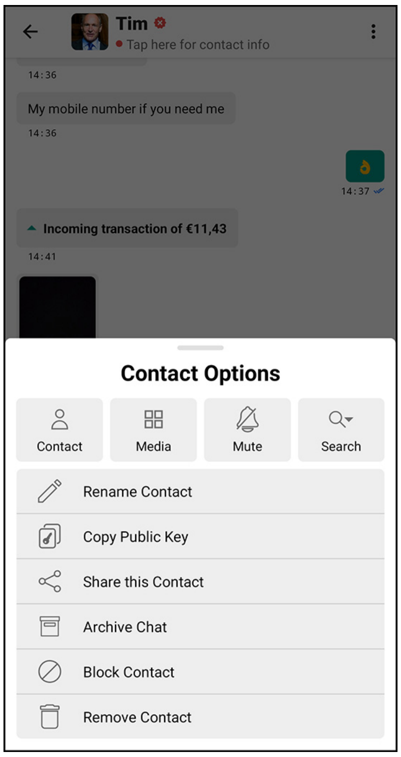
A contact is verified when during import both the camera and NFC chip are used to ensure legitimacy and authenticity of the document. When a contact is not verified, it's identity name below may be forged.

Identity Name T.J. BERNERS-LEE

The identity name consists of the initials and surname of the contact that are on the official identity document. If the contact is unverified, the contact can choose it's own name.

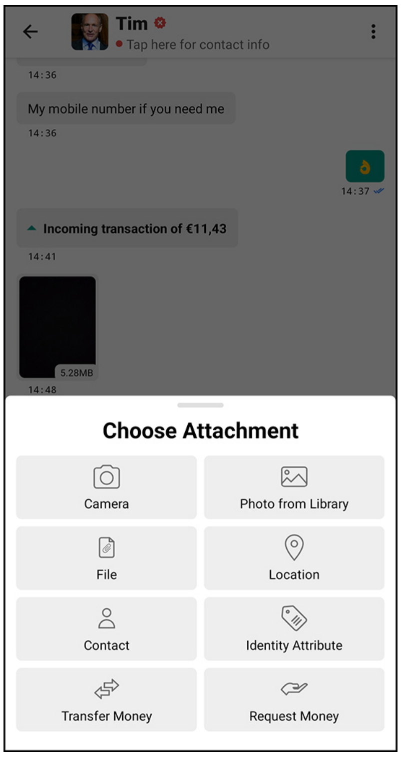
Phone Number 07-02-2022
0612345678

(d) Dialog View Contact



14:36
My mobile number if you need me
14:36
14:37
Incoming transaction of €11,43
14:41

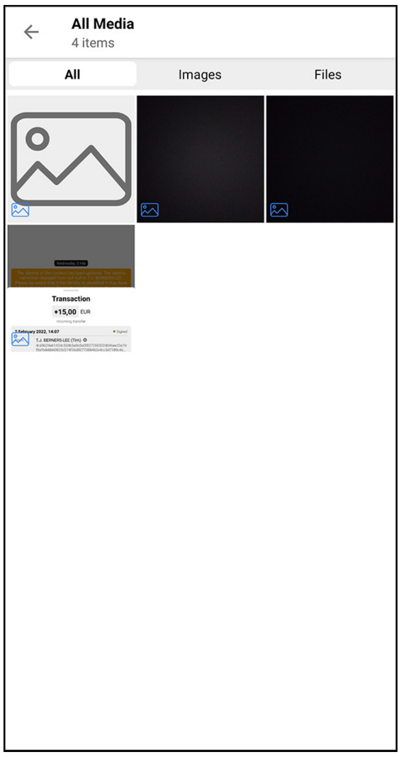
(e) Dialog Chat Options



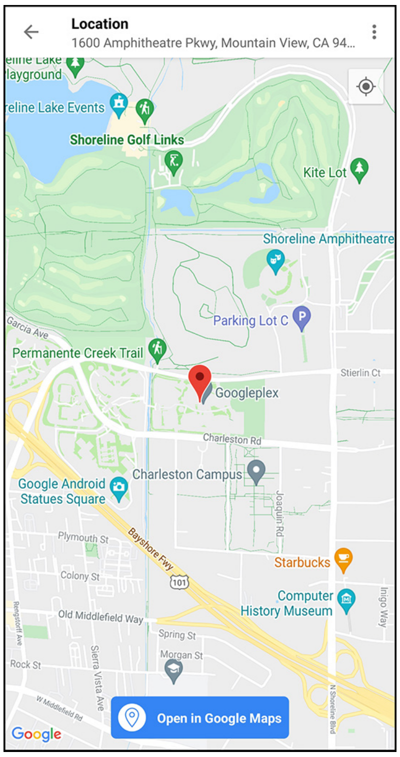
14:36
My mobile number if you need me
14:36
14:37
Incoming transaction of €11,43
14:41
5.28MB
14:48

(f) Dialog Chat Attachments

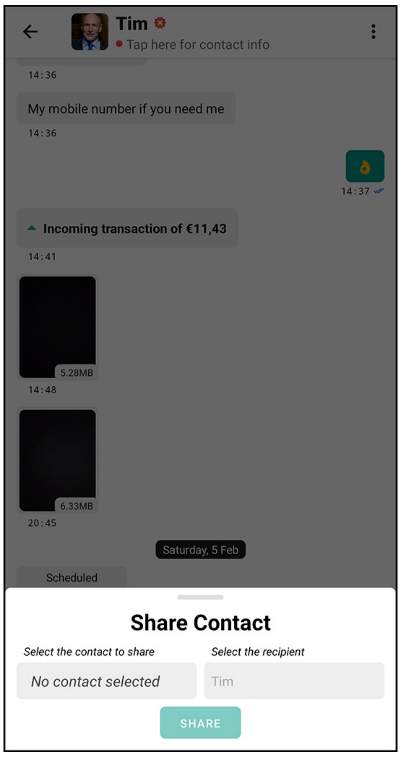
Figure 1.5: Dialogs used in platform (3)



(a) Dialog Chat Media



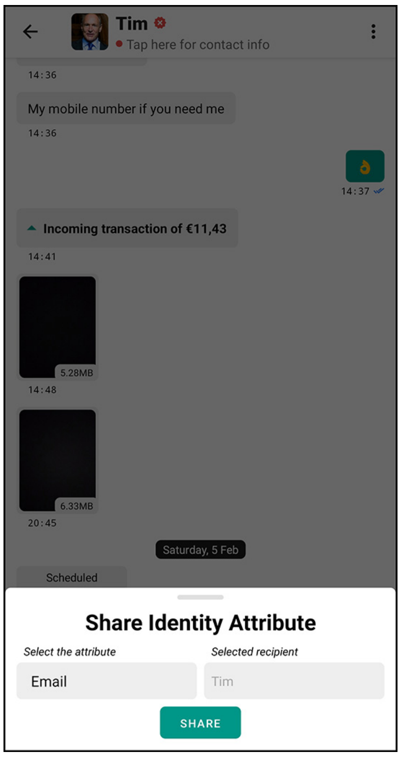
(b) Dialog Chat Location



My mobile number if you need me

Share Contact

(c) Dialog Share Contact

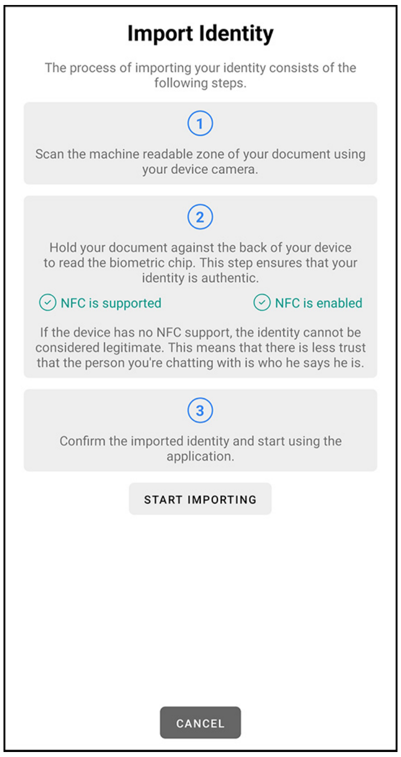


My mobile number if you need me

Share Identity Attribute

(d) Dialog Share Identity Attribute

Figure 1.6: Dialogs used in platform (4)



Import Identity

The process of importing your identity consists of the following steps.

1

Scan the machine readable zone of your document using your device camera.

2

Hold your document against the back of your device to read the biometric chip. This step ensures that your identity is authentic.

✓ NFC is supported ✓ NFC is enabled

If the device has no NFC support, the identity cannot be considered legitimate. This means that there is less trust that the person you're chatting with is who he says he is.

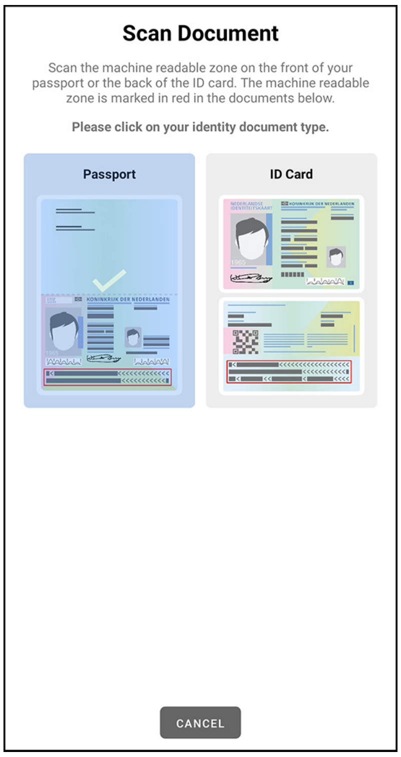
3

Confirm the imported identity and start using the application.

START IMPORTING

CANCEL

(a) Start View

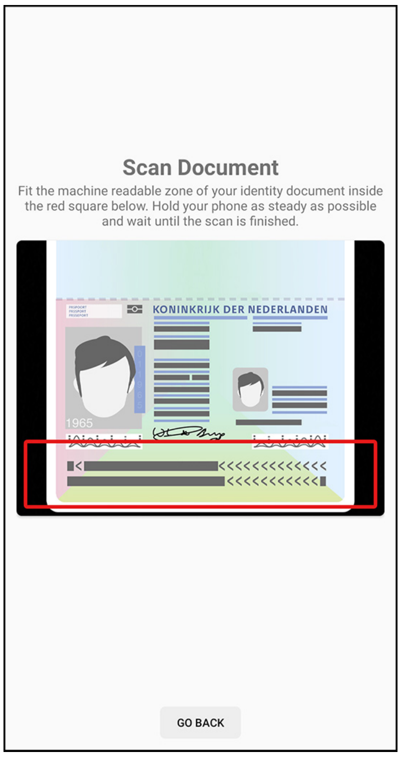


Scan Document

Scan the machine readable zone on the front of your passport or the back of the ID card. The machine readable zone is marked in red in the documents below.

Please click on your identity document type.

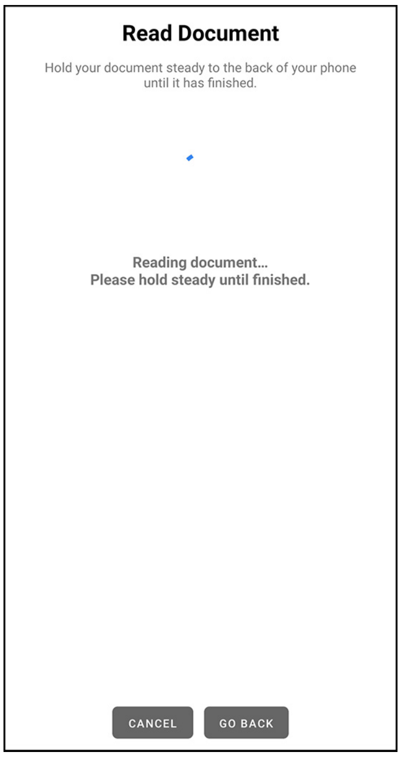
(b) Document Type View



Scan Document

Fit the machine readable zone of your identity document inside the red square below. Hold your phone as steady as possible and wait until the scan is finished.

(c) Document Scan View



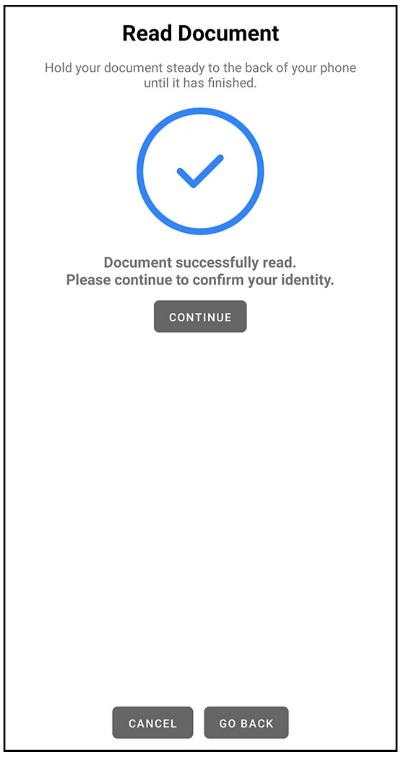
Read Document

Hold your document steady to the back of your phone until it has finished.

Reading document...
Please hold steady until finished.

CANCEL GO BACK

(d) Reading View



Read Document

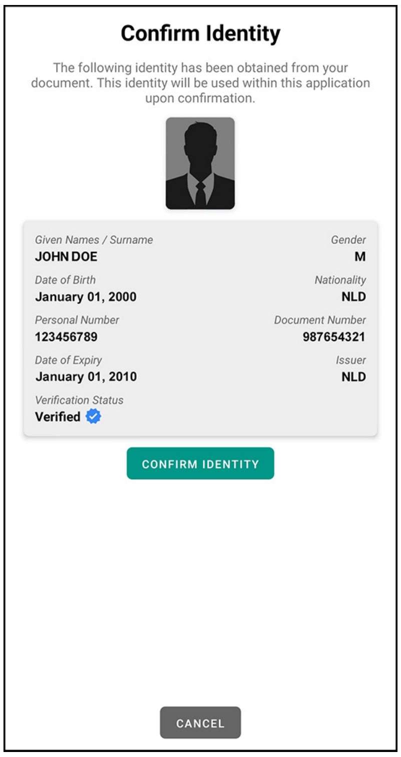
Hold your document steady to the back of your phone until it has finished.

Document successfully read.
Please continue to confirm your identity.

CONTINUE

CANCEL GO BACK

(e) Reading Finished View



Confirm Identity

The following identity has been obtained from your document. This identity will be used within this application upon confirmation.

Given Names / Surname **JOHN DOE**	*Gender* **M**
Date of Birth **January 01, 2000**	*Nationality* **NLD**
Personal Number **123456789**	*Document Number* **987654321**
Date of Expiry **January 01, 2010**	*Issuer* **NLD**
Verification Status **Verified** ✓	

CONFIRM IDENTITY

CANCEL

(f) Identity Confirm View

Figure 1.7: Identity onboarding views from start to end

2 Implementation Details

2.1. Data Transfer Protocol

The data transfer protocol is specifically designed for use in the P2P network IPv8[1]. In this section, we briefly discuss every component separately, as well as its integration in the protocol itself. The 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 community and not the protocol. The protocol can be found in the `kotlin-ipv8` repository[2].

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 the use of 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 retranmission of packets

retransmitInterval (4000) the allowed interval (msec) before retranmission

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

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

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

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 makes use of 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 transfers that have been stopped manually by the peer

TimedOut set contains for each transfer the count of how many times it has timed out

ScheduledTasks set contains the tasks that are scheduled to be executed at a specific time. A priority queue makes sure the tasks are ordered accordingly

2.1.3. Callbacks

Callbacks, pieces of executable code that are executed after the occurrence of specific events, 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

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 specific tasks at specific times. If enabled, the protocol is able to schedule execute transfer timeouts and retransmission of windows and acknowledgments, after the allowed interval has expired. Scheduled tasks are added to the set 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 `scheduledSendIntervalInSec` parameter. Transfers in the `scheduled` set can only be started if they satisfy the following conditions, 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`. Transfers that satisfy these requirements are started.

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, 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 it satisfies the same requirements as in the above `sendScheduled` method. If not, the transfers are scheduled to be started at a later time.

Transmit Data

The protocol determines during the transfer which blocks have to be sent in the next window. This is a combination of the blocks from the next window and possibly unacknowledged 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, there appear to be some missing blocks, it embeds these block numbers in the acknowledgment. The `sendAcknowledgement` method creates an acknowledgment packet and transmits it over the community endpoint. It also schedules a task to terminate the transfer and to resend the acknowledgement if no response returns within the `timeoutIntervalInSec` and `retransmitIntervalInSec`, respectively.

Terminate

A transfer, outgoing or incoming, can be 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 make sure the transfer has a better chance to succeed in the next attempt. This does have a (small) negative impact on the transfer speed as fewer blocks are sent in one window, requiring more windows in general.

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
- `unAckedBlocks`: block numbers that are missing

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 an idle function that waits until a specific event occurs, 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 the transfer is not already incoming, meaning the packet is retransmitted, if the transfer has not already been transferred, and if the transfer is currently stopped. Additionally, the size of the transfer is validated to be within defined boundaries. If all those requirements are met, the transfer is accepted by sending an `Acknowledgement` packet back to the sender of the transfer and placing the transfer in the set of incoming transfers, including its transfer-specific settings announced by the sender. The receiver also schedules the retransmission of the acknowledge and schedules the terminate execution. 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 a `Acknowledgement` packet. The receipt of this packet confirms receipt of the previously sent packet. It is a response on either a `WriteRequest` or `Data` packet. Before knowing what to do, it has to check to which transfer this acknowledgment belongs. Also, if the transfer has been stopped by the receiver, there will be no response. Using the `outgoing` set of the peer and the transfer `nonce` announced 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 announced in the 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 has a progressive nature as it always aims to move to the next window, even if there are unreceived blocks. The only situation in which the protocol is not progressing to the next window is when no response is received from the receiver.

onData

This listener belongs to the recipient of the transfer only and is triggered upon receipt of a `Data` packet. The transfer is located using the `incoming` set, the peer and the `nonce`. The data in the payload is added to the correct location in the total data file, identified by its block number. Even blocks that arrive in a different order are able to append their data to the total data file. If after this block there are no unreceived blocks left, an acknowledgment is sent and the incoming transfer is finished, added to the `transferred` set, and terminated. This includes the callback function that handles the conversion of the file and the removal of the incoming transfer in memory. If there are still unreceived blocks left, and this block appears to be the last block of its window, an acknowledgment is sent. If the block is not the last block of the window, no acknowledgment is sent.

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 is notified to the protocol and the corresponding transfer is terminated.

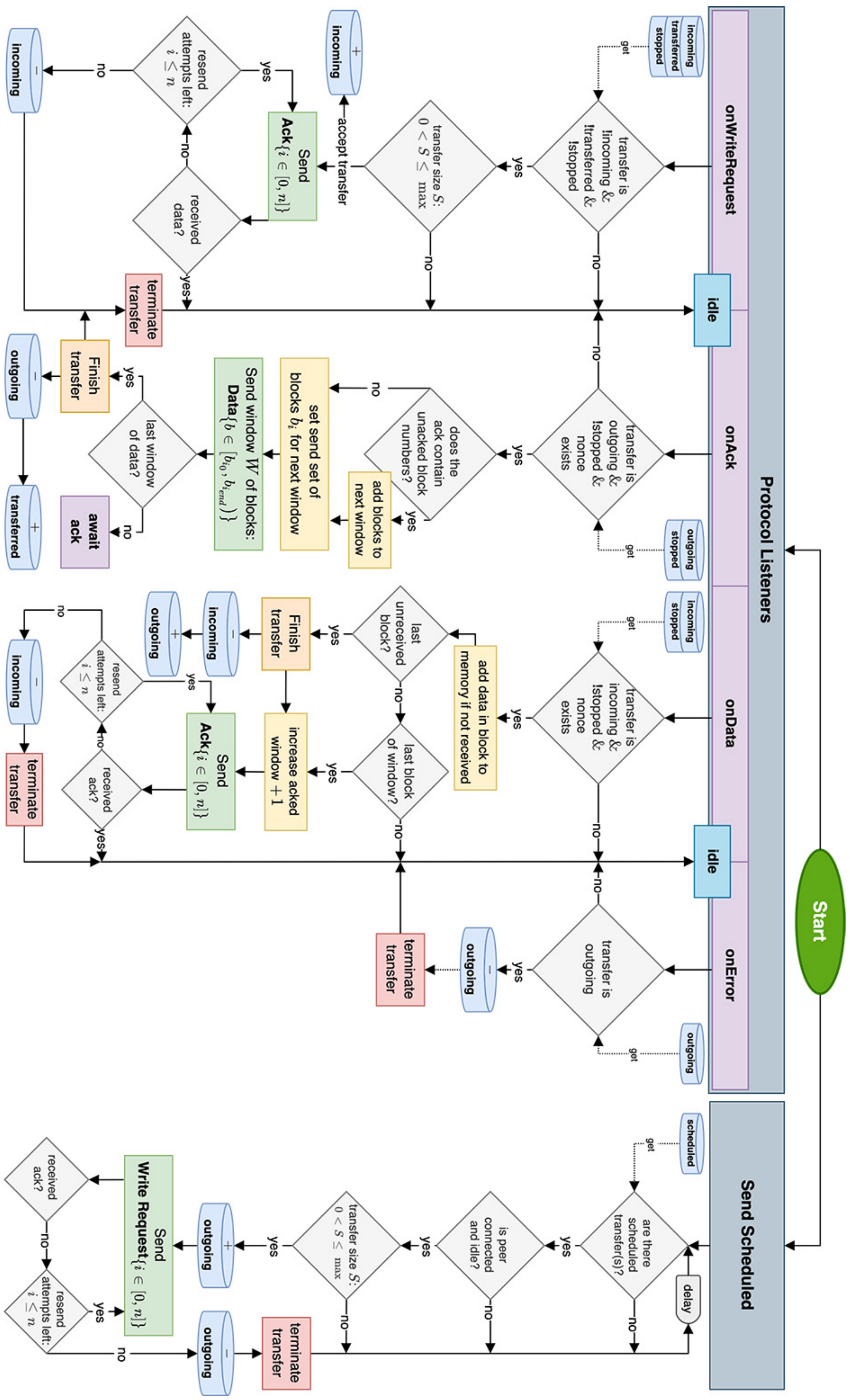


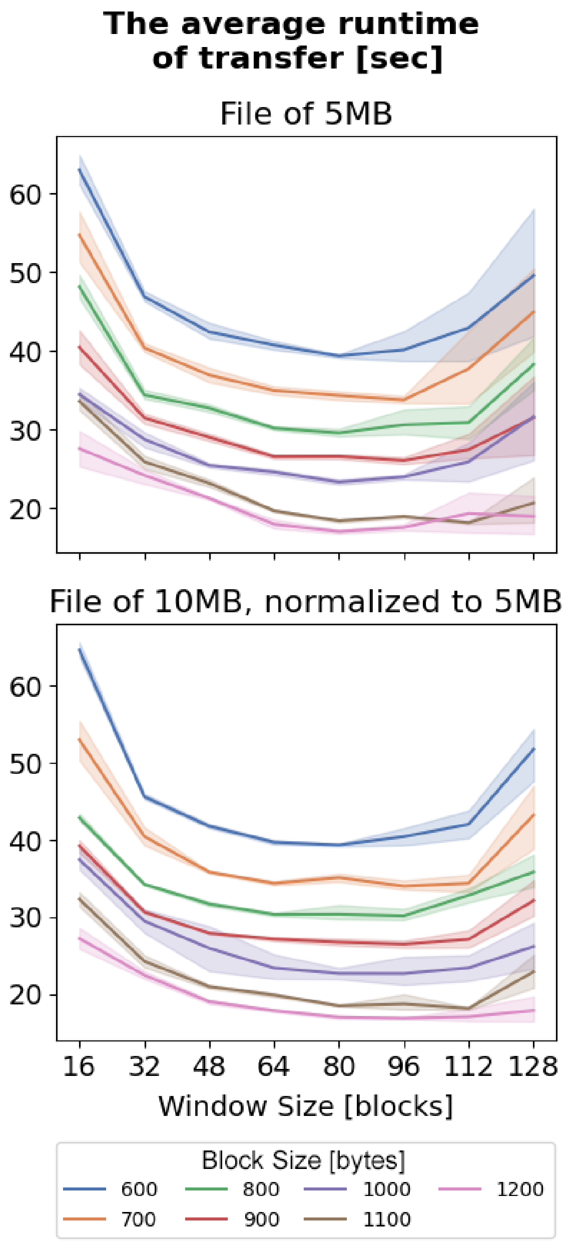
Figure 2.1: Flow diagram of data transfer protocol

2.1.8. Experimental Analysis

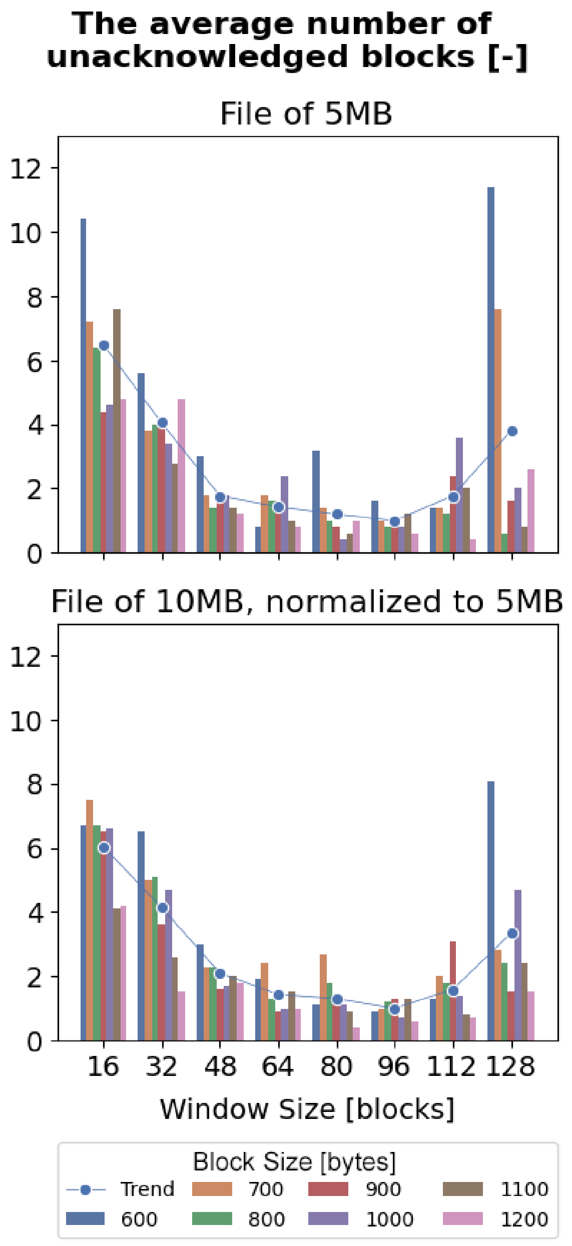
The purpose of the experimental analysis is to find the optimal settings in terms of performance. We define the protocol to be optimal if (I) the runtime, the time between the start and end of the start, is as low as possible, (II) the number of unacknowledged blocks is as low as possible, and (III) the number of retransmits of 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 both file sizes of 5MB and 10MB. Figures 2.2 and 2.3 contain the results of all executed tests. 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.

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 and delays using the same indicators as in the experimental analysis. The exchange of the files in the experimental analysis was executed within the same 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 2.4 contains the results of the evaluation of the performance and delays.

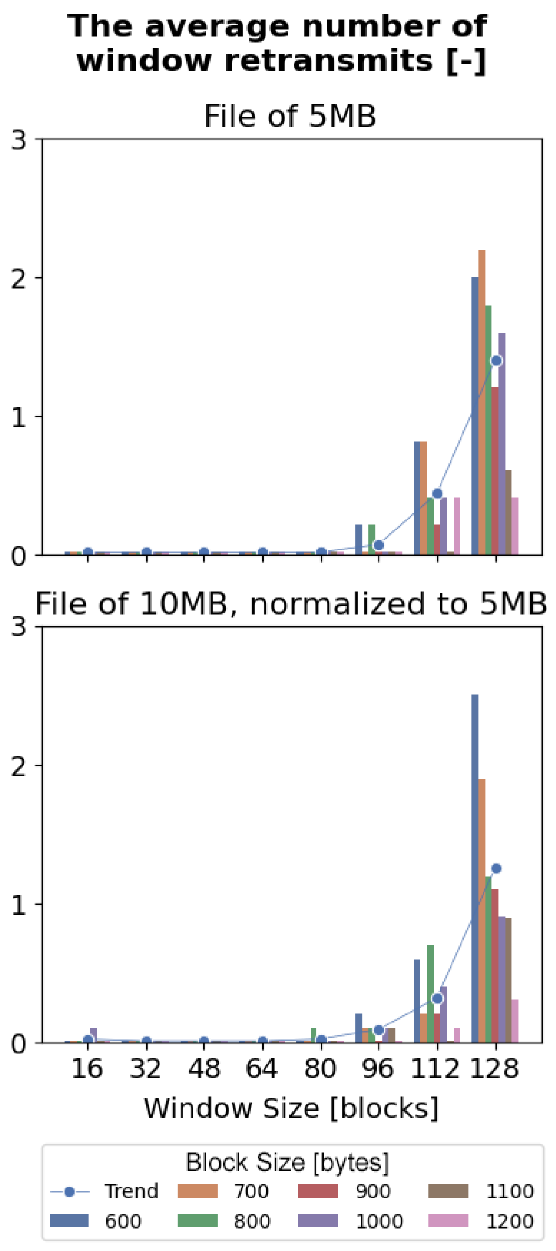


(a) Runtime analysis

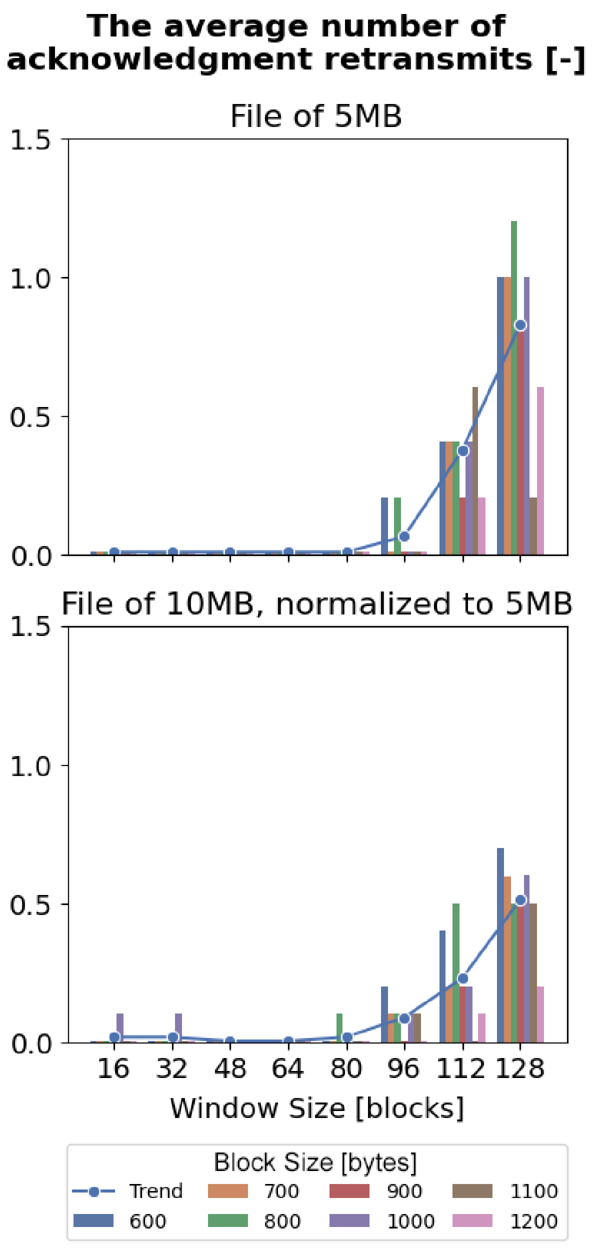


(b) Unacknowledged blocks analysis

Figure 2.2: Experimental analysis (1). The analyzed results for the runtime and unacknowledged blocks of all executed tests, for each variable window and block size. Each experiment is averaged over five iterations and executed for a file size of 5MB and 10MB, the latter is normalized to 5MB.

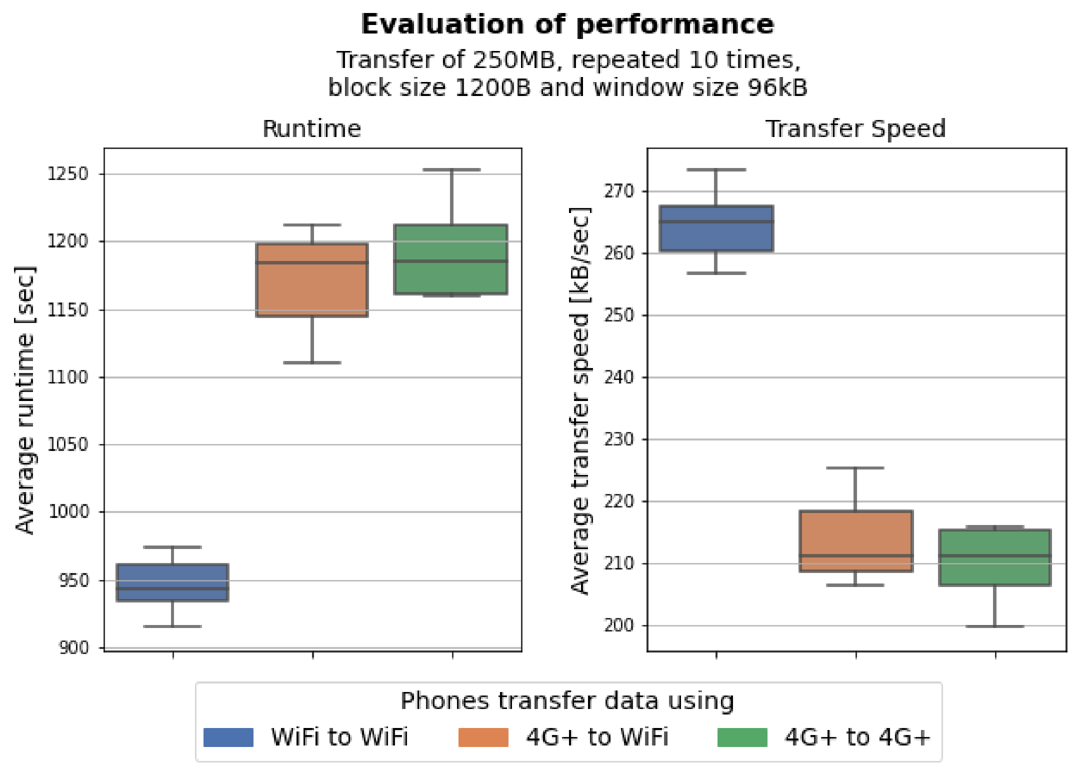


(a) Window retransmit analysis

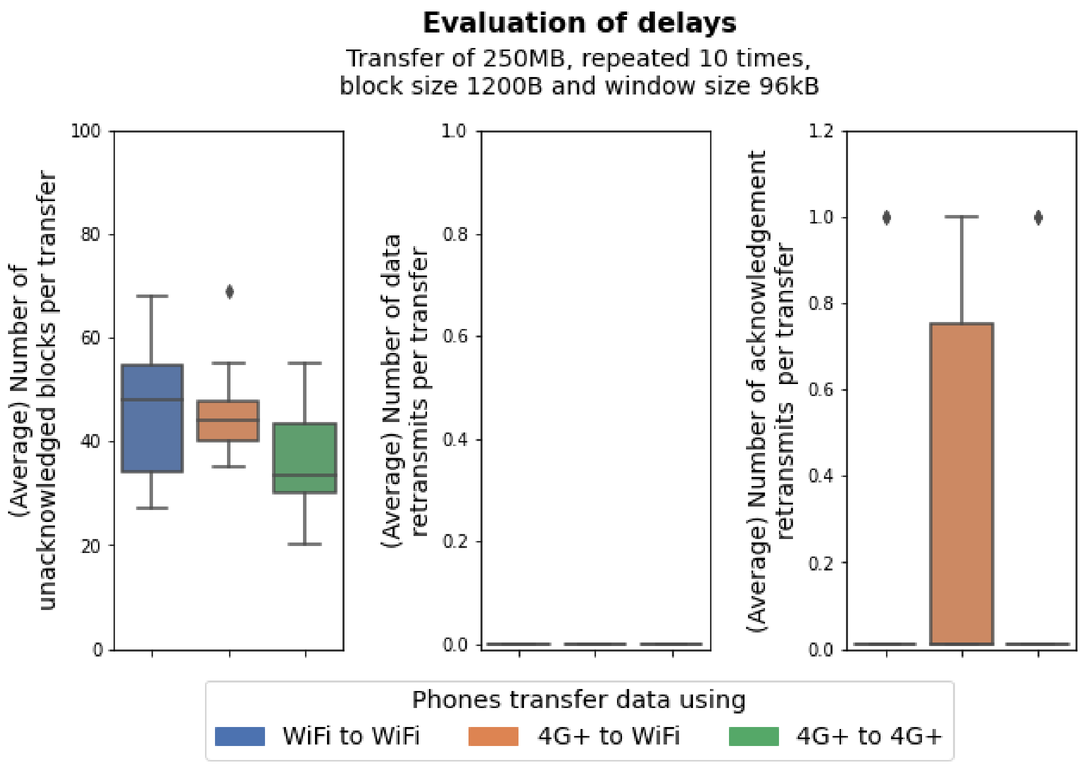


(b) Acknowledgment retransmit analysis

Figure 2.3: Experimental analysis (2). The analyzed results for the window and acknowledgments retransmits of all executed tests, for each variable window and block size. Each experiment is averaged over five iterations and executed for a file size of 5MB and 10MB, the latter is normalized to 5MB.



(a) Evaluation performance



(b) Evaluation delays

Figure 2.4: Evaluation of a transfer of 250MB

Bibliography

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

Tribler. (2021). Ipv8 documentation. https://py-ipv8.readthedocs.io/_/downloads/en/latest/pdf/