TrADS: a Trust-Aware Decentralized Social Network

Valentin Siegert✉, Dirk Leichsenring, Alejandro Wurts Santos, and Martin Gaedke

Chemnitz University of Technology, Chemnitz Germany
{valentin.siegert, dirk.leichsenring, alejandro-wurts.santos, martin.gaedke}@informatik.tu-chemnitz.de — orcid.org/0000-0001-5763-8265, 0009-0008-6354-0019, 0009-0002-9434-2740, 0000-0002-6729-2912

Abstract

In today's data-driven world, people are increasingly prioritising data privacy and control. This awareness has sparked an initiative for a decentralised web, where web applications no longer rely on centralised data storage. Solid, a prominent approach for a decentralized web, allows users to store their data in decentralised pods of their control. However, the integration of data from various and potentially untrusted sources can lead to malicious or harmful results and impair user experience. To solve this problem for social networks, we propose a trusted and decentralised web application called TrADS. It utilises the MVC pattern to integrate external data from Solid pods based on trust evaluations. In this article, we extend our first paper on TrADS with further details on related work assessment, concept and implementation. We are also extending our evaluation to a second international user study with another 64 participants. We measure the user experience instead of pure usability and form two separate groups of participants, one of which experiences the trust awareness of TrADS and one of which does not. The results do not yet show significant improvement in the user experience but show that after using trust awareness in a social network, users favour a network with such features.

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

River Journal, 1–32.

1 Introduction

The web became a collection of closed data silos due to commercialization [10]. Recently, data sovereignty has become increasingly important [4], as the original data owners lose full control over their data in closed data silos. This loss of control restricts user privacy, as it is difficult to completely delete data from such silos and it cannot be moved one-to-one to another web application. For this reason, various projects and actors are working on the decentralization of the web, with both political and technical motivation. To improve data privacy and sovereignty, the EU is driving forward the decentralization of the web due to political reasons. The Next Generation Internet Initiative[1], the project Gaia-X[2] and also the General Data Protection Regulation (GDPR)[3] are some examples of the EU's activities towards a decentralized web. Since the web was originally designed as a decentralized system [2], the change towards closed data silos brought up various technical projects to decentralize the web again. The most-known projects are Solid [20] and Fediverse applications like Mastodon[4]. Solid enables a decentralization of data over the web by allowing data owners to keep control of their data due to storing it within its pod structure. In contrast, the Fediverse is a set of applications that are designed to be hosted as many instances and are able to synchronize content between these instances as well as between instances of different applications.

Decentralized web applications obtain data not only from their own data spaces, but also decentrally, e.g. from decentralized knowledge graphs, Solid pods or solid data spaces [19]. The decentralized data is accessed dynamically at runtime with the help of semantic web technologies [22, 24] Therefore, the user's data can be stored anywhere on the web and the user can retain control over the access management of the data and its content. In this way, a decentralized web application accesses data in a decentralized manner as well as its own data spaces, as application-internal data is most likely still kept internally [22]. In addition, caches of decentralized data within decentralized web applications are required to enable the fast response times generally expected by web users today.

[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/`

Similar to the expected fast response times, users will only use a web application that offers them a high level of user experience [21]. This applies equally to decentralized web applications. However, obtaining decentralized data means that a web application relies on externally stored data. Since external data can also be malicious or harmful, the user experience can be deteriorated by compromised external data. This poses a major challenge for decentralized web applications to improve privacy and data control for users [22]. To reduce the risk of compromise, people rely on information that they can trust in their subjective opinion. Similarly, a web application could filter out malicious or harmful data once it is aware of trust about web data. The web application would thus counteract a deterioration in the user experience. Classic centralized web applications achieve the trustworthiness of the data they use by completely controlling their data space and only inserting data that is known to the operators or is validated user input [16]. Decentralized web applications no longer have complete control over their data space. Data that has already been used once can be changed externally without the application controlling or knowing about the change before it uses the data again. Since the amount of web data is constantly growing [25], it is impossible to moderate the external data obtaining by experts [1]. Instead, a trust awareness component is needed [22], which assesses the extent to which data is trustworthy enough to be used by a decentralized web application.

There is not yet a component for trustworthiness for decentralized web applications. Related work shows that trustworthiness can be calculated differently depending on which trust model is used [23, 29]. A trust model calculates trust depending on various factors, e.g. based on the origin of the data, its reputation, popularity and more. Based on these calculations, a trust-based decision can be made as to what extent the data is trustworthy enough to be used. However, it is not yet defined how the interfaces of such a trust awareness component are structured, which processes within a web application require such a component and how often such a component needs to be used within a typical web application workflow. In order to clarify these questions and to enable trust awareness for decentralized web applications, we describe in this article a framework that enables trust awareness for a decentralized social network. We extend our initial work on our framework presented at ICWE 2024 [24], based on the comments and feedback from reviewers during the conference, in the following areas:

1. We have extended the published related work with a detailed discussion of the requirements and the analysis of how the related work fulfills the requirements.
2. The concept of our application TrADS, a **Tr**ust-**A**ware **D**ecentralized **S**ocial network that has a Solid-based decentralized data layer, is extended with further processes of a social network.
3. The prototypical implementation of TrADS is detailed in terms of how data are stored within the pods and how the trust awareness component is implemented. Its code[5] and demo[6] is available online.
4. We have extended the originally published evaluation with a second user study[7] conducted in German and English to internationalize the first published results including 64 new participants. Instead of evaluating usability, we assess user experience. In addition, the new study splits the participants into two test groups to compare TrADS with the trust awareness component activated and deactivated.

The rest of the article begins in section 2 with an extended state of the art of decentralized social networks. Section 3 describes our framework for a trust-aware, decentralized, and Solid-based social network called TrADS including architecture and the online available prototype. In section 4, we present an extended user study that builds on the existing study presented at ICWE 2024 and compare both study results. Finally, we draw conclusions in section 5.

2 State of the Art

In this section, we present a detailed analysis of the state of the art, including a qualitative comparison between various decentralized social networks that already exist today. For this purpose, we first address the requirements of a decentralized social network in subsection 2.1. In subsection 2.2, we present existing decentralized social networks and subsequently analyze the extent to which related work meets the requirements. In subsection 2.3, we summarize the results of the state of the art analysis and draw conclusions.

[5] code: `https://zenodo.org/records/10641771`
[6] demo: `https://vsr.informatik.tu-chemnitz.de/projects/2024/trads/`
[7] evaluation data: `https://zenodo.org/records/14199062`

2.1 Requirements

In order to comprehensively examine the existing research, we identified two different groups of requirements: the requirements of the end users of the social network and the requirements of its operators. The requirements based on these two roles are presented and explained, whereby the roles are defined first. All requirements are mapped to a four-level evaluation scheme, whereby not all requirements include all levels: ○ not satisfied, ◔ partially satisfied, ◕ mostly satisfied, ● fully satisfied.

2.1.1 End User Requirements

Whether or not a system is actually used on the market depends to a large extent on whether the end user's requirements are met. End users have an interest in using decentralized social networks freely and independently and at the same time protecting their privacy and personal data. They expect to be able to use the network to communicate with their friends and family, discover new content and use the social network with ease. In addition, they are interested in being able to use the decentralized social network in the same way as a conventional social network and therefore expect a similar range of functions. End users usually use exactly one access point to the network, i.e. an instance provided by an operator. It is also possible for end users to be the operator of their own instance at the same time. However, this role duality has no effect on the separation of requirements.

Trust awareness. With the large amount of content that users consume every day, it is almost impossible for operators to determine the exact origin of all data in order to evaluate it in terms of trustworthiness and reject it if necessary [1]. Nevertheless, the trustworthiness of the decentralized content should be checked by the application before it is presented to end users. The requirement is assessed to be:

● fully satisfied if content is always checked for trustworthiness in a comprehensible way for end users.

◔ partially satisfied if instances check content for trustworthiness, however this check might be optional and not comprehensible to the end users.

○ not satisfied if no content is checked for trustworthiness.

Data control. With centralized social networks, users have only limited control over their data and are largely dependent on the operator. Depending on the architecture of a decentralized social network, control over the data may lie with the operators or with the end users themselves. End users should always have full and sole control over who can access their data. In addition,

they should be able to independently and completely remove their own data from the social network. They should also have control over the location of data storage. Thus, it should be possible to change the storage location without affecting their own presence on the social network. The requirement is assessed to be:
● fully satisfied if end users have full and sole control over access and storage location of their data.
◕ mostly satisfied if end users have control over data access, but not over where the data is stored.
◔ partially satisfied if the end users have no influence on data storage and only indirect control over access to the data.
○ not satisfied if end users have no control over the data after publication.

Resistance to censorship. A social network should enable a free exchange of opinions within the legal framework. It should not be possible for operators to delete content created by users within their data. The non-display of content through so-called shadowbans has already led to political discussions in the past[8]. The power to decide which content is displayed to a user should be under the control of the respective user, provided that legal framework conditions are not undermined in the process. The requirement is assessed to be:
● fully satisfied if it is technically impossible for third parties to influence the content of the network within the scope of censorship.
◔ partially satisfied if content could be deleted or fully blocked by third parties, however only in a limited amount.
○ not satisfied if operators and third parties have full control on created and shown content.

Networkability. Due to the multitude of different social network services, end users require to use the same platform if they want to stay in contact with each other. To avoid having to actively use multiple services, a selected platform should offer the option of interacting with other platforms. Networkability is limited to other platforms that offer corresponding interfaces or use common standards. Therefore, this requirement measures the degree of possible communication beyond the user's social network. The requirement is assessed to be:
● fully satisfied if networkability with other social networks is explicitly provided for.

[8] `https://www.nytimes.com/2018/07/26/us/politics/twitter-shadowbanning.html`

◕ mostly satisfied if networkability is not explicitly provided for, but possible in principle.
◔ partially satisfied if networkability is not explicitly provided for and difficult to implement.
○ not satisfied if networkability with other social networks is impossible.
Independence. When using traditional social networks, end users are dependent on the operator. They can only use the service provided as intended by the operator. Even in a decentralized social network, the influence of third parties, e.g. an operator, on one's user experience cannot be completely ruled out. End users may be restricted in their communication to specific instances or be bound by terms and conditions of use. The ability to change the instance used for a social network allows end users to escape this influence. However, they should still be able to use their existing presence on the social network without any restrictions. The requirement is assessed to be:
● fully satisfied if there are no dependencies on third parties.
◔ partially satisfied if only the appearance in the social network depends on external authority, e.g. linked to the domain.
○ not satisfied if there is a dependency on the operator of the instance.

2.1.2 Operator Requirements

The individual instances of a social network are set up and maintained by operators. Depending on the architecture of the network, operators can decide on the storage of end user data, communication with other instances, the range of functions offered and the control of published content. An instance can be used by a large number of end users or just one. The operator requirements are non-functional and are primarily related to their workload.
Adaptability. There are many different types of social networks, such as microblogging services, discussion platforms or multimedia platforms. These use cases also place different demands on the underlying architecture. A generic social network that can be used for any use case with few modifications is desirable from the operator's point of view in order to be able to react to the respective needs of the users. The requirement is assessed to be:
● fully satisfied if the social network can be adapted to a wide variety of use cases with little effort.
◔ partially satisfied if customization is possible, but involves considerable effort.
○ not satisfied if the use for different use cases is not possible.
Manageability. When operating an instance, there are recurring tasks to maintain the function of the service. Such tasks may, for example, be related

to content moderation or managing which other instances can be interacted with. Operators are interested in keeping the recurring effort and unavoidable manual interactions as low as possible. Manageability evaluates the maintenance effort of an instance, regardless of the technical resources required. In addition, the consequences that insufficient maintenance of the instance can have on functionality are also considered. The requirement is assessed to be:
● fully satisfied if no recurring tasks are required to maintain operation.
◔ partially satisfied if there are moderate administrative costs that have consequences if neglected.
○ not satisfied if there is considerable effort required for operation.
Scalability. Both the increasing number of users of an instance and the growth of the decentralized social network through the addition of new instances must be feasible. The need for resources such as storage space, computing power or network capacity plays a key role here. Hereby, the need for resources such as storage space, computing power or network capacity plays a key role. Scalability also plays a role for special user groups such as influencers, public officials or commercial users. Their appearances on a social network are often much more extensive and are followed by a large number of users. An architectural limitation on the size of the decentralized social network would hinder its general usability. The requirement is assessed to be:
● fully satisfied if the demand for resources scales linearly in the network.
◕ mostly satisfied if the demand for resources for the individual instances of the network scales linearly.
◔ partially satisfied if there is an exponential increase in the resources required.
○ not satisfied if the decentralized social network cannot be scaled at will.

2.2 Related Work

The most successful decentralized social networks at the moment are those from the Fediverse [12]. The Fediverse is mainly based on federated networks, each of which is a combination of communicating instances. The distribution of the network to a large number of independently operating instances ensures the distribution of control over the network. Instances can be operated privately or publicly with a variable number of end users, who are free to decide which instance they use. In this article, we take a closer look

at four federated networks, including diaspora* [9], PeerTube[10], PixelFeed[11] and Mastodon[4], which is one of the most successful federated networks with 1.8 million active users in 2022 [17]. Communication between instances, as well as between clients and individual instances, is mainly realized with ActivityPub [15]. In a system that uses ActivityPub, each user has an inbox through which they can receive information and an outbox whose content can be accessed externally. Other notable communication protocols in the Fediverse are OStatus [27] and diaspora* federation protocol[12]. Since each approach participating in the Fediverse must first and foremost provide a suitable interface for networkability, the Fediverse is not limited to federated networks based on the multiple instance approach.

In terms of the requirements we have identified, the four federated networks mentioned are the same. All's trust awareness about the content created and displayed is the responsibility of the individual instance. Since end users are dependent on the instance used, full data control is only possible by operating an own instance. Identities are tied to an instance and only movable to another if the instances cooperate [18]. Censorship of content by the individual instances is only possible within an instance, but not globally. Networkability is one of the basic ideas of the Fediverse, and is realized with the mentioned protocols like ActivityPub [15]. Federated networks are suitable for all types of social networks and can be adapted without restriction. Manageability is low, as the operator is responsible for controlling the content and managing the instances to be communicated with. The operator of an instance is responsible for controlling the content and managing the instances that can be communicated with. Both of these tasks can involve considerable effort and, if neglected, can cause problems for the users of the instance. This is an unsolved problem, especially for larger instances, which can only be partially solved by automation [1]. The growth of such networks can be achieved by increasing the number of end users of an instance and by increasing the number of instances. The resource requirement increases linearly with local growth, but overlinearly in the entire federated network due to intermediate storage on several instances.

Some decentralized social networks are based on the peer-to-peer model (P2P), in which all participants have equal rights to provide and receive services. In this article, we take a closer look at three P2P-based social networks,

[9] `https://diasporafoundation.org/`
[10] `https://joinpeertube.org/`
[11] `https://pixelfed.org/`
[12] `https://diaspora.github.io/diaspora_federation/`

including Secure Scuttlebutt[13], Sone[14], and Peergos[15]. Secure Scuttlebutt users host their own node, which stores their own data as well as the data of all their friends and their friends' friends. The reach and view of users is therefore primarily limited to their respective social environment, but can be expanded through publicly accessible nodes. To be part of Sone, end users require a Freenet[16] access point. The data is stored in encrypted form on various nodes, while the location is determined by a routing algorithm. In order not to be disadvantaged by the strong data distribution in terms of access rights, all content in Sone is public and is accessed anonymously and indirectly. Peergos end users have the choice of using an existing node or hosting one themselves. Once they follow other users, they can write messages and exchange data unilaterally.

None of the P2P-based implementations presented offers trust awareness for the data displayed. If the data is only stored on a self-operated node, the user may retains full data control like in Peergos. Censorship is indirectly possible by blacklisting other nodes by operators, but there is no direct influence on other nodes. However, due to the enforced encryption of content in Sone, this influence through blacklisting is not possible. P2P makes networking with other decentralized networks more difficult, as the provision of a suitable communication interface is missing and has not yet been taken into account in the approaches presented. Independence from third parties in P2P-based approaches is possible, as a presence does not necessarily depend on specific nodes of the network. The adaptability of a P2P-based decentralized social network is unrestricted. Node operators have no manual effort, as they merely act as distributed data storage. Scalability depends on the implementation, but P2P networks scale very well in principle. Finding the desired resources in the network is also theoretically always possible in $O(n^2 \log n)$ due to the *small-world phenomenon* [11].

Well-known representatives of blockchain-based decentralized social networks are SteemIt[17] and Minds[18] [6]. The focus of these approaches is on resistance to censorship and the monetization of content. The use of a blockchain ensures freedom from censorship, as the blocks created cannot be subsequently manipulated. Content can be monetized through the use of one

[13] `https://www.scuttlebutt.nz/`
[14] `https://github.com/Bombe/Sone`
[15] `https://peergos.org/`
[16] `https://www.hyphanet.org/`
[17] `https://steemit.com/`
[18] `https://www.minds.com/`

or more cryptocurrencies [9]. The blockchain serves as a record of all actions in such decentralized networks. Access to content is usually realized via the P2P-based data distribution protocol IPFS [13]. The blockchain approaches do not include any trust awareness about data displayed. Although the content in such decentralized social networks is due to the blockchain resistant to censorship, users do not have full control over their data, which also severely restricts adaptability. There are no dependencies on operators in a blockchain-based decentralized social network, as all nodes work independently of each other. Networkability is not planned and would be a scaling problem, as all interactions from other networks would also have to be included in the respective blockchains. 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 [6] like scalability, content visibility and decentralization of content. Although the number of interactions increases linearly, it cannot be distributed, as each node must update the ledger independently. Thus, the size of the network is limited by the amount of data that can be processed by the nodes.

With Movim[19] there is also an XMPP-based decentralized social network. The Movim instances provide an XMPP client that offers the functionalities of the social network. Therefore, it does not matter which XMPP servers are used by Movim users. All personal data, including content, is stored on the end user's preferred XMPP server, so that a change of Movim instance is possible at any time. Content is cached on Movim instances to make created public content available via HTTP. The structure of Movim is similar to federated networks, but communication takes place between XMPP servers instead of between instances. There is no provision for checking the content for trustworthiness or for networking with others. Full user control over their data is only possible by operating both Movim instance and XMPP servers. Censorship is only possible to a limited extent, as operators can, for example, prevent communication with certain users. The user is not tied to one instance when using it, but is dependent on the XMPP server used. Movim is largely adaptable to all desired types of social networks and free of administrative effort. Since the XMPP server only contain the data of their own users, the Movim network can scale linearly.

In decentralized social networks, there are also minimalist approaches such as twtxt[20]. The entire network consists of users publishing a twtxt file

[19] `https://movim.eu/`

[20] `https://twtxt.readthedocs.io/en/latest/user/intro.html`

via URL, which serves as their identity. The twtxt client uses the twtxt file of all followed users to create a microblogging feed. There are also extensions to twtxt such as Yarn.social[21], which extends twtxt with hashtags and metadata from social networks. Because of the direct access, there is no trust awareness with twtxt. In the case of independent hosting, users can control their already published content, but without any access control. As the control over the published data lies with the respective users and other users have direct access to it, censorship of the content is not possible and independence from third parties is guaranteed. Networking with other systems is not planned and not easily possible. The simple structure of twtxt makes it easy to operate an instance or to host a twtxt file directly, which leads to a high level of manageability. There is linear scalability, as the number of accesses to the file increases linearly with the number of followers.

There are also approaches based on relay systems such as Notes and Other Stuff Transmitted by Relay (nostr)[22] in decentralized social networks. Its aim is to build a globally independent and censorship-resistant social network. It consists of a large number of relays with which users can communicate via a client. A user sends his signed content to be published to a selection of relays for publication. Content is published after relays have accepted it and other users request it. There is no way to notify other users about news in nostr, as everything is call-based. No trust awareness is provided in nostr, as it relies solely on the relay system. In addition, end user lose all control over data after sending it to relays. Networkability with other networks is not planned. Resistance to censorship and independence from third parties are the two central goals of nostr achieved through the use of independent relays. As nostr focuses on the exchange and not on the content, the adaptability is given accordingly. The operation of the relays does not represent any administrative effort for the operators, as they only receive and provide data. The respective storage space and network bandwidth requirements of the individual relays increase linearly with the number of relay users, which supports scalability.

2.3 Qualitative Comparison

The approaches examined each have individual strengths and weaknesses and each only fulfill a subset of the requirements. This can also be seen visually in the table 1, which summarizes the results and contains TrADS in the last row. The biggest gap is in the trust awareness requirement. Due to modera-

[21] `https://yarn.social/`
[22] `https://nostr.com/`

Table 1: Qualitative comparison between decentralized social networks

Decentralized Social Network	T	D	R	N	I	A	M	S
Mastodon [4]	◔	◕	◔	●	◔	●	○	◕
PeerTube [10]	◔	◕	◔	●	◔	●	○	◕
diaspora* [9]	◔	◕	◔	●	◔	●	○	◕
Pixelfed [11]	◔	◕	◔	●	◔	●	○	◕
Secure Scuttlebutt [13]	○	◕	◔	◔	◔	●	●	●
Sone [14]	○	◕	●	◔	◔	●	●	●
Peergos [15]	○	●	◔	◔	●	●	●	●
SteemIt [17]	○	○	●	○	●	○	●	○
Minds [18]	○	○	●	○	●	○	●	○
Movim [19]	○	◕	●	◕	●	◕	●	●
twtxt [20]	○	◕	●	○	●	◔	●	●
nostr [22]	○	○	●	◔	●	◔	●	●
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.

tion, which is particularly problematic for larger instances [1], the federated networks can fulfill part of this requirement, but are far from automated trust awareness. Data control is only fulfilled very well by Peergos[15], while all others usually do not give the user a free choice of data storage location. Networkability and Adaptability in particular are very well fulfilled by the Fediverse. Resistance to cencorship and independence, on the other hand, are only fulfilled by applications that are not part of the Fediverse or are usually not a P2P solution. The models based on P2P have the best results in terms of requirements, especially for operators, although Movim, twtxt and nostr also excel here with the exception of adaptability. It is remarkable that none of the related work is based on Solid [20]. TrADS is therefore alone in the related works with the idea of a decentralized social network based on Solid.

3 TrADS

To improve related work in terms of requirements and to take advantage of the benefits of Solid, we have created TrADS, a **Tr**ust-**A**ware **D**ecentralized **S**ocial network. Solid allows end users to retain control over their own data [20] by using so-called pods as personal data storage. Content in the pods

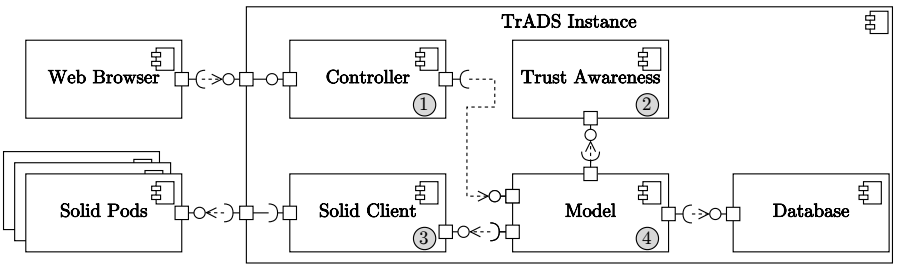


Figure 1: A TrADS Instance Architecture

is organized using containers, which are functionally comparable to folders in a file system. The content can be saved in the form of linked data in the Resource Description Framework (RDF) [5] or any other format. The data remains under the control of the end users in Solid, as they can decide on changes to access rights at any time. The use of end-user-specific data stores enables a high degree of reuse of the stored data, as it can be used by all permitted applications and services. Furthermore, the end user can move the data at any time, as applications only need access to the URL of the user-desired pod. In addition, the technical implementation of Solid is promising for broad adaptation due to the use of established W3C standards [20]. Solid-based applications do not exchange data directly as in federated networks or the P2P-based approaches, but read or write it in pods. This also increases the resistance to censorship.

In the remainder of this section, TrADS is conceptually detailed by first describing its architecture in subsection 3.1 and then presenting the common workflows of a social network including Solid and the trust awareness component of TrADS in subsection 3.2. The section finishes by detailing the prototype of TrADS in subsection 3.3.

3.1 Architecture

In order to maintain a high networkability similar to the federated networks, TrADS adapts their instance-based approach. TraDS instances can thus be hosted by anyone and used by any end user owning a Solid Pod. Its architecture is shown in detail in the UML component diagram in Figure 1. In general, a TrADS instance follows the Model-View-Controller (MVC) pattern [14]. Thus, it contains the classic Controller (1) and Model (4) components. To simplify the figure, the View is not shown but used in the end user's web

browser. The Model (4) accesses two data spaces, an internal database and any Solid pods via the Solid Client (3). The database is used for internal application data, such as the output of the trust awareness component (2), which provides information on how trustworthy the data stored in the solid pods are. In addition, the database serves as a cache for data from solid pods. The cache functionality is needed to realize a fast response time of TrADS for end users. As the pods are located outside the TrADS instance, they may be overloaded or unavailable at the time of one end user request. To still provide the user with a usual response time of a social network, the data is cached for read purposes but always stored in the solid pods.

Due to the potentially malicious or harmful data and the constantly growing amount of data on the web [25], TrADS has a trust awareness component (2) in accordance with the position paper by Siegert and Gaedke [22]. It is responsible for evaluating the trustworthiness of external content before the content is displayed to end users. TrADS does not filter out any content, but marks untrustworthy content accordingly in order to convey a sense of trust to end users. They therefore have the freedom to decide what they interact with, but are supported by the TrADS instance. Since only the TrADS instance bundles the data from different pods, the component is present in every instance. This allows each instance to function independently of other instances. The trust awareness component can deliver different results for the same content on two instances. It may be that the instances have different data (amounts) available as a basis for the evaluation, use different trust models to realize trust awareness [29] or because instance users and operators have different preferences for trust awareness. As mentioned in the introducing section 1, TrADS deliberately leaves the definition of the trust model to be used within the trust awareness component open. This article focuses on the framework conditions in which such a component can be integrated into a social network, both in the workflows and in the web user interface.

As the content is completely under the control of the respective author, technical challenges arise in TrADS when interacting with the content of other users. In a social network, this content includes posts, chat messages or interactions with posts such as likes or reposts. If a user comments on another user's post, they cannot ensure that this content will not be changed or remain available later. By using checksums when referencing content in TrADS, the referenced content can be checked for changes at a later point in time despite decentralized storage. For every interaction that contains an internal reference, TrADS saves the current checksum of the referenced content at the time of creation. If other users later request the post including the

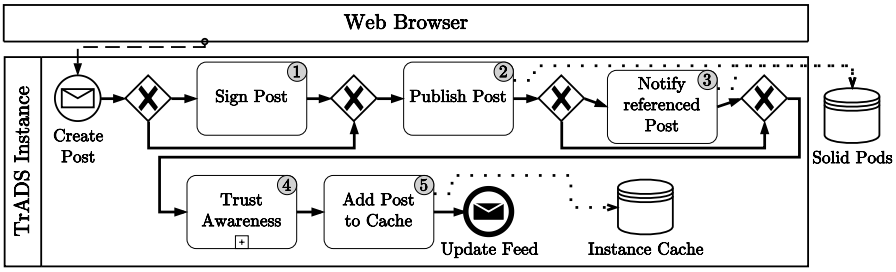


Figure 2: BPMN of Creating a Post in TrADS

comments, TrADS can recognize whether the comments were created for the current post content or not. In order to ensure the integrity of posts, TrADS also enables the optional signing of posts using OpenPGP in accordance with RFC 9580 [28]. Signing allows posts to be saved in other users' pods, e.g. when they interact with the post, to facilitate the distribution of content in TrADS. However, to ensure that such distribution does not undermine data control over the author of the post, it is only possible if the post is to be visible to third parties. If a post has been distributed to a pod other than the author's, it is not technically possible to delete it afterwards. Nevertheless, the author has control over the data, as he decides on replication by TrADS. Furthermore, replication of content on the web cannot be completely prevented anyway.

3.2 Workflows

As TrADS is a social network for microblogging, the creation of posts is an essential function. Figure 2 shows the process of how a new post is created in TrADS. After receiving the new post via the web user interface, the post is optionally signed (1). The user has defined this in the form for creating posts using a toggle option. TrADS signs the mail with the user's PGP key. To be able to do so, the key must be accessible to TrADS from the user's solid pod. After the article has been published (2), a referenced article is also informed about the referencing in case of a repost (3). Both of these steps interact with the corresponding solid pods of the respective authors of the posts. TrADS then evaluates the new post in terms of trustworthiness (4) and saves the post and the trust awareness rating in the instance's own cache (5).

The main page in social networks is usually a user feed with the latest posts and interactions from users or interests that the user follows. The process for creating such a feed is detailed in figure 3. When a user requests their

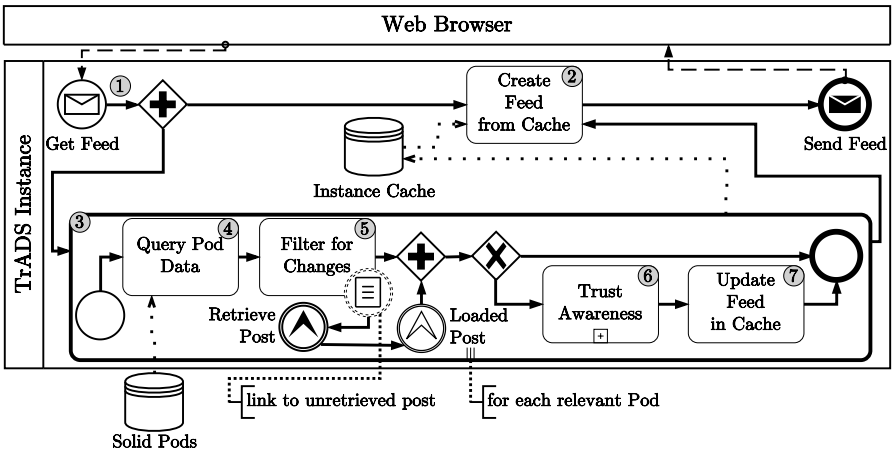


Figure 3: BPMN of Getting an End User's Feed in TrADS

feed (1), a feed is created based on the data in the instance's cache (2). Thanks to this caching, the user can expect the usual short response time. Parallel to the delivery of the user feed via the cache, a sub-process is carried out in the TrADS instance for each relevant solid pod to request new content (3). After querying the pod's TrADS data (4), TrADS filters the data for changes (5). During filtering, it may be discovered that new references have been set to locations that are not yet known to the instance. In such cases, TrADS starts the retrieve post process for each post unknown to the instance and waits for it to finish. Only after all referenced posts have been loaded can the TrADS instance add the changes to the cache. If no changes to the data have been detected, the sub-process of loading content for the pod ends. Otherwise, Trust Awareness checks the trustworthiness of each new or updated content in the queried pod (6) so that the user feed can be updated according to the changes (7). Subsequently, the updated user feed is created again (2) and sent to the user's web browser after all known pods have been checked, even without a further user request.

The mentioned process for retrieving referenced but yet unknown posts is detailed in Figure 4. Before the post can be retrieved from the solid pod that stores it (2), the pod's author information is retrieved from TrADS (1) if the pod is also unknown. Since the retrieved post can itself reference a post, the process calls itself recursively (3) until a post does not reference another post. When the entire chain of referenced posts is loaded, the retrieved post is

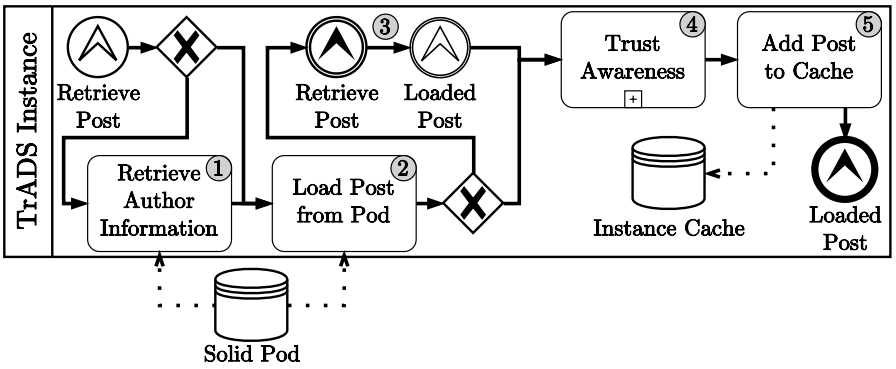


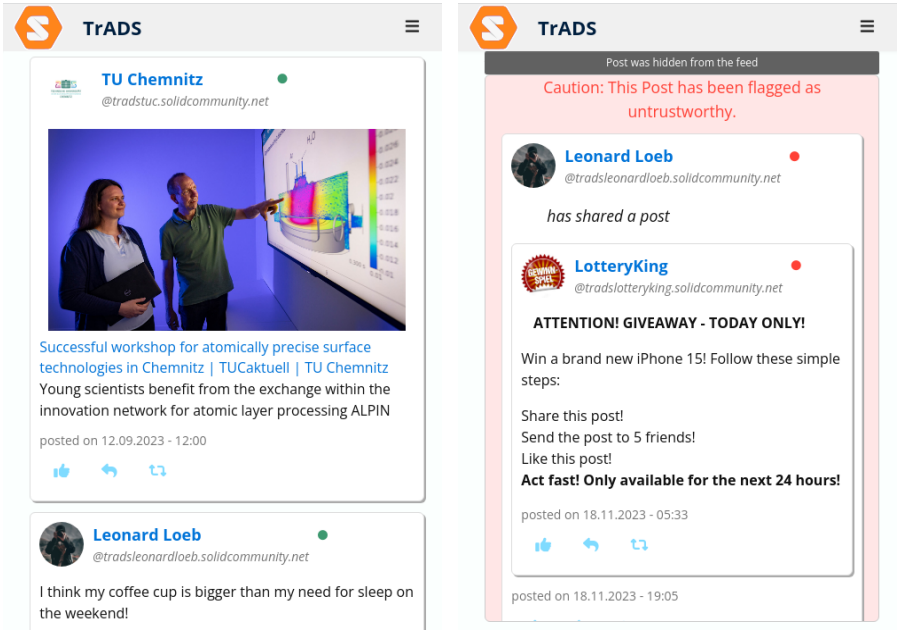
Figure 4: BPMN of Retrieving a Post in TrADS

checked by Trust Awareness (4) and added to the instance cache (5). In this way, the trustworthiness of each post that refers to another can be determined using the chain of post references.

3.3 Prototype

We made the prototype of TrADS available online[6], and published its code[5] for the purpose of reproducibility already for the first paper on TrADS at ICWE 2024. Similar to other social networks, the TrADS prototype contains typical functions of a social network. When accessed, users are presented with their personalized feed consisting of posts and reactions from related pods. The related pods are selected based on the social network accounts a user follows. Each post within TrADS contains the author's name, profile picture, the date the post was published and the number of likes received. The typical options for sharing and replying to the post are also displayed for each post. An example of a post is shown in Figure 5a. TrADS also supports checking the profiles of other accounts, managing followed accounts and editing your own profile, as well as basic chat functionality. To illustrate the results of the trust awareness component each post is marked with a colored circle. The circle is located at the top right of each post. If you move the mouse over a circle, a tooltip is displayed explaining the meaning of the individual colors, where the possible colors are: ● untrustworthy, ● very likely untrustworthy, ● maybe untrustworthy, ● neutral, ● trustworthy, ● very trustworthy.

If a post is ● untrustworthy, it will be hidden from the feed by TrADS. The dark gray indicator visible at the top of the figure 5b is displayed instead.



(a) Examplary TrADS Feed, including two posts

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

Figure 5: TrADS Front-end Examples

Once the user clicks on the indicator, the user can see the expanded view of the figure on an untrusted post. In order to not completely filter out the content within TrADS and thus automatically enforce a certain level of control over the content within the network, TrADS only hides untrusted posts. Users therefore retain control over the content they wish to consume. The prototype does not contain a fully-fledged trust model within the trust awareness component. While frameworks such as ConTED [23] or other trust models [29] could be used once migrated to the decentralized web, the prototype's trust awareness assigns trust awareness results randomly. For demo purposes, however, the results in the running demo were influenced by us in order to obtain meaningful results.

The solid pod data of TrADS can be individually restricted by the access management of solid pods, while the container structure of TrADS must be publicly available. Otherwise, TrADS instances unknown to the user cannot insert any of the user's content into the instance, which would contradict the

basic function of the network. The pod data is all stored in the container named *social*. Within this container, there are five other containers to make it easier to find the various data in TrADS. While all media content such as images, GIFs and videos are stored in the *media* container, there is the *reactions* container for likes and comments on posts. The two containers *inbox* and *outbox* serve TrADS as data storage for the chat functionality per user. A message always exists in the network in the sender's outbox and the recipient's inbox. All posts are stored in the pod within the *posts* container. Inspired by the human-readable formats of twtxt [20], TrADS saves the content of chat messages, comments and posts in Markdown format. By storing each content in separate Markdown files, each content is identifiable and retrievable via its URL. Metadata of any kind is stored in the Markdown files as headers in YAML format. In the case of a signed post, the content of the Markdown file including the header is the PGP Signed Message, followed by the PGP signature.

4 Evaluation

To investigate and understand the impact of the trust component on the user experience, we have already conducted a user study on the front-end of TrADS in the paper presented at ICWE 2024 [24]. To further investigate our findings and overcome the limitations of the German participants in the first study, we conducted a second, improved user study for this article. In this section, we present the process of the second user study and then discuss its results and possible implications. We do this by comparing the changes we made in the second survey compared to the first and by comparing the changes in the results of the two studies. All data of the evaluation is accessible online[7], including the raw survey data as well as all numbers we discuss in the results subsection.

4.1 Procedure

We used the same tool for the second user study as in the first one in order to be able to embed TrADS directly into the survey again using IFrames [24]. In order to internationalize the user study, we offered the second study in German and English, with users being able to choose the language. Similar to the first user study, each participant was asked about their gender, age and social network usage frequency after a welcome message. Each participant then had time to explore TrADS. The tool randomly filtered which prototype

version each participant saw to create two different but almost equal groups of participants. One group of people saw TrADS as it was presented in this article, including an active trust awareness component. We refer to this group with enabled trust awareness in TrADS as Group 1 below. The second group of participants, on the other hand, saw a second running instance of TrADS without an activated trust awareness component. Thus, the colored circles next to the posts and the function for hiding untrustworthy posts are deactivated. We refer to this group with disabled trust awareness in TrADS as Group 2 below.

To encourage each participant to interact with the demo, they were given a small series of interaction tasks, including searching for the latest post from a particular magazine, liking a post and writing their own post. However, the extent to which participants completed each task was not tested. Instead, participants had to answer whether they noticed the colored circles next to the posts and the hiding of untrustworthy posts. Those control questions though only were asked past the evaluation of the participants' user experience.

While the first user study determined the usability of TrADS in its current state using the System Usability Score (SUS) [3], we evaluated the user experience of TrADS in our second one. Since TrADS' trust awareness is intended to counteract the possible deterioration of user experience through external stored data, the evaluation of the user experience instead of only its usability improves the evaluation. We used the UEQ+ question catalogue [21], which extends the User Experience Questionnaire (UEQ). UEQ+ provides a set of scales, which can be combined to create a questionnaire. We hence used the following 10 scales to evaluate the user experience: (1) Attractiveness, (2) Perspicuity, (3) Stimulation, (4) Dependability, (5) Intuitive Use, (6) Trust, (7) Quality, (8) Clarity, (9) Value, (10) Trustworthiness of Content. In UEQ+, each participant answers four questions per scale and rates how important each scale is to them in a fifth question. All these questions and the importance are indicated on a 7-point Likert scale. Based on the importance rating, UEQ+ is able to calculate an individual KPI value for each participant [21].

Subsequently, the control questions are asked, which are in total a set of four questions. The participants respond those questions about to which extent they noticed Trust Awareness elements in the user interface with one of *always*, *sometimes*, and *not at all*. The first question is about whether the participant noticed the circles besides posts in general and similarly a second one asks about them noticing posts which got hidden. Only if the participant responds with more than *not at all*, each question gets a subsequent question

whether they noticed the circles had different colors, and whether they ever extended a hidden post to read the original post.

Further on, the participants rated 9 statements on the integration of trust awareness into a social network and 7 statements on the importance of data protection in social networks, each on a 5-point Likert scale. The survey explains shortly the trust awareness features, including a table to map colored circles to trustworthiness meanings. Since Group 2 not even saw those circles, it is explained in a way how it could be potentially, and in Group 1 how it was in the shown demo.

As the participants interacted with TrADS during the user study, we used the same TrADS demo mode for both groups that we had already used in our first user study. Thus, no permanent changes are possible for the participants, as all likes or self-written posts are removed after 30 minutes. In addition, the participants' interactions are not saved in a solid pod, but only in the instance cache. Furthermore, user management is deactivated to prevent users in the TrADS cache from changing, whether by adding new users or removing existing ones. To fill the two instances with content, we have prepared 72 solid pods with the content of posts and interaction with other posts, in our previous study it were only 25 pods [24]. Both instances used in the user study contain the same content. Most of the solid pods are profiles of existing websites that populate the TrADS instances with news. News articles only contain links to the corresponding articles, which are visualized by means of a preview in TrADS. In order to avoid socially controversial topics in our user study, we only selected articles from the areas of entertainment, sports and animals. Similar to the first user study, there are three profiles that post fraudulent content and 17 profiles that depict other users on the social network talking about content such as restaurant visits or funny GIFs.

4.2 Results

In our second user study, we received a total of 64 complete responses, taking into account only those who answered the control questions correctly for their group. Thus, we only take those of Group 1 who noticed the circles besides the posts and that some posts got hidden in the feed. In Group 2 we respectively only take participants into account if they did not see any circles besides the posts and also did not notice any hidden posts. This filtering reduces the number of responses to 29. As to be seen in table 2, most of the filtered participants were males (22), most were between 25-34 years old (19) and most use social networks on a daily basis (22).

Table 2: Participants Details of the second user study conducted on TrADS, which only contains responses after filtering

	Group 1	**Group 2**	**Group 1&2**
Number	10	19	29
Gender: Female	2	4	6
Gender: Male	8	14	22
Gender: Diverse	0	0	0
Gender: Not Disclosed	0	1	1
Age: Below 18	0	0	0
Age: 18-24	0	1	1
Age: 25-34	9	10	19
Age: 35-50	1	6	7
Age: 51-70	0	2	2
Age: Above 70	0	0	0
Usage: Daily	9	13	22
Usage: Several Times a Week	1	3	4
Usage: Once a Week	0	1	1
Usage: Less Often	0	1	1
Usage: Not at All	0	1	1

In our first user study, we had a significant difference in the SUS scores [3] between participants noticing in minimum one trust awareness front-end element and participants that did not notice any trust awareness in TrADS. Based on a significance level of $\alpha = 0.05$ and 51 degrees of freedom, the critical value for the t-test of the first user study is 2.01 and results with $(95\% - CI[1.06, 21.23])$ in $t(51) = 2.22$, $p = 0.03$. The SUS score distribution is again shown in figure 6.

However, in our second user study, we did not measure SUS scores but UEQ+ KPIs [21]. Additionally, we had two well distinct groups of participants in our second user study, seeing different versions of TrADS. In the first user study all participants saw TrADS with enabled trust awareness component. Respectively, the KPI value distribution of the second user study is shown in figure 7. Group 1 has an expected KPI value of $\mu = 1.25$ with a standard deviation of $\sigma = 0.98$. On the other hand, Group 2 has an expected KPI value of $\mu = 0.47$ with a standard deviation of $\sigma = 1.04$. Overall, the second user study revealed that the UEQ+ KPI for TrADS has an expected value of $\mu = 0.74$ with a standard deviation of $\sigma = 1.07$ over both groups.

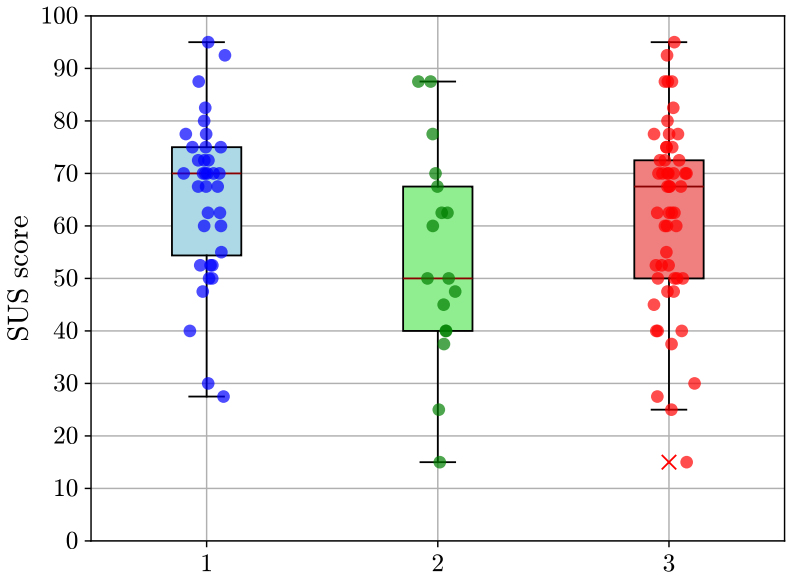


Figure 6: SUS Distribution in first user study of (1) participants noticed in minimum one trust awareness front-end element (2) participants did not notice any trust awareness front-end element (3) all participants.

Comparing expected values of the two groups' KPIs, it is noticable that Group 1 with enabled trust awareness got a quite better score than group 2.

To investigate whether the difference of KPI values in Group 1 and Group 2 is also significant, we examined both KPI value data sets in an independent t-test. Based on a significance level of $\alpha = 0.05$ and 27 degrees of freedom, the critical value for our t-test is 2.05. This results with $(95\% - CI[-0.04, 1.60])$ in $t(27) = 1.96$, $p = 0.06$. We could thus in our second user study not find a significant difference in the user experience between participants having an enabled trust awareness component in TrADS and those who had the trust awareness disabled. However, it is positive to see that our confidence lower boundary indicates that the enabled trust awareness does not decrease user experience of TrADS noticeably. In addition, we checked the effect size to quantify the different between the two KPI data sets with Hedges' g [7] resulting in $g = 0.74$, indicating a noticeable difference between the two groups.

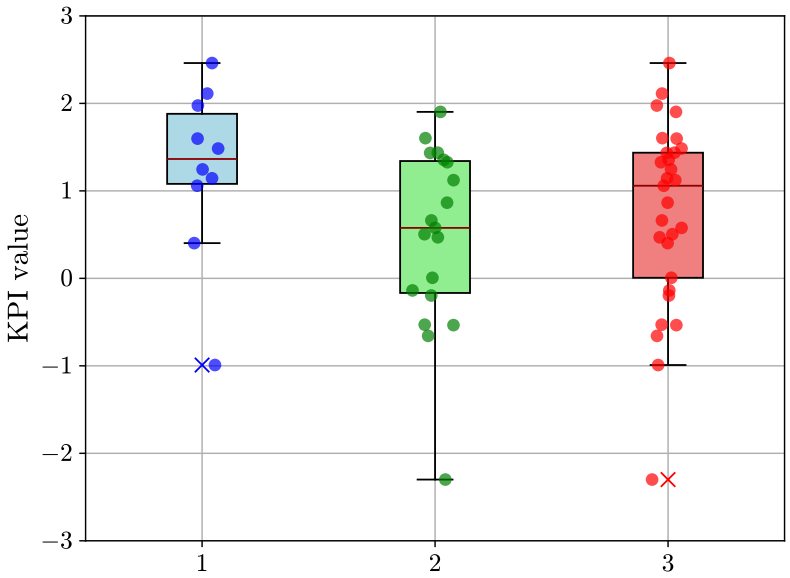


Figure 7: UEQ+ KPI Distribution in second user study of (1) participants in Group 1 (2) participants in Group 2 (3) participants in Group 1 and Group 2.

Since the authors of the UEQ KPIs [8] state a possible range in practical applications as $[-0.286, 2.143]$, we also carried out a sensitivity analysis of the KPI data sets. To do this, we remove all data series that do not lie within the specified possible range. Although we use UEQ+ and not UEQ, we wanted to show how this additional filtering would change the result. The expected value and standard deviation of the KPIs for group 1 is $\mu = 1.38, \sigma = 0.55$, for group 2 $\mu = 0.86, \sigma = 0.66$ and for both groups together $\mu = 1.04, \sigma = 0.66$. The KPI values included in this sensitivity analysis are shown as a boxplot in Figure 8. Based on a significance level of $\alpha = 0.05$ and 21 degrees of freedom, the critical value for our t-test is 2.08. This results in $(95\% - CI[-0.05, 1.08])$ in $t(21) = 1.89$, $p = 0.07$. Thus, the t- and p-values of the t-test show even less of a significant difference between the two groups than with all KPIs. However, Hedges' g [7] of this t-test results in a higher $g = 0.80$. Such an effect size of 0.8 or more indicates that there is a substantial difference between the two groups that is likely to be significant.

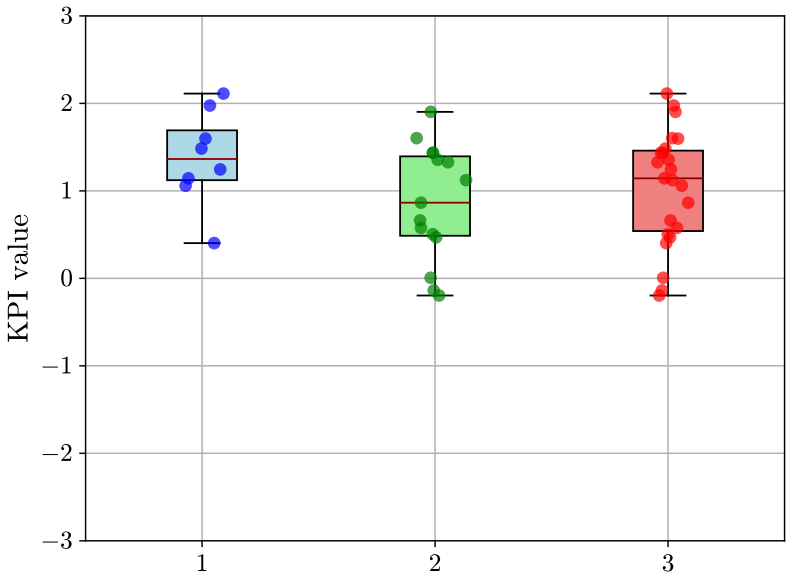


Figure 8: UEQ+ KPI Distribution in second user study without KPI values that are not in most-likely range of UEQ KPIs of (1) participants in Group 1 (2) participants in Group 2 (3) participants in Group 1 and Group 2.

The remaining survey statements are the same as in the first user study. All statements are rated by the participants on a 5-point Likert scale, where 1 stands for full rejection and 5 for full approval. While the statements were overall when combining both Groups rated similar to the results of our first user study, some ratings differ within Group 1 and Group 2. Participants that experienced the trust awareness component actively in Group 1 agree to use a social network providing such trustworthiness information about posts more than other social networks ($\mu = 4.30, \sigma = 0.95$), while those who did not experience it but only read about it tend to disagree on that ($\mu = 2.68, \sigma = 1.16$). Similarly, Group 1 rates it helpful that untrustworthy posts are hidden ($\mu = 4.50, \sigma = 0.71$) and Group 2 again tends to disagree ($\mu = 2.89, \sigma = 1.24$). However, Group 2 also distrust their shown network way more ($\mu = 2.32, \sigma = 1.20$) than Group 1 does ($\mu = 4.20, \sigma = 0.92$). In addition, Group 1 does not have a clear idea of how social networks store their data ($\mu = 4.70, \sigma = 0.67$), while Group 2 feels to have a little more knowledge ($\mu = 3.95, \sigma = 0.91$).

We expected our first user study to be confirmed by a second international study using UEQ+ instead of SUS to measure user experience rather than usability. However, the present results do not confirm that the trustworthiness component significantly improves the user experience at the current state. Thus, while our results could mean that the presence of our trust awareness component actually has no effect on user experience, several indicators point to a Type II error in our results, where there is insufficient power to reject the null hypothesis. The indicators include: The medium to high Hedges' $g = 0.74$ effect size of our results, a t-value close to the critical t-value, a $p = 0.06$ close to the $\alpha = 0.05$, and higher KPI values of $\mu = 1.25$ in Group 1 than in Group 2 $\mu = 0.47$ with a standard deviation around 1 in both groups. In addition, the t-test shows that the perception of trust hardly lowers the user experience due to the lower threshold of the 95% confidence interval of -0.04. In order to discuss the results transparently and mitigate possible publication bias [26], the publication of these results is nevertheless important for future work.

5 Conclusion

In this article, we present TrADS, a **Tr**ust-**A**ware **D**ecentralized **S**ocial network. Following the MVC pattern [14] and the concept of a Trust Awareness within decentralized web applications [22], TrADS integrates decentralized web data sourced from solid pods in a trustworthy way. As this article extends our presentation of TrADS at ICWE 2024 [24], we have extended the already published related work with a detailed analysis of the state of the art, including requirements descriptions and insights into the requirements evaluations per related approach. We have also detailed the processes including the trust awareness component of TrADS and explained how the data is stored in the solid pods and how trust awareness is implemented. Since the focus of this work is on the expected improvement of the user experience through the trust component, the component itself is only a mockup that does not yet include a trust model like our previous work ConTED [23]. Instead, the trust awareness in the current prototype is output randomized or set by experts for demo purposes. Due to the expected improvement of the user experience, we have extended the evaluation we conducted for the contribution at ICWE 2024. We evaluated TrADS with a new user study with 64 participants, which is not only German but international and evaluates the user experience instead of just the usability, as the first user study did. Our extended evaluation also includes two separate groups and not just one

as in the first paper on TrADS [24]. Group 1 experienced TrADS with Trust Awareness features enabled and Group 2 experienced a version without these. We also filtered out valid responses more strictly than in the first user study in order to only include participants who either perceived trustworthiness in group 1 or correctly did not perceive it in group 2. This filtering reduced our degrees of freedom within the t-test to 27. Unfortunately, the new evaluation does not confirm our initial findings from the ICWE paper. Based on the measured UEQ+ KPIs [21], there is no significant improvement in user experience with active trust awareness, but several indicators point to a type II error of our performed t-test. A type II error means that our resulting data does not have enough power to reject the null hypothesis, while an evaluation with more participants would likely do so. In addition, our extended evaluation shows that users who have interacted with the trust awareness component prefer to use social networks with similar features than those who have only read about such features. The indication that participants in Group 2 also distrust their demo without a trust component more than those with an active trust component is another positive finding that emerges in our extended evaluation.

In future work, we will work on the optimisation of trust models migrated to the decentralised web in order to be able to integrate the most promising candidates into TrADS. ConTED [23] is a promising migrated trust model that has not yet been finalised for integration into TrADS. Apart from the optimisation in terms of the requirements for a trust awareness component presented in the position paper by Siegert and Gaedke [22], the extent to which the current rather passive intervention of the trust awareness component in TrADS can also be extended to other use cases in the decentralised web remains to be explored. This is because as soon as the use case of a web application involves, for example, directly paying customers, as in the case of a booking platform, data that is obviously malicious or harmful would have to be completely filtered out by trust awareness and not just hidden like untrustworthy posts in TrADS. Otherwise, customers could fall victim to a scam and no longer want to use the application. In such an extreme case, the user experience would be at its lowest point, which is exactly what the trust awareness component is designed to prevent. In addition to the integration of a complete trust model in TrADS, further work is also required to improve the trust awareness features in the user interface. While the usability in our first TrADS paper was already significantly better for those who were aware of the features, we were not yet able to confirm this for the user experience with this article. On the one hand, a significantly larger evaluation with more

participants can provide a clearer picture of the current status, but on the other hand, the initial ideas for visualising trust awareness are not necessarily sufficient. Improving visualisation is therefore just as important for improving trust awareness in TrADS.

Acknowledgements

The authors would like to thank Stefan Heyder for his valuable contributions to the discussion on the statistical analysis of the user study conducted for this article. This article 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).

References

[1] Ishaku Hassan Anaobi, Aravindh Raman, Ignacio Castro, Haris Bin Zia, Damilola Ibosiola, and Gareth Tyson. Will admins cope? decentralized moderation in the fediverse. In *Proceedings of WWW '23*, page 3109–3120, 2023.

[2] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The world-wide web. *Commun. ACM*, 37(8):76–82, August 1994.

[3] John Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3):189–194, 1996.

[4] Stephane Couture and Sophie Toupin. What does the notion of "sovereignty" mean when referring to the digital? *New Media & Society*, 21(10):2305–2322, 2019.

[5] Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C, February 2014.

[6] Barbara Guidi. When blockchain meets online social networks. *Pervasive and Mobile Computing*, 62:101131, 2020.

[7] Larry V. Hedges. Distribution theory for glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, 6(2):107–128, 1981.

[8] Andreas Hinderks, Martin Schrepp, Francisco José Domínguez Mayo, María José Escalona, and Jörg Thomaschewski. Developing a ux kpi based on the user experience questionnaire. *Computer Standards & Interfaces*, 65:38–44, 2019.

[9] Mahamat Ali Hisseine, Deji Chen, and Xiao Yang. The application of blockchain in social media: A systematic literature review. *Applied Sciences*, 12(13), 2022.

[10] Luis Daniel Ibáñez, Elena Simperl, Fabien Gandon, and Henry Story. Redecentralizing the web with distributed ledgers. *IEEE Intelligent Systems*, 32(1):92–95, jan 2017.

[11] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170, 2000.

[12] Lucio La Cava, Sergio Greco, and Andrea Tagarelli. Understanding the growth of the fediverse through the lens of mastodon. *Applied Network Science*, 6(1):64, Sep 2021.

[13] Protocol Labs. IPFS Standards. `https://specs.ipfs.tech/`, Last Access: 27.03.2025.

[14] Avraham Leff and James T. Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, pages 118–127, 2001.

[15] Christine Lemmer-Webber, Jessica Tallon, Erin Shepherd, Amy Guy, and Evan Prodromou. Activitypub. first edition of a recommendation, W3C, January 2018.

[16] Xiaowei Li and Yuan Xue. A survey on server-side approaches to securing web applications. *ACM Comput. Surv.*, 46(4), mar 2014.

[17] Mastodon gGmbH. Mastodon annual report 2022, 2023. `https://joinmastodon.org/reports/Mastodon%20Annual%20Report%202022.pdf`, Last Access: 27.03.2025.

[18] Mastodon gGmbH. Moving or leaving accounts, 2023. `https://docs.joinmastodon.org/user/moving`, Last Access: 27.03.2025.

[19] Sascha Meckler, Rene Dorsch, Daniel Henselmann, and Andreas Harth. The web and linked data as a solid foundation for dataspaces. In *Companion Proceedings of WWW '23*, page 1440–1446, 2023.

[20] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: A Platform for Decentralized Social Applications Based on Linked Data. Technical report, MIT CSAIL & Qatar Computing Research Institute, 2016.

[21] Martin Schrepp, Jessica Kollmorgen, Anna-Lena Meiners, Andreas Hinderks, Dominique Winter, Harry B. Santoso, and Jörg Thomaschewski. On the importance of ux quality aspects for different product categories. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(2):232–246, 2023.

[22] Valentin Siegert and Martin Gaedke. Trust awareness for redecentralized web applications (position paper). In *Joint Proceedings of ESWC 2023 Workshops and Tutorials*, 2023.

[23] Valentin Siegert, Arved Kirchhoff, and Martin Gaedke. Conted: Towards content trust for the decentralized web. In *Proceedings of WI-IAT 2022*, pages 604–611, 2022.

[24] Valentin Siegert, Dirk Leichsenring, and Martin Gaedke. Trusting decentralized web data in a solid-based social network. In Kostas Stefanidis, Kari Systä, Maristella Matera, Sebastian Heil, Haridimos Kondylakis, and Elisa Quintarelli, editors, *Web Engineering*, pages 230–245, Cham, 2024. Springer Nature Switzerland.

[25] Petroc Taylor. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025, nov 2023.

[26] Alison Thornton and Peter Lee. Publication bias in meta-analysis: its causes and consequences. *Journal of Clinical Epidemiology*, 53(2):207–216, 2000.

[27] W3C. Ostatus community group, 2012. `https://www.w3.org/community/ostatus/`, Last Access: 27.03.2025.

[28] Paul Wouters, Daniel Huigens, Justus Winter, and Niibe Yutaka. OpenPGP. RFC 9580, July 2024.

[29] Han Yu, Zhiqi Shen, Cyril Leung, Chunyan Miao, and Victor R. Lesser. A survey of multi-agent trust management systems. *IEEE Access*, 1:35–50, 2013.

Biography

Valentin Siegert is a research associate and doctoral candidate at Chemnitz University of Technology. His research interests include Web Engineering, Cybersecurity, Semantic Web and Web of Things. With his research, he is currently working on a Trust Awareness of Decentralized Web Applications using Linked Data in a dynamic and automated manner.

Dirk Leichsenring, an alumnus of Chemnitz University of Technology, has a keen interest in open source software, web technologies, and leveraging trust awareness to enhance user experience on the web. His master's thesis focused on trust in decentralized Solid-based social networks.

Alejandro Wurts Santos is a Full Stack Developer currently working at AI Conver. He is pursuing a Master of Science in Web Engineering at Chemnitz

University of Technology. His professional journey includes experience at Oracle, where he worked in enterprise application development, and earlier work at a metrology technology firm where he developed industrial software solutions. With an undergraduate degree in Mechatronics from Universidad de las Américas Puebla, Wurts combines engineering fundamentals with web development expertise.

Martin Gaedke is a pioneering researcher in the fields of Web Engineering and Smart Collaboration. As Full Professor of Computer Science and Director of the Computing Center at Chemnitz University of Technology, he leads research activities that empower people, software, and machines to collaborate more intelligently in our hyper-connected world. With over 30 years of experience, a PhD from the Karlsruhe Institute of Technology (KIT), and visiting roles at leading institutions like TU Wien, Martin's research explores new approaches to Web-based solution development, Data Engineering, and Trustworthy AI. His work doesn't stay in the lab — in close collaboration with enterprises such as Daimler, Microsoft Research, and Hewlett-Packard, his research helps shape secure, reliable, and human-centered digital ecosystems. He has co-authored over 250 scientific publications and actively mentors early-career researchers and future technology leaders. His commitment to bridging academic rigor with real-world relevance continues to strengthen the foundations of digital infrastructures that are open, ethical, and intelligently collaborative.