Distributed Discovery of User Handles with Privacy

Thomas Paul and Marius Hornung and Thorsten Strufe
Peer-to-Peer Networking Group
TU Darmstadt
[thomas.paul, mhornung, strufe][at]rbg.informatik.tu-darmstadt.de

***Abstract*—Decentralized Online Social Networks (DOSN) seek to increase their users' privacy by removing centralized data storage and control. Their lack of usability and competitive features has severely hindered their success. Even a privacy preserving user discovery, which seems paramount for a social networking service, so far has not been implemented. Aiming at a hybrid architecture of decentralized servers, as it is suggested by the currently most successful DOSN Diaspora amongst others, we suggest a distributed discovery of users. Our discovery mechanism protects privacy of users and does not create vulnerabilities to mass collections of profiles or SPAM.**

The scheme adapts known techniques like DHT, indirection, and proof of knowledge, to meet the service specific requirements. A general implementation for a popular XMPP server furthermore gives proof of our concept and represents a first step towards constructing a reliable distributed social networking service with user discovery on decentralized servers, without leaking any information about its users.

I. Introduction

User trust in Online Social Networks (OSN) is waning as a result of public discussions about provider-side security problems and mass surveillance of users. Decentralized Online Social Networks (DOSN) aim to provide social networking functionality without requiring a (centralized) OSN provider. Thus there is no need in DOSN to trust an omni-potent provider for both: to efficiently protect user data against malicious actors as well as to abstain from misusing or secretly sharing the available information about their users.

However, Narayanan et al. [1] doubt that federated or distributed social networks are a promising alternative because of technical as well as economical disadvantages. We take the following two facts as supportive motivations for our work: First, that technical issues can be solved since large OSN like Facebook are technically distributed and do not run on a single machine (but are controlled by one authority). Second, that the email system, which is also a realization of the architecture that we want to improve, is widely adopted.

Decentralization removes any single entity with complete knowledge about users as well as about the social graph connecting them. However, several appealing functions in today's OSN, like Facebook, require knowledge about user data and the social graph. These functions include not only advanced inference or recommendation features, but even paramount functions like discovering other users. Even the most popular DOSN Diaspora consequently relies on out-of-band communication for exchanging user handles, which are required to connect to other users [2].

Existing technology does not seem to offer appropriate solutions to the privacy preserving user discovery problem:

1) Creating a central index requires a trusted party, or leak information about the participants. Beyond privacy concerns, economical reasons prevent its setup in a decentralized system, as it would cause costs that are unlikely to be paid for by the users, in the current Internet ecosystem.
2) Classical P2P search assumes a public, distributed search index, which at least partially is stored on and forwarded by presumably untrusted nodes [3], and commonly even spreads information by replicating its content.
3) Public key encryption with keyword search (PEKS) [4] requires possession of the cipher, and hence a centralized, encrypted index, which is unviable as mentioned above. Applying PEKS to a DHT does not seem viable since the cipher isn't known to the requesting party and hence can not easily be addressed and discovered.
4) Secret database querying approaches do not solve the problem of avoiding to build a centralized database that contains user-linkable information.

In this work, we propose a mechanism that adds the functionality of user handle discovery in RFC 822[1] based systems, to make a step towards providing usable DOSN with a competitive feature set, while maintaining the privacy of their users. This implies that no information that can be linked to any participating individual may be leaked by the scheme. Because of potential abuse of profile discovery, the scheme must also prevent mass collection of profiles and their identifiers to avoid SPAM or other types of illegitimate messaging.

To tackle this issue, we propose a solution that divides the service into three separate parts:

- The collection of distributed information about servers that host profiles which potentially include a specific user, identified by separate user attributes.
- A privacy preserving negotiation protocol to prove knowledge about the sought user.
- The provision of ephemeral handles with limited validity, which allow for a single message within limited time.

The scheme is applicable to arbitrary hybrid DOSN architectures. These consist of decentralized servers that are equal by design and operated by diverse parties, and which are selected by participants to host their profiles. Hybrid

[1] https://www.ietf.org/rfc/rfc0822.txt, accessed: 18 th of February 2014

978-1-4799-3512-3/14/$31.00 ©2014 IEEE

DOSN architectures circumvent the implications of leveraging unreliable resources (P2P) while still operating without a central authority. Examples for this type of approaches are Diaspora, Vis-a-Vis [5], Vegas [6] and SoNet [7]. Without global knowledge about other servers, a common choice of identifying users and their profiles is to use addresses of the form [user]@[host], following RFC 822. The host part uniquely identifies the server (usually using its DNS name), and the user part the respective participant registered on the server.

The first step towards building our search scheme is to map all atomic user properties on their registering servers. A DHT spanning all participating servers then is used to register the user properties under their server address. Hence, there is no link between the attributes that describe an individual and the user herself, but only a link to her server. Her server consequently can be discovered, when searching for the user, and contacted for further negotiation.

The contacted server then can verify the validity of the discovery request. Demonstrated with knowledge about the target subject (in a privacy preserving manner), the server can decide whether to create a valid temporary handle (Search_ID) for contacting the subject once, or to create an invalid one, pointing nowhere. The participants hence have the liberty to define a selected set (or subset) of knowledge that is needed to discover a valid temporary handle for their profile.

Our contribution in this paper is to adapt well-known techniques like DHTs, indirection schemes and proof of knowledge algorithms in an innovative and beneficial way to allow finding user handles in decentralized communication systems without facilitating SPAM. In contrast to previous solutions that leverage lookup services, we avoid to build a search index that can be maliciously exploited. The rest of the paper defines the requirements and the protocol, explains our design, gives a detailed evaluation and finally summarizes the main contributions.

II. Requirements

The central objective is to enable users in a distributed client-server environment to find user handles of communication partners without knowledge about the partner's responsible server. State of the art services are not widely adopted (e.g. public e-mail address catalogs) because of their potential facilitation of copious undesired messaging. Our scheme hence needs to meet the following requirements:

1) The service must not jeopardize the privacy of any participant, and hence no linkable data may be published.
2) The discovery scheme has to be scalable to handle large numbers of users (comparable with popular OSN) and servers.
3) The search protocol must be resistant to illegitimate user discovery. Permanent addresses, or user handles, may be retrieved through the system only upon explicit approval by the related individual.
4) The search protocol must be resistant to unsolicited mass communication. It specifically has to prevent uninformed mass address retrieval.
5) Supporting requirements 3 and 4, the scheme shall merely provide ephemeral alias addresses for single use only.

III. System Overview

This Section describes the system design and illustrates the solution space. We explain the mechanism to discover the server that is responsible for a targeted identifier, while meeting requirements 1 and 2 and subsequently specifying the registration process of profile attributes. To address requirement 3, we introduce an access control mechanism. Requirement 4 is fulfilled by restricting the served identifiers to be valid within a short term (minute scale). Requirement 5 is met by introducing an ephemeral user handle.

A. Discovery Mechanism

As a consequence of requirement 1 as well as the absence of a central authority in our system environment, we can neither build a central index to locate user handles, nor use a distributed lookup service which publishes or replicates data to discover them. Since the servers in our scenario are equal in functionality, a potential search request sender does not have a specific location to start the search procedure. Furthermore, privacy preserving negotiation for demonstrating knowledge about the search target to distinguish between legitimate and illegitimate requests is an expensive procedure (with respect to communication overhead).

Thus, the basic idea is to first locate servers which potentially host the sought user handle and subsequently perform the negotiation with a small set of servers. We use an efficient discovery mechanism which allows us to register tuples of attributes and server addresses for that purpose. Promising candidates are DHTs (e.g. CHORD [8], CAN [9]), since they are scalable and churn resistant.

Our approach implies the need to *register* a tuple of every field (just once per attribute value no matter how often it is occurring on the server) of the user profile (e.g. name, city) and the server address at the lookup service by its own.

Thus, a request to the DHT returns a list of servers, matching the field content. The servers which are part of every single list are candidates which may be the server hosting the desired contact. We will call this list the *candidate list* for the rest of the paper. This list can then be condensed by intersecting candidate lists for several search requests while increasing the detail of the search. But it terminally may yield more than a single server.

To mitigate the issue of index poisoning, the tuple of server address and the piece of search content can be signed by the server. Registering this signature together with the tuple allows us to validate whether the registration of an item is originated by the legitimate server or not.

B. Access Control

Each user defines a set or range of knowledge that is necessary to discover herself. A requesting individual then has

to demonstrate at least this minimum knowledge about the sought subject in order to get a valid Search_ID. We define a privacy preserving negotiation protocol which allows the search request sender to prove knowledge about the search target without disclosing the query to anybody but the servers, sharing the same knowledge. This prevents the untrusted nodes in the lookup service from being able to learn valid attribute combinations from search requests.

C. Ephemeral User Handles

Each provided valid Search_ID can only be used for a single message and just for a short period of time. We decided not to provide the long term valid addresses (IDs) of the subject to fulfill the misuse and access control requirements. Being contacted via short term identifier, the sought user handle owner may decide whether to reply or not on the message for disclosing the permanent user handle.

IV. Protocol

This section gives a detailed description of our protocol, which consists of the two parts of user registration and discovery. It relies on a lookup service (DHT), which is adopted by the communication servers (Diaspora, SMTP, XMPP, etc.) of all participating users.

A. Definitions

This subsection defines terms for later use in the formal protocol description: *Servers* are nodes, participating in the DHT as well as providing the underlying communication service (e.g. e-mail). *Search Fields* are tuples of properties comprising of a field name and its content, which describe a detectable individual (field "First Name" containing "Bob", for instance). Search fields are filled by participants in order to describe themselves. *The Client* is a part of the software installation. It is installed on the user's machine and is responsible for communicating with the assigned server. *The Ephemeral User Handle* is a string, which is a valid address for just one message in a short predefined period. It consists of a random string which is not guessable.

B. User Registration

An individual user $\bar{A}$, registered at the responsible server A, fills m of n search fields $f_1, f_2...f_m$, describing its profile with strings $a_1, a_2...a_m$. $\bar{A}$'s client then sends the data to the server A, which in turn registers itself for each of the descriptors (Algorithm 1).

C. User Discovery

A requesting user $\bar{A}$ describes the target subject by providing as many specified search fields $(\{(a_1, f_1), (a_2, f_2), \ldots, (a_j, f_j)\})$ as possible. $\bar{A}$'s client subsequently submits the entered information to its responsible server A, which in turn retrieves a list of candidate servers, which are responsible for users matching any of the specified Search Fields (Algorithm 2).

Having a list of servers with potential matches (the length is depending on the popularity and distribution of the search items), the user can submit requests for receiving short term IDs to servers from the candidate list. These requests contain all available knowledge about the target individual. Servers receiving this request check if a matching individual is registered and if this individual's access control policy is met by the request, i.o.w. whether the presented knowledge is sufficient to generate a Search_ID.

Algorithm 1: Registering Search Fields

Data: m of n search fields $f_1, f_2...f_m$ of a user profile with strings $a_1, a_2...a_m$ as their values
Result: m in the lookup service registered search fields

foreach (f_i, a_i) **do**
  $\tilde{a}_i = concatenation(f_i, a_i)$;
  $h_1 = hash(\tilde{a}_i)$;
  $register\ h_1\ at\ the\ DHT$
end

Algorithm 2: Search algorithm, conducted by the searcher's hosting server

Data: search fields $f_1, f_2...f_j$ with contents $a_1, a_2...a_j$
Result: List of servers which potentially host the target subject

foreach (f_i, a_i) **do**
  $\tilde{a}_i = concatenation(f_i, a_i)$;
  $h_1 = hash(\tilde{a}_i)$;
  $request\ h_1\ at\ the\ DHT$;
  $receive\ list\ a_i^l\ of\ servers,\ assigned\ to\ a_i$;
end
$candidate\_List = \bigcap_{i=1}^{n} a_i^l$

The search protocol is depicted in Fig. 1, with the following variables:

j	number of filled out search fields in the search form at the first step
k	number of hosts having at least one user matching one search field
l	number of hosts having at least one user matching all search fields; the value is the sum of all hosts resulting from the *buildIntersectionList*-operation
m	number of filled out search fields in the search form for a specific host (second step)
n	number of matching users on the specific server
p	sum of all public profile fields

D. Privacy Preserving Negotiation Algorithm

In case an adversary runs a server and registers popular data items (e.g. popular names) for advertising herself to become part of the candidate list of a user's request, she could misuse the negotiation process for learning more attributes about users. We counter this potential data leakage by suggesting the following negotiation protocol, consisting of two algorithms: one on the server side and one on the searcher's client side.

In this protocol, we propose to define a set of obligatory base attributes: name, first name and city. Our experiments

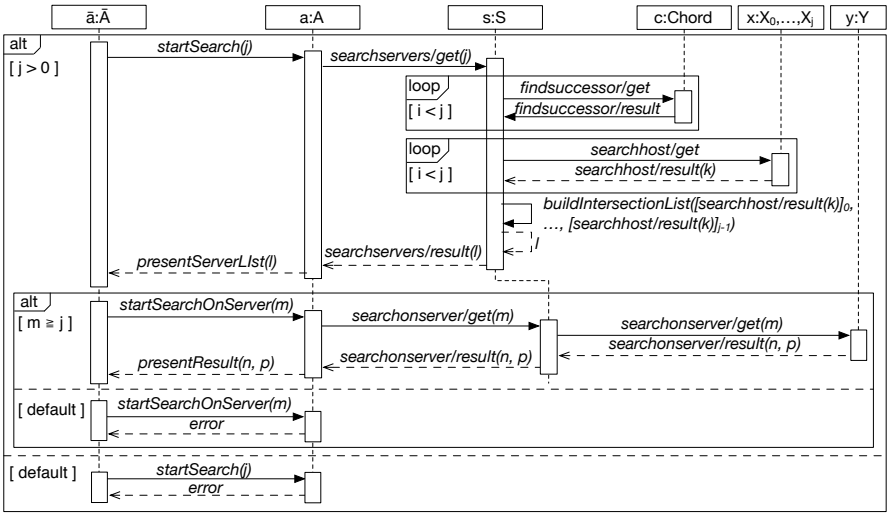

Fig. 1. Sequence diagram of the message flow during the search phase. $\overline{A}$ represents the user's Client and S the user-assigned server. $CHORD$ illustrates the lookup service. $X_n ... X_{n-1}$ are the servers, selected during the *findSuccessor* operation. j, k, l, m, n and p are the numbers of the variable parts of each message.

(Section V) show that this combination is a good choice for a real world user identifier. Furthermore, the entropy of a single attribute is small and an attacker could guess the existence of combinations of popular attributes. Our attribute concatenation (Algorithm 3) increases the number of possible combinations per server to the total number of first names multiplied by the number of last names and the number of cities. Hashing the concatenated strings keeps them secret but still allows conducting pattern matching operations on the server.

Please note that if a user realizes attackers guessing the combinations due to the fact of a popular name and big city etc., she can easily define the minimum knowledge to contain more attributes (e.g. interests, employer) and thus make guessing harder.

Algorithm 3: Client-side negotiation algorithm

Data: m of n search fields $f_1, f_2 ... f_m$ of a user profile with strings $a_1, a_2 ... a_m$ as their values and the Server_ID of the request recipient

Result: Request message content for a temporal user handle

$base = concatenation(a_1 ... a_3, Server\_ID)$;
$P(a_i) = powerset(a_i), i > 3$;
foreach $(x \text{ in } P(a_i))$ **do**
  $conc_x = concatenation(base, x)$;
  $knowledgeproof_x = hash(conc_x)$;
  $add\ knowledgeproof_x\ to\ the\ knowledgeprooflist$;
end
$add\ hash(base)\ to\ the\ knowledgeprooflist$;
$Request\ message = knowledgeprooflist$

We expect the server to have a table of user handles and the corresponding hashes, related to those users which are registered at this location. The hashes are computed according to the user defined minimum knowledge. Subsequently, the server algorithm runs a local lookup for the hashes in the "knowledgeprooflist" (the hashed representation of the requester's knowledge and a result of Algorithm 3) from the negotiation request. If a match happens, the request is replied with a valid temporal user handle, otherwise with an invalid one.

V. Evaluation

With respect to our requirements, we evaluate the functionality, the mentioned privacy properties and the scalability of our search protocol in this Section. Regarding functionality, we answer the following questions: Is it possible to find persons with three to five attributes (e.g. name, first name, city, employer) in the real world? Is the approach feasible to achieve this? Our privacy discussion answers the questions whether the protocol leaks private data. Regarding scalability we discuss: How many servers need to be contacted for negotiating with them? How much data is exchanged by applying the protocol?

A. Functionality

The only real world data collections we are aware of and which are suitable for estimating the search success of a public user handle search algorithm are phone number search engines[2,3], social networks and web search engines (like Google). Thus, we used them for estimating the amount of data which is necessary to identify a person.

Our experiments with the authors as an example, showed that the first name and the last name as search strings reduce the result list to the length of two (Thorsten Strufe) and seven (Marius Hornung). Thomas Paul is a common name, leading to the necessity of taking the city information into account.

[2]http://www.dastelefonbuch.de/
[3]http://www.teleauskunft.de/

Further real world experiments in existing user handle lookup services (see above) with about two dozens of popular German names showed that the combination of first name, last name and city usually is sufficient to find the subject. We argue that this sample size is big enough, since drawing two dozen strangers is unlikely and a search service is still useful even if a negligible minority of subjects could not be uniquely identified, just using commonly available knowledge (e.g. city, name, employer). The answer to the evaluation question whether our profile description is applicable hence is true.

To show the feasibility of our scheme, we introduce an example scenario: We assume the server landscape to have a similar structure like the e-mail system or the XMPP system because of both: a lack of reliable user and usage quantifications from DOSN and the similarity of the architectures. That means for our scenario to assume some big hubs with plenty of users, providing public-available services (e.g. Google) as well as a significant number of smaller servers[4,5] maintained by companies or non-profit organizations like universities. Algorithm 2 builds on the assumption that not every server has registered every search string (e.g. name) which occurs in the system. We assume that the server candidate list usually contains the big hubs, but just a small fraction of small servers.

We make the following assumptions for our scenario:

- The lookup service (e.g. DHT) works well in low-churn environments like ours, since this concept is well known, evaluated in previous work and tested within our prototype in a small scale
- The server popularity has the following properties which reflect the situation in the e-mail provider landscape:
  - 10 big hubs (like Google or Microsoft in the e-mail environment) concentrate the majority of users on them,
  - 10,000 smaller (about 100 users in average) servers exist, maintained by companies, organizations and communities,
  - 10,000 different values of each: the first names, last names and cities are known in the system.

The worst case assumption, is to assume that every first name, last name and city name occurs in each of the big hubs. Subsequently, applying our algorithms in our scenario would mean to have a candidate list, consisting of lass than 110 servers (out of 10,010) before starting the negotiating process.

The affordability of the negotiation process is shown in Section V-C. Contacting those (less than) 110 servers results in a list of ephemeral user handles where the overwhelming majority is pointing nowhere. Sending a contact request with a short reasoning for the contact request may motivate the desired search target to answer the request and hence disclose the permanent user handle. Please note, as shown before, the combination of first name, last name and city describes a person very well. Thus it is not likely to receive a plenty of contact requests, addressed to the wrong subject. Nevertheless in case of suffering a lot of requests, caused by e.g. a popular name in a big city, the burden of proving knowledge can be raised by adding more fields to the knowledge proof.

The presented search algorithm mitigates the problem that the peers, responsible for the most popular keys, become a bottleneck (Zipf distribution of the request popularity [10], [11]), by saving only the server's address once per popular attribute instance (e.g. popular name on one server) instead of saving each instance of the user attribute separately. Furthermore, our approach circumvents the need to store replicas of fully qualified user descriptions at other peers in the network, as proposed in [12] or [13].

B. Privacy and Security

This Section contains some privacy and security contemplations of the proposed protocol as well as some further improvements which address security concerns beyond the scope of the system design to mitigate service availability attacks. The scope of this part of the evaluation is to answer the question whether the use of the search scheme can be harmful for the user or server authority. Leaking data could affect user's privacy, coming with negative side effects far beyond the search functionality.

Attacking additional functionality just brings back the actual status before the search protocol was available. Hence, we consider attacks on the search functionality not to be an obstacle for adding this search protocol to a communication system. We thus, assume an attacker which aims to leverage the search scheme to illegitimately access private information. Since we trust friends not to publish user handles, arbitrary nodes in the network, except the own server or trusted friends, can be attacker.

The proposed search protocol does not include actions, sending any personal related information to anybody but the server, the user is assigned to. Other nodes, like the nodes in the DHT, just learn that a user named "Bob" is assigned to server A. The link between the public string "Bob" and the ID of the subject, which was sought after, has to be made by the server A. Hence we meet the requirement 1.

Even blind testing of attribute combinations does not give reliable information about single users, since knowing that a user with first name "Bob" and a user with a last name "Brown" exist, does not yield concluding that "Bob Brown" exists. The first name item and the last name item may belong to different subjects, using the same server. Armknecht et al. [14] show that the entropy of the user attributes is high enough to make guessing (dictionary attacks) unfeasible as soon as a combination of attributes is used as a key.

Requirements 3 and 4 are met by introducing the short term valid ID, since a requesting node needs to negotiate for receiving this ID, which is invalidated after a short time.

Since the design of our protocol is based on the assumption that attackers have less knowledge about the search target than legitimate request senders, this mechanism is not able to prevent well informed attackers from getting a short term

[4]http://www.mailradar.com/mailstat/

[5]http://www.zdnet.com/jabber-numbers-overtake-icq-3039117160/

valid ID. We argue that this is not an obstacle, since popular OSN like Facebook allow strangers to send friend requests as well. To mitigate that issue, we suggest to allow just one short term valid ID per requester ID. Please note that the well informed attacker still does not have access to the permanent user handle (e.g. email address) until she gets a reply message.

Two types of index poisoning could be possible. An attacker could register plenty of entries for another server to increase its load due to being involved in unsuccessful search requests. Furthermore, an attacker could create fake (sybil) IDs to register plenty of fake targets aiming at hiding the real search target. The first poisoning attack can easily be tackled by signing the DHT entries, the latter one can be mitigated by reputation schemes and tackled by abolishing the zero cost environment.

Further security limitations are the following:

- Attackers can learn users not to be part of the system if one of her attributes is not popular by sending a request to the DHT. If the server list is empty, the search target is most probably not registered.
- The number of participating servers can be estimated by requesting popular attributes and calculating the super set of all server list items.
- The popularity of servers can be estimated by evaluating the relative frequency of servers being part of the candidate list resulting from arbitrary requests to the DHT.

To summarize: Users can register to the search infrastructure without fearing to publish their user handles or any other private information. However, the functionality of the search scheme can be attacked like in other distributed search schemes as well. We argue that the presented limitations are not an obstacle for this scheme to be adopted in DOSN or XMPP systems.

C. Communication and Computational Costs

Compared with centralized approaches for finding user handles, more communication is necessary, since the searcher does not know both: the user handle as well as the server which has that knowledge. Thus, the search procedure contains the additional step for finding the responsible server. Nevertheless, our requirement 2 is that a search scheme must not become prohibitively expensive in terms of network and system load. In the latter of this Section, we show that the network load is not a prohibitive issue.

Having a given search field distribution (e.g. names and city inhabitants) and user-server affiliation frequency distribution, the final network load per search request thus depends on the efficiency of the chosen lookup service, the complexity of the search request, the number of participating servers and the number of system users (and hence are registering attributes).

The load increases less than linear with a growing number of users, since duplicate attribute instances per server do not have an impact. A growing number of servers has a linear impact as well as the length of the search field content. Nonlinear growth of load is caused by the privacy preserving negotiation algorithm while calculating the power set of the atomic parts of the search request. This is given by 2^{n-2} (minus two, because of the concatenation of three fields) where n is the number of search fields. We argue that this is not an issue (Figure 2), since we expect the number of search fields to be usually small (less than 6).

Giving a second example scenario, we assume the profile attribute length, the profile field name length, the sender address length, the host name length and the key length to be 255 bytes (=255 characters) each, which represents the worst case. Based on our prototype, we assume a Chord DHT to be the lookup service. Furthermore we assume users to fill three search fields, a fraction of 30% of the servers having at least one user with one matching attribute field and a fraction of 5% of these servers having at least one user hosting a matching user profile. Subsequently, we assume that 2 users on each server in the candidate list are matching all search fields, and each of these users filled out 5 profile fields.

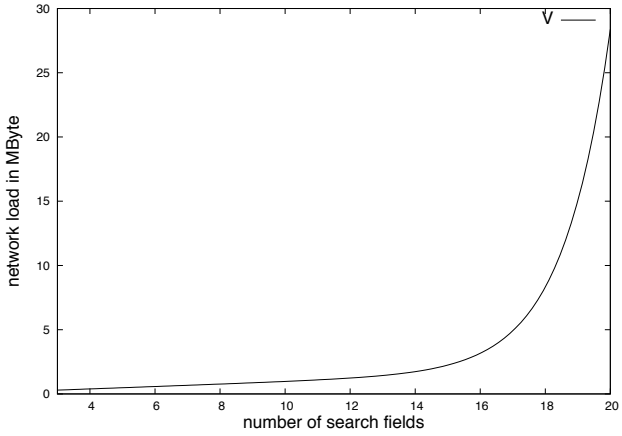

Fig. 2. Impact of growing number of search fields on our example scenario; 1,000 servers

Since we assume the maximum field lengths (URL restrictions) in our example and do not expect users to use more than three to five attributes per average search operation in reality, we assume our scenario to be a rather worse case. As a consequence of this scenario evaluation, we argue that the resource consumption is not an obstacle for deploying this protocol and thus the privacy is preserved at an affordable amount of costs. Hence we meet our requirement 2.

To prove the functionality of our scheme in a testbed, we implemented a proof-of-concept prototype by adding components to the XMPP server Tigase. The choice was driven by the openness of the latter as well as by the fact that motivating existing XMPP users to help us testing is easier than motivating people to join Diaspora* for that purpose. Creating enough load to the system for exploring its performance limits, was done by a small script for sending random requests.

Beside the result that the idea works well, we learned that the performance is sufficient not to be prohibitive even though we did no performance optimizations in our code and used comparably slow machines (2.2 GHZ AMD cores). Assuming that users may want to find in average a dozen friends in a

month, this server setup can handle more than 50,000 users at one core without further code optimizations.

VI. Related Work

The authors of Vis-a-Vis[5] suggested an DOSN approach based on virtual individual servers. Every user runs her own virtual server for profile data availability and interaction handling. A hierarchical scheme for finding user handles and group management is proposed, realized by spanning a DHT across the individual servers. Data which is used to characterize users in the search scheme is defined as "searchable" which means to be public. We consider this search scheme to be the closest to our proposed solution but it does does not meet our privacy requirements.

To the best of our knowledge, no current work exploits the properties of decentralized server architectures for finding user handles with privacy. Several approaches implement client-server based DOSN like e.g. Diaspora* and Jabbix[6], but they do not provide any integrated comprehensive user discovery. XMPP addresses can globally be found by using central user directories, based on XEP-0055 (Jabber Search), but there are "no security features or concerns related to this proposal"[7]. Server-local JID (Jabber Identifier) search is possible with the "Net::XMPP::JID - XMPP JID Module" [8] but again comes without tackling privacy and SPAM concerns.

The situation for finding e-mail addresses is even worse. Attempts to provide system-wide search functionalities did not succeed. For example, the leading German telecommunication company took the e-mail directory[9] (public, without privacy protection) offline. Hence, finding e-mail addresses causes the need for side channel communication via e.g. web pages or social networking sites.

Since finding user handles is a subproblem of finding arbitrary resources, general purpose search approaches are feasible for finding user handles as well. Common ground of these approaches (e.g. keyword search [15], XPath [16], ROAR [3], SplitQuest [17] or Bubblestorm [18]) is the public-accessibility of the search strings. Thus, these solutions do not meet our requirements 1, 3 and 4. Our solution, in contrast allows for the privacy preserving publication and discovery of users, which we achieve by two cascaded indirection schemes.

VII. Conclusion

This paper presents the first approach that allows to find user handles in systems of decentralized client-server architectures, without disclosing any data that is linked to the permanent user handle (ID or address). The novelty is to avoid building a user-linkable search index. This is realized by cutting the search strings into atomic parts, which are linked to the server handle instead of the user handle and separately registered at the lookup service. The reunification of the search results is done locally and the subsequent access to a temporal user handle is limited to requests, which demonstrate a minimum knowledge about the search target. The permanent ID (e.g. email address) is not published by this mechanisms until the user is manually responding the request and thus confirming the contact to be desired.

It hence renders offline or other side channel communication for the purpose of discovering users unnecessary and thus increases the usability of email, or jabber-like IM services. It additionally represents a step towards creating a usable distributed social networking service, based on decentralized servers since finding friends is a core functionality in today's OSN.

Our extensive evaluation includes the functionality, the privacy and security as well as the communication costs, coming with the usage of our approach. A prototype implementation based on XMPP, extending the popular jabber server Tigase, underlines feasibility and scalability of the system.

[6]https://jabbix.com/

[7]http://xmpp.org/extensions/xep-0055.html

[8]http://search.cpan.org/dist/Net-XMPP/lib/Net/XMPP/JID.pm

[9]http://www.email-verzeichnis.de/

VIII. Acknowledgements

This work has been [co-]funded by the German Research Foundation(DFG) in the Collaborative Research Center (SFB) 1053 "MAKI - Multi-Mechanism-Adaptation for the Future Internet".

References

[1] A. Narayanan and V. Toubiana, "A Critical Look at Decentralized Personal Data Architectures," *arXiv preprint arXiv:* . . . , 2012. [Online]. Available: http://arxiv.org/abs/1202.4503

[2] S. Schulz and T. Strufe, "Diaspora: Practical attacks for profile discovery and deletion," in *ICC*, 2013.

[3] C. Raiciu *et al.*, "Roar: Increasing the flexibility and performance of distributed search," in *SIGCOMM*, 2009.

[4] D. Boneh *et al.*, "Public key encryption with keyword search," in *Advances in Cryptology-Eurocrypt*, 2004.

[5] A. Shakimov *et al.*, "Vis-a-vis: Privacy-preserving online social networking via virtual individual servers," in *COMSNETS*, 2011.

[6] M. Durr *et al.*, "Vegas – a secure and privacy-preserving peer-to-peer online social network," in *SocialCom*, 2012.

[7] L. Schwittmann *et al.*, "Sonet–privacy and replication in federated online social networks," in *HotPOST*, 2013.

[8] I. Stoica *et al.*, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM*, 2001.

[9] S. Ratnasamy *et al.*, "A scalable content-addressable network," in *ACM SIGCOMM*, 2001.

[10] K. P. Gummadi *et al.*, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *SOSP*, 2003.

[11] L. Breslau *et al.*, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM*, 1999.

[12] L.-A. Cutillo *et al.*, "Safebook: a privacy preserving online social network leveraging on real-life trust," *IEEE Communications Magazine*, 2009.

[13] K. Rzadca *et al.*, "Replica placement in p2p storage: Complexity and game theoretic analyses," in *IEEE ICDCS*, 2010.

[14] F. Armknecht *et al.*, "Protecting Public OSN Posts from Unintended Access," in *ICC 2014*.

[15] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *Middleware*, 2003.

[16] A. Bonifati *et al.*, "Xpath lookup queries in p2p networks," in *WIDM*, 2004.

[17] P. Lopes and R. A. Ferreira, "Splitquest: controlled and exhaustive search in peer-to-peer networks," in *IPTPS*, 2010.

[18] C. Leng, "Bubblestorm: Replication, updates, and consistency in rendezvous information systems," Ph.D. dissertation, Technische Universität Darmstadt, 2012.