# Tawki: Towards Self-Sovereign Social Communication

Martin Westerkamp, Sebastian Göndör and Axel Küpper
Service-centric Networking
TU Berlin | Telekom Innovation Laboratories
Berlin, Germany
Email:{westerkamp | sebastian.goendoer | axel.kuepper}@tu-berlin.de

***Abstract*—As of today, web-based social communication platforms, such as WhatsApp, Twitter, or Facebook, are almost exclusively realized via centralized platforms, based on proprietary interfaces, protocols, and data formats. In consequence, even though social communication being a decentralized, peer-to-peer phenomenon, web-based communication today is implemented via closed, proprietary data silos, which not only lock-in users into their service platforms, but also control and own exchange information and data, including personal and sensitive data such as photos, messages, or contact information.**

**In this paper we present Tawki, a decentralized service architecture for social communication. Using Tawki, users remain in full control of their personal data, which is stored and managed by personal data storages. Each data storage is accessible via a unified Tawki API, which allows users to send and request data to and from other users' personal data storages. Following this approach, social communication is again peer-to-peer without involving a third party controlling and monitoring the process. Tawki uses the Ethereum Name Service (ENS) for both the management of user identities and resolving identifiers to the respective user's personal storage location. Leveraging the immutability of the Ethereum Blockchain, identity management and discovery of personal data storages is secured against censorship and control through any third party.**

***Index Terms*—Blockchain, Distributed Ledger Technology, Distributed Social Networks, Online Social Networking.**

## I. Introduction

Since the early days of the social web, Online Social Network (OSN) services allowed people to connect to each other and communicate and since then gained more and more importance in our everyday lives [1]. As of today, people have started to organize their lives using web-based communication and collaboration platforms, such as Facebook, Twitter, or WhatsApp. These platforms offer a worry-free way of communicating with friends, relatives, or coworkers, as the service manages and organizes all data and information while providing convenient ways to use and access the services everywhere and anytime.

Most social web services require their users to store user profiles on their platforms, including personal and sensitive data such as photos, messages, or contact information. As OSN providers often automatically claim ownership of the managed data in order to use it for targeted advertisement and data analysis [2] [3], users of web services started to feel apprehensive about disclosing personal information [4]. Possession of and access to vast amounts of personal information enables OSN service providers to create highly detailed profiles of their users, which are based on personal information most users wouldn't want to be used by anyone. As operators of OSN service platforms claim ownership for all content uploaded to or created on their servers, users essentially lose control over their data privacy [5]. This prevalent loose handling of personal data by OSN providers resulted in a *"well documented frustration"* of users [6] who wanted to *"jump out of the walled gardens"* [3].

Web-based social communication platforms today are almost exclusively designed in a centralized, proprietary fashion. As a result, users can only communicate and share content with users of the same service, as proprietary communication protocols and APIs restrict interoperability with other services. This way, users are being locked into the service they signed up with in the first place, unable to freely communicate with users of other OSN services [7].

To alleviate the issue of OSN services being proprietary, isolated platforms, peer-to-peer (P2P) and federated social communication architectures have been proposed that distribute control and information between multiple servers or devices [8] [9]. Still, existing decentralized OSN services such as Mastodon[1] or Diaspora[2] again lock their users into another, yet in itself distributed service. As communication with outside services is again mostly not possible [10], adaption of decentralized OSN services is rather low.

We envision a truly open and free service architecture for social communication services that allows users of any kind of software to freely communicate and collaborate with each other. Such an ecosystem of social communication platforms must allow users to remain in full control of their data, including the possibility to migrate their entire user account and all associated data to an individually selectable location. In order to prevent such an ecosystem to be controlled by a central entity or organization, no central part of the architecture can be run or maintained in a centralized fashion. The result is a truly open and heterogeneous ecosystem of social communication platforms that are seamlessly connected via a common open protocol.

[1]Mastodon website: http://mastodon.social
[2]Diaspora website: https://diasporafoundation.org

The contribution of this paper therefore proposes solutions for the following research questions: How can social text-based communication be implemented in a fully decentralized fashion without relying on any central authority? How can users remain in full control of their own data?

In this paper we propose Tawki, a decentralized and distributed service architecture for social communication and collaboration. Users are able to select any storage provider for their user data, including self-hosted solutions and corporate-run managed servers. All personal data storages implement a common API for data exchange, thus supporting direct Peer-to-Peer (P2P) communication between users without involvement of a central server orchestrating the process. In order to discover endpoints of individual users, Tawki leverages the decentralized nature of the Ethereum Blockchain to implement a global mapping system for user accounts. Using the Ethereum Name Service (ENS), users are able to register a username, which can then be resolved to the respective owner's data storage. This way, both the identity and data management are controlled by the users themselves without the need for a central entity or organization.

The remainder of this paper is organized as follows. Chapter II presents the technological background and discusses similar service architectures based on Distributed Ledger Technology (DLT). Chapter III then presents the architecture and communication protocol of Tawki, where the implementation is discussed in Chapter IV. Chapter V discusses challenges of the proposed solution that are yet to be addressed and solved. Finally, Chapter VI discusses the proposed architecture and concludes the paper.

## II. Related Work

In this section, we first introduce Online Social Network services to provide a common understanding of the goals and feature sets of such networks. Based on this, we provide an overview of related Blockchain-based solutions for OSN services and their advantages in comparison to centralized solutions as well as obstacles they currently face. Thereafter, an overview of Blockchain-based identity solutions is presented to evaluate their applicability for user-owned identities in the context of OSN services.

### A. Online Social Networks

OSN services became popular in the late 1990s, when services such as Sixdegrees.com or Friendster allowed people to create a profile page for themselves and create links to other users's profile pages. This network of interconnected digital representations of individuals became increasingly popular and ultimately led to implementation of today's dominant OSN services such as Twitter or Facebook [1]. Existing literature employs various terms for social web services, including *Online Social Network* (OSN), *Social Web Site* (SWS) and *Social Networking Site* (SNS). One of the most accepted and used definitions of OSN services was given by Boyd and Ellison in [11], who defined a Social Network Site (SNS) as a web based service that allows *"[...] individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system"*. Many other definitions of OSNs exist, which vary in details. Anyhow, certain recurring aspects are mentioned in most definitions, being (1) a public social profile, (2) the ability to create, maintain, and publish a list of connections to other users, (3) means to create and publish content in various forms, and (4) communication with other users.

To address the issue of OSN services mostly being closed and centralized services, decentralized OSN architectures were proposed [3] [12], which either relied on P2P technology (e.g. [13] [14]), a federation of loosely coupled servers (e.g. [15] [16]), or hybrid approaches (e.g. [17]). Yet, even with the existing federated or P2P solutions, users are still confined within one OSN system where communication with other OSN systems is not possible at all or limited to few options [10].

In order to implement communication between multiple, mutually independent OSN servers, several communication protocols exist. The Extensible Messaging and Presence Protocol (XMPP) was introduced in 1998 as the first decentralized Instant Messaging (IM) protocol [18]. Being an open and decentralized alternative to the plethora of mutually incompatible IM services of that time, XMPP allows users to register with any XMPP server and connect to users, including users registered on different XMPP servers. The architecture of XMPP comprises a distributed client-server infrastructure, where XMPP servers connect to each other in a loosely coupled fashion. Users in XMPP are identified via Jabber IDs (JID), which comprise a local-part and a domain-part, where a JID's domain-part identifies the Fully Qualified Domain Name (FQDN) of the XMPP server the JID is registered with [19]. Messages in XMPP are sent to the server a user is registered with, which then forwards them to the recipient's server identified by the domain-part of the recipient's JID. The communication protocol uses XML streams for data exchange for client-to-server (c2s) and server-to-server (s2s) communication, where information is encapsulated in *stanzas*. In order to support versatility and extensibility, XMPP supports the XMPP Extension Protocol (XEP) [20], which allows the definition of additional functionality on top of the standard.

As XMPP was deemed to be *"not particularly web-friendly"* and lacked support for mobile use-cases [21], the Matrix protocol [22] was proposed as an open, federated protocol for IM, Voice over IP (VoIP), and the Internet of Things (IoT) for real-time communication. Similar to XMPP, Matrix consists of a federation of loosely coupled servers, to which users connect via a REST-based API. Users in matrix are addressed via identifiers that comprise both a local and a domain-part, while messages are routed via the client's Matrix server to the recipients' servers, from where they can be accessed. By implementing REST-based APIs, Matrix is easy to be integrated into web and mobile applications.

Still, as XMPP and Matrix both implement decentralized communication between multiple mutually independent service operators, user accounts are still bound to the service and domain they signed up with in the first place and cannot be moved to another service location without changing the identifier [23].

To address the underlying problem of incompatible communication standards and identity management in OSN services, Sonic was proposed as an open and heterogeneous social communication framework [1]. Sonic allows users to host their social profile pages on any server, while a Distributed Hash Table (DHT)-based directory service manages discovery of social profiles and identity management. Sonic allows arbitrary OSN services to seamlessly connect to each other, therefore eradicating gaps between different services altogether. Still, user-identifiers in Sonic are based on base36-encoded hash values, making them not human-readable and hard to remember and therefore less suitable for use in social communication services.

### B. Blockchain-based Social Networks

While P2P service architectures facilitate applications and services without central control entities, reliability and authenticity of the managed information is hard to guarantee. With the advent of Blockchain technology, immutability of the stored information (state) was introduced, where only verified and authenticated transactions can change the global state. Distributed consensus algorithms ensure that manipulation of any information managed by a DLT-based system is rendered practically impossible [24].

Since the introduction of Blockchain networks, such as Bitcoin or Ethereum, a plethora of decentralized services and applications have been proposed that build on or use Blockchain technology to implement independence from centralized components [25]. Being important services in our daily digital routines, OSN services have been proposed as Blockchain-based services. Chakravorty and Rong proposed a concept for a social network that utilizes Blockchain technologies for guaranteeing authenticity of blog entries called Ushare [26]. While data is proposed to be stored in a DHT, the user is enabled to limit the degree to which it is shared in the network using a so called relationship system. Published data is encrypted and hashes referring to its ownership are stored in the Blockchain. The proposed solution depicts a promising concept for restricting access to propagated data. Nevertheless, it cannot secure republishing once decoded information and does not provide sufficient incentives for running a node or storing data in the DHT. Furthermore, the proposed utilization of a permissioned Blockchain raises questions regarding potential centralization and lack of trust.

In contrast, Peepeth[3] is a microblogging service based on the Ethereum Blockchain. To decrease gas costs and guarantee scalability, data is stored on the InterPlanetary File System (IPFS) while only references to hashes are kept on-chain. Another measure for reducing the amount of Blockchain transactions are bulk postings. A user may post up to 15 messages or interactions before a transaction is submitted, bundling all actions into a single transaction. Peepeth uses a smart contract for all Blockchain related interactions, taking a share of ether for actions such as donations. Third parties are enabled to utilize the smart contract for own front-ends as the corresponding Application Binary Interface (ABI) is freely accessible. However, the smart contract itself is closed source, circumventing trust and transparency. Due to the utilization of IPFS, messages can never be changed or deleted safely which reduces users' control over their own data. The front-end provided does not show messages that are declared inappropriate. While these messages are still available in the network, there exists a risk of censorship by the central access point that is currently used.

Status.im[4] focuses on building a decentralized platform around a messaging application that is based on the Ethereum Blockchain and its sub-protocols. Peer-to-peer communication is implemented using Ethereum's Whisper protocol with delegate notes for offline inboxing. As Status focuses on messaging, data is kept on the user's device. While this is a viable solution for the target use-case, microblogging requires data availability when user devices are offline.

### C. Identity on Blockchain

Before the advent of Blockchain technology, the paradigm of Zooko's triangle suggested that one would have to choose two out of the three name system properties "distribution", "security" and "human-readability" [27]. Security is defined by the ability to verify the correctness of a value to a given key. The Domain Name System (DNS), for instance, guarantees secure resolving of human-readable names, but is not distributed. Instead, it follows a hierarchical approach with central authorities. Blockchain based solutions such as Namecoin [28] and Blockstack [29] claim to overcome this restriction by referencing human-readable names on the ledger, benefiting of the technology's distributed and secure characteristics. As a result, all three properties of Zooko's triangle can be achieved.

In most Blockchain implementations, users create cryptographic key pairs for interacting with the ledger. While this mechanism creates anonymity to a certain degree, it hampers usability as byte sequences have to be handled by users. To mitigate this problem, human readable identifiers have been introduced. For instance, the EOS Blockchain natively supports accounts which are identified by a 12 character long name [30]. In the Ethereum ecosystem, the ENS[5] creates a link between human readable identifiers and attributes such as address, name or swarm[6] hash. The schema used for identifiers is comparable to the DNS, there are top-level domains owned by registrar contracts and sub-domains. Each domain entry in a registrar references a resolver contract that holds all attached information. While there is a default resolver, custom contracts that implement the corresponding interface can be deployed to define additional attributes.

[3]Peepeth website: https://peepeth.com/

[4]Status.IM website: https://status.im/

[5]ENS website: https://ens.domains/

[6]Swarm Github repository: https://github.com/ethersphere/swarm

In contrast to Blockchains that provide in-built naming mechanisms or use smart contracts for this purpose, Blockstack is a naming solution that aims to build a virtual layer on top of existing Blockchains to implement novel functionality while maintaining the security of the underlaying Blockchain [29]. In its current implementation, Blockstack builds an identity system based on Bitcoin that does not only provide DNS functionality but also user identities. Identifiers declare storage locations in zone files which are saved by all Blockstack nodes, only their hashes are stored in the underlaying Blockchain for verification purposes [31].

Sovrin [32] and uPort [33] target solving more sophisticated identity use-cases like proving claims in a decentralized, secure and privacy-preserving manner. Sovrin relies on a public, permissioned ledger that is operated by a set of trusted entities accross the globe, where nodes are authorized and governed by the non-profit Sovrin foundation. Although this architecture guarantees low operational costs as consensus is found using Redundant Byzantine Fault Tolerance (RBFT) instead of Proof-of-Work (PoW) and no cryptocurrency is involved as of now, it relies on trust in the involved governance mechanism. In contrast to Sovrin, uPort is based on the Ethereum Blockchain and does not implicate external governance but requires payments in Ethereum's cryptocurrency Ether for interacting with the ledger. uPort is based on smart contracts and abstracts user accounts through proxy contracts that are owned by the user but can be reclaimed in case keys are lost. To achieve this, trusted entities are defined in controller contracts for attesting ownership. While Sovrin and uPort focus on elaborate use-cases with multiple entities involved, such as claim issuers and verifiers, they are not well suited for simple human-readable user identification.

## III. Tawki Architecture

In order to facilitate web-based communication and collaboration without the need for any central or semi-central entity controlling or monitoring the process, we propose Tawki. Tawki leverages Blockchain technology to manage identities in a decentralized and distributed fashion, building on a Blockchain-based identification mechanism that links self-hosted data storages to a distributed, secure and human-readable identifier. Hereby, Tawki poses an alternative to current centralized or federated social networks that suffer from isolated data storage and lock-in effects in so-called "walled gardens".

### A. Personal Data Storage

To achieve data sovereignty, user generated content is only stored in a personal data storage to ensure full control over created content. This backend server does not only offer data storage but also supplies a REST API for interacting with the client and exchanging messages with other users' personal data storages. Users can either deploy an instance on their own device such as a Network-attached storage (NAS) or use a trusted entity's service. As personal content is only stored on a device that is controlled by the user and distributed to other contacts when intended, it can be deleted or altered by its owner at any time, while allowing the owner to define fine-grained access control policies for other users. Hereby, sensitive data is not only protected from unintended processing by a third party, but it is also fully compliant to the European General Data Protection Regulation (GDPR) [34].

In contrast to client devices, the personal data storage server has to provide high availability to be accessible by other contacts who retrieve public or shared information and messages. Notifications about new private or public messages are stored by the backend and delivered to clients when going online. Hereby, a user-experience similar to centralized OSNs is guaranteed as information can be exchanged without all clients having to be available at the same time. Even though decentralization introduces a certain communication overhead compared to centralized service architectures, the general applicability of such a federated approach has been found to be feasible [35] [36].

In order to verify the validity of interactions with the server, for instance in server-to-server communication, a key pair is generated. The respective public key is stored and linked to the user on a distributed ledger. Server responses are signed using the private key and therefore verifiable after retrieving the key from the ledger.

### B. User Identification

In Tawki, user-specific identifiers are stored on the Blockchain in order to achieve secure, distributed and human-readable names. Unique identifiers are attributed with a URI that refers to a user's API server handling message exchange and personal data storage. As the link between a user's identifier and storage is decoupled, the server instance can be shifted to any location at any time. To achieve this, only the URI attribute set on the Blockchain needs to be adjusted. This characteristic distinguishes Tawki from federated approaches as Diaspora, where users are able to choose an instance in the first place, but are incapable of moving their profile to another server without loosing personal information and connections. Following Tawki's approach, connections to other users remain intact and no data is lost.

The user login in the client is deduced from the cryptographic key pair that is used for Blockchain communication. These keys are handled by Blockchain-specific wallets and are accessible through browser plug-ins, mobile applications or standalone browsers. As a result, users do not have to handle username and password, but are identified by their Blockchain account's key pair and do not have to register at a central service.

### C. Unique Object Identifiers

In order to allow individual data objects in Tawki to be uniquely identifiable and distinguishable, unique object identifiers (UOID) are used. To avoid intentional and unintentional collision of identifiers, UOIDs comprise two parts where the first part represents the user identifier of the user creating the data object. The second part of a UOID is a random hash, which is separated from the user's identifier via a colon (":"). The resulting UOID in the format `user-id:random-id` allows to uniquely identify any data object as well as its creator, allowing to resolve a UOID to the linked data object by resolving the first part to the respective user's server and user account.

### D. Message format

In Tawki, messages exchanged between two or multiple users are encoded as message objects. When a new message is composed, a message object is created with a *message-id*, which is used to identify the data object. To allow other messages to reference an existing message, each message object specifies an optional *reference-id*, where both *message-id* and *reference-id* are UOIDs. If a message is a direct answer or comment on another message, it references this message's message-id by specifying it as it's own reference-id. Messages that do not refer to another message specify a blank reference-id. This way, messages can be created as individual messages as well as as comments or responses to previous messages. Furthermore, messages can be grouped in logical channels called conversations by specifying an optional parameter *conversation-id*, allowing for nested conversations between two or multiple communication partners. To create a new channel, a UOID is created as a conversation identifier, which is then specified in all message objects sent to the same channel. Messages with the same conversation-id can then be grouped, allowing the creation of distinct channels. Hereby, messages can be assigned to different target topics. The resulting format of a Tawki message object is illustrated in Table I.

TABLE I
TAWKI MESSAGE FORMAT

| Parameter | Description |
|---|---|
| `message-id` | Specifies the identifier of the message object. |
| `reference-id` | Specifies the optional referenced message. |
| `conversation-id` | Optional ID to group messages in logical channels. |
| `from` | ENS name of the message's author. |
| `to` | ENS name of the message's recipient. |
| `date` | Timestamp |
| `message` | The messages content |

### E. Tawki Communication Protocol

*1) Registration:* The registration process involves three entities; a unique Blockchain based identifier linking to a user and his/her backend instance, the backend itself and the client that accesses both of the former, including a Blockchain wallet such as Metamask. The process consists of the following three steps:

1) Users are required to register a human-readable name on a distributed ledger. The registration process depends on the identity and naming mechanisms supplied by the underlying implementation. For instance, while EOS supplies a built-in method for human-readable names, the smart-contract based ENS provides similar features for Ethereum.
2) After deploying a server instance, its Uniform Resource Identifier (URI) is stored and linked to the user's unique name on the ledger. As this URI can be changed by the user at any time, user data can be migrated easily to other locations without loosing any (social) connections. In addition to the URI, the user's and server's public keys are stored for verifying signed requests in communication.
3) After registering a name and deploying a backend, users create a public profile that is stored on their private storage. It includes a display name, picture and description.

*2) Friend Request:* As discovering and connecting with friends and other contacts is the key aspect in social networks, this process has to be transparent to users. Tawki retains the distributed characteristics of social graphs in its technology stack while providing human readable names for identifying individuals. The procedure of building connections in Tawki is exemplified for two users, namely Alice and Bob, in Figure 1 and consists of the following six steps:

1) Before adding new contacts, users can lookup people by using their unique identifiers. As these identifiers may - depending on the underlaying implementation - be stored in a hashed form, they have to be known by the users. This can be achieved by searching for user profiles via a distributed flooding search or passing another user's identifier directly, e.g. via email.
2) To retrieve another user's profile, the ledger's name service is used to resolve an identifier to the corresponding user's backend URI. The server hosting the user's backend is then queried directly from the client to receive user information. Based on the public profile, Alice decides if she would like to send Bob a friend request.
3) To send a new friend request to Bob, Alice instructs her own server instance to send a friend request message to Bob using his unique identifier. The user-specific signature is generated using the blockchain wallet's key pair and used for two purposes. Firstly, Alice authenticates herself at her own server. Secondly, it enables Bob's server to verify the request's authenticity in Step 6.
4) Alice's request is stored on her own backend to provide valid feedback on the current status to Alice and to process Bob's reply.
5) The request is forwarded to Bob's backend server.
6) Upon reception of a new friend request, Bob's backend server queries the Blockchain to retrieve Alice's public key in order to verify the certificate's authenticity.
7) In case the request is valid, it is stored for further processing and availability to the user.

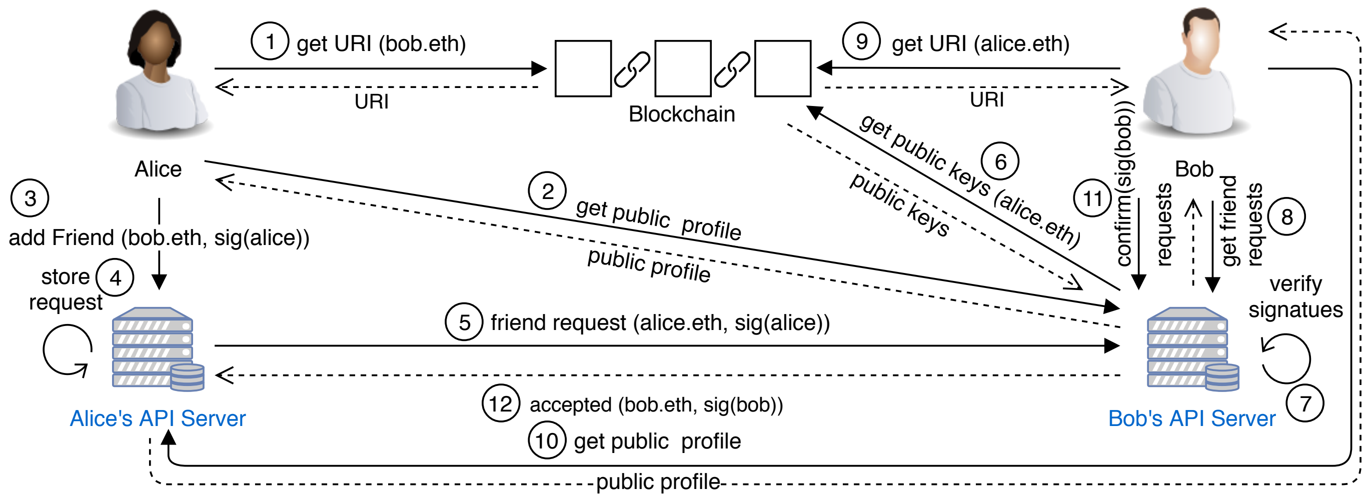


Fig. 1. Sending a friend request in Tawki

8) When Bob goes online, his client queries the backend for all pending friend requests and retrieves Alice's inquiry.
9) To display Alice's public profile, the client receives her backend server's URI from the blockchain.
10) The profile is received from Alice's server and presented to the Bob.
11) Bob either accepts or declines the request by sending an answer to his server.
12) Bob's server processes the answer by adding Alice to Bob's friend list in case the request was accepted and removing it from the pending list. The decision is then forwarded to Alice's server which follows the same pattern by removing the request from the pending list and adding Bob to Alice's friends list in case it was accepted.

*3) Messaging:* Conceptually, Tawki does not differentiate between private messaging and public posts as used in microblogging. The main difference is in the definition of recipients in the message object. If no recipients are defined, messages are declared public and subscribers are notified about the new public post. Otherwise, the private message is delivered to all declared recipients. We showcase the protocol's procedure using a private messaging example with multiple participants, Alice, Bob, and Carol, as presented in Figure 2. The process consists of the following steps:

1) In order to create a new conversation, Alice creates a message object, as defined in Section III-D. A newly generated conversation ID indicates a new conversation. In case the ID is already taken, it indicates an answer in a conversation. To comment a message, the corresponding message ID should be the referenced.
2) Alice sends the created message object to her own server instance.
3) The user-specific backend server persists the message object. From this point it can be queried by Alice and all receivers.
4) All recipients' server URIs are retrieved from the blockchain.
5) A notification is sent to the recipients' servers. The notification may include the message object, but does not have to. As private messages are likely to be viewed by users, including the message in the notification and caching it on the recipients' servers prevents the overhead of querying Alice's server as soon as a message is to be read. On the other hand, for public posts, there is a lower probability of potential recipients to read the message so that it should be requested on demand. The example presented in Figure 2 shows a private message. Therefore, the message content is included in the notification.
6) Upon retrieval of a notification, the sender's public key is retrieved from the blockchain.
7) The signature sent with the notification is verified using the received public key. Furthermore, the notification is persisted on the server.
8) When going online, participants retrieve private messages and the server creates an ordered list of public posts. If the message is not cached or the defined cache duration is overdue, the message content is retrieved from the sender's server by the recipient's personal data storage server before forwarding it.
9) For composing a reply, Carol creates a message object using the same conversation-id.

## IV. Implementation

The prototypical implementation of Tawki consists of two main components. Firstly, the user-specific backend supplies an API for accessing and storing data as well es exchanging messages. It is implemented using Express.js[7], a web application framework for Node.js[8]. The API is following the REST

[7]ExpressJS website: http://expressjs.com/
[8]NodeJS website: https://nodejs.org/en/

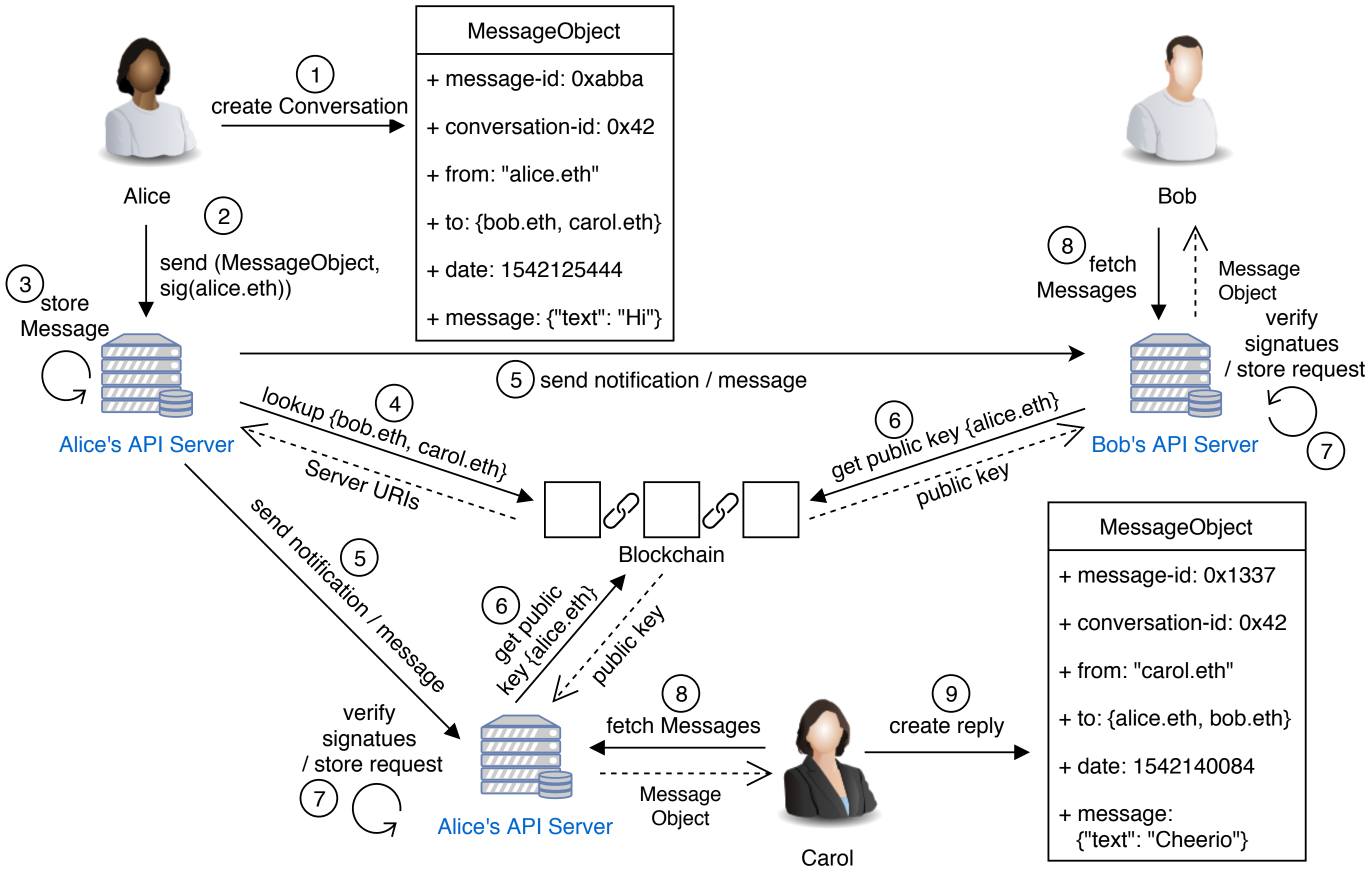


Fig. 2. Creating a private group conversation with three participants

architecture and is therefore implemented fully stateless and web-based. All user-generated data is stored in a MongoDB[9] which is accessed via Mongoose[10].

Secondly, a web-based client application, depicted in Figure 3, provides a user interface to users that interacts with the Blockchain as well as the user's storage server. The application logic of the client application is implemented using react.js[11]

For the management of user identities, Tawki uses the Ethereum Blockchain. Ethereum has been chosen due to its comprehensive software support for wallets on various platforms and its mature implementation of the ENS. Blockchain interactions such as name registrations and lookups are conducted using Ethereums's JavaScript library Web3js[12]. As the Web3js library requires a web3 instance to be injected to use the webpage, a browser plugin such as Metamask[13], a standalone Ethereum browser like Mist[14] or a mobile application like the Status[15] client can be used.

As an ENS name is used for identification, users first have to register such name where the registration process depends on the top level domain's registrar. For example, while the ".test" domain on the Ropsten testnet uses a first-come-first-serve mechanism, the mainnet's ".eth" registrar utilizes a bidding mechanism that requires multiple steps for placing bids and revealing them. Locking Ether in the registrar contract prevents spam and provides a more elaborate method for assigning popular names. After a name has been registered, a resolver contract has to be set in the registrar contract, with the resolver supplying a set of attributes for each name it holds. While there is a default resolver containing features such as address or name, users may also define or deploy their own contracts to supply further information. In the registration process, the Tawki-specific resolver has to be set as it enables linking a URI to a name. Furthermore, the user's public key is stored to enable the verification of certificates in messaging. As names are hashed in the ENS, names cannot be resolved from an address by default. To permit the front-end to retrieve the user's name from the address that is supplied by the wallet, a reverse registrar is used, linking the address to the before registered name.

When first visiting Tawki, users are lead through a registration process that permits them to register a ENS name. It automatically applies the correct resolver contract and reverse resolver and sets the user's respective backend API. Users can

[9]MongoDB website: https://www.mongodb.com/
[10]Mongoose website: https://mongoosejs.com/
[11]ReactJS website: https://reactjs.org/
[12]Web3JS GitHub repository: https://github.com/ethereum/web3.js/
[13]Metamask website: https://metamask.io/
[14]Mist GitHub repository: https://github.com/ethereum/mist
[15]Status website: https://status.im/

adjust their server location at any time via a configuration panel, supplying full control over data storage. Other contacts' profiles can be viewed using their respective ENS names. When following another user, their microblogging messages are presented in a timeline on the homepage.

Blockchain-related transaction costs only accrue during the registration of an identifier and when changing the respective URI pointer in case user data is migrated to another storage. Despite fluctuating gas costs on Ethereum and changing exchange rates for the required cryptocurrency, the actual costs depend on the registrar that is used. For instance, the ENS registrar responsible for the '.eth' domain requires users to bid on names for their distribution. The minimum bid for a name is 0.01 ether[16], not including required gas costs for executing related transactions. When winning an auction, the amount of ether is locked in the registrar's smart contract for at least one year. Therefore, the cryptocurrency used to claim a name is not spent but locked in a smart contract for a certain time frame. However, other registrars may apply different rules. The '.test' registrar deployed on the Ropsten testnet, for instance, permits registering names in a first-come-first-serve manner. Thereafter, the user owns the name for 28 days before it is freed and open for registration by any user. To provide an indicator of the involved costs for registering and maintaining a name on the ENS, we measure the gas costs generated by respective transactions. During the registration process, five methods are called to sign up for a name, referencing the Tawki-specific resolver and setting the API-server's URI. In addition, the name is registered on the reverse registrar for creating a bidirectional link. In total, the required gas sums up to 207,101 gas, as depicted in Table II. As of November 24, 2018, the gas price for a transaction to be mined within five minutes was at 2.5 gwei[17], according ETH Gas Station[18], a web page predicting gas costs based on recent blocks mined in the Ethereum chain. Following this assumption, 207,101 gas * 2.5 gwei = 0.000517752 ether are required for completing the registration process on Tawki. While exchange rates highly fluctuate for Ether [37], Ethereum's native cryptocurrency, we calculate the respective fiat currency to provide a reference. According to CoinMartketCap[19], 1 Ether equals 131.35 USD at the time of writing, resulting in sign up costs of 0.068 USD. Hereby, the involved expenses are within low bounds even when considering fluctuating exchange rates, making Tawki a feasible solution for a decentralized, privacy focused social network.

## V. Discussion and Future Work

The proposed approach for storing data requires deploying a backend interface for each user in the system. As this involves some overhead for the end-user, it might influence a broader adoption for users who are used to the comparably convenient registration processes of centralized web services.

[16]ENS website: https://registrar.ens.domains/

[17]1 ether = 1000000000 gwei

[18]ETH gas station website: ethgasstation.info

[19]CoinMartketCap website: https://coinmarketcap.com/

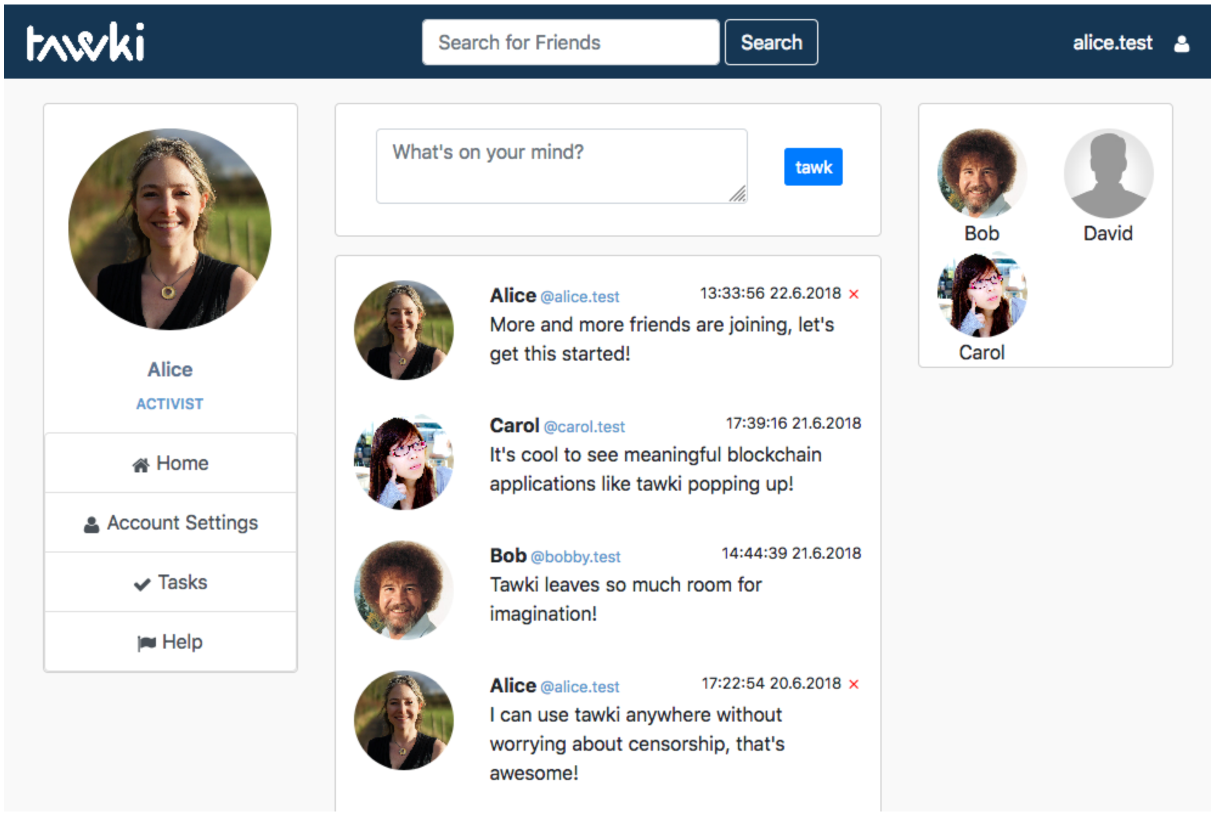
TABLE II
Gas cost analysis

| Operation | Gas costs |
|---|---|
| Name registration | 38,254 |
| Set Resolver | 32,419 |
| Set URI | 41,344 |
| Claim reverse address | 38,254 |
| Set name on reverse resolver | 56,830 |
| | 207,101 |

To mitigate this issue, service providers could offer hosting data for users. In case of privacy concerns or a lack of trust in a chosen provider, users could transfer their data at any time without loosing data or any connections to contacts. As a result, several service providers would compete for users by offering trusted services.

Another approach would be to rely on P2P storages such as IPFS and Swarm which have been proposed in the Blockchain space. While users do not necessarily have to run their own nodes, data can be published in the networks and accessed by their hashes. As a result, referenced data cannot be tampered even though it is not self-hosted. While this approach provides easy access to data in the network, data objects cannot be deleted securely due to its potential replication in the network, limiting users' data sovereignty. As data in social networks is typically sensitive to users, even when encrypting data, missing revocation mechanisms result in privacy issues in case private keys are disclosed. Furthermore, the networks currently lack incentivization and therefore limit data persistence. While Swarm is to be incentivized within the Ethereum ecosystem in the future [38], Filecoin proposes a unique Blockchain for such purposes on IPFS [39]. However, these mechanisms have not yet proven to be applicable to date and cannot ensure the deletion of data. Consequently, these P2P technologies may only constitute a viable solution for data which is not sensitive and does not have to be deletable or modifiable.

A key feature in centralized OSNs are suggestions of new contacts and topics including network wide discussions using hashtags, as implemented in Twitter and Facebook. In distributed systems, indexing is a difficult endeavor due to limited data access. While on the one hand, such limited access of personal data is an intended characteristic of Tawki, restrictions discovering novel information may hamper its user acceptance. To alleviate such implications, indexing services may evolve in the future to provide such functionality. Users could freely choose from a range of indexing services based on privacy aspects, quality of suggestions and applied filters. For instance, emerging index providers could either claim to be completely free of censorship or filter certain content that is defined improper. After registering for an index, users do not only notify connected followers about new messages, but also the index service to include it for global discussions, trending topics and so forth. In contrast to centralized OSNs, however, the index is independent of the user's data storage location and can be changed at any time.



Alice @alice.test 13:33:56 22.6.2018 ×
More and more friends are joining, let's get this started!

Carol @carol.test 17:39:16 21.6.2018
It's cool to see meaningful blockchain applications like tawki popping up!

Bob @bobby.test 14:44:39 21.6.2018
Tawki leaves so much room for imagination!

Alice @alice.test 17:22:54 20.6.2018 ×
I can use tawki anywhere without worrying about censorship, that's awesome!

Fig. 3. Screenshot of the Tawki webapp

## VI. Conclusion

Today's web-based social communication services are mostly built in a closed, proprietary fashion. While communication itself is a decentralized, independent, and universal phenomenon, web-based social communication is mostly managed by and routed via central servers. As a consequence, users are forced to disclose their entire profile data and communication contents to the respective service's operator. As business models of social communication platforms are mostly built around analyzing the mostly very personal data for the purpose of targeted advertisement, service operators have shown to be reluctant to opening their service platforms and allowing competitors to connect to their users. This creates lock-in effects, as user who would like to switch to a competitor's service are in risk of losing all their data and connections to other users.

While multiple federated alternatives have been proposed, they mostly fail to eliminate lock-in effects due to isolated data storage APIs and protocols, as well as due to user identities being bound to the service providers' domains. Furthermore, current blockchain-based implementations do not provide data sovereignty, as data is either stored on immutable ledgers or distributed in peer-to-peer networks. This way, control over one's personal data is lost, as the information is replicated to multiple, possible insecure nodes in the system.

In this paper we proposed Tawki, a decentralized social communication architecture that allows users to freely choose a personal data storage for managing their data. Tawki data storages implement a common REST-based API that allows other users to send and request messages and information to a user's endpoint - regardless of the implementation being used. User identities are managed by the Ethereum Blockchain, where the ENS is used to resolve a user's identifier to his (current) personal data storage's location. This way, communication in the Tawki ecosystem is entirely peer-ro-peer with no central entity being able to control or censor the information flow. A prototypical web-based client has been implemented that demonstrates the applicability of the proposed solution.

## References

[1] S. Göndör, "Seamless interoperability and data portability in the social web for facilitating an open and heterogeneous online social network federation," Ph.D. dissertation, Technische Universität Berlin, June 2018.

[2] O. Malik, "Is Facebook Beacon a Privacy Nightmare?" Tech. Rep., 2007, https://gigaom.com/2007/11/06/facebook-beacon-privacy-issues/. Accessed: 6.6.2017.

[3] C. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee, "Decentralization: The Future of Online Social Networking," in *W3C Workshop on the Future of Social Networking Position Papers*, vol. 2, 2009.

[4] F. Stutzman, R. Gross, and A. Acquisti, "Silent Listeners: The Evolution of Privacy and Disclosure on Facebook," *Journal of Privacy and Confidentiality*, vol. 4, no. 2, p. 2, 2013.

[5] B. Fitzpatrick and D. Recordon, "Thoughts on the Social Graph," 2007, http://bradfitz.com/social-graph-problem/ Accessed: 15.5.2017.

[6] T. Berners-Lee. (2009) Socially Aware Cloud Storage. [Online]. Available: https://www.w3.org/DesignIssues/CloudStorage.html

[7] S. Gilbertson, "Slap in the Facebook: It's Time for Social Networks to Open Up," 2007, https://www.wired.com/2007/08/open-social-net/ Accessed: 21.5.2017.

[8] T. Paul, A. Famulari, and T. Strufe, "A Survey on Decentralized Online Social Networks," *Computer Networks*, vol. 75, Part A, pp. 437–452, 2014.

[9] D. Koll, J. Li, and X. Fu, "The Good Left Undone: Advances and Challenges in Decentralizing Online Social Networks," *Computer Communications*, 2017.

[10] S. Göndör and A. Küpper, "The Current State of Interoperability in Decentralized Online Social Networking Services," in *Proceedings of the 2017 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2017.

[11] D. M. Boyd and N. B. Ellison, "Social Network Sites: Definition, History, and Scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[12] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, "Decentralized Online Social Networks," in *Handbook of Social Network Technologies and Applications*. Springer, 2010, pp. 349–378.

[13] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "PeerSoN: P2P Social Networking: Early Experiences and Insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*. ACM, 2009, pp. 46–52.

[14] M. Durr, M. Maier, and F. Dorfmeister, "Vegas - A Secure and Privacy-Preserving Peer-to-Peer Online Social Network," in *International Conference on and 2012 International Conference on Social Computing (SocialCom) Privacy, Security, Risk and Trust (PASSAT)*. IEEE, 2012, pp. 868–874.

[15] Diaspora, "An Introduction to the Diaspora Source," 2015, https://wiki.diasporafoundation.org/An_introduction_to_the_Diaspora_source, Accessed: 9.1.2017.

[16] Friendica, "Develop," http://friendica.com/develop, Accessed: 15.1.2017.

[17] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A Privacy-Preserving Online Social Network Leveraging on Real-Life Trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94–101, 2009.

[18] P. Saint-Andre, K. Smith, and R. Tronçon, *XMPP: The Definitive Guide*. O'Reilly, 2009.

[19] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Address Format," IETF, Tech. Rep., 2011, https://tools.ietf.org/html/rfc6122. Accessed: 31.7.2017.

[20] P. Saint-Andre and D. Cridland, "XEP-0001: XMPP Extension Protocols," XMPP Standards Foundation, Tech. Rep., 2016, https://xmpp.org/extensions/xep-0001.html. Accessed: 1.8.2017.

[21] "Matrix FAQ," Matrix.org, Tech. Rep., 2017, https://matrix.org/docs/guides/faq.html. Accessed: 1.8.2017.

[22] "Matrix Specification," Matrix.org, Tech. Rep., 2017, https://matrix.org/docs/spec/. Accessed: 1.8.2017.

[23] S. Göndör, F. Beierle, E. Küçükbayraktar, , H. Hebbo, S. Sharhan, and A. Küpper, "Towards Migration of User Profiles in the SONIC Online Social Network Federation," in *Proceedings of the 10th International Multi-Conference on Computing in the Global Information Technology (ICCGI)*. IARIA, 2015.

[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, http://www.bitcoin.org/bitcoin.pdf. Accessed: 23.9.2018.

[25] M. Westerkamp, F. Victor, and A. Küpper, "Blockchain-based Supply Chain Traceability: Token Recipes Model Manufacturing Processes," in *2018 IEEE International Conference on Blockchain (Blockchain 2018)*, 2018.

[26] A. Chakravorty and C. Rong, "Ushare: User Controlled Social Media Based on Blockchain," in *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, ser. IMCOM '17. New York, NY, USA: ACM, 2017, pp. 99:1–99:6.

[27] Z. Wilcox-O'Hearn, "Names: Distributed, secure, human-readable: Choose two," 2010. [Online]. Available: http://web.archive.org/web/20011020191610/http://zooko.com/distnames.html

[28] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," in *WEIS*, 2015.

[29] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A Global Naming and Storage System Secured by Blockchains," in *USENIX Annual Technical Conference*, 2016, pp. 181–194.

[30] block.one, "EOS.IO Technical White Paper v2," Tech. Rep., 2018, https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. Accessed: 7.9.2018.

[31] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Bootstrapping trust in distributed systems with blockchains," *login: USENIX Mag.*, vol. 41, no. 3, 2016.

[32] Sovrin Foundation, "Sovrin: A Protocol and Token for Self-Sovereign Identity and Decentralized Trust," Tech. Rep., 2018, https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf. Accessed: 24.9.2018.

[33] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, "uPort: A Platform for Self-Sovereign Identity," Tech. Rep., 2016, http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf. Accessed: 24.9.2018.

[34] European Parliament, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 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)," 2016, http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN. Accessed: 15.9.2017.

[35] M. Marcon, B. Viswanath, M. Cha, and K. P. Gummadi, "Sharing Social Content from Home: A Measurement-Driven Feasibility Study," in *Proceedings of the 21st International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2011, pp. 45–50.

[36] A. Koopmans, "Decentralized Social Networking Site," Bachelor Thesis, Universiteit van Amsterdam, 2015.

[37] S. Ammous, "Can cryptocurrencies fulfil the functions of money?" *The Quarterly Review of Economics and Finance*, 2018.

[38] V. Trón, A. Fischer, D. A. Nagy, and Z. Felföldi, "swap, swear and swindle: incentive system for swarm," Tech. Rep., 2016, http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw%5E3.pdf. Accessed: 20.8.2018.

[39] Protocol Labs, "Filecoin: A Decentralized Storage Network," Tech. Rep., 2017, https://filecoin.io/filecoin.pdf. Accessed: 20.8.2018.