ProofBook: An Online Social Network Based on Proof-of-Work and Friend-Propagation

Sebastian Biedermann[1], Nikolaos P. Karvelas[1], Stefan Katzenbeisser[1], Thorsten Strufe[2], and Andreas Peter[3]

[1] Security Engineering Group
Technische Universtität Darmstadt
{biedermann,karvelas,katzenbeisser}@seceng.informatik.tu-darmstadt.de
[2] P2P Networking Group
Technische Universtität Darmstadt
strufe@cs.tu-darmstadt.de
[3] Distributed and Embedded Security Group
University of Twente
a.peter@utwente.nl

Abstract. Online Social Networks (OSNs) enjoy high popularity, but their centralized architectures lead to intransparency and mistrust in the providers who can be the single point of failure. A solution is to adapt the OSN functionality to an underlying and fully distributed peer-to-peer (P2P) substrate. Several approaches in the field of OSNs based on P2P architectures have been proposed, but they share substantial P2P weaknesses and they suffer from low availability and privacy problems. In this work, we propose a distributed OSN which combines an underlying P2P architecture with friend-based data propagation and a Proof-of-Work (PoW) concept. ProofBook provides availability of user data, stability of the underlying network architecture and privacy improvements while it does not limit simple data sharing based on social relations.

Keywords: Online Social Network, Peer-to-Peer, Proof-of-Work.

1 Introduction

Popular Online Social Networks (OSNs) use a centralized design which sometimes leads to intransparent providers that can be the single point of failure. In order to solve this problem, current research aims at either content confidentiality through encryption of posts, or at privacy through the removal of centralized data storage control by introducing an underlying peer-to-peer (P2P) substrate. However, P2P architectures lead to new problems which can have a strong impact on design principles of these OSNs. Availability and freshness of the users' published content, is a very important property of any successful OSN. User data has to be stored on the devices of other users alternating between an online and offline status. In order to increase user data availability within the OSN, the user data has to be stored on a great number of other users, which finally results in the fact that each OSN participant has to provide a lot of storage. In general, OSNs which use other unknown P2P participants to store the users' sensitive data do not enjoy great popularity even if the stored data is encrypted. Furthermore, P2P substrates suffer from stability and security problems like Denial-of-Service (DoS) attacks (in which large amounts of requests are sent), Eclipse attacks [15], systematic content pollution (by introducing large amounts of fake data) or misuse for collusive piracy, many of which are exacerbated by the possibility to create a number of different zero-cost identities in the system (sybils).

In this paper, we assume an attacker model in which malicious users can perform DoS attacks, for example in the form of message flooding, and propose a fully distributed OSN architecture which nevertheless ensures privacy of each participant. We focus on the mitigation of these DoS attacks and propose a new OSN architecture, based on an underlying P2P substrate and an incentivised Proof-of-Work (PoW) concept which we call ProofBook. ProofBook has a decentralized architecture which nevertheless can ensure availability of published content, can increase anonymity and privacy of each user and can prevent manipulations and insider attacks of malicious participants as well as can mitigate DoS attacks. We can summarize the contributions of ProofBook as follows:

– *Availability:* User data is continuously available and as up-to-date as possible. This is realized with an update-on-request concept in which each user stores only the data of friends.
– *Stability:* Systematic content pollution, which is a major problem in P2P substrates, is mitigated in ProofBook by design, due to the implemented PoW concept which operates like a stamp used to pay for delivery of requests.
– *Privacy:* Friend relationships among ProofBook users are private and can neither be manipulated nor is any private user data revealed. This is enforced with the help of cryptographic techniques and hiding sources of delivered requests.

2 Related Work

2.1 Consolidations of P2P Substrates and OSNs

In general, data sharing performance in P2P networks can be increased based on social information about the participants. Chen Hua et al. [10] proposed "Maze", a hybrid P2P architecture which benefits from social information to help peers discover each other. Pouwelse et al. [13] proposed "Tribler" which is a consolidation of an underlying P2P architecture and social network data in order to increase usability. Graffi et al. [9] investigated security problems in P2P-based social networks. They proposed a P2P based social network with fine-grained user- and group-based access control to shared content.

In order to avoid the centralized architectures of OSNs, different OSN architectures based on an underlying P2P substrate have been proposed. The most popular architecture is Diaspora [5] which is an OSN based on a network of independent servers that are maintained by users who allow other users to store their data. Buchegger et al. [6] introduced PeerSoN, a distributed OSN based on a two-tiered P2P architecture which consists of peers communicating with each other and a separate look-up service. Cutillo et al. [7] proposed Safebook which exploits real-life trust relationships and maps these social links to a decentralized P2P network. Based on this, security mechanisms are implemented, while data integrity and availability are provided. In summary, these consolidations can achieve availability, but they often lack privacy.

2.2 Proof-of-Work Based Architectures

A lot of work has been done in the field of incentivised P2P substrates based on certain proofs for more accountability or e-cash ([3],[4]). Proof-of-Work (PoW) schemes are variants of cost-functions which are difficult to produce but trivial to verify. The degree of difficulty can vary depending on factors like the amount of participants. A PoW can be used to verify the existence of remote hardware resources that are controlled by a remote client. In particular, PoW approaches are used to combat spam mail, to mitigate DoS attacks or to control access to a shared resource [8]. Back [1] proposed "Hashcash" which is a PoW-based architecture that throttles systematic abuse of remote network resources. Bitcoin [12] is an electronic currency system based on a P2P substrate which prevents double spending of digital cash by adding a transaction with a PoW into a globally distributed chain of participants. The PoW is to find a hash for a given content which has a previously defined amount of initial zeros. This hash is calculated over data that includes the history of transactions. Since the currently most efficient way to find a valid hash are brute forcing techniques, the difficulty exponentially increases with the increasing amount of required initial zeros. Different hashes are created by changing an included nonce.

3 Overview of the Key Scheme

The architecture of ProofBook combines an underlying P2P substrate with a protocol that makes use of a PoW concept and incentivised cooperation to mitigate misuse. ProofBook provides standard OSN operations, integrates an incentive for participants without disrupting the utilization of the OSN and propagates new user data with the help of the user's friends.

3.1 User Registration and Friendships

ProofBook does not have a database which stores account information about users. Joining ProofBook and accordingly the underlying P2P substrate can be achieved by receiving the IP address of a participating peer from a secondary channel (for example via a web site). New friendships can be established by directly exchanging initial information. More precisely, each user U is associated with a public and private key pair (k_p, k_s) that will be used for signing certain information like U's data container. Additionally, U creates a symmetric "friend-key" k_f. The latter is distributed among U's group of friends and enables them to reply on requests which target U's data. Based on different "friend-keys", U can also maintain different group of friends (close friends, colleagues, etc). A friend relationship to U is established by retrieving U's public key k_p and subsequently U's data container which also includes the symmetric friend-key k_f. Obtaining U's public key enables a friend to identify oneself as member of U's group of friends. We do not treat the exchange of this data further, as this can be done with standard cryptographic techniques using a secondary channel.

3.2 User Data Propagation and Availability

The data of each ProofBook user U is saved in a container structure which can include up-to-date status information, personal information and pictures. U's data container is identified by U's ID and a signature on the content under U's secret key k_s to verify U as the owner. The data container's structure is illustrated in Figure 1. U's data container also includes the IDs of U's current friends. This way, a friend of U can contact other friends of U even if they have no established friendship themselves. A ProofBook container is separated into a redundant array of 8 data blocks (block-level striping with double distributed parity). This offers the opportunity to restore the whole data of U's container even if 2 sub-containers are not available while the storage efficiency is still 75%. Furthermore, each sub-container includes a timestamp (last-modified). The key scheme of ProofBook is based on a simple fact which can guarantee availability of U's data even if U is offline: *If U has retrieved and viewed the data container from an arbitrary friend once, U can locally save this data forever and this event cannot be undone.* Based on this, users directly save all data containers of their friends locally and ProofBook benefits from this approach, because users can not only request the friend's data from the target friend itself, but also from other friends of this friend who have a locally stored the data containers of their friends as well. In addition, there is no need to encrypt a container's data since it is only stored on users who are allowed to access this data anyway.

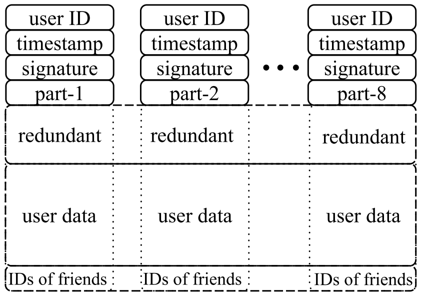

Fig. 1. A container which is published by U and can be requested by friends of U

ProofBook users can perform update requests when viewing the content of a friend's data container. If a newer data container of this friend is currently available, it can be requested and used to replace the locally saved one. The updated data container is again available for update-requests of other friends of this friend. The functionality of ProofBook is based on the simple update requests which are delivered by the underlying P2P substrate and which can be answered by friends of the target friend or the target friend himself. The content of a ProofBook update request can be seen in Figure 2.

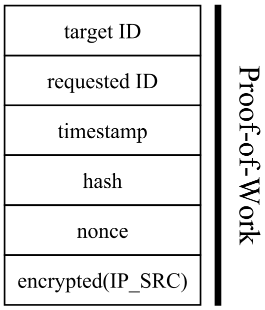

Fig. 2. A ProofBook update request

A ProofBook request includes a target user ID which can be the ID of the target friend or the ID of a friend of this friend, retrieved from the target friend's data container. Furthermore, there is the ID of the friend whose data container is requested, a timestamp and a hash (SHA256) which actually represents a PoW that is calculated over the content of the request and that needs to have a previously defined amount of initial zeros. A valid hash can be found by changing the nonce which is included in the content. Additionally, the request includes the encrypted source IP address of the requesting user which can only be decrypted by users who have the target friend's friend key k_f. The request is anonymous since there is no plain information about its source, only about the target. To mitigate DoS attacks, systematic content pollution and to limit users obtaining multiple identities in parallel, the requests are only forwarded by the underlying P2P substrate if the PoW, which was calculated by the source user, can be verified by the delivering peers. Otherwise, the requests are dropped. The request cannot be changed by peers without redoing the PoW and successful modifications are not possible since the friend key k_f is not available to arbitrary delivering peers. To ensure the freshness of the PoW, the peers use the timestamp and a synchronized clock (for example retrieved from a fixed clock server). New data of U is propagated stepwise among the group of U's friends based on these update requests.

Figure 3 shows the initial steps of a request sent by a user Alice to retrieve a new data container of her friend Bob. In an update procedure, Alice sends multiple requests targeting Bob as well as Bob's friends. If the PoW of the request

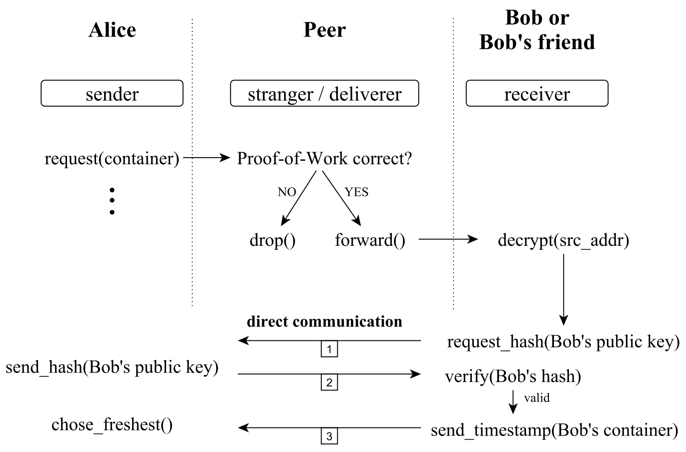

Fig. 3. Alice performs an update request which is forwarded to retrieve timestamps of Bob's currently available data containers

can be verified by the peers which are on the route, they forward the request to the target user. Receivers decrypt Alice's source IP address using Bob's friend key k_f and directly reply with a request for a hash over Bob's public key to verify that Alice is a friend of Bob. If Alice replies with a valid hash, the receiver sends the timestamp of Bob's stored data container. Alice chooses the most up-to-date data container and directly downloads all sub-containers from the chosen target. This target can be Bob himself (if Bob is currently online). Alice verifies the signature of each sub-container of Bob's container with the help of Bob's public key k_p. The reply procedure uses directly established connections and the P2P substrate is only used in the update request delivering procedure (one-way).

3.3 The ProofBook Payment Scheme

In Bitcoin [12], the amount of initial zeros of the required hashes which is used as degree of difficulty for the PoW is dynamically arranged. In ProofBook, there are only three different static levels $L_i, i \in \{1, 2, 3\}$ implemented. They enforce three degrees of difficulty on two different points.

First, since the update requests from a specific source are usually delivered on equal paths in the underlying P2P substrate within same periods of time, L_i depends on the amount of requests a peer has already delivered to the same target. A peer will only deliver a specific amount of requests to a target peer with the initial PoW difficulty L_1 within a fixed implemented time slot. If more requests have to be delivered within the same time slot, the difficulty to send a request to the same target becomes harder (L_2) and finally very hard (L_3). The increasing difficulty levels do not depend on the sources of the requests, rather on the target. Sending update requests to a specific target gets more expensive, the more requests are sent to the same target within the same time slot. This way, several attacks based on huge amount of requests can be mitigated.

Second, the last peer on the path to an arbitrary target obtains information about the target's hardware device, retrieved from the client once enrolled in the P2P substrate. If the target device is a desktop computer, the peer delivers every request, if it is a notebook only with L_2 or higher and if it is smartphone only with L_3.

This way, there is another load balance, which enforces more load on devices having better performance than others because they are cheaper to request.

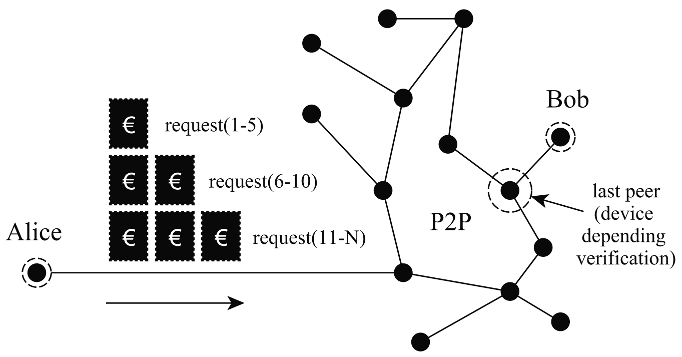

Fig. 4. Utilization of different example PoW levels L_i comparable to amounts of required stamps in the real world

Figure 4 illustrates the utilization of L_i at the two different points in Proof-Book. Like a stamp which is used to send a letter in the physical world, these values are used to pay different prices for the update requests to a specific target. The PoW acts as a buffer and reduces the amount of requests which can be delivered to the same target within a time slot depending on the amount of requests and the target's device.

It shall be noted that a botnet owner could compute the PoWs which he could use for continuous high amounts of requests to a single target in order to make it also costly for others to reach this user. However, the underlying P2P substrate is dynamic and uses different paths for the requests, depending on the entry point of the sender. That way, an attacker would require a high degree of knowledge about the current internal topology as well as many user IDs of users who are not necessarily connected with each other to mount such an attack successfully.

In order to implement an incentive for the replier of an update request, who finally sends the data container to the requesting user, we use a stepwise strategy in a commitment scheme. For each sub-container Bob sends to Alice, Bob receives a PoW-token from Alice (L_1) which he can use to decrease PoWs of own requests at a later point in time (Figure 5). If it so happens that Alice or Bob skips during the container transfer, Bob has at least retrieved some usable tokens and Alice can restore the data container even if up to two of the eight sub-containers are

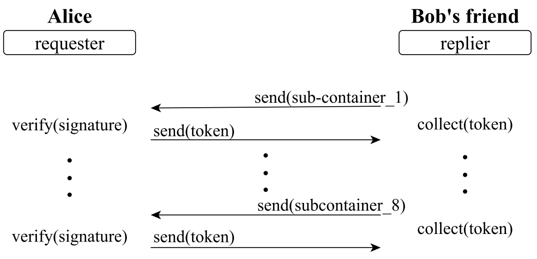

Fig. 5. For each verified received sub-container, Alice sends a PoW token to Bob

missing using the redundant structure of the containers. Bob can use the PoW-tokens in later requests to decrease his required PoW. This can be reached by extending an update request (Figure 6) with these tokens and calculating the PoW over the request and the extending tokens (max 8). The tokens can be used within a period of time in the future, defined by their timestamps. Furthermore, they can only be used in requests targeting the data container of the same user who Alice has requested and with whom Bob is a friend as well.

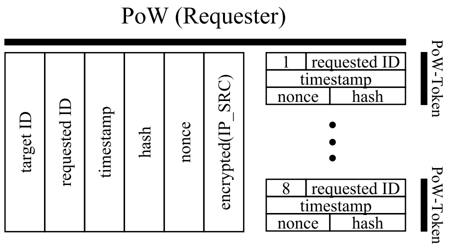

Fig. 6. An update request extended with previously earned PoW-tokens

Delivering peers only accept two different extended update requests, either with at least 6 or 8 additional PoW tokens targeting the same requested user ID. 6 tokens can decrease one level of PoW (e.g. from L_2 to L_1) and 8 tokens can decrease two levels (from L_3 to L_1). With this strategy, ProofBook actually gives an incentive for Bob to send that much sub-containers which Alice needs since Bob can perform these extended requests faster.

4 Evaluation

In the evaluation, we investigated two questions: First, how much time is required for an update request, since delays are caused by the PoW calculations and verifications. Second, in which periods of time data containers are propagated among a group of friends based on the update-on-request approach with the help of friends.

4.1 ProofBook Update Request Timings

First, we evaluated how much time the PoW computations require in our setup (Intel(R) Core(TM) i7 M620 with 2.67Ghz). We calculated PoWs for random update requests depending on different levels (amount of required initial zeros). Since the calculations become exponentially more difficult, the required time becomes quickly very large (Figure 7).

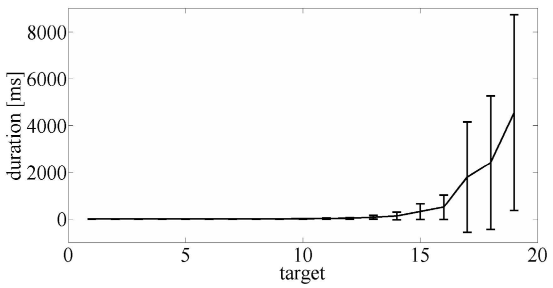

Fig. 7. Timings required to find a PoW with different targets (1024 test-runs)

A PoW with at target of 16 requires around 0.5 second on average having a standard deviation of 0.5 second. We implemented these timings as well as their deviations in a ProofBook simulation based on the discrete event-based P2P network simulation engine Oversim [2] which is based on Omnet++ [16]. For the test-runs, we used a Pastry [14] P2P substrate. The expected amount of forwarding peers in a Pastry overlay substrate is $O(\log N)$ where N is the number of participating peers. We investigated how much time a ProofBook update request requires including the PoW calculations and the "on-the-way" verifications.

Finally, we compared the results to other test-runs with normal request in the same Pastry P2P substrate. We used a P2P overlay with 1000 peers each of them running the ProofBook application on top of the Pastry P2P substrate and a ProofBook update request message size of 256 Bytes. A random ProofBook user is continuously chosen and sends an update request to another random ProofBook user. We have chosen two levels L_1 with a target of 17 and L_2 with a target of 19. Results of these test-runs can be seen in Table 1.

Table 1. Mean of delivered requests/s between random users (1000 participants)

PoW L_i (target)	L_0 (0)	L_1 (17)	L_2 (19)
Delivered requests/s	0.99301	0.35602	0.24580

The amount of involved peers lead to an average hop count of 2.51 ± 0.03 peers for a one-way delivered request. Table 1 shows the requests in comparison to requests without any PoWs in the same P2P Pastry substrate. With L_1, an update request can reach the target only every three seconds in average and with L_2 an update request can reach the target only every four seconds in average. The PoW scheme successfully limits the amount of delivered requests.

4.2 User Data Container Propagation

We executed other test-runs to investigate how fast a randomly chosen updated data container of an arbitrary user U propagates within U's group of friends. Since the network protocols are negligible in this case, we developed a ProofBook data container propagation simulation in Java. For a realistic scenario, we first needed to determine several values:

First, based on [11], we assume a power law probability distribution (Zipf's law) for the amount of friends each user has. We use a maximum of 500 and an α of 0.5 which leads to a numerical mean of 341 friends which is consistent with a recent Facebook study[1], where the median of the users' friends was found to be 342. Second, in reality, OSN users are continuously interested in up-to-date data of just a few of their "best" friends and in the data of most of their other friends only sporadically. We subdivide each user's friends into three equal subgroups. A random friend from the first subgroup (best-friends) is requested with a probability of $p = 0.5$, from the second subgroup with $p = 0.35$ and from the third subgroup with $p = 0.15$. Third, a user is not continuously online. Because of the increasing number of involved mobile online devices, we assumed a probability P_{on} of 33% for U to be online in any time slot of 6 minutes which leads to an average online time for a user of 7 hours a day.

In the test-runs, we monitored a randomly chosen user U_m who updated his data and we monitored all of U_m's friends. In a time slot (6 minutes), each friend F_i of U_m can perform an amount of R_n update request. Requests of F_i are only performed if F_i is online as well as replies are only sent if the target is online. F_i's chosen targets depend on F_i's "best-friends" subgroups. After each slot, all friends of U_m have adapted their online status. In multiple test-runs, we monitored the time which is required to distribute U_m's new data container among at least 33% of U_m's friends. This is feasible in our scenario since we also assume that not more than 33% of U_m's friends belong to U_m's best friends. Table 2 shows results. In average, U_m's new data container was distributed in 11.91 hours if U_m's friends request their friends for updates every minute depending on their best-friend subgroups. In order to improve the container propagation, we enforced additional requests of U_m's friends in each time slot. Multiple update requests are feasible since the most requests do not finally lead to the transfer of a data container. If we assume that the clients of U_m's friends additionally perform a request to one of their friends (randomly chosen) every minute ($+\ random_6$), the container of U_m is distributed within 5.47 hours. If we

[1] `http://blog.stephenwolfram.com/2013/04/`

Table 2. Time required to propagate an arbitrary user's data container within 33% of this user's friends (100 test-runs)

R_n within time slot	mean [h]	std [h]
6	11.91	5.86
6 + $random_6$	5.47	2.67
6 + $random_{24}$	2.45	2.15

increase the number of enforced requests to one request every $15s$ (+ $random_{24}$), U_m's container is distributed in 2.45 hours.

5 Conclusion

To the best of our knowledge, ProofBook is the first OSN architecture which is based on a combination of two approaches: First, a Proof-of-Work (PoW) architecture is implemented which mitigates certain network attacks like Denial-of-Service as well as allows preferring users with better network performance in data transfers. In order to combine OSN mentality and the PoW, incentives are introduced and up-to-date data containers can be only retrieved unhindered if the users follow the underlying protocol. Second, ProofBook benefits from a friend-based data propagation approach which is based on the ability of friends of a user replying to update requests targeting the data container of this user. The privacy level is enhanced since the users' data is only stored on the users' friends rather than on unknown peers. The sources of requests are anonymous and can only be decrypted by the target group of friends. In an evaluation, we showed that the PoW scheme helps to limit the amount of requests that can be successfully performed and accordingly mitigates certain attacks. We showed that the friend-based data propagation scheme is feasible under realistic conditions. An implementation and a user-interface are crucial for the success of any OSN and required to perform further evaluations of the proposed architecture. This as well as a more large-scaled evaluation belong to our future work.

Acknowledgments. Thorsten Strufe is partially supported by MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013. Andreas Peter is supported by the THeCS project as part of the Dutch national program COMMIT.

References

1. Back, A.: Hashcash: A denial of service counter-measure (2002)
2. Baumgart, I., Heep, B., Krause, S.: OverSim: A flexible overlay network simulation framework. In: Proceedings of 10th IEEE Global Internet Symposium (GI 2007) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA, pp. 79–84 (2007)

3. Belenkiy, M., Chase, M., Erway, C.C., Jannotti, J., Küpçü, A., Lysyanskaya, A.: Incentivizing outsourced computation. In: Proceedings of the 3rd International Workshop on Economics of Networked Systems, NetEcon 2008, pp. 85–90. ACM, New York (2008)
4. Belenkiy, M., Chase, M., Erway, C.C., Jannotti, J., Küpçü, A., Lysyanskaya, A., Rachlin, E.: Making p2p accountable without losing privacy. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society, WPES 2007, pp. 31–40. ACM, New York (2007)
5. Bielenberg, A., Helm, L., Gentilucci, A., Stefanescu, D., Zhang, H.: The growth of diaspora - a decentralized online social network in the wild. In: 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 13–18 (2012)
6. Buchegger, S., Schiöberg, D., Vu, L.-H., Datta, A.: Peerson: P2p social networking: early experiences and insights. In: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems, SNS 2009, pp. 46–52. ACM, New York (2009)
7. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a privacy preserving online social network leveraging on real-life trust. IEEE Communications Magazine 47(12) (December 2009), Consumer Communications and Networking Series
8. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)
9. Graffi, K., Mukherjee, P., Menges, B., Hartung, D., Kovacevic, A., Steinmetz, R.: Practical security in p2p-based social networks. In: IEEE 34th Conference on Local Computer Networks, LCN 2009, pp. 269–272 (October 2009)
10. Hua, C., Mao, Y., Jinqiang, H., Haiqing, D., Xiaoming, L.: Maze: a social peer-to-peer network. In: IEEE International Conference on E-Commerce Technology for Dynamic E-Business, pp. 290–293 (September 2004)
11. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, pp. 29–42. ACM, New York (2007)
12. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
13. Pouwelse, J.A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D.H.J., Reinders, M.J.T., Steen, M.R.V., Sips, H.J.: TRIBLER: a social-based peer-to-peer system. Concurrency and Computation: Practice and Experience 20, 127–138 (2008)
14. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems (2001)
15. Singh, A., Castro, M., Druschel, P., Rowstron, A.I.T.: Defending against eclipse attacks on overlay networks. In: SIGOPS European Workshop (2004)
16. Varga, A.: Using the omnet++ discrete event simulation system in education. IEEE Transactions on Education 42(4), 11 (1999)