MoP-2-MoP – Mobile private microblogging

Marius Senftleben[1,2], Mihai Bucicoiu[1,2], Erik Tews[1], Frederik Armknecht[3], Stefan Katzenbeisser[1,2], and Ahmad-Reza Sadeghi[1,2]

[1] CASED/Technische Universität Darmstadt, Germany
[2] Intel ICRI-SC at TU Darmstadt, Germany
[3] Universität Mannheim, Germany

Abstract. Microblogging services have become popular, especially since smartphones made them easily accessible for common users. However, current services like Twitter rely on a centralized infrastructure, which has serious drawbacks from privacy and reliability perspectives. In this paper, we present a decentralized privacy-preserving microblogging infrastructure based on a distributed peer-to-peer network of mobile users. It is resistant to censorship and provides high availability. Our solution allows secure distribution of encrypted messages over local radio links to physically close peers. When redistributing messages, each peer re-randomizes encryptions to achieve unlinkability. Moreover, we show the feasibility of our solution using different synchronization strategies.

Keywords: Microblogging, privacy, anonymity, censorship-resistance, mobility, peer-to-peer, delay-tolerant networking

1 Introduction

Exchanging small text messages in a publish-subscribe manner from one publisher to many subscribers — also known as microblogging — has become a popular form of Online Social Networking (OSN) activity. Microblogging services allow users to send out clear, succinct and informative messages. The communication is in plaintext, and all widely adopted services (such as Twitter) follow the client-server model. Unfortunately, these design decisions imply numerous privacy and security problems, particularly in oppressive political environments.

Current microblogging services are prone to censorship. Due to the centralized nature of the services and the messages in plain text, acts of censorship can easily be performed either by the service providers themselves or external parties. Furthermore, once a central server becomes unavailable, e.g., due to regional Internet shut-downs, messages can no longer be sent or received, which again provides censorship potential. Moreover, all user interactions are known to the provider, among them all messages sent, all existing subscriptions, the entire query-patterns of users, etc. Complete data retention facilitates traffic analysis as well as data mining on the unencrypted messages.

These problems create a demand for privacy friendly microblogging services. In this paper, we propose a private mobile microblogging architecture that respects the users' privacy and is resilient to censorship. We rely on the fact

that smartphones are becoming ubiquitous communication devices, which are equipped with local communication facilities (such as NFC links and ad-hoc Wifi networks), while having Internet connectivity. Instead of relying on a centralized infrastructure to exchange messages, our solution is based on a peer-to-peer architecture, involving mobile peers who exchange messages with each other using local radio links and a best-effort message synchronization strategy.

Our architecture allows peers to microblog short messages privately using their smartphones while on-the-go. Messages are transmitted to a group in encrypted form, so that only peers who are authorized group members can access them; messages that cannot be read by a peer will nevertheless be forwarded to guarantee message spread. Multiple replications of messages stored at peers due to the decentralized message dissemination over point-to-point links increase censorship-resistence. The use of re-randomizable encryption provides message unlinkability as well as sender anonymity, because messages get re-randomized each time before being forwarded.

In the present paper we first introduce our new microblogging architecture, which uses universal re-encryption to facilitate the unlinkability of exchanged messages. Subsequently, we argue through simulations that microblog messages are sufficiently spread within the network of peers and that our architecture achieves the desired security and privacy goals.

The rest of this paper is structured as follows: In Section 2 we describe the proposed architecture with its functionality, state its privacy and security goals and outline the adversary's capabilities. In Section 3 we discuss our simulation results. Section 4 elaborates on the fulfillment of privacy, anonymity and censorship-resistance. Section 5 deals with related work and Section 6 concludes.

2 Mobile private microblogging

Our proposed microblogging solution *Mobile Peer to Mobile Peer (MoP-2-MoP)* builds upon mobile peers that interact with their smartphones using point-to-point communication links once they are physically close (technically they form an unstructured peer-to-peer overlay network). We first give an overview of the architecture, then state the privacy goals along with the adversary model, followed by descriptions of the functional components.

2.1 Infrastructure overview

In our scenario, we consider mobile peers that form a dynamically changing peer-to-peer network, where the movement patterns correspond to the natural movements of the smartphone owners. All peers maintain a local buffer of encrypted messages. Whenever two peers are physically close to each other they exchange messages based on a fixed strategy (described in Section 2.6). By this local message exchange, the system aims at propagating new messages through the entire network. We refer to this process as peer synchronization. In case network segmentation occurs, peers can — as a backup solution — also download

messages from servers using a wide-area communication network (called server synchronization); note that there can be multiple servers, as they only serve as additional channel to transfer messages.

In order to guarantee confidentiality and unlinkability of messages, they are encrypted using a variant of the ElGamal encryption scheme that offers the possibility of re-randomizing ciphertexts without knowledge of the public key [9]. The sender of a message can designate the message to a certain group by encrypting it with an appropriate group key; the exchange of group keys is based on social trust and outlined in Section 2.5. Whenever a peer receives messages, it checks whether he can decrypt them using any available group keys. By default messages get re-randomized before being sent to other peers during peer synchronizations.

Figure 1 shows a schematic overview of the architecture. Peers are depicted

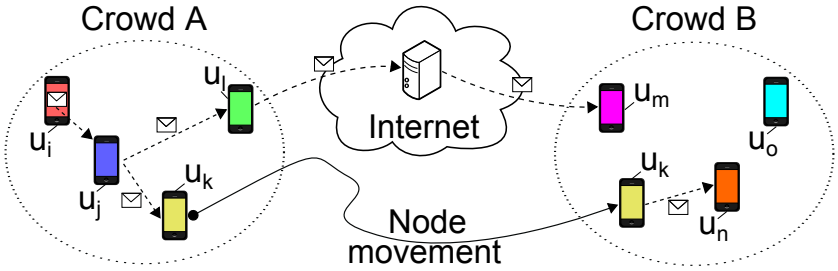

Fig. 1. Schematic overview of the MoP-2-MoP architecture.

by a smartphone-like shape, where u_{index} represent different peers. The dotted ellipses named Crowd A and Crowd B represent clusters of peers which are physically close to each other so that peer message synchronization is possible. Consider the following example: User u_i acts as originator of a message; as soon as u_i and u_j are close, they initiate a synchronization of their message buffers; this results in the new messages being spread. This way, the message finally reaches all other physically close peers in Crowd A (i.e., u_l and u_k) through peer synchronizations, depicted as dashed lines. The original encrypted message gets re-randomized at each hop so that a global adversary cannot link exchanged messages. A physically distant Crowd B can get the messages via two different mechanisms. Firstly, peer u_k can physically move close to a peer in Crowd B and initiate a peer synchronization there. Secondly, one member of Crowd A (u_l) can upload his local messages to a server, which offers the possibility of a server synchronization with any member u_m of Crowd B. In summary, we assume peer-based intra-crowd message dispersal, where encrypted messages are spread in an opportunistic manner using peer synchronization, supplemented by optional inter-crowd dissemination using server synchronization.

2.2 Privacy goals and adversary model

This section details the goals of the architecture and states the adversary model assumed. The *privacy and security goals* of the microblogging architecture are:

Anonymity. Sender anonymity is required. An attacker should have no information on the originator of a message. Unlinkability of encrypted messages and opportunistic synchronization achieve this goal.

Privacy. Group memberships of a peer and all messages should be kept confidential. This amounts to some form of receiver anonymity.

Censorship-resistance. No central entity should have the power to censor messages based on their content, and the message propagation should not be fully subverted by technical means.

In a *basic adversary model* we distinguish between wide-area network (WAN) communication for the server synchronizations, and local point-to-point communication deployed in the peer-to-peer (P2P) network. We assume that the adversary is *not* able to break cryptographic primitives. The considered capabilities in our adversary model are:

- All WANs are under full *passive* control of the respective operators. The adversary can monitor and log all such communication channels used.
- Shut-downs of WAN infrastructures ("kill-switches") occur.
- Limited local jamming, monitoring or logging of P2P links is possible.
- P2P peers are not compromised.

In an *extended adversary model* we assume the existence of a limited number of malicious peers that control their own communication link and have access to all local keys. For example, such devices could be compromised by malware.

2.3 Universal re-encryptable ElGamal

Peers who are not members of a group should not be able to decrypt messages sent in that group; furthermore, a peer should not be able to learn anything from messages he cannot decrypt, in particular the used public key. Nevertheless, re-randomizations of all encrypted messages are needed to achieve unlinkability. In order to achieve all these requirements, we use a variant of the ElGamal encryption scheme introduced in [9], which is summarized below.

In a cyclic, multiplicative group of prime order p with neutral element 1, the ciphertext of a message m encrypted under an ElGamal public key $pk = (g, h)$ is composed of two parts: (i) an ordinary ElGamal encryption of m and (ii) a random encryption of the neutral element 1. More precisely, two integers $r_1, r_2 \in \mathbb{Z}_p$ are chosen uniformly at random and the ciphertext is computed as $c = (c_1, c_2, c_3, c_4) := (g^{r_1}, h^{r_1} \cdot m, g^{r_2}, h^{r_2})$. The first component (c_1, c_2) is a textbook ElGamal encryption of m, while (c_3, c_4) is a random encryption of the neutral element 1. To compute a re-randomized ciphertext c' one chooses two new random integers $t_1, t_2 \in \mathbb{Z}_p$ and computes $c' = (c'_1, c'_2, c'_3, c'_4) := (c_1 \cdot c_3^{t_1}, c_2 \cdot c_4^{t_1}, c_3^{t_2}, c_4^{t_2})$. Due to the homomorphic properties of ElGamal the tuple (c'_1, c'_2) is again an ElGamal encryption of m and (c'_3, c'_4) is another encryption of 1 (both under the same pk). Thus, re-randomization is possible using the two ciphertext tuples, while no knowledge of pk is required. During decryption one can test whether the ciphertext c' was encrpyted under a certain key pk by decrypting the tuple (c'_3, c'_4) and checking if the decrypted value equals one.

2.4 Message format and storage

The message format follows the hash-and-encrypt concept to achieve integrity. A Message Authentication Code (HMAC) is used to achieve authenticity within the set of group members. A message m consists of a timestamp t concatenated with the message text msg and their HMAC, i.e., $m = t \,||\, msg \,||\, \text{HMAC}(t \,||\, msg)$.

Subsequently, the message is cut into blocks so that each one fits in the message space of the cipher introduced in Section 2.3; all blocks are then encrypted independently. (Note that due to the required properties of ElGamal, any kind of hybrid encryption is not possible here; due to the randomized chiphertext ECB mode is sufficient.)

Each peer maintains a message buffer filled with incoming messages. Upon receipt of any new message, the peer checks whether it belongs to a group the peer is a member of. This is done by brute-forcing: the peer tries to decrypt the message with all private group keys it possesses; if decryption works, the message belongs to a group the peer is a member of. This brute-force decryption step is necessary because we refrain from tagging messages with any sort of group identifier, which would open the possibility of message linking attacks.

2.5 Group and key management

We assume that the global system parameters of the encryption scheme are already present in the implementation of any client. Whenever a new message group is formed, the responsible peer creates an asymmetric ElGamal key pair, which identifies the new group. All members of a group are given both the public key and the secret key of the ElGamal key plus a group-specific secret required to compute the HMAC for their group messages.

We propose a key propagation mechanism that is based on social trust that uses existing real world trust relationships between people. Whenever two nodes are close to each other, one node can "introduce" the other one to a group by initiating the exchange of the group ElGamal key pair. This key can be sent from one device to another one using NFC transmission or an optical channel (such as a barcode that is scanned by the other device). Key revocation is done by forming new group keys and discarding the old ones. Nevertheless, key management in our approach is treated as a replaceable black box; more advanced group key agreement schemes can be implemented in the future. For example, approaches such as LoKI [3] can be added to the infrastructure, where a key exchange app in the background automatizes the collection of shared secrets between mobile devices, and users can post-hoc establish group keys based on Online Social Networking friends whom they physically met before.

2.6 Synchronization of messages

Our architecture supports two synchronization methods, peer syncs and server syncs, both of which are described subsequently.

Peer synchronization (or peer sync) is the bidirectional exchange of multiple encrypted messages over a point-to-point communication link between two nodes. Nodes in our scenario do not manage any routing information for peer-sync operations, but use local communication only for peers within range. A send buffer is a subset of all messages of a node containing the messages to be transmitted upon the next peer sync event. We stress that this buffer does not only contain messages the peer can decrypt to ensure an appropriate spread of all messages. Four strategies to prioritize messages during peer syncs are proposed:

Best Effort A fraction p of the send buffer is allocated to messages of groups the sending peer is a member of; the fraction $1 - p$ of slots holds other messages. Both fractions are filled randomly.

Random The send buffer is filled uniformly at random.

Round Robin The send buffer is sequentially and block-wise filled with new messages each time.

Latest Only Only the latest messages received are put into the send buffer and sent at the next sync operation.

The used technology for the point-to-point communication allows peers to initiate a peer sync either manually, automatically, or semi-automatically. A manual peer sync would require peers to consciously connect their smartphones with other peers, e.g., by a short-ranged optical link, based on existing social trust relationships. In automatic mode the peer sync runs in the background on discovery mode and syncs whenever another peer is available and in reach. In semi-automatic mode a user maintains either a white-list or black-list of other peers. Users are thus able to synchronize messages more or less restrictively, based on how risk-averse they are.

Server synchronization (or server sync) is the option of downloading encrypted messages from servers to provide an alternative means of message transportation, since different crowds of nodes are likely to get separated if they are not geographically close or socially connected. During a server synchronization a node uploads its send buffer to the server and downloads new messages from it. Servers are only data sinks that can neither decrypt the messages, nor determine group memberships of messages.

3 Simulation

For an assessment of the message propagation in our architecture we implemented a discrete, event-based simulation *excluding* central servers. If servers were included, nodes would obtain messages whenever they server-sync. In our simulations peers only synchronize their messages amongst themselves using local point-to-point communication links established when close to each other. We briefly give an overview of the simulation and discuss its results.

3.1 Simulation overview

The nodes move on a plane according to generated mobility traces. Their message group assignments, message synchronizations and message creation events are

simulated based on empiric data. Results are presented for 300 nodes moving with $1.4m/s$ on an area of 0.16 km^2 over a duration of 7 days. The simulation runs over 2016 rounds, each representing a time frame of 5 minutes.

Network initialization. Node mobility traces are computed with the BonnMotion package [1], using its ManhattanGrid mobility pattern, which creates movements on a regular grid, resembling a street map.[4] A square sized $400m \times 400m$ is used to restrict the mobility. It is populated by 300 nodes, which results in $530m^2$ per node, a value in between the population density of a small ($253m^2$, e.g., Darmstadt) and a larger ($820m^2$, e.g., Berlin) city.

Group assignment. The group memberships of each node are drawn out of a discrete power-law distribution with exponent $\alpha = 2.276$ and $x_{min} = 2$. The same applies to the number of groups a node is a member of – values which have empirically been computed in [10]. They remain fixed once they got initialized.

Peer syncs and local storage. Based on the mobility patterns, the node connectivity is derived, i.e., we determine whether a given pair of nodes is eligible for a peer sync. We set the maximum distance over which a peer sync can take place to 25 meters (similar to the Bluetooth standard). We limit the number of peer syncs per node to 2 for each round in order not to exceed the Bluetooth transmission rate. When peer-syncing, nodes exchange a send buffer of 100 messages drawn according to a fixed synchronization strategy of Section 2.6. For the best effort strategy the probability p was varied and then fixed as $p = 0.4$ to obtain most expressive results. Each node's local storage saves up to 10,000 messages. The latest incoming messages from a peer sync shift out the oldest messages received by a node.

Message creation. For the actual simulation runs we sample the message creation events based on Twitter microblogging data [19]. Each group creates messages according to the Poisson-distribution with $\lambda_{Group} = 0.21 \cdot |Group|$, $|Group|$ being the number of peers in that group. The sampled values for each group are distributed uniformly at random across the group members.

3.2 Simulation results

Group message spread is our main metric, which we define for one group as $\sigma = \frac{msg_r}{msg_c \cdot |Group|}$, where msg_r is the number of group messages received across the group (including senders), and msg_c is the sum of all messages created in the group. (Thus, if all group members receive all messages, we have $msg_c \cdot |Group| = msg_r$ and $\sigma = 1$.) Figures 2 and 3 present the average group message spread for all messages created in a six hour window between rounds 72 and 144, monitored over 300 rounds, together with error bars for selected points in time.[5] We averaged over all groups and ran 10 independent simulations. Thereby, we distinguish large and small groups. The large group class contains two groups of 190 and 291 members, while the small group class contains the average over

[4] We used the package's mobility patterns RandomWaypoint and GaussianMarkov as well, but due to similar results we only show the results for ManhattanGrid here.

[5] In this period the network is in its operational window, having most buffers filled.

21 groups with 4–6 members. The figures also show different synchronization strategies.

Overall, both the large and the small groups have a peak message spread of close to 100%, although the propagation is significantly slower for the small groups across all but the Latest Only synchronization strategy.

The Best Effort strategy favors large groups, as expected. An increase of p to values higher than 0.4 yields extreme degradations in the performance of the smaller and smallest group, whereas the largest groups profit from a highly selfish selection of their messages. The Random and the Round Robin strategy are similar (both of them implement a form of uniform drawing). Interestingly, however, the Round Robin outperforms the Random strategy most of the time. The Latest Only strategy has similar propagation dynamics across the small and the large groups. In summary, the simulations show the feasibility of the

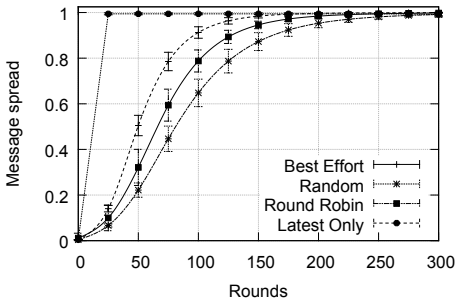

Fig. 2. Large group message spread.

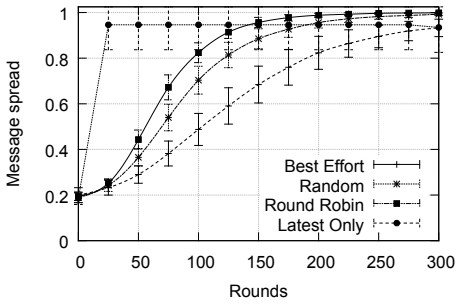

Fig. 3. Small group message spread.

MoP-2-MoP architecture. Different synchronization strategies impact the message spread across groups of different sizes. The Best Effort strategy favors large groups and disrupts the message propagation for small groups. The Latest Only strategy achieves fast propagation of the messages but might have a detrimental effect due to the last-in-first-out principle and fixed local storage size.

3.3 Computation complexity

A major concern is the computational complexity needed for the cryptographic operations. To test the ElGamal encryption's computation needs, we have developed an Android application that decrypts 100 messages, using spongycastle[6], a repackage for Android of the BouncyCastle Java crypto-library. (Note that only the second part of the ciphertext needs to be decrypted in order to determine if the message belongs to a specific group.) The power consumption of the app was measured using LittleEye[7]. During our test runs on a Samsung S3 smartphone, the decryption used around 1200 mW and up to 40% of the CPU. The decryption time was 1.55 seconds. Depending on the transfer technology used to establish pairing between peers, the total amount of energy drained will increase by 750 mW for Bluetooth and by 2500 mW for WiFi [13].

[6] `https://github.com/rtyley/spongycastle`
[7] `www.littleeye.co`

4 Privacy, anonymity and censorship-resistance

In this section we discuss the extent to which the stated privacy goals are met for our basic adversary model of Section 2.2. Furthermore, we comment on the extended model. We focus on local communication during peer synchronizations, since the anonymity of server synchronizations can be achieved through classical means such as the use of Tor.

Privacy. Privacy refers to the confidentiality of group messages and the group memberships of a node. Peer syncs guarantee these properties, since the exchanged messages look completely random and no group memberships can be derived from the encrypted messages themselves. Moreover, due to the unlinkability and re-randomization of the transmitted ciphertexts, messages can not be identified, linked or traced by the adversary. This achieves receiver anonymity.

Yet, local monitoring and communication logging could create a communication graph of who is performing syncs with whom. Community detection then reveals communication patterns by showing the connectedness of the nodes and the frequency of their peer syncs, thus possibly yielding a side-channel for group affiliations. We leave this for future research.

Sender anonymity. Peer synchronizations are beneficial for sender anonymity. The original sender as the creator of a new message achieves k-anonymity for a significant value of k only after a small amount of time. We analyzed the sender anonymity set sizes obtained in our simulation by retrospectively calculating the number of nodes from which a receiver could potentially have obtained a message, assuming a global passive adversary. Figure 4 depicts the development of the sender anonymity set size over time, averaged over our simulation runs. The anonymity set size develops according to logistic growth, asymptotically reaching 100%, that is the totality of all nodes in the network – and it reaches about 95% of the full crowd size after 9 rounds.

Censorship-resistance. The distributed and decentralized peer-to-peer architecture and redundant message stores are key to make the infrastructure censorship-resistant. Filtering on a semantic content level is not possible, because of the indistinguishability of all ciphertexts. A censor is thus required to block the communication of a node independently of the messages it sends. For

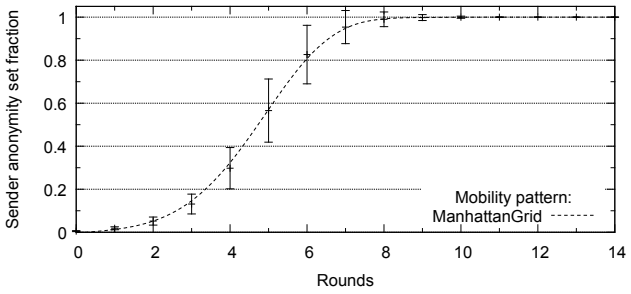

Fig. 4. Sender set fraction of total number of nodes over time.

example, the adversary can deploy local jammers to render peer synchronizations over local radio links in its sphere of influence impossible.

We conducted simulations that emulated local jamming. By p_{censor} we denote the fraction of the nodes disabled in each round by jamming. For $p_{censor} = 0.50$ the group message spread over time is shown in Figures 5 and 6, again averaged over 10 simulation runs. The results show that the architecture remains functional with half the nodes disabled per round, albeit with a slower message spread.

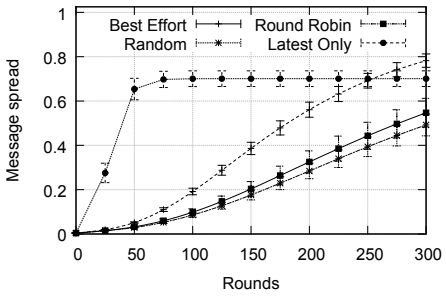

Fig. 5. Large group with $p_{\text{censor}} = 0.5$

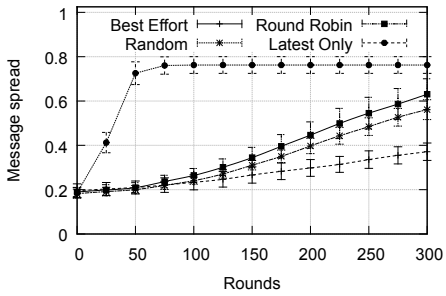

Fig. 6. Small group with $p_{\text{censor}} = 0.5$

Extended adversary model. An attacker able to compromise devices via malware (such as a trojan with keylogger and root-access) or via social engineering (e.g., a government bribing group members or introducing its own agents into a group) leaks the compromised nodes' group memberships and the content of messages encrypted under all compromised keys. Group infiltrations thus allow adversaries to read all the messages in the affected groups. These messages can subsequently be traced during peer syncs. Note that this also affects the k-anonymity of the uncompromised groups in case messages of affected and unaffected groups are simultaneously exchanged in a peer sync. Infiltrated groups need to be re-keyed; that is, a new group key needs to be set up and the old keys need to be discarded.

Finally, spam messages can have detrimental effects by flooding the send buffer of nodes by spam messages. A possible countermeasure for in-group spamming is the use of special spam filters (such as [11]) locally at each node; nodes would then not forward messages marked as spam. Junk ciphertexts injected into the network that cannot be decrypted under any group's key are a denial-of-service attack that ultimately affects all nodes and cannot be prevented easily. Devices performing DoS attacks might be blacklisted; the blacklists could be distributed as special messages in our system.

5 Related work

Approaches facilitating anonymity in Internet communication include Tor[8], using the concept of onion routing, and Crowds [14], which follows a peer-to-peer

[8] `https://www.torproject.org`

approach. In the latter, peers are used to create a cascade over which web transactions are routed in order to achieve sender anonymity; transport encryption is established between each pair of nodes. However, they both are developed for unicast fixed-line communication only. In a client-server setting, the Hummingbird server [5] provides a private microblogging service by obliviously matching messages to subscribers without learning about plaintext messages. In contrast, we aim at building a decentralized and peer-to-peer based microblogging infrastructure.

Solutions that address anonymity in mobile ad hoc networks (MANETs), where routing between mobile devices is done in a self-configured manner, include ALARM [6], a secure-link state based routing protocol which achieves anonymity and untraceability together with security properties by leveraging group signatures, and MASK [20], an anonymous on-demand routing protocol with unlocatability and untrackability. In the context of vehicular ad hoc networks (VANETs), where revocable anonymity is needed for liability issues, pseudonymity schemes are typically used to provide anonymity during normal operation [8].

In delay-tolerant networks (DTNs), which maintain no explicit routing information, human mobility and its characteristics (cf. [15]) are leveraged for message propagation. Su et al. [18] argue based on collected mobility data that effective routing decisions can be made by only knowing the pair-wise contacts that took place between nodes, irrespective of mobility models or location information. Chaintreau et al. [4] empirically show, that the distribution of the intercontact time between devices carried by humans can be approximated by a power law distribution for durations of up to a day. In Humanets [2], smartphone-to-smartphone communication is used to more efficiently propagate messages, while at the same time avoiding the use of mobile telephony networks. In our approach, we leverage these observations both in our microblogging architecture and in the simulation.

A number of solutions for content distribution in DTNs have been proposed (cf. [12], [17]). However, propositions focusing on anonymity in such scenarios are scarce: Fanti et al. [7] propose a mobile microblogging solution with trusted message propagation by the use of social-graphs and private-set intersection protocols, but do not focus on unlinkability. Rogers et al. [16] focus on secure communication over diverse networks achieving forward security, but do not specifically address anonymity. In contrast to these approaches, our architecture targets strong privacy and anonymity properties.

6 Conclusion

We presented a novel approach for mobile private microblogging, combining mobility, the peer-to-peer paradigm and local point-to-point communication links over a delay-tolerant opportunistic network. Our architecture achieves sender and receiver anonymity and is censorship-resistant. At the same time it ensures a sufficient message spread. Future work will address scalability and performance

issues, both in terms of networking load and on-device computation. Efficient re-keying of groups is an issue that will be addressed by incorporating broadcast encryption schemes. Furthermore, we will investigate means to enhance the robustness of the scheme against malicious nodes.

Acknowledgments. We thank the anonymous reviewers and our shepherd Urs Hengartner for their valuable comments. Mihai Bucicoiu was funded by the Romanian Ministry of Labour through grant POSDRU 76903.

References

1. Aschenbruck, N., Ernst, R., Gerhards-Padilla, E., Schwamborn, M.: Bonnmotion: a mobility scenario generation and analysis tool. In: ICST'10. pp. 51:1–51:10
2. Aviv, A.J., Sherr, M., Blaze, M., Smith, J.M.: Evading cellular data monitoring with human movement networks. In: HotSec'10. pp. 1–9
3. Baden, R.: LoKI: Location-based PKI for social networks. SIGCOMM Comput. Commun. Rev. 41(4), 394–395 (Aug 2011)
4. Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., Scott, J.: Impact of human mobility on opportunistic forwarding algorithms. IEEE Trans. Mob. Comput. 6(6), 606–620 (2007)
5. De Cristofaro, E., Soriente, C., Tsudik, G., Williams, A.: Hummingbird: Privacy at the time of twitter. In: S&P'12. pp. 285–299
6. El Defrawy, K., Tsudik, G.: Alarm: Anonymous location-aided routing in suspicious manets. IEEE Trans. Mobile Comput. 10(9), 1345–1358 (2011)
7. Fanti, G., Ben David, Y., Benthall, S., Brewer, E., Shenker, S.: Rangzen: Circumventing government-imposed communication blackouts. In: UCB/EECS-2013-128
8. Fonseca, E., Festag, A., Baldessari, R., Aguiar, R.L.: Support of anonymity in vanets - putting pseudonymity into practice. In: WCNC'07. pp. 3400–3405
9. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal re-encryption for mixnets. In: CT-RSA'04. pp. 163–178
10. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: WWW'10. pp. 591–600
11. McCord, M., Chuah, M.: Spam detection on twitter using traditional classifiers. In: ATC'11, pp. 175–186
12. McNamara, L., Mascolo, C., Capra, L.: Media sharing based on colocation prediction in urban transport. In: MobiCom'08. pp. 58–69
13. Perrucci, G.P., Fitzek, F.H.P., Widmer, J.: Survey on Energy Consumption Entities on the Smartphone Platform, pp. 1–6 (May 2011)
14. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1(1), 66–92 (Nov 1998)
15. Rhee, I., Shin, M., Lee, K., Hong, S., Chong, S.: Human mobility patterns and their impact on delay tolerant networks. In: HotNets'07
16. Rogers, M., Saitta, E.: Secure communication over diverse transports: [short paper]. In: WPES'12. pp. 75–80
17. Stanford MobiSocial project: `http://mobisocial.stanford.edu`
18. Su, J., Chin, A., Popivanova, A., Goel, A., de Lara, E.: User mobility for opportunistic ad-hoc networking. In: WMCSA'04. pp. 41–50
19. Weng, J., Lim, E.P., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential twitterers. In: WSDM'10. pp. 261–270
20. Zhang, Y., Liu, W., Lou, W., Fang, Y.: Mask: anonymous on-demand routing in mobile ad hoc networks. IEEE Trans. Wireless Commun. 5(9), 2376–2385 (2006)