Trusting Decentralized Web Data in a Solid-based Social Network*

Valentin Siegert✉[0000−0001−5763−8265], Dirk Leichsenring[0009−0008−6354−0019], and Martin Gaedke[0000−0002−6729−2912]

Distributed and Self-organizing Systems,
Chemnitz University of Technology, Chemnitz Germany
{valentin.siegert,dirk.leichsenring,martin.gaedke}@
informatik.tu-chemnitz.de

Abstract. In our current data-centric society, there is a rising concern among individuals regarding their privacy and the degree of control they have over their personal data. In response to this growing demand for transparency and control, a recent initiative for web re-decentralization has emerged, wherein web applications no longer store data centrally. One of the prominent approaches aimed at re-decentralizing the web is Solid. Solid facilitates data storage in decentralized pods secured with web access control, enabling seamless connectivity of diverse data. However, the integration of decentralized data from various third-party pods in web applications poses a significant challenge, as the data's trustworthiness may be compromised, potentially leading to malicious or harmful outcomes. Therefore, data integration necessitates a trust-aware decision, i.e., the extent to which the data is trustworthy enough to use despite its high heterogeneity. To enable such trust awareness for web applications utilizing decentralized data, we propose a trust-aware framework called TrADS for decentralized social networks. TrADS leverages the MVC pattern of web applications to integrate external data from Solid pods trust aware. The potentially malicious and harmful status of external data is an end-user concern. A web application must therefore be trust aware to provide a better user experience. Therefore, we evaluate the proposed trust-aware decentralized social network in terms of usability and transparency with 53 end users through an empirical study. The results demonstrate that the trust awareness component in TrADS can support end users in their trustworthy user experience.

Keywords: Trust · Social Linked Data (Solid) · Re-decentralization of the Web · Decentralized Web · Responsible Web.

1 Introduction

The web has increasingly become a collection of closed silos of resources [5] due to commercialization. However, more and more attention is being paid to data

* This paper is supported by the European Union's HORIZON Research and Innovation Programme under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI)

sovereignty [3], as full data control is lost as soon as the data is placed in a closed silo. As a result of this loss, data privacy is also restricted accordingly. Various projects are working on decentralizing the web for this reason, but also because of its originally decentralized architecture. For example, with its Next Generation Internet Initiative[1], with Gaia-X[2] and also with the General Data Protection Regulation (GDPR)[3], the EU is driving decentralization forward for political reasons. However, projects such as Solid [13] and Fediverse projects such as Mastodon[4] are also working on the decentralization of the web. The Fediverse is built across multiple instances that can communicate with each other, allowing users to exchange their data between them and thus use it in a decentralized manner [6]. In contrast, Solid's pod structure allows data owners to not only decentralize their data, but also allows them to fully control it.

Decentralized web applications can obtain any data from the web with the help of semantic web technologies and thus use data in a decentralized manner. Such web applications therefore no longer store their users' personal data in a data store controlled by the application, but for example in a Solid pod or in Solid data spaces [11]. However, data storage in decentralized web applications is not fully decentralized, since application-internal data are also kept internal with such a decentralization. In addition, correctly implemented caches in a decentralized web application are required for technically smooth data access to decentralized data in order to ensure fast response times in practice.

As discussed in the position paper by Siegert and Gaedke [14], integrating external data storage creates a significant challenge for decentralized web applications, since user experience could be negatively affected by malicious or harmful data. Classic web applications offer a trustworthy user experience by homogenizing data based on the data model used and reusing this collected data later accordingly. The data comes either from sources known to the application or from user input, whereby this input is validated for security reasons [9]. With decentralized web applications, however, the web application no longer has complete control over the data storage. For example, data once stored on a decentralized Solid pod can be changed without the knowledge or control of a web application before it is read out later. External data can also be malicious or harmful, but should neither be neglected by default, nor used unfiltered in a decentralized web application. Integration of untrustworthy data can result in a deterioration of the user experience. To ensure a trustworthy user experience in decentralized web applications, the trustworthiness of external data must be taken into account. However, external data cannot be continuously monitored on changes by the web application. In addition, the amount of data in the web is constantly growing [16], which makes it impossible to moderate the integration of external data by experts [1]. Thus, decentralized web applications require a

[1] `https://digital-strategy.ec.europa.eu/en/policies/next-generation-internet-initiative`
[2] `https://gaia-x.eu/`
[3] Regulation (EU) 2016/679 of the European Parliament and of the Council
[4] `https://joinmastodon.org/`

trust awareness component [14], which assesses the extent to which the data is trustworthy enough to be used.

A trust awareness component is yet not existing for decentralized web applications. Based on related work [17], trust awareness is calculated on different information about the data, e.g. its origin or reputation, depending on the trust model used. Based on the results of these calculations, a trust-based decision is made about the extent to which the data is trustworthy enough to be used. However, it is currently unclear how the interfaces of a trust awareness component are structured, what typical processes in a web application including such a component look like and how often the component is integrated into the application's workflow. To enable decentralized web applications being trust aware and to address these questions, we introduce in this paper a trust aware decentralized web application. The use case for our application is a social network where a large number of user pods and additional data are merged into a typical social network platform. Furthermore, we evaluate the usability of the prototype and the impact of the trust awareness component on the usability to draw conclusions on how such a component can impact the user experience to be more trustworthy. The contributions of our paper are as follows:

1. We present a framework named TrADS, a **Tr**ust-**A**ware **D**ecentralized **S**ocial network, which has a Solid-based decentralized data layer.
2. We demonstrate its feasibility with a publicly accessible prototype[5] whose code is also accessible[6].
3. We evaluate the usability of our prototype and the impact of the trust awareness component on usability with a 53-participants large user study.

The remainder of the paper begins in section 2 with a qualitative discussion of related work. In section 3, we present our framework for a trust-aware, decentralized, and Solid-based social network called TrADS, along with insights into its prototypical implementation and its qualitative comparison with related work. We detail the 53-participant user study on the usability of TrADS and the impact of the trust awareness component on the usability in section 4, and draw conclusions in section 5.

2 Related Work

Several decentralized social networks exist today already. To systematically analyze and assess them, we identify two sets of requirements, one motivated by the end-users of the social network and the other motivated by the social network's operators. The five end-user requirements are: *trust awareness*, *data control*, *resistance to censorship*, *networkability*, and *independence*. The three operator requirements are: *adaptability*, *manageability*, and *scalability*. The decentralized social networks are selected on the basis of the technology they use in order to

[5] demo: `https://vsr.informatik.tu-chemnitz.de/projects/2024/trads/`
[6] code: `https://zenodo.org/records/10641771`

Table 1: Qualitative comparison between decentralized social networks

Decentralized Social Network	T	D	R	N	I	A	M	S
Mastodon [4]	◔	◕	◔	●	◔	●	○	◕
PeerTube [8]	◔	◕	◔	●	◔	●	○	◕
diaspora* [7]	◔	◕	◔	●	◔	●	○	◕
Pixelfed [9]	◔	◕	◔	●	◔	●	○	◕
Secure Scuttlebutt [10]	○	◕	◔	◔	◔	●	●	●
Sone [11]	○	◕	●	◔	◔	●	●	●
Peergos [12]	○	●	◔	◔	●	●	●	●
SteemIt [14]	○	○	●	○	●	○	●	○
Minds [15]	○	○	●	○	●	○	●	○
Movim [16]	○	◕	●	◕	●	◕	●	●
twtxt [17]	○	◕	●	○	●	◔	●	●
nostr [19]	○	○	●	◔	●	◔	●	●
TrADS	●	●	●	●	●	●	●	◕

T, D, R, N, I, A, M, S respectively stand for: **T**rust awareness, **D**ata control, **R**esistance to censorship, **N**etworkability, **I**ndependence, **A**daptability, **M**anageability, and **S**calability.

cover the widest possible range of different decentralized social network technologies. All requirements are mapped onto a four-level assessment scheme: ○ not satisfied, ◔ partially satisfied, ◕ mostly satisfied, ● fully satisfied. All results of the qualitative comparison are also summarized in table 1, which we detail in the following of this section. In addition, we list the requirement assessment of TrADS, our own solution, which is described in detail in subsection 3.3.

Currently, the most successful approach to decentralized social networks are federated networks building the Fediverse [6], being a combination of individual instances that communicate with each other. Instances can be operated privately or publicly and can be used by a variable number of users, who are free to decide which instance they use. By distributing the network across a large number of independently operating instances, it is ensured that nobody has sole control over the social network. The best-known representatives of federated networks include diaspora* [7], PeerTube[8], PixelFeed[9] and Mastodon[4], which is one of the most successful federated networks with 1.8 million active users in 2022 [10].

In a peer-to-peer (P2P) model, all participants (nodes) have equal rights and can both provide and receive services. In this paper, we consider three P2P-based decentralized social network approaches: (1) Secure Scuttlebutt[10], (2) Sone[11], and (3) Peergos[12]. Users of Secure Scuttlebutt operate their own node on which,

[7] `https://diasporafoundation.org/`
[8] `https://joinpeertube.org/`
[9] `https://pixelfed.org/`
[10] `https://www.scuttlebutt.nz/`
[11] `https://github.com/Bombe/Sone`
[12] `https://peergos.org/`

in addition to their own data, the data of all friends and the data of their friends are stored. Communication beyond those is achieved through the use of so-called pubs (publicly accessible nodes). In Sone, however, users operate their own Freenet[13] access point nodes. Data is encrypted and stored on different nodes, whereby the storage location is determined by a routing algorithm over which users have no direct influence. Users of Peergos, in turn, need access to a node operated by themselves or by others on which they can store their data. They then have the option of following other users, giving them the opportunity to write messages and share data unilaterally. If the data is only stored on a self-operated node, the user may retains full control.

Well-known representatives of blockchain-based decentralized social networks are SteemIt[14] and Minds[15] [4]. The blockchain serves as a record of all actions in the decentralized network. Access to content, such as images or texts, is often realized via a P2P-based data distribution protocol called IPFS [12], as otherwise the required storage capacity of the blockchain would be too large. As the blocks in the chain cannot be subsequently modified or deleted, the use of blockchain results in unresolved problems in the context of social networks [4] like scalability, content visibility and decentralization of content.

Movim[16] is an XMPP-based decentralized social network. Using an abstraction layer, a Movim instance provides an XMPP client that offers the functionalities of a social network. Such an instance can be used by any number of users, regardless of the XMPP servers they use. All personal content is stored on the user's XMPP server allowing users to switch instance at will. The used Movim instance keeps a copy of the content as a cache and makes the created public content available via HTTP. twtxt[17] is a very minimalistic microblogging system approach. Each user makes a twtxt file publicly available via a URL, which contains the user's posts line by line with the URL of this file serving as the identity. A twtxt client retrieves the files of all followed users, similar to an RSS feed reader, and creates a feed from them. Yarn.social[18] is a decentralized social network based on twtxt files, providing optional extensions to support hashtags and metadata for social networks. Notes and Other Stuff Transmitted by Relay (nostr)[19] is a protocol for decentralized social networks based on a relay system. The aim is to create a global independent and censorship-resistant social network consisting of a large number of relays with which users communicate using a client. The relays accept user content and pass it on when it is requested by other users. To publish content, a user, who is identified by a publicly known key, signs the content and sends it to a selection of relays. A nostr client then creates a feed for a user, which consists of all publications by other users that

[13] https://www.hyphanet.org/
[14] https://steemit.com/
[15] https://www.minds.com/
[16] https://movim.eu/
[17] https://twtxt.readthedocs.io/en/latest/user/intro.html
[18] https://yarn.social/
[19] https://nostr.com/

are followed by the client's user. nostr is completely call-based, so there is no way to inform other users about interactions, and once a user has sent data to a relay, they lose all control over it, which makes it impossible to delete any.

While trust awareness on decentralized data is not supported in any of the approaches mentioned, the Fediverse applications at least consider moderation of the data per instance. However, this moderation in the Fediverse cannot be fully automated and is a known and unsolved problem [1], especially for larger instances. Data control is mostly supported, if at all, by its own independent hosting, which has the disadvantage of hosting effort. With approaches such as Movim[16] or twtxt[17], the effort is less, but they still require their own hosting and, like twtxt, sometimes have no access control. Due to the independent distribution of data storage, some of the approaches mentioned are already resistant to censorship. However, many approaches are also limited in their resilience, as individual instances can restrict communication with others. In particular, the Fediverse performs very well in terms of networkability through common data exchange protocols such as ActivityPub [8]. Protocols such as IPFS [12] also support the exchange of data to a high degree, but the using approaches lack networkability due to their blockchain architecture. All of the above approaches give users a choice of instances, which partially supports the idea of independence. However, the better solution is when data and/or identities are not bound to instances, making approaches completely independent of third parties. Adaptability is a given in some approaches but is limited in approaches such as Movim[16], twtxt[17] and nostr[19], and it is not given at all in blockchain-based decentralized social networks. While most approaches already perform well in terms of manageability and scalability, the Fediverse is poor in terms of manageability [1], and also has some disadvantages in terms of scalability. These are due to the caching principle in each instance, which results in overlinear growth of the entire system as a network grows. The blockchain-based approaches are even worse than the Fediverse in terms of scalability, as they face the problem of scalable blockchain technology [4].

3 TrADS

To improve the related work and to especially make social networks trust-aware by integrating a trust awareness component [14], we present in this section TrADS, a **Tr**ust **A**ware **D**ecentralized **S**ocial network. We first explain details on TrADS architecture including its typical processes, and then proceed to present details about its prototype including details on the front-end. To be able to list TrADS in the table 1 of related work, we conclude the section with a brief analysis of TrADS using the requirements mentioned in section 2.

3.1 Architecture

The architecture of such a social network is significantly influenced by the technology used for decentralized storage and data exchange. In order to maintain

the networkability of the Fediverse, the separation of data and instance as in Movim[16], and in particular a high level of data control similar to that in Peergos[12], we use Solid [13] in our approach. The Solid pods allow end users to control their own data through the web standards used and to freely transfer the data to other platforms, thus minimizing the dependency of end users on the actual instance of the social network. The high level of data control of the end user in Solid and the web standards used also enable a high degree of adaptability, as with P2P-based approaches or the Fediverse. Similar to Movim[16] and the blockchain approaches, Solid promotes the independence of the instances, but in contrast, data control is not restricted by a lack of access control.

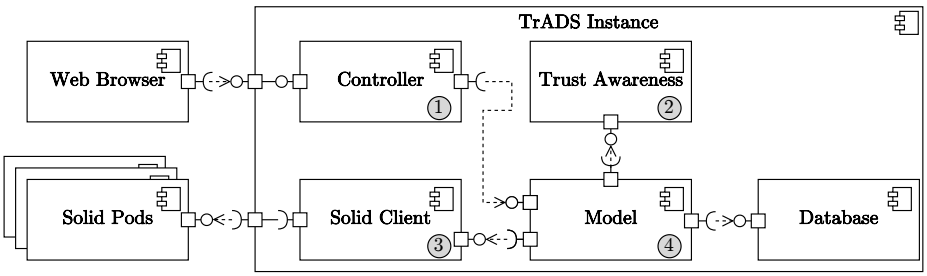


Fig. 1: A TrADS Instance Architecture

A TrADS instance is independent of a central architecture and therefore resistant to censorship by others. The UML component diagram in figure 1 provides an overview of the architecture of a TrADS instance. TrADS is a web application based on the Model-View-Controller (MVC) pattern [7] and therefore has the classic components Controller (1) and Model (4). The View is not shown separately for simplification and is used in the end user's web browser in particular. The model has access to a database component, which provides both a database for application-internal data and a cache, which is used for the technical realization of a fast response time for user requests. The decentralized nature of TrADS therefore does not slow down the interaction between the user and the TrADS instance. Apart from the classic database connection of the model, it has access to a Solid Client (3), which enables TrADS to access Solid pods. The newly introduced Trust Awareness component (2) is used in TrADS by the model component to neither blindly accept nor directly reject external content originating from the Solid pods. This means that externally sourced content is evaluated in a trust aware manner before it is displayed to the user. This type of integration as a local component on the individual instance rules out manipulation by third parties and thus makes the TrADS instance independent about whether decentralized data is trustworthy enough. This also makes it possible to display the assessments and to allow future assessments to be influenced by the user. The trust awareness result can vary from instance to instance, whether due

to a different data situation, the use of a different trust model in the component or different preferences of the instance users and operators.

To ensure that users can assume the integrity and origin of the displayed data despite the decentralized storage, TrADS uses checksums for referencing and offers optional content signing. For example, when commenting on a post and thus referencing the post, a user cannot be sure that the post will still be available (unchanged) later. The use of checksums when referencing content ensures that it is possible to check later whether the content that can now be accessed is identical to the content referenced in the initial post. For this purpose, the checksum associated with the content is also specified together with the reference to the content. Users can thus check whether the referenced content still exists in exactly this form. Otherwise, the content was referenced incorrectly or the referenced content has since been manipulated. The optional signing of content makes it possible to ensure that only the displayed content creator has published the currently saved content if a unique public key can be assigned to him. If a post creator later decides to remove a post or restrict access to it, partial discussions could exist in TrADS as the replies to the post content are in turn controlled by their creators.

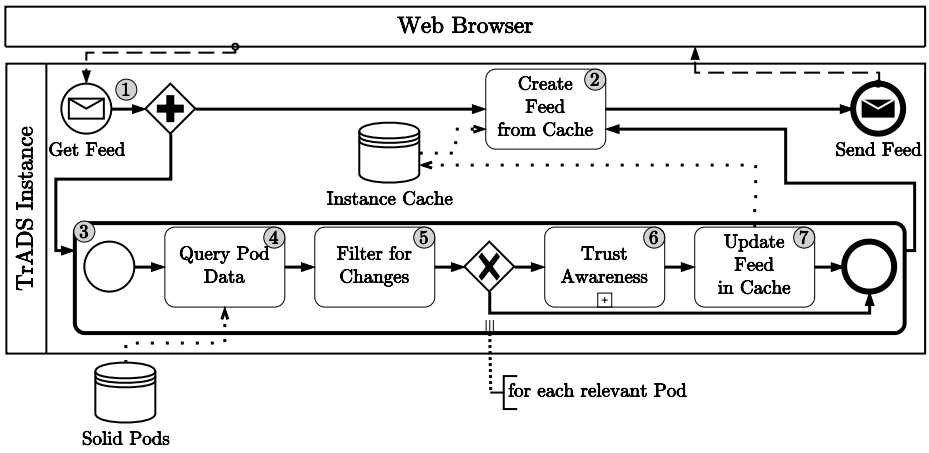


Fig. 2: Getting a User's Feed in TrADS

Trust Awareness is always requested as soon as new data from any Solid pod is included in a TrADS instance. For technical reasons, such inclusion often results in caching at the instance. An example of this is shown in figure 2 using the query of a user's feed. The process begins with the user's request to retrieve their feed (1). To ensure that the request can be answered as quickly as possible, a quick initial response is sent to the user after the request (2) and a query is sent to all relevant Solid pods to update the data (3). The fast response to the user is made possible by querying the instance cache and thus shortens the response

time for the user instead of awaiting all included pods. The data update subprocess is executed for each relevant pod and begins with the data query (4). The retrieved data is then filtered for changes (5) to determine whether and what should be updated in the instance cache. If an update of the cache is necessary, the trust awareness component (6) will determine the extent to which the data is trustworthy before the update is executed (7). TrADS does not detail anything about the metrics on the basis of which the trust-aware decision is made. In this paper, the component of trust awareness is intentionally only implemented as a prototype so that the architectural integration and its effects in particular can be both presented and examined. Frameworks such as ConTED [15] or other trust models migrated to the decentralized web [17] can be used to implement the trust awareness component use case independently.

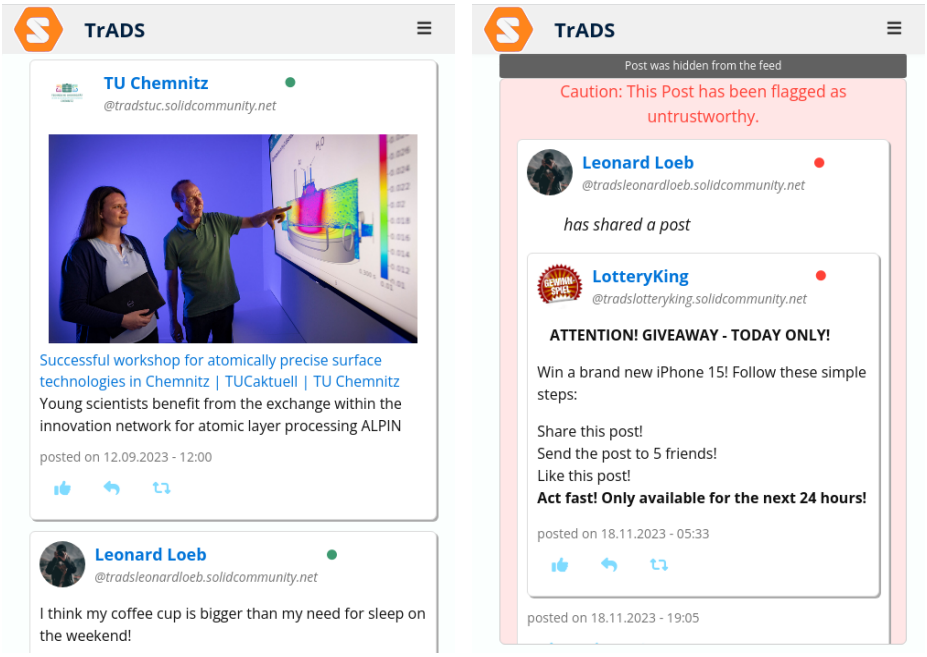
3.2 Prototype

A demo[5] of TrADS as well as the implementation code[6] are available online. TrADS offers users functions that they are already familiar with from other social networks. When called up, the feed is displayed, which consists of posts and reactions from related pods. These are either friends or any other social network account the user follows. Figure 3a shows the structure of posts in the TrADS feed, as well as the menu. For a post, the name of the author, their profile picture, the date the post was published and the number of likes received is displayed together with typical options to share and respond to it. Additional functions in TrADS are typical of social networks, such as adding new friends or following accounts, as well as viewing profiles and editing your own profile. Furthermore, a chat functionality is also implemented in TrADS.

A trust rating of the individual posts is displayed in the feed. A colored circle next to the user's name indicates the trustworthiness of the respective post. The meaning of the color is explained with the help of a tooltip as soon user hoovers over the circle. The colors of the circle stand thereby for: ● untrustworthy, ● very likely untrustworthy, ● maybe untrustworthy, ● neutral, ● trustworthy, ● very trustworthy.

Posts that are classified as untrustworthy by TrADS are hidden in the feed. However, an indication appears at the location of the post writing that the post has been hidden from the feed. Users have the option to click on this indication to view the distrusted post with a corresponding warning. Figure 3b shows how this post is displayed after the user decides to view it despite poor trustworthiness.

As a prototypical implementation, TrADS does not use a fully-fledged trust model in its trust awareness, but simple functions that serve as placeholders for a future implementation. Trust values are initialized for posts by retrieving the author's trust value from the cache and modifying it with a random-based variance. The trust value of the authors is initialized with a neutral value and also provided with a random variance for scattering. Likes of a post increase the trust value of the post if the trust value of the like creator is greater than that of the post. Sent chat messages also lead to an increase in the trust value of the recipient of the message.



(a) Examplary TrADS Feed, including two posts

(b) Expanded untrustworthiness warning, showing the post within a TrADS feed

Fig. 3: TrADS Front-end Examples

The cache enables the trust awareness to use stored content as the basis for the trust evaluation. In addition, the evaluation of individual content can also trigger a change in the evaluation of other content already stored in the cache. A strongly negative evaluation of a content or author does not lead to its rejection in TrADS, but only to the corresponding indication and hiding from the feed. This gives users control over the content they consume, and additionally TrADS' trust awareness can make use of such negatively rated content as a good source of information for later trust evaluations.

3.3 TrADS Qualitative Comparison

TrADS is a very good approach for a decentralized social network with regard to the requirements mentioned in section 2. The **Trust awareness** introduction into decentralized web applications is the main contribution of the paper. It is realized by TrADS as a component that assesses the trustworthiness of all external data from Solid pods. Even though TrADS does not implement a fully functional trust model, the architecture enables the necessary integration of further work. Most importantly, TrADS is the first to use trust awareness for external decentralized data. The use of Solid provides a high level of data control in TrADS.

This includes control of data access with the ability to revoke permission, as well as control over where the data is stored. Thanks to the free choice of instances, the ability to self-host an instance and direct user access to their pods, TrADS offers complete **resistance to censorship**. Also, **networkability**, **independence**, **customizability** and **manageability** are all met by using the instance concept, decentralized storage of data in Solid pods and web standards used in Solid, such as WebID. With projects such as ActivityPods[20], it is even possible to connect to related approaches from the Fediverse with TrADS. However, scalability is only mostly satisfied, as an instance scales linearly, but the network of instances scales quadratically in the worst case due to the caching of a TrADS instance.

4 User Study

To be able to draw conclusions from the existing prototype of TrADS about the influence of the trust awareness component on the user experience, we conducted a user study on the front-end of TrADS. In this section, we present the design and process of the study and then discuss its results and the implications of the user study. All data of the evaluation is accessible online[21], including the raw survey data as well as all numbers we discuss in the results subsection.

4.1 Procedure

The user study was conducted in German using a tool that made it possible to embed TrADS directly into the survey using IFrames. After a welcome text and the collection of general information about the participant, each participant was given time to explore TrADS independently. Each participant was then asked to first read the latest post from a local newspaper, then like a post from a specific account and finally write their own post. In this way, it was initiated that the participants interacted with TrADS. Although it was not checked to what extent the participants completed the tasks.

After the interaction with TrADS, the usability of the demo implementation was determined using the System Usability Score (SUS) [2]. This uses 10 statements, which are evaluated by the user on a 5-point Likert scale and then systematically combined to form a score. In order to look specifically at the use of the Trust Awareness component, 4 questions are asked as to whether and to what extent the participant was aware of Trust Awareness in the front-end, always, sometimes or not at all. However, 2 of these questions are only displayed if the participant indicated more than not at all in the previous always-indicating question. For example, the participant is only asked whether the different colors of the trust circles are perceived if they have indicated that they always or sometimes see the circles. The participant then rated 9 statements on the integration

[20] `https://activitypods.org/`
[21] data available at `https://zenodo.org/records/10641724`

of trust awareness into a social network, again on a 5-point Likert scale. In this way, the general acceptance of a trust awareness component can be examined. The survey ends with 7 statements on the importance of data protection in social networks, which the participant again rates on a 5-point Likert scale.

For the survey, TrADS was put into demo mode to ensure that no persistent changes can be made to the instance and that all users are shown an identical version. To this end, user management is deactivated, preventing the addition and deletion of users in the cache of the TrADS instance. Interactions, such as creating or liking posts, are only stored in the cache and not in the corresponding Solid pods. All new content stored in the cache is automatically deleted every 30 minutes. Furthermore, personal posts are not displayed in the feed, as the participants have no connection to the demo user.

To bring the social network to life for the demo, 25 pods are created on a separate Solid pod provider for user representation. There are 5 profiles of companies that fill the network with news. News articles only contain links to the corresponding articles, which can be read by means of a preview. The topics of the articles were chosen from the areas of entertainment, sport and animals in order to avoid socially controversial topics as far as possible. Three other users are used in the demo for fraudulent content, which should not be trusted and should therefore be visualized accordingly by the front-end. The remaining 17 users are private users of the network who have been assigned different trust values after creating posts in order to have different users.

4.2 Results

Only the 53 fully completed responses were taken into account for the evaluation[21], of which 26 were female, 19 male and 8 without gender information. According to the participants, the age distribution was as follows: under 18: 1; 18 to 24: 4; 25 to 34: 9; 35 to 50: 27; 51 to 70: 11 and one participant over 70. While 40 people stated that they use social networks several times a week, one participant stated that they use them weekly, three use them less than once a week and two participants do not use them at all.

27 of the 53 participants stated that they had noticed the circles indicating trustworthiness next to the names of the authors of the posts. More than half of this group of participants noticed that these dots had different colors. 26 of the participants stated that they had noticed the indications on untrustworthy feed removals, yet only two participants tried to click on these. In total, 36 of the 53 participants noticed at least one of the two trust awareness front-end elements, of which 17 even noticed both elements. However, there were also 17 participants who did not notice either of the two trust awareness front-end elements.

Figure 4 shows the SUS distribution of the survey as a boxplot (1). The figure also contains the boxplots for two different user groups. The boxplot (2) represents the group of users who noticed at least one of the two trust awareness front-end elements, while the boxplot (3) represents the group of users who did not notice any of these elements. The SUS of the survey have an expected value of $\mu = 68$ with a standard deviation of $\sigma = 17.5$. According to SUS, such an

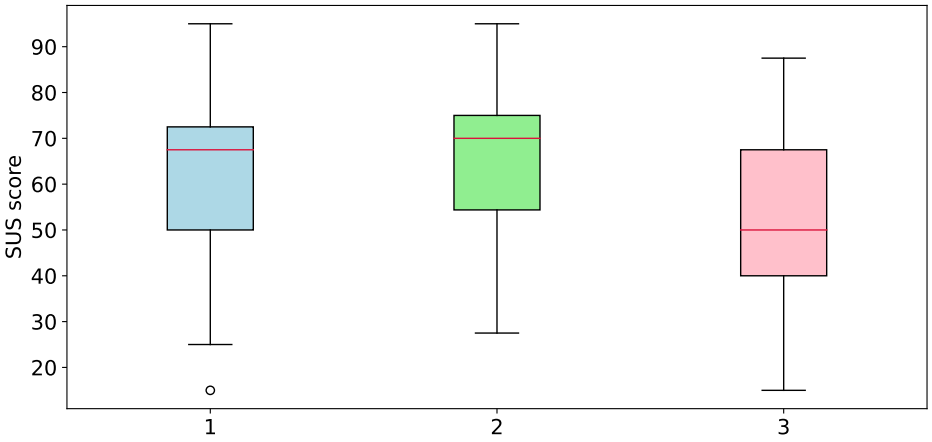


Fig. 4: SUS Distribution of (1) all participants (2) participants noticed in minimum one trust awareness front-end element (3) participants did not notice any trust awareness front-end element

expected value is to be classified as an average score. Using the Shapiro-Wilk test, we determined a p-value of 0.234 for the survey's SUS data points. As the p-value is below the significance level of $\alpha = 0.05$, a normal distribution of survey's SUS can be assumed.

To investigate the extent to which the participants' perception of trust awareness of the application influences the usability of the system, we examined the box plots (2) and (3) in a two-sample t-test. Based on a significance level of $\alpha = 0.05$ and 51 degrees of freedom, the critical value for our t-test is 2.01. This results with $(95\% - CI[1.06, 21.23])$ in $t(51) = 2.22$, $p = 0.03$. Thus, a significant difference exists in the SUS distribution of end users noticing trust awareness front end elements and those who did not. The significant differences in the usability scores indicate an improvement in the user experience as a result of trust awareness. However, it should be noted that the groups were not randomly selected, which limits the reliability of the t-test.

The reaminding survey statements provide further insights on 5-point Likert scales with values from 1 to 5: The presentation of an assessment of the trust awareness of the content was rated as useful ($\mu = 4.25$, $\sigma = 0.75$) and helpful ($\mu = 4.11$, $\sigma = 0.86$). It was further agreed ($\mu = 3.89$, $\sigma = 0.92$), that a social network in which this additional information is provided is more likely to be used than another. The results of the statements on dealing with untrustworthy posts show a greater discussion among the participants with $3 < \mu < 4$ and $\sigma > 1$. The protection of personal data ($\mu = 3.3$, $\sigma = 0.88$) and the knowledge of what data is stored about oneself ($\mu = 3.13$, $\sigma = 0.99$) is rather important for our participants. Additionally, most generally do not know where their personal data is stored ($\mu = 2.4$, $\sigma = 1.15$) or who has access to it ($\mu = 2.45$, $\sigma = 1.19$).

5 Conclusion

In this paper, we present a trust awareness framework called TrADS for decentralized social networks. TrADS leverages the MVC pattern [7] of web applications to integrate external data from Solid pods [13] into decentralized web applications in a trust aware manner. We evaluate the proposed trust-aware decentralized social network TrADS in terms of usability and transparency with 53 end users in an empirical study. The results indicate that the trust awareness component in TrADS can support end users in their trustworthy user experience. Furthermore, user study participants agree that a social network improves if it provides information on the trustworthiness of data. The results were less clear when it came to dealing with content that was classified as untrustworthy, including opposing opinions. Contradictory results were obtained from the survey on participants' attitudes towards the handling of their data on social networks. On the one hand, privacy and data control is important to the participants, but on the other, the majority of them stated that they do not know what happens to their data. The qualitative comparison based on 8 requirements of related work of decentralized social networks with TrADS shows that the use of Solid and its web standards makes TrADS an above-average solution. Solid combines the advantages of other approaches such as the Fediverse, blockchain or peer-to-peer (P2P) with regard to the requirements mentioned. Only the scalability of P2P approaches is slightly better with regard to all instances of a decentralized social network. In particular, however, TrADS introduces trust awareness, which is not present in any of the existent approaches. The Fediverse already recognizes the problems of malicious or harmful data, but only solves this with manual moderation, which leads subsequent problems in larger networks [1].

In terms of future work, we suggest further research to determine the generalizability of this work. This includes the back-end use of the trust awareness component of decentralized web applications, as well as the extent to which the results of the component are used in the front-end to improve the user experience. Further targeted investigations are required for the generalization of trust awareness functionality in decentralized web applications. Trust awareness should not have to be developed by the developers themselves for each application or set up at great expense. It will only be widely used in many web applications if the set-up effort is limited, e.g. similar to the effort required today to deliver a website with end-to-end encryption via TLS. The use case of social networks used in this paper is a special one in terms of data access by end users, as they create content themselves, which is then used by others for interaction. In other web applications, end users are often only consumers of a service. The integration of external data is then often motivated by suppliers of resources for the service of the web application. Such a change in the use case can also change the impact on the user experience than TrADS does in a very transparent way with the visible front-end elements without removing content fully. Such a change to the use case can, for example, result in the application having to intervene more strongly than TrADS does in the case of untrustworthy data.

References

1. Anaobi, I.H., Raman, A., Castro, I., et al.: Will Admins Cope? Decentralized Moderation in the Fediverse. In: Proceedings of WWW '23. p. 3109–3120 (2023). https://doi.org/10.1145/3543507.3583487
2. Brooke, J.: SUS: A 'Quick and Dirty' Usability Scale. Usability evaluation in industry **189**(3), 189–194 (1996)
3. Couture, S., Toupin, S.: What does the notion of "sovereignty" mean when referring to the digital? New Media & Society **21**(10), 2305–2322 (2019). https://doi.org/10.1177/1461444819865984
4. Guidi, B.: When Blockchain meets Online Social Networks. Pervasive and Mobile Computing **62**, 101131 (2020). https://doi.org/10.1016/j.pmcj.2020.101131
5. Ibáñez, L.D., Simperl, E., Gandon, F., Story, H.: Redecentralizing the web with distributed ledgers. IEEE Intelligent Systems **32**(1), 92–95 (jan 2017). https://doi.org/10.1109/MIS.2017.18
6. La Cava, L., Greco, S., Tagarelli, A.: Understanding the growth of the Fediverse through the lens of Mastodon. Applied Network Science **6**(1), 64 (Sep 2021). https://doi.org/10.1007/s41109-021-00392-5
7. Leff, A., Rayfield, J.T.: Web-application development using the Model/View/Controller design pattern. In: Proceedings Fifth IEEE EDOC. pp. 118–127 (2001). https://doi.org/10.1109/EDOC.2001.950428
8. Lemmer-Webber, C., Tallon, J., Shepherd, E., Guy, A., Prodromou, E.: Activitypub. first edition of a recommendation, W3C (Jan 2018), `https://www.w3.org/TR/2018/REC-activitypub-20180123/`
9. Li, X., Xue, Y.: A survey on server-side approaches to securing web applications. ACM Comput. Surv. **46**(4) (mar 2014). https://doi.org/10.1145/2541315
10. Mastodon gGmbH: Mastodon Annual Report 2022 (2023), `https://joinmastodon.org/reports/Mastodon%20Annual%20Report%202022.pdf`
11. Meckler, S., Dorsch, R., Henselmann, D., Harth, A.: The Web and Linked Data as a Solid Foundation for Dataspaces. In: Companion Proceedings of WWW '23. p. 1440–1446 (2023). https://doi.org/10.1145/3543873.3587616
12. Protocol Labs: IPFS Standards, `https://specs.ipfs.tech/`
13. Sambra, A.V., Mansour, E., Hawke, S., et al.: Solid: A Platform for Decentralized Social Applications Based on Linked Data. Tech. rep., MIT CSAIL & Qatar Computing Research Institute (2016), `http://emansour.com/research/lusail/solid_protocols.pdf`
14. Siegert, V., Gaedke, M.: Trust Awareness for Redecentralized Web Applications (Position Paper). In: Joint Proceedings of ESWC 2023 Workshops and Tutorials (2023), `https://ceur-ws.org/Vol-3443/ESWC_2023_TrusDeKW_paper_7938.pdf`
15. Siegert, V., Kirchhoff, A., Gaedke, M.: ConTED: Towards Content Trust for the Decentralized Web. In: Proceedings of WI-IAT 2022. pp. 604–611 (2022). https://doi.org/10.1109/WI-IAT55865.2022.00095
16. Taylor, P.: Volume of Data/Information Created, Captured, Copied, and Consumed Worldwide from 2010 to 2020, with Forecasts from 2021 to 2025 (nov 2023), `https://www.statista.com/statistics/871513/worldwide-data-created/`
17. Yu, H., Shen, Z., Leung, C., Miao, C., Lesser, V.R.: A Survey of Multi-Agent Trust Management Systems. IEEE Access **1**, 35–50 (2013). https://doi.org/10.1109/ACCESS.2013.2259892