

# Distributed and Domain-Independent Identity Management for User Profiles in the SONIC Online Social Network Federation

Sebastian Göndör, Felix Beierle, Senan Sharhan, and Axel Küpper

Telekom Innovation Laboratories, TU Berlin  
Service-centric Networking

Email: sebastian.goendoer@tu-berlin.de, beierle@tu-berlin.de,  
senan.mh.sharhan@campus.tu-berlin.de, axel.kuepper@tu-berlin.de

**Abstract.** As of today, communication habits are shifting towards Online Social Network (OSN) services such as *WhatsApp* or *Facebook*. Still, OSN platforms are mostly built in a closed, proprietary manner that disallows users from communicating seamlessly between different OSN services. These lock-in effects are used to discourage users to migrate to other services. To overcome the obvious drawbacks of proprietary protocols and service architectures, SONIC proposes a holistic approach that facilitates seamless connectivity between different OSN platforms and allows user accounts to be migrated between OSN platforms without losing data or connections to other user profiles. Thus, SONIC builds the foundation for an open and heterogeneous *Online Social Network Federation* (OSNF). In this paper, we present a distributed and domain-independent ID management architecture for the SONIC OSNF, which allows user identifiers (GlobalID) to remain unchanged even when a profile is migrated to a different OSN platform. In order to resolve a given GlobalID to the actual URL of a social profile the Global Social Lookup System (GSLs), a distributed directory service built on peer to peer technology is introduced. Datasets called *Social Records*, which comprise all information required to look up a certain profile, are stored and published by the GSLs. Following this approach, social profiles can be migrated between OSN platforms without changing the user identifier, or losing connections to other users' social profiles.

## 1 Introduction

As of today, a strong trend can be observed that shows that communication habits are shifting towards Instant Messaging (IM) and Online Social Networks (OSN). While old fashioned communication habits such as voice calls are declining, usage of OSN and IM services is steadily rising [1][2]. OSN platforms allow their users to communicate via text, audio, and video, share content, or just stay in contact with friends and relatives. While a large number of competing OSN platforms with a broad variety of features exist as of today, *Facebook*, which was founded in 2004, managed to overcome its predecessors and competitors by far

in terms of number of users and popularity [3], and continues to be the world leader in terms of users accessing the service [4]. Competitors were forced out of the market or had to focus on niche markets such as modeling relations to business partners (e.g., *LinkedIn* and *Xing*) or to address different aspects of social interactivity (e.g., communication via *WhatsApp*) or activities (e.g., publishing images via *Instagram*). Most OSN designs promote a closed, proprietary architecture that disallows users from communicating seamlessly between different OSN services. The well-calculated lock-in effects of proprietary platforms are used to bind users to the service, as migrating to another OSN platform would result in a loss of connections to friends and the data one has accumulated as part of his social profile [5]. Alternative OSN architectures propose a federation of servers or make use of peer to peer technology to distribute control over the social graph and associated data [6][7]. Still, communication between different OSN platforms is mostly not possible, or just enabled via special plugins or services, which are used to replicate data between different accounts of the same user on different OSN platforms [8]. To overcome the obvious drawbacks of proprietary protocols and architectures in the area of OSN services, SONIC [9] proposes a holistic approach that facilitates seamless connectivity between different OSN platforms and allows user accounts to be migrated between OSN platforms without losing data or connections to other user profiles. Following the *Interop* theory [10], the vision of SONIC proposes an open and heterogeneous *Online Social Network Federation* (OSNF), in which social profiles are managed independently from the platform they are hosted on [11]. To allow seamless and transparent communication between different OSN platforms, identification of user profiles as well as resolving identifiers to a profile’s actual location is a crucial task. As profiles may be migrated at any time, identifiers that are bound to a domain name of a OSN platform cannot be employed. Hence, identifiers in SONIC need to be domain agnostic and created in a distributed fashion. This allows users to keep their identifier even after migrating to a new OSN platform on a different domain. Anyhow, introducing domain agnostic global identifiers requires for means of resolving an identifier to the current network location of the respective social profile. For this reason, SONIC introduces the Global Social Lookup System (GSLS), a distributed directory service built on peer to peer technology using distributed hash tables (DHT).

In this work, we present an identification architecture for decentralized OSN ecosystems. The architecture features GlobalIDs as domain agnostic, globally unique identifiers, which can be generated in a distributed fashion without the need for a central authority. The architecture introduces a distributed directory service, the GSLS, which is utilized to resolve GlobalIDs to a user profile’s actual location. The GSLS manages a digitally signed dataset, the *Social Record*, which comprises information about the social profile identified by the GlobalID. Following this paradigm, social user profiles can be identified independently of the OSN platform’s operator. Furthermore, users can change the location of their profile at any time without losing connections between other social profiles [11]. The architectural requirements for the SONIC federation have been defined

in [12], comprising a decentralized architecture, the use of open protocols and formats, the option for users to migrate their social accounts, seamless communication, the use of a single social profile, and global user identification. The remainder of this paper is organized as follows: The following chapter provides an overview about existing approaches, protocols, and standards in the area of identity management. Chapter 3 gives an overview of the concept of the SONIC federation, followed by a description of the identity management architecture in Chapter 4. Chapter 5 describes the implementation of the proposed solution, which is evaluated in Chapter 6. Chapter 7 concludes the paper.

## 2 Related Work

Services that manage multiple users or objects require a measure of identification to distinguish between individual users or objects. For this purpose, an identifier is assigned to each entity, where an identifier is a name that can be a word, letter, number, or a combination of those, with the usual intent of being unique in a certain domain. This assures that each user or object can be uniquely addressed via its identifier, and two equal entities can be distinguished. In applications and services that are used by multiple users, each user is traditionally assigned a user name, which is unique in the domain of the application or service. A well known example is the Linux operating system, where each user gets to chose a unique user name and a serial number (uid). The uid is used by the system to identify users, while the actual user name is mostly used for authentication and displaying purposes. Social applications and services also usually identify users by a numerical user identifier, which in most cases has to be unique within the domain of this service or application. In addition, most services allow their users to pick a display name, which is shown to other users. This display name is then not necessarily used as an identifier, but as a normal name. As of this, the display name is not necessarily unique and functions similar to a given name.

While issuing and resolving user identifiers within the same domain is comparatively easy, identifying entities across different domains is a more complex task. Here, usually composed identifiers are used that comprise a local identifier, which is unique in its issuing domain, and a domain identifier that uniquely identifies the domain. This way, a local user name "Marc" can exist in two separate domains at the same time, while only the domain name is required to be unique. This kind of composed identifiers is used by most Internet-based services or applications, where the domain identifier is the full qualified domain name (FQDN) of the service. One example are email addresses [13] or jabber-ids (JID) as employed by XMPP in the format `local-id@domain-id` [14]. Resolving this kind of composed identifiers depends on the Domain Name Service (DNS) [15], which is required to resolve the domain part of the identifier, while the local user name is resolved by the service itself.

Similar to this identifier format, Unified Resource Identifiers (URI) or International Resource Identifiers (IRI) [16] can be used to uniquely identify an entity or person. Here, a path can be specified to further describe categories or a classes

of identities, e.g., `http://company.com/berlin/employees/alice`. By utilizing actual URLs as identifiers, users and services can easily resolve an identifier to e.g., a document, which provides further information about the linked entity. This approach is employed by e.g., WebID [17], where a URI is resolved to a profile document using the DNS. The protocol WebID+SSL [18] further involves exchange and verification of encryption keys to establish a trusted and secure connection between two individuals. Also the authentication protocol OpenID employs URIs as user identifiers [19]. While the advantage of these kinds of composed identifiers are that services can freely assign user names for identification purposes, identifiers created in this fashion are bound to the domain they were created in, and hence cannot be migrated to another domain.

In scenarios, where entities need to be identified independently of a fixed domain or service, different approaches have to be applied. To avoid collision of identifiers created without coordination of the id generating services, randomness can be used to make a collision unlikely. Following this approach, cryptographic hash functions are used to create a random number from a combination of deterministic or random input values. Universally Unique Identifiers (UUID) - also known as Globally Unique Identifiers (GUID) - are 128 bit identifiers created by using hash algorithms [20]. The UUID standard defines 4 types of identifiers. Depending on the type of the UUID, different data is used for its creation. For example, a version 1 UUID uses the machine's MAC address and datetime of creation, while version 5 uses SHA1 with a namespace part. The uniqueness of UUIDs is based on the assumption that generating the same UUID twice is very unlikely. In 2003, the OASIS group introduced eXtensible Resource Identifiers (XRI) as an identifier scheme for abstract identifiers [21]. XRIs are designed to be domain-, location-, and platform-independent and can be resolved to an eXtensible Resource Descriptor Sequence (XRDS) document via HTTP(S). Work on the XRI 2.0 specification was discontinued in 2008 by the XRI Technical Committee at OASIS. Twitter Snowflake [22] is an identifier schema based on hashing a timestamp, a preconfigured machine number, and a sequence number. Twitter Snowflake was built for fast and distributed id generation without the need for the machines generating the ids to coordinate with each other. Snowflake was discontinued in 2010, but other implementations of the approach exist, e.g. PHP Cruftflake [23]. Boundary Flake, which follows a similar approach as Twitter Snowflake, is a "decentralized, k-ordered id generation service" [24]. Here, the machine's MAC address, a UNIX timestamp, and a 12 bit sequence number are hashed to create a 128-bit identifier. In comparison to composed identifiers, distributed identifiers can be generated in a distributed fashion, i.e., without a central control entity. Anyhow, verification of an entity's identity might be problematic, as any entity can assume any ID. To circumvent this, distributed entity's need to be resolvable in a trusted and secure manner.

To verify an identity, identifiers are usually resolved to a data record or a document comprising further information about the identified entity. Usually, such data records are maintained in a network-based database and made accessible to authorized clients by a directory service. In directory services, data

records (entries) are organized in a hierarchical structure, where each entry has a parent entry. Each entry is identified by a distinguished name (DN), which is not necessarily unique. Therefore, each entry is uniquely identified by its path from the root entry, the relative distinguished name (RDN). As entries might be shifted to another branch or level in the tree-like structure, its RDN is not guaranteed to remain stable. An existing and widely used standard for directory services is the Lightweight Directory Access Protocol (LDAP) [25] [26] [27] based on the ITUT standard X.500 [28]. One of the most used and well known directory services is the Domain Name System (DNS) [15] [29]. The DNS is a hierarchically and decentrally organized directory service, that allows users and services to resolve human readable domain names into IP addresses, therefore mapping a name to a location. The data is stored in resource records (RR), which are replicated throughout the system. Still, both LDAP and the DNS build on a hierarchical design, which requires one organization or company to maintain control. To circumvent certain drawbacks and security issues in the DNS, Distributed Hash Tables (DHT) have been adopted for the use of directory services. In [30], Ramasubramanian and Sirer propose a DHT-based alternative for the DNS. This approach provides equal performance as the traditional hierarchical DNS, but showed a far better resilience against attacks [31].

### 3 The SONIC OSN Federation

As today's OSN platforms are mostly closed solutions that keep users from freely communicating and connecting with each other, several alternative solutions and architectures have been proposed over the last years. Here, either alternative centralized OSN platform solutions were built or ones relying on federated or completely decentralized architectures [6]. Anyhow, all proposed alternatives require a user to create a new user account within the new system, while seamless interaction with other OSN platforms is mostly still not possible. The motivation for users to abandon one closed system for another closed solution is therefore limited. In contrast to the approaches mentioned above, SONIC proposes a different paradigm. Here, a common protocol is used to allow different kinds of OSN platforms to interact directly. Following this approach, OSN platforms support a common API and protocol, which allows to exchange social information across platform borders, while addressing remotely hosted user accounts directly. This way, users can choose the OSN platform they prefer while staying seamlessly connected to all friends using other OSNs. Hence, it is rendered irrelevant whether a user's friends are using the same or a different OSN platform. The result is an *Online Social Network Federation (OSNF)* defined as a *heterogeneous network of loosely coupled OSN platforms using a common set of protocols and data formats in order to allow seamless communication between different platforms* [12]. Prerequisites for an OSNF ecosystem comprise a *decentralized architecture*, the use of *open protocols and formats*, *seamless communication* between platforms, *migration* of user accounts to other OSN platforms [11], and a *single profile* policy with *global user identification* [12].

## 4 User Identification

In the SONIC OSNF, every user and every platform is identified by a globally unique identifier, the GlobalID. GlobalIDs are domain and platform independent and remain unchanged even when a user account is moved to a new domain. This way, a user account can be addressed regardless of where it is actually hosted. Furthermore, migration of user profiles is made possible without losing connectivity between social user accounts - even when the location of a profile is changed frequently [11]. A user's GlobalID is derived from an PKCS#8-formatted RSA public key and a salt of 8 bytes (16 characters) length using the key derivation function PBKDF#2 with settings SHA256, 10000 iterations, 256bit output length. The result is converted to base36 (A-Z0-9), resulting in a length of up to 50 characters length (see Figure 1). An example of a GlobalID is 2UZCAI2GM45T160MDN440IQ8GKN5GGCK096LC9Z0QCAEVAURA8. Each entity in the SONIC ecosystem maintains two RSA key pairs, the *PersonalKeyPair* and the *AccountKeyPair*. While the *PersonalKeyPair* is used to derive the GlobalID, the *AccountKeyPair* is used to sign and verify all communication payload data within SONIC. As a result, the *PersonalKeyPair* can never be changed while *AccountKeyPairs* can be revoked and exchanged with a new key pair. GlobalIDs are registered in a global directory service, the Global Social Lookup System (GSLS). By resolving a GlobalID via the GSLS, the actual network location (URL) of a user's account can be determined. Information about the actual profile's location, as well as other information required for verification of authenticity and integrity are stored in a dataset called *Social Record*.

### 4.1 Global Social Lookup System

Following the idea of a fully decentralized OSN ecosystem that does not depend on any entity or service controlled by a single corporation or group, the GSLS was designed as a directory service built on DHT technology. Similar to the DNS, any participant in the SONIC ecosystem is able to host a GSLS server that is automatically integrated into the DHT, forming a dynamic, heavily distributed directory service. The GSLS operates as a global directory service with a REST-based interface for read and write operations as described in Table 1. As data in the GSLS is public and may be overwritten by unauthorized entities, the data

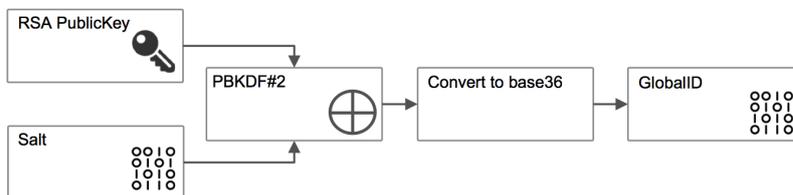


Fig. 1: Creation of a GlobalID.

is digitally signed using the user’s *PersonalKeyPair*. As the GlobalID is derived directly from the enclosed public key and the salt, unauthorized changes in the payload would result in either an invalid digital signature or - in case the key pair is exchanged - an altered GlobalID.

## 4.2 The Social Record

The GlobalID and associated information is published in a dataset called *Social Record*, which comprises information that is required to resolve the GlobalID to the profile’s location. The actual contents of the *Social Record* are described in Table 2. For security reasons, the GSLS API requires data to be formatted as a signed JSON Web Token (JWT [32]) using RS512 to digitally sign the payload using the owner’s *PersonalKeyPair*. The *Social Record* data itself is a private claim named `socialRecord` and has to be a serialized, Base64URL-encoded JSON object. The digital signature of the JWT is created using the *PersonalPrivateKey*, so the signature can be verified by everyone using the *PersonalPublicKey*, which is included in the signed dataset. In case that the *AccountKeyPair* should be revoked, a key revocation certificate is created. Similar to [33], this certificate comprises the revoked public key, date and time of the revocation, a numerical indication of the reason for the revocation, and a digital signature. All revocation certificates are published in the *Social Record*, while the outdated *AccountKeyPair* is replaced with a new one.

GlobalIDs are generated in a distributed fashion. Hence, without a central authority controlling the process, attacks are possible that aim at taking over a SONIC identity. As GlobalIDs are derived directly from the *PersonalKeyPair* and salt, an attacker would need to create a valid signature for a crafted *Social Record*. This would mean that an attacker would need to get access to the *PersonalPrivateKey* itself. Replacing the *PersonalKeyPair* itself is not possible, as exchanging the key pair would result in an altered GlobalID. As the GlobalID is used for resolving the *Social Record*, this would deflect the attack. As GlobalIDs are derived from an RSA public key using SHA256, it’s uniqueness depends on the security of the key generation. To prevent attackers from creating rainbow tables for all available GlobalIDs in order to find a collision for a random *Social Record*, a cryptographic salt has been introduced. This salt is used by PBKDF#2 in

Method	Path	Description
GET	/	Request a status message from the GSLS node
GET	/:globalID	Retrieves the <i>Social Record</i> as a signed JWT for the specified GlobalID.
POST	/	Sends a new <i>Social Record</i> as a signed JWT to be stored in the DHT.
PUT	/	Sends a new version of an already existing <i>Social Record</i> as a signed JWT to the DHT. The already existing version will be overwritten.

Table 1: GSLS REST interface

Attribute	Description
<code>type</code>	Type of the <i>Social Record</i>
<code>globalID</code>	The identifier for the user profile
<code>platformGID</code>	GlobalID of the associated OSN platform
<code>displayName</code>	Human-readable username for on screen display
<code>salt</code>	Cryptographic salt of 16 characters length
<code>accountPublicKey</code>	RSA public key
<code>personalPublicKey</code>	RSA public key
<code>datetime</code>	XSD DateTime timestamp
<code>keyRevocationList</code>	List of revoked account key pairs
<code>active</code>	Flag that describes the current status of the <i>Social Record</i>

Table 2: Contents of the *Social Record*

the generation process of the GlobalID. The usage of the salt, which is randomly created for each *Social Record*, aggravates brute force attacks as a new key cannot be checked against multiple *Social Records* for a collision, but needs to be hashed again for each GlobalID. Anyhow, as generating an RSA key pair is the most time consuming task in creating a GlobalID, an attacker might chose a key and just alter the salt in order to find a collision. To limit the possibility of this attack to succeed, the length of the salt has been fixed to 8 bytes. By limiting the length of the salt, only  $4.2 \times 10^9$  possible salts can be used, thus effectively eliminating the chance of creating a collision through manipulation of the salt. Using the birthday bound, an attacker would need to create  $4.8 \times 10^{37}$  key pairs and salts for a 1% chance of a collision, thus rendering an attack extremely unlikely.

## 5 Implementation

This section describes the implementation details of the GSLS. It has been implemented as a Java server daemon based on Eclipse Jetty, a lightweight application server capable of handling REST requests. The application is run via Jsrv to run as a server daemon. The GSLS exposes a REST-based interface on port 5002 that allows clients to commit and request *Social Records*. The interface features operations for retrieving and writing *Social Records* as described in Table 1. For storage of the *Social Records*, the GSLS implements TomP2P, a Kademlia-based DHT implementation written in Java [34]. Kademlia is based on a reactive key-based routing protocol, which uses other node's search queries to update and stabilize the routing tables. As a result, Kademlia-based DHTs are very robust and performant, as separate stabilization mechanisms are not necessary [35].

To prevent manipulation of the dataset by malicious participants, the dataset is stored as a signed JSON Web Token (JWT). The token is signed using RS512. For compatibility reasons, the dataset is encoded using Base64URL and stored in the JWT as a private claim named `data`. The token is then signed with the private key matching the enclosed public key. This way, the integrity of the

dataset can always be verified. *Social Record* datasets sent to the GSLS will be validated by the service regarding integrity and format to ensure that no faulty datasets are managed or delivered by the GSLS. The API allows no DELETE requests, as a hard delete would allow a previously occupied GlobalID to be reused by a new *Social Record* with a matching GlobalID. Even though being unlikely, identity theft would be made possible this way. As of this, the GSLS only supports a soft delete, where the `active` flag of the *Social Record* is set to 0 to mark the dataset as inactive.

## 5.1 SONIC SDK

In order to ease the integration of the SONIC protocol into both existing OSN platforms as well as to support the development of new OSN projects, the SONIC SDK has been implemented. The SDK features a set of classes that provide functionality for formatting, parsing, signing, and validating SONIC data formats, as well as handling requests to and from other SONIC compliant platforms. For resolving GlobalIDs, the SONIC SDK incorporates an API to retrieve *Social Records* from the GSLS, as well as for creating, publishing, and updating *Social Records*. The SONIC SDK automatically resolves GlobalIDs in the background, including an automated integrity verification process. The SDK has been integrated in a proof of concept implementation of the SONIC project as well as in the well-known OSN platform *Friendi.ca*.

## 5.2 SONIC App

To ease the management of the *Social Record* and the associated key pairs for the user, the SONIC App has been implemented as a mobile application based on Android (see Figure 2). The SONIC App allows a user to create both the *Social Record* and associated key pairs and is able to synchronize the data with the GSLS, as well as with the user's SONIC platform. While the *AccountKeyPair* needs to be accessible by the platform in order to sign data as well as requests, the *PersonalKeyPair* is only used to sign the *Social Record*. Using the SONIC App, the *PersonalKeyPair* is managed by a device owned by the user and is not made available to the platform or any other - possibly untrusted - third party. To ease the setup process when creating a user account on a SONIC platform, the SONIC App automates the creation of keys. Here, the platform displays a QRCode encoding the login credentials of the platform, which can be scanned by the SONIC App. After creating a new *Social Record* and the associated keys, the SONIC App uploads the necessary data to the platform. Besides creating a new *Social Record* or editing an existing one, the SONIC App also automates the process of migrating a social profile to a new platform as described in [11]. Here, a new user account is created at the target platform and all profile information is copied to the target location. As part of this migration protocol, the SONIC App will automate the process of updating the *Social Record* in the GSLS. Further features of the SONIC App include exporting a *Social Record* to a text file,

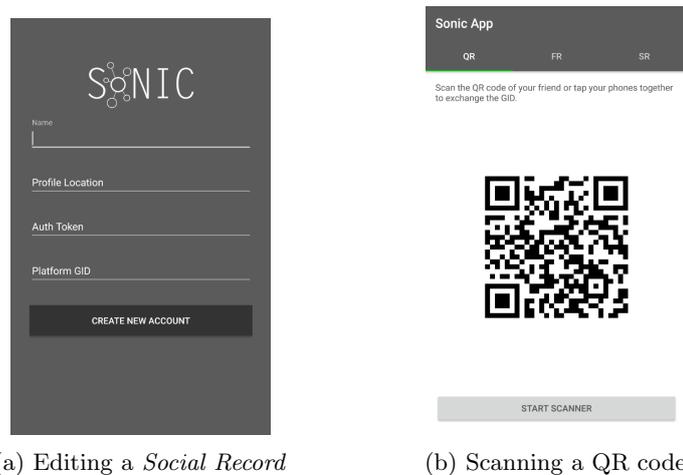


Fig. 2: User Interface of the SONIC App.

importing a *Social Record* from a text file, and scanning a QRCode encoding the GlobalID of another user in order to directly send a friend request to him.

## 6 Evaluation

For the evaluation of the GSLS, a testbed with 3 virtual machines has been set up. Each node was configured to use 1 virtual CPU and 1 GB of RAM, running Debian Linux "Wheezy". To perform the evaluation of the writing performance of the system, 50,000 unique *Social Records* were created by a script and directly pushed to the GSLS. For each *Social Record* dataset being sent to the GSLS, the total duration of the request to complete was measured and logged to a database for later analysis (see Table 3). Each write request comprised a payload of approximately 4KB depending on the *Social Record's* contents.

Analysis of the logged data showed that most requests were fully processed in approximately one second, with a minimum of 0.956 seconds and an average of 2.312 seconds (median value 1.032 seconds). While 30.8% of all requests were processed in less than a second, 89.6% of all requests were processed in less than 2 seconds. Only 4.9% of the requests took more than 3 seconds and 3.6% of the requests took more than 6 seconds. Even though the overall writing performance of the GSLS can be considered good, a small fraction of requests took a - partly significant - longer amount of time to complete. As no request timeout was configured on both server and client side during the test, the client waited until a response was received. Here, response times of up to 227.548 seconds were measured. To perform an evaluation of the reading performance of the GSLS, 10,000 requests for randomly chosen GlobalIDs for existing *Social Records* were sent to the one of the nodes. Again, all requests were answered successfully. The average response time for the requests was found to be 0.034 seconds with

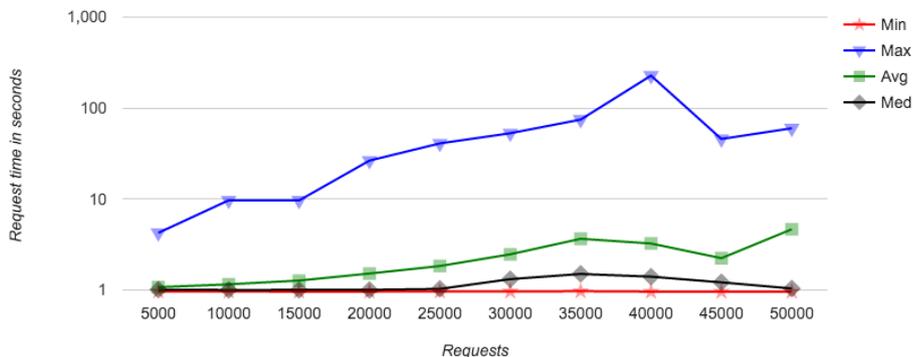


Fig. 3: GSLS writing performance for 50,000 consecutive write requests.

a minimum of 0.009 seconds and a maximum of 4.085 seconds. The median time to answer a request took 0.014 seconds. While the reading performance of the GSLS while accessing stored *Social Records* showed to be stable and fast, writing new datasets to the DHT showed a slower performance. Still, the median response time for a successful request was 1.032 seconds, with few requests that took longer to complete.

## 7 Conclusion

To overcome the obvious drawbacks of proprietary protocols and service architectures, SONIC proposes a holistic approach that facilitates seamless connectivity between different OSN platforms and allows user accounts to be migrated between OSN platforms without losing data or connections to other user profiles. Thus, SONIC builds the foundation for an open and heterogeneous *Online Social Network Federation*. In this paper, we presented a distributed and domain-independent ID management architecture for the SONIC OSNF, which allows user identifiers to remain unchanged even when a profile is migrated to a different OSN platform. These so called GlobalIDs are derived from a public key pair using PBKDF#2 and are therefore domain-agnostic. In order to resolve a given GlobalID to the actual URL of a social profile the GSLS, a distributed directory service built on DHT technology has been introduced. Datasets called *Social Records*, which comprise all information required to look up a certain profile, are stored and published by the GSLS. For security reasons, *Social Records* are digitally signed using the user's private key. For easing key management and exchanging of GlobalIDs, a mobile SONIC app has been implemented based on Google Android. This application allows to create, edit, import, and export *Social Records*, exchange GlobalIDs, and directly send friend requests using the SONIC protocol [9].

## Acknowledgment

The work described in this paper is based on results of ongoing research and has received funding from the research projects SONIC (<http://sonic-project.net>) and reThink (<http://rethink-project.eu>). SONIC (grant number 01IS12056) is funded as part of the *SoftwareCampus* initiative by the *German Federal Ministry of Education and Research (BMBF)* in cooperation with *EIT ICT Labs Germany GmbH* and *Deutsches Luft- und Raumfahrtzentrum (DLR)*. reThink (grant number 645342) is funded as part of the European Union's research and innovation program *Horizon 2020*.

## References

1. Perrin, A.: Social Media Usage 2005-2015 (2015) [http://www.pewinternet.org/files/2015/10/PI\\_2015-10-08\\_Social-Networking-Usage-2005-2015\\_FINAL.pdf](http://www.pewinternet.org/files/2015/10/PI_2015-10-08_Social-Networking-Usage-2005-2015_FINAL.pdf).
2. Ofcom: Communications Market Report 2012 (2012) [http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr12/CMR\\_UK\\_2012.pdf](http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr12/CMR_UK_2012.pdf).
3. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The Anatomy of the Facebook Social Graph. arXiv preprint arXiv:1111.4503 (2011)
4. Cosenza, V.: World map of social networks (2016) <http://vincos.it/world-map-of-social-networks/>.
5. Yeung, C., Liccardi, I., Lu, K., Seneviratne, O., Berners-Lee, T.: Decentralization: The Future of Online Social Networking. In: W3C Workshop on the Future of Social Networking Position Papers. Volume 2. (2009)
6. Paul, T., Famulari, A., Strufe, T.: A Survey on Decentralized Online Social Networks. *Computer Networks* **75, Part A** (2014) 437–452
7. Heidemann, J.: Online Social Networks - Ein sozialer und technischer Überblick. *Informatik-Spektrum* **33**(3) (2010) 262–271
8. Hu, P., Fan, Q., Lau, W.C.: SNSAPI: A Cross-Platform Middleware for Rapid Deployment of Decentralized Social Networks. arXiv preprint arXiv:1403.4482 (2014)
9. Göndör, S., Beierle, F., Sharhan, S., Hebbo, H., Küçükbayraktar, E., Küpper, A.: SONIC: Bridging the Gap between Different Online Social Network Platforms. In: Social Computing and Networking (SocialCom), 2015 IEEE 8th International Conference on, IEEE (2015)
10. Palfrey, J.G., Gasser, U.: Interop: The promise and perils of highly interconnected systems. Basic Books (2012)
11. Göndör, S., Beierle, F., Küçükbayraktar, E., , Hebbo, H., Sharhan, S., Küpper, A.: Towards Migration of User Profiles in the SONIC Online Social Network Federation. In: ICCGI, IARIA (2015) 1–2
12. Göndör, S., Hebbo, H.: SONIC: Towards Seamless Interaction in Heterogeneous Distributed OSN Ecosystems. In: Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on, IEEE (2014) 407–412
13. Resnick, P.: Internet Message Format (2008) <https://tools.ietf.org/html/rfc5322>.
14. Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence (2004) <http://tools.ietf.org/html/rfc3921>.
15. Mockapetris, P.: Domain Names - Concepts and Facilities (1987) <https://tools.ietf.org/html/rfc1034>.

16. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax (2005) <https://www.tools.ietf.org/html/rfc3986>.
17. W3C: WebID 1.0 Web Identity and Discovery (2013) <http://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html>.
18. Story, H., Harbulot, B., Jacobi, I., Jones, M.: FOAF+SSL: Restful Authentication for the Social Web. In: Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009). (2009)
19. Recordon, D., Reed, D.: OpenID 2.0: A Platform for User-Centric Identity Management. In: Proceedings of the Second ACM Workshop on Digital Identity Management. DIM '06, ACM (2006) 11–16
20. Leach, P., Mealling, M., Salz, R.: A Universally Unique Identifier (UUID) URN Namespace (2005) <http://tools.ietf.org/html/rfc4122>.
21. Reed, D., McAlpin, D.: Extensible Resource Identifier (XRI) Syntax V2.0 (2005) <https://www.oasis-open.org/committees/download.php/15377>.
22. Demir, B.: Twitter Snowflake (2010) <https://github.com/twitter/snowflake>.
23. Gardner, D., Vasconcelos, L.: Cruftflake (2015) <https://github.com/davegardnerisme/cruftflake>.
24. Featherston, D., Debnath, S., Nyman, T., Veres-Szentkirlyi, A., Countryman, M.: Boundaryflake (2015) <https://github.com/boundary/flake>.
25. Wahl, M., Howes, T., Kille, S.: Lightweight Directory Access Protocol (v3) RFC 2251 (1997) <http://www.ietf.org/rfc/rfc2251.txt>.
26. Zeilenga, K.: Lightweight Directory Access Protocol (LDAP) Transactions RFC 5805 (2010) <http://tools.ietf.org/html/rfc5805>.
27. Sermersheim, J.: Lightweight Directory Access Protocol (LDAP) The Protocol RFC 4511 (2006) <http://tools.ietf.org/html/rfc4511>.
28. International Telecommunication Union (ITU-T): X.500: Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services (2012) <http://www.itu.int/rec/T-REC-X.500/en>.
29. Mockapetris, P.: Domain Names - Implementation and Specification (1987) <https://tools.ietf.org/html/rfc1035>.
30. Ramasubramanian, V., Sirer, E.G.: The Design and Implementation of a Next Generation Name Service for the Internet. ACM SIGCOMM Computer Communication Review **34**(4) (2004) 331–342
31. Massey, D.: A Comparative Study of the DNS Design with DHT-Based Alternatives. In: Proceedings of IEEE INFOCOM'06. (2006)
32. Jones, M., Bradley, J., Sakimura, N.: JSON Web Token (JWT). Technical report, IETF (2015) <http://tools.ietf.org/html/rfc7519>.
33. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polkk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (2008) <https://tools.ietf.org/html/rfc5280>.
34. Bocek, T.: TomP2P, a P2P-based high performance key-value pair storage library (2012) <http://tomp2p.net>.
35. Maymounkov, P., Mazières, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. IPTPS '01, London, UK, UK, Springer-Verlag (2002) 53–65

## Additional Information

<b>Bibliographic Data</b>	S. Göndör, F. Beierle, S. Sharhan, and A. Küpper, "Distributed and domain-independent identity management for user profiles in the sonic online social network federation," in <i>International Conference on Computational Social Networks (CsoNet)</i> . Springer, 2016, pp. 226-238.
<b>Pre-print from</b>	<a href="https://beierle.de">https://beierle.de</a>
<b>Online at</b>	<a href="http://dx.doi.org/10.1007/978-3-319-42345-6_20">http://dx.doi.org/10.1007/978-3-319-42345-6_20</a>
<b>Authors</b>	Sebastian Göndör    Felix Beierle    Senan Sharhan   Axel Küpper
<b>BibTeX</b>	<pre>@inproceedings{Goendoer2016CSoNet, title = {Distributed and Domain-Independent Identity Management for User Profiles in the SONIC Online Social Network Federation}, author = {Göndör, Sebastian and Beierle, Felix and Sharhan, Senan and Küpper, Axel}, booktitle = {{International Conference on Computational Social Networks (CsoNet)}}, publisher = {Springer}, year = {2016}, pages = {226-238}, isbn = {978-3-319-42344-9}, doi = {10.1007/978-3-319-42345-6_20}, url = {http://link.springer.com/chapter/10.1007/978-3-319-42345-6_20} }</pre>