WARP: A ICN architecture for social data

Fabio Angius*, Cedric Westphal[†‡], Mario Gerla*, Giovanni Pau*[§]

*Department of Computer Science, University of California, Los Angeles, California (USA)
[†]Innovation Center, Huawei, Santa Clara (USA)
[‡]Department of Computer Engineering, University of California, Santa Cruz
[§] Universitè Pierre et Marie Curie (UPMC) - LIP6, Sorbonne Universites - Paris, France.
<fangius, gerla>@cs.ucla.edu, cedric.westphal@huawei.com, giovanni.pau@lip6.fr

***Abstract*—Social network companies maintain complete visibility and ownership of the data they store. However users should be able to maintain full control over their own content. For this purpose, we propose WARP, an architecture based upon Information-Centric Networking (ICN) designs, which expands the scope of the ICN architecture beyond media distribution, to provide data control in social networks. The benefit of our solution lies in the lightweight nature of the protocol and in its layered design. With WARP, data distribution and access policies are enforced on the user side. Data can still be replicated in an ICN fashion but we introduce control channels, named *thread updates*, which ensures that the access to the data is always updated to the latest control policy. WARP decentralizes the social network but still offers APIs so that social network providers can build products and business models on top of WARP. Social applications run directly on the user's device and store their data on the user's *butler* that takes care of encryption and distribution. Moreover, users can still rely on third parties to have high-availability without renouncing their privacy.**

I. Introduction

Online social networks (OSNs) such as Facebook, Twitter, Google+, Myspace, LinkedIn, etc. have become irreplaceable to form virtual communities. The data shared over an OSN can be very personal and demand immediacy from the provider. As a consequence, OSNs gather huge amount of data about their users, and are now the real custodians of people's identity. Some OSNs even allow third party's software to use this knowledge. Social data can also be used to infer people's interests in order to win their attention and deliver messages more effectively, such as advertisements. The OSN platform can also be used by governments and legal agencies to monitor the user's activities. This sharing of the users' data may happen without the explicit consent of the user.

Recent research efforts [25], [23], [27], [9], as well as commercial initiatives (say, Tent.io), have proposed to decentralize the social network so that users can maintain control over their data. Concurrently, the research community has developed new ideas for an Internet architecture that can better serve the needs of users and developers. In particular, Information Centric Networking (ICN) architectures focus on connecting users to data instead of setting up connections between machines.

It is our belief that ICN architectures are now mature enough to overcome the network issues that distributed social networking has faced so far. Therefore, we propose a solution that implements all the functionality common to social networks while leaving the users in full control of their data. Furthermore, our solution is compatible with the value add of current OSN providers. In this paper we present the WARP framework, that is: an architecture that allows users to share data, potentially over the infrastructure of an OSN provider, but which prevents the OSN provider to access to the user's data without the explicit consent of the user.

Consider Facebook: right now, the "Facebook application" is a software that runs on their servers in their data center, and the end-user chooses how to interface to it, either via a web-browser or a mobile application. In other words, data is generated at the source by the user, and then moved to Facebook. Then Facebook manages and distributes the data. We want to reverse this scheme in an ICN fashion by having a Facebook application running on the source of the data, e.g. the user cellphone. The application must have the duty of collecting the data from the users, whatever this data is, and pass it to the WARP framework which then takes care of encrypting and distributing the data to the network as an authenticated source.

Note that Facebook can still distribute the data, for instance to improve the performance or offering the distribution service in exchange of showing their advertisement. WARP only guarantees to the user that his/her data will not be accessible unless he/she decides to grant to Facebook the right to access it. Our ultimate goal is to show that, with minor changes, ICN does this by design, and that additionally, it is possible to use WARP to build a shared social infrastructure as an overlay of the current Internet that is able to serve both the interests of OSNs and the rights of the users.

The rest of the paper is organized as follows, in the next section we review the related work from literature then we describe our requirements in Section III, WARP's architecture in Section IV, and the protocol details in Section V. We evaluate the framework in Section VI before concluding the work in Section VII.

II. Related Work

Information Centric Networking [1][7][26] describes network architectures that use data retrieval primitives, i.e. *put/get*, in place of the primitives for machine-to-machine message delivery, i.e. *send/listen*. The final goal is to decouple applications from topology and fetch data from anywhere in the network, including transparent caches. ICNs usually rely on two types of packets, data and requests. Data packets simply contain the content object. The content is uniquely named, signed and, since there is no access control on the network caches, encrypted for security reasons. Requests, intuitively, are the messages used to fetch data. By design, ICN architectures secure the data instead of the communication channel. However, since the producer has no control over the replicas of its data, their application to social networking is very limited. A trivial example is the following: let us imagine that a content, say A, is initially shared by Alice with all her friends. After a few of her friends have downloaded A, A is replicated on several distributors. If now Alice wants to withdraw the permission of reading A from Bob, she must first ensure that all the copies of A stored in caches are voided and replaced with the new version of A. This is required to avoid that Bob fetches the first version of A that he is able to decrypt. The task is made difficult for Alice since the data could have been replicated several times on caches that are unknown to her.

Privacy Preserving Social Content Distribution PrPl [25] and Musubi [9], have deeply inspired this work since, to the best of our knowledge, these were the first research projects that aimed at decentralizing modern social networks. Despite sharing the same goal, though, their nature is very different from WARP's. In fact, they focus on the application support, and specifically on distributed search indexing. WARP proposes a new network infrastructure for distributing data in an efficient and secure manner. For this reason WARP has to be considered complementary to these projects and integrating all solutions together is the final goal. Cachet [20] addresses a similar problem as WARP. It implements distributed social feeds, policy based encryption and even offline persistency (which WARP does not entirely support at the moment). However, Cachet has three substantial limitations: (1) by admission of its authors, the computational overhead needed to secure access and updates on the DHT is not sustainable; (2) it relies on proxies to re-encode the content and to revoke keys; and (3) it is limited to the news feed type of content whereas WARP allows to develop any kind of software and to enhance it with social functionality. For the sake of precision, Cachet boosts its performance by implementing a gossip-based social caching, although it cannot outperform WARP's structured cache search algorithm. Other relevant projects include Tribler [23], Diaspora [5], PeerSon [6], Safebook [8], LotusNet [2] and SCOPE [18]. These projects implements subsets of Cachet's functionality and for this reason will not be discussed further in this paper. The reader is referred to the original papers for the details.

III. Architectural Requirements

A. High level view of WARP

We propose an open infrastructure to decentralize the distribution of social data whilst enforcing the users' privacy. WARP allows to disseminate social data in an ICN fashion, letting the user decide the right tradeoff between performance, security and control over his own data. The protocol is based on a hierarchical name resolver to facilitate routing without lacking information about users' activities.

Data security is enforced using a combination of symmetric cryptography and Attribute Based Encryption (ABE) as in [15], [14] but with the difference that WARP does not depend on proxies for content updates.

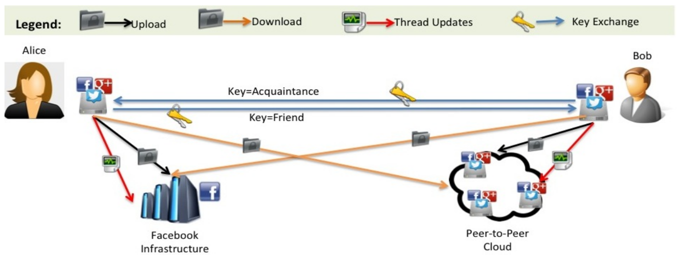
Content updates are taken care of by implementing a two-way agreement between the producer of the original content and the producer of the update. Namely, assuming the case of a comment on a Facebook wall post, the first user (who owns the post) will agree to link the content update to his Facebook wall as long as the second user gives a signed certificate of what its comment contains and what the name of the data is. By creating this link, the owner of the wall prevents two problems, first: it links only comments that it consider appropriate and reputable; second, it certifies to his audience that they have not been compromised afterwards; at the same time the content update is bound with its actual producer, who is also responsible for its distribution.

Instead of implementing a key-revocation protocol, whenever security policies change, the cached objects are deleted, or substituted with a newly encrypted version. This is made possible by using control channels, named *thread updates* that notify content distributors of modifications to their content.

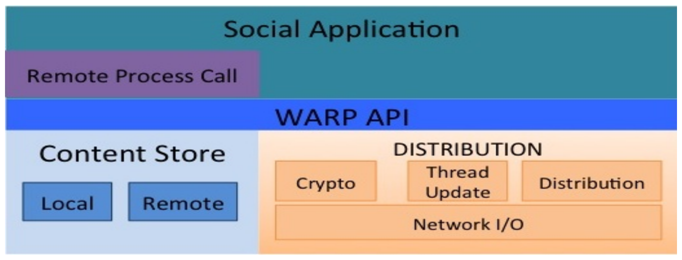
B. A Motivating Example

With respect to Fig. 1a, let us consider the case of two users, Alice and Bob, who want to use a Facebook-like application (FL) built for the WARP architecture.

1) Both the users must have control of a *butler* that will be the guardian and the host of their data. See Section IV-B.
2) Independently of the social software Bob and Alice are going to use, they are already able to establish a social relationship as a mere exchange of keys between their butlers. Friendships are defined by assigning a user to one or more categories - e.g. ”friend”, ”colleague” or ”family”. Let us assume that Alice categorizes Bob as friend whereas Bob categorizes Alice as ”colleague”.
3) Alice and Bob install on their butler the FL software which has the following duties: offer a user interface, compile the data in the application format, interface with the butler's database to store the data, choose what groups can access to the content created.
4) Alice has no privacy issues; however she does not want to use her own bandwidth to distribute her FL data. Therefore she chooses to distribute everything named under *Alice/facebook/* using the FL infrastructure.
5) Bob is not concerned about bandwidth consumption and does not want to pay a cloud service to distribute his data. Instead he organizes a P2P network with his closest friends that become distributors of his content. In exchange he does the same for these friends.



(a)



(b)

Fig. 1: (1a) An example of decentralized OSN where Alice chooses to distribute her facebook content using the Facebook infrastructure while Bob chooses a P2P solution using his friends' butlers as distributors. (1b) The internals of a butler. Social applications interact locally or remotely via RPC.

6) Alice shares her newsfeed with all her friends and so does Bob. In this case, Bob will be able to follow Alice's feed whereas Alice cannot do the same with Bob's. Moreover Alice will not even be able to see if Bob is writing on his wall because she will not be able to identify what the data objects are.
7) When Bob wants to see Alice's stream, the FL application takes care of identifying all the chunks of the stream, download them, decrypt them and display them on the screen aggregated with all news of Bob's friends.
8) Whenever Alice wants to publish something that Bob should not see she can add to the feed a post with a content specific policy, that excludes Bob from the audience. For instance by replacing the group "friend" with the id of every single friend beside Bob, or creating a new group and distributing the new keys to its members. In this case Bob will be able to see that the network object (because he has access to the stream). However he will not be able to decrypt the post.
9) In order to change the access to a content that has already been published, the user has nothing else to do than simply change the policy on it. The butler will take care of voiding every cached copy. The new copy will be distributed by the butler upon request.

IV. Architecture

A. Data Naming

WARP organizes content in *network objects* that are uniquely named using the following structure:

```
x.y/<folder>/<sub-folder>/.../<appendix>
```

There must be at least two segments divided by a backslash. The first one is always in the form $x.y$ where x is the user name of the producer and y is the Identity Authority (IA) that certifies the identity of the user and his public keys. The remaining part of the name is organized as a file-system, *folders* and *sub-folders* that serve for organization and routing purposes. The last segment, the *appendix*, identifies the network object within its folder. Applications have the duty of obscuring content names, e.g. hashing them, in order to avoid leaking information, as for instance a picture named "Paul and Me in Hawaii" would do.

B. Butlers and Distributors

A WARP network is a two-tier overlay composed of ***butlers*** and ***distributors***. Every machine in the network must refer to an IA that guarantees for its identity, public keys and network address. Note that the IA does not work as a name resolver for that purpose WARP relies on the DNS service.

Every user in the social network must have a **butler** that serves four fundamental tasks: 1) establishing social links with the other butlers, 2) interfacing with the social applications; 3) storing the user's data; and 4) organizing data distribution. A user exists in the network only via his butler and, from the distribution point of view, all his content is originated there.

Users establish their social relationship directly from the butler interface. The OSN accounts will be dismissed, so that there will be only one global social graph. A relationship is established when the user categorizes the identity of another user. A sketch of a butler's internals is showed in Fig. 1b. At the top of the stack, there is the social application that interfaces with the butler locally or via Remote Process Call. The latter is a strong requirement of our design since content is often generated and consumed from mobile devices. WARP APIs only allow the social application to store their data in the local storage and communicate whether they should be made available to the network.

A butler is always the root source of the user's content. It can distribute the data as a stand-alone server or using *distributors* that behave similarly to ICN caches. A distributor can be: any trustworthy butler, if users want to share their data in a pure peer-to-peer fashion; a paid cloud service, if performance are a concern; or an ad-based service offered, for instance, by the company that developed the social application. WARP distributors differ from ICN caches because they must monitor the updates of the content they store via control channels, named *thread update*, that deliver information about the cached content. Thread updates deliver two types of information: (1) notify security updates, e.g. a newly encrypted version of the content, and (2) notify the creation of new related content in order to allow prefetching.

C. Cryptography and Credentials

Content is authenticated with signatures generated using any public-key scheme, such as DSA or ECDSA [16], that guarantees confidentiality (Identity Based Encryption (IBE) [19], for instance, does not). At the current state of the art we rely on the IA to authenticate the public keys. However we conjecture that, given the nature of WARP, PGP [11] and a web-of-trust [24] can suffice for the task. In any case we consider the problem of key authentication beyond the scope of this paper.

Privacy is enforced using a combination of symmetric and attribute based encryption (ABE) [12]. ABE is an asymmetric scheme adopted for broadcast encryption [10]. It gives fine control over the audience of a private message. The cyphertext is encrypted with a *policy* expressed as a boolean expression. Variables, named *attributes*, are connected by the operators $AND(\wedge)$, $OR(\vee)$ and *k-of-n*. Every user possesses a private-key that is associated with some of such attributes and is able to decrypt the cyphertext if these satisfy the encrypting policy. Performance-wise, ABE, as most of asymmetric encryption schemes, is sensibly slower than symmetric cryptography thus WARP secures the data using a secret key (sk) that is encrypted with ABE and distributed together with the cyphertext.

1) Key-distribution: Butlers exchange keys on an as-needed basis, only when some content has to be decrypted and not when the users define their relationship. The reason for doing so is that, as in [21], keys have an expiration date, in the order of a week or a month. If two users do not interact often it is pointless to exchange keys. For the same reason, users should maintain a pool of recent keys in the eventuality that they have to decrypt a content that was encrypted with one of the previous keys.

The butler automatically assigns a random *alias* to every known user so that content can be encrypted for a single user only.

2) Key-Revocation: Key revocation at the scale of a social network is a critical matter due to the fact that users have often hundreds to a thousand friends [4]. Without loss of generality, let us consider the case of a content c encrypted using a single-attribute policy $p = friend$. In order to oust a user from the audience its producer must: (1) create a new version of the attribute, say $friend'$, (2) for each key containing the attribute $friend$ distribute a new one containing the new attribute $friend'$, (3) the butler re-encrypts c with policy $p' = friend'$. Similarly to [17], we work around this problem by using aliases of users' identities.

The butler assigns an extra attribute, named ***bucket***, to each known user. A bucket is an integer between 0 and K where K is a constant parameter. When a user, say u_i, is revoked of the attribute, say a, the new policy will be such that the presence of a will be conditioned to the presence of every bucket beside the one assigned to u_i or the individual identity of every other user that shares his bucket with u_i. In practice, let us consider a content C encrypted using a policy $P = att_1$, and four users u_1, u_2, u_3, u_4 distributed into two buckets, respectively $B_1 = \{u_1, u_2\}$ and $B_2 = \{u_3, u_4\}$. In order to prevent the user u_4 from decrypting C this will be re-encrypted using a new policy $P' = (att_1\, AND\, B_1)\, OR\, u_3$. This approach prevents from the redistribution of new keys to all the users whose keys contains att_1 and prevents the number of attributes from growing linearly with the number of users as long as the parameter K is chosen adequately. Moreover, since keys are redistributed periodically, the impaired policy will be enforced only until the new attributes will be generated.

D. Content

Data in the network is organized as a collection of *network objects*, represented in Fig. 2a. An object can contain data, reference another object or both. Each object virtually belongs to a doubly-linked list thus it maintains a pointer to its successor and predecessor. Note that, while connecting objects in a list is convenient for publishing unbounded streams of data, such as the Facebook Timeline, it is not efficient for random access or to download multiple objects in parallel. One way of solving the problem is to make object names enumerable, for example, given a folder `/x.y/photos/AFAFA/)`, a seed s and a hashing function h the object could be named `/x.y/photos/AFAFA/h(seed,1))`, `/x.y/photos/AFAFA/h(seed,2))` and so forth. This naming convention is more efficient but has the disadvantage that if one wants to ban a member of the audience, making it impossible for him to trace new content, all the names would have to be changed. At this stage of research we want to keep our design as flexible as possible and for this reason we leave to the application developer to choose an appropriate naming convention and wether to facilitate sequential or random access. Public fields contain general information about the object, they are: **Content Name**,**Version** which specifies what decryption key to use, and **Signature** that authenticates the content. Followers fields are accessible only to the audience of the content, i.e. people who are granted a valid key, they are: **Application** that specifies what application created the content, **Secret Key** the key used to encrypt the data, and **Reference** that is used to link another object in the network. **Next** and **Previous** are, respectively, a pointer to the successor and the predecessor of the object in its stream. Ultimately, the field **Thread Update Pointer** is an identifier, readable only by distributors, that specifies at what moment the object was published, distributors use it to monitor future changes to the object or its folder.

V. Protocol

A. Protocol Messages

As discussed in Section II, ICN protocols generally use two types of messages, requests and data. WARP introduces two additional types of messages: ***RESOLVE*** and ***NOTIFY***.

A **RESOLVE** message is used to obtain a list of distributors for a given folder. This type of query is answered authoritatively only by the producer of the content or a distributor for a parent folder, which behave similarly to a BitTorrent tracker [22]. In other words, considering the name `x.y/folder1/folder2/name1` a list of its distributors can be obtained by its butler `x.y` or any distributor for

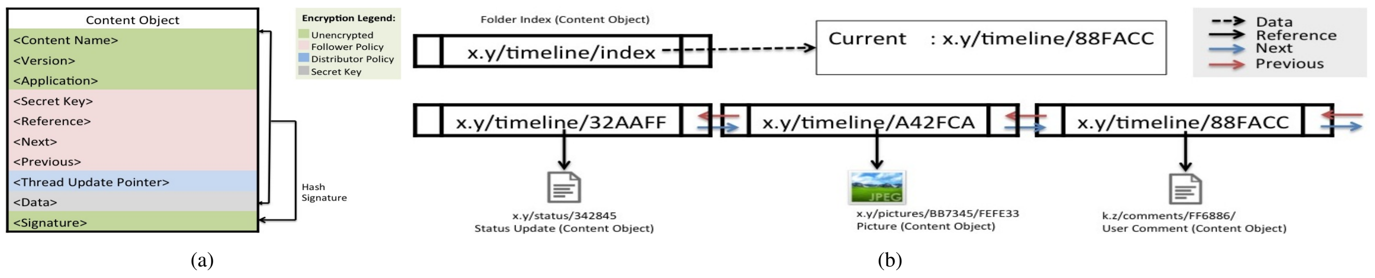


Fig. 2: (2a) A network object (2b) A possible organization of network objects to recreate a Facebook-like timeline. The index of the folder `x.y/timeline` keeps track of the latest segment of the news feed which is organized as a WARP feed. Each entry of the timeline reference an external object, from left to right: a status update from the user, a picture from another application, and the comment of another user. The social application "timeline" will take care of aggregating the content and show it to the user.

`x.y/folder1/` or `x.y/folder1/folder2`. Machine addresses are resolved using the normal DNS infrastructure since domain names can be certified by the IA at the same way of identities.

NOTIFY messages are one of the core ideas of WARP. They are sent by a butler to another to inform that a content has been created. They are fundamental to implement common functionality - such as comments, re-post, likes and re-tweets - since in WARP a network object cannot be modified by anybody that is not its producer. By design, content aggregation is operated at the application level rather than at the storage level. Namely, assuming the case of Alice adding a post on her Facebook wall and Bob commenting on it, in WARP there would be three distinct objects. Alice's post, Bob's comment and Alice's link to Bob's comment. A notify message contains a content-name, a signed checksum of the related data and an application identifier that says which application has generated the content.

B. Thread Updates

Thread Updates (TUs) are simple publisher/subscriber channels between a user butler and its distributors. Every TU binds to a folder and delivers messages containing the changes made to that folder and all its sub-folders. Changes made to the butler storage are univocally named using always increasing $identifiers$ so that the evolution of the content can be placed on a timeline. Distributors declare from what point in time the want to receive the messages using the TU pointers in their content. A distributor chooses which channel to subscribe based on the content that it has cached, e.g. if it has stored two objects, say `x.y/folder1/object1` and `x.y/folder2/object2`, it can choose whether to follow their directory exclusively or the whole tree `x.y/`, in any case changes are never forwarded twice since they are univocally identifiable.

C. Application

Social Applications create content and store it into the butler via common CRUD[1] functionalities. Additionally, they subscribe to be notified whenever a NOTIFY message arrives to the butler. Due to space constraints, we cannot give an extensive discussion on WARP's APIs although in Fig 2b we outline the content design of Facebook-like timeline. The figure shows two entities, and index file that keeps track of the latest post, and a stream of objects that refers to external objects that populates the timeline. Note that everything, beside the index, is being anonymized.

[1]common acronym for Create, Read, Update and Delete

VI. Evaluation

1) Key-Revocation: As opposed to what was done by [20], [15], [14], [13] for social networking and by [28] for cloud storage, we decided not to use proxies for the key revocation process. While we do not consider the trustworthiness of proxy a real issue, since all the centralized part of such systems are generally assumed reputable, our concern are centered around scalability. Proxies are intermediaries that take part in every decrypt operation. This would imply that every data transfer in the network would at one point end in a proxy transaction, and this would include very simple operations such as, for instance, likes on Facebook. Moreover, some proxies need part of the cyphertext in order to complete the transaction which would cause an additional cost in terms of redundant network traffic that we are trying to avoid. In WARP key revocation maps into voiding cached content and re-encoding the content upon request. It is important to remember that interest for social content has generally a very short life-span. Therefore, as long as access rules are properly set when the content is generated, re-encoding might not even be used without the need to pay the cost of a proxy transaction every time the content is accessed.

Another observation that can be made about WARP's revocation procedure is whether the dimension of the encoding policy grows linearly with the number of attributes, as discussed in section IV-C2. Attributes are expanded only and if the content has to be re-encoded during its *popular* period. Since keys are periodically changed, attributes can be shrunk again at the first key renewal. Furthermore we load-balance the problem by uniformly sub-dividing the users into buckets. As long as the number of buckets is properly chosen [3], the chances of using high-number of attributes are low. As a ground rule we do not consider policy changes to be frequent enough to justify the network and computational overhead of using proxies.

2) Thread-Updates: Every cache should periodically ping a control channel to verify that the content of its cache is still valid. There might be a concern regarding the overhead induced by the TUs. To answer this question, the reader must consider that thread updates can be cached and redistributed as well. This implies that on the butler side the cost of delivering such information can be very low if the distribution chain is properly set. Moreover, on the distributor side, the cost is still fairly low considered that policy are not updated frequently and that request for updates can be sent with a granularity of choice, not to mention that they have the bandwidth occupation of only a few packets. For instance, a distributor can commit to enforce new policies only once every 30 minutes.

3) Distribution: Data distribution in the WARP network is intentionally kept as flexible as possible. The solutions mentioned earlier - i.e. centralized, peer-to-peer and multi-tier - are not offered by any previous work. Cachet has the advantage of offering offline data distribution which WARP does not offer because the butler has to take action after receiving a NOTIFY message, although we consider this a minor problem considered that the whole point of WARP is to prevent personal data to be replicated without control. Beside this aspect though, performance-wise there is not a real comparison between the two protocols. Being DHT-based, Cachet requires $O(logN)$ messages to locate the content, where N is the number of machines in the network that can be in the order of tens of millions and more. WARP on the other hand can resolve the list of distributors in constant time with no regard to the network dimension. Their use of social caching, while effective, relies on the number of connections opened and, as shown in their result, is effective only as long as requests are sent to about 40% of the contacts. WARP structured approach never sends a request to more than one distributor.

VII. Conclusion and Future Work

In this paper we presented WARP: a scalable ICN architecture that supports social networking with no limitations on both the user and the vendor side. The main contribution of WARP consists in offering a unique solution to the most recent privacy violations and to the current limitations of Information Centric Networking. Our priority is to explore the scientific aspects of ICN and social network: we believe that there is a lot that can be done, especially for caching, leveraging on the information given by the social graph. Ultimately, while bulk data distribution will be most likely operated by third parties, we believe that real-time content such as status updates, tweets and messages can be distributed in a P2P fashion.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.

[2] L. M. Aiello and G. Ruffo. LotusNet: Tunable privacy for distributed online social network services. *Computer Communications*, Dec. 2010.

[3] F. Angius, C. Westphal, M. Gerla, J. Wei, and G. Pau. Prefix Hopping: Efficient Many-To-Many Communication Support in Information Centric Networks. *IEEE NOMEN Workshop - Infocom 2013*, 2013.

[4] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. *ACM SIGCOMM Computer Communication Review*, 39(4):135–146, 2009.

[5] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang. The growth of Diaspora - a decentralized online social network in the wild. In *IEEE INFOCOM Workshops*, pages 13–18, Mar. 2012.

[6] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. PeerSoN: P2P social networking: early experiences and insights. In *ACM EuroSys Workshop SNS'09*, pages 46–52, 2009.

[7] A. Chanda and C. Westphal. Contentflow: Mapping content to flows in software defined networks. In *Proc. of IEEE Globecom*, Dec. 2013.

[8] L. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 47(12):94–101, Dec. 2009.

[9] B. Dodson, I. Vo, T. J. Purtell, A. Cannon, and M. Lam. Musubi: disintermediated interactive social feeds for mobile devices. In *ACM World Wide Web conference WWW'12*, 2012.

[10] A. Fiat and M. Naor. Broadcast Encryption. *Lecture Notes in Computer Science*, 1994.

[11] S. Garfinkel. *PGP: pretty good privacy*. O'reilly, 1995.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. ACM CCS*, pages 89–98, 2006.

[13] S. Guha, K. Tang, and P. Francis. NOYB: privacy in online social networks. In *in ACM WOSN'08*, 2008.

[14] S. Jahid, P. Mittal, and N. Borisov. EASiER: Encryption-based access control in social networks with efficient revocation. In *ACM Symposium on Information, Computer and Communications Security*, pages 411–415, 2011.

[15] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia. DECENT: A Decentralized Architecture for Enforcing Privacy in Online Social Networks. In *ArXiv Tech. Rep. 1111.5377*, Dec. 2011.

[16] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.

[17] P. Junod and A. Karlov. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In *Proc. ACM DRM'10*, pages 13–24, 2010.

[18] M. Mani, A.-M. Nguyen, and N. Crespi. SCOPE: A prototype for spontaneous P2P social networking. In *IEEE PERCOM'10*, Mar. 2010.

[19] L. Martin. Identity-Based Encryption and Beyond. *IEEE Security & Privacy Magazine*, 6(5):62–64, Sept. 2008.

[20] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia. Cachet: a decentralized architecture for privacy preserving social networking with caching. In *Proc. ACM CoNext'12*, pages 337–348, 2012.

[21] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Attribute-based Systems. In *Proc. ACM CCS'06*, pages 99–112, 2006.

[22] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-Sharing System: Measurements and Analysis. In M. Castro and R. van Renesse, editors, *Peer-to-Peer Systems IV*, volume 3640 of *Lecture Notes in Computer Science*, chapter 19, pages 205–216. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2005.

[23] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. Van Steen, and H. J. Sips. TRIBLER: a social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 20(2):127–138, 2008.

[24] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *The Semantic Web-ISWC 2003*, pages 351–368. Springer, 2003.

[25] S.-W. Seong, J. Seo, M. Nasielski, D. Sengupta, S. Hangal, S. K. Teh, R. Chu, B. Dodson, and M. S. Lam. PrPl: a decentralized social networking infrastructure. In *ACM Workshop on Mobile Cloud Computing & Services*, 2010.

[26] K. Su and C. Westphal. On the benefit of information centric networks for traffic engineering. In *IEEE ICC Conference*, June 2014.

[27] B. D. Te-Yuan and H. M. Lam. The Junction Protocol for Ad Hoc Peer-to-Peer Mobile Applications. In *Tech. Rep. Stanford University*, 2011.

[28] Z. Xu and K. M. Martin. Dynamic User Revocation and Key Refreshing for Attribute-Based Encryption in Cloud Storage. In *IEEE TrustCom'12*, pages 844–849, 2012.