Cryptography Enhanced Ad-Hoc Approach to P2P Overlays

Michal Zima
Faculty of Informatics
Masaryk University
Brno, Czech Republic
E-mail: xzima1@fi.muni.cz

Eva Hladká
Faculty of Informatics
Masaryk University
Brno, Czech Republic
E-mail: eva@fi.muni.cz

***Abstract*—We address the problem of a secure direct communication of two arbitrary peers in a P2P network without knowing each other's IP addresses. An efficient solution to this problem will provide a practically usable way of communication to privacy-sensitive P2P applications. Traditionally, P2P architectures view this problem as a problem of mutual anonymity of a message sender and receiver, but usual solutions suffer from various inefficiencies or complexities. By looking at the problem from a perspective of an ad-hoc network, we are able to apply a familiar approach of multihop communication and ad-hoc routing algorithms to a P2P overlay. Introduced usage of a public key as a node's identifier adds further security features, including data integrity through digital signatures, or end-to-end encryption. Proposed P2P overlay has been successfully used for building a decentralised cryptocurrency exchange Coincer.**

I. Introduction

Many applications need arbitrary two peers to be able to communicate with each other, nonetheless some of them extend this requirement to also minimising a risk of exposure of peers' IP addresses (identities). The motivation might be just to tackle privacy concerns or to prevent targeted attacks against users of the application. In our case, we develop a decentralised exchange of cryptocurrencies: Coincer [1]. By hiding IP addresses we want to provide users with a similar level of privacy as cryptocurrencies usually offer. Moreover, as Coincer's trading protocol may span from tens of minutes (due to inherent properties of the cryptocurrencies in order to maintain security of the trade) up to several hours (depending on degree of cooperation between the respective users) it is very beneficial to conceal IP addresses of peers, since it hinders efforts of attackers to prevent their trading partner from finishing the trade or even directly attacking them.

In this paper we propose a simple unstructured anonymous P2P overlay with efficient communication. We treat an unstructured P2P network as an ad-hoc network. Hence, we modify for this purpose a simple ad-hoc routing algorithm DSDV [2], which provides the overlay with an any-to-any anonymous communication channel with low setup, maintenance and communication overhead. By using space efficient elliptic curve cryptography (ECC) public keys as peer identifiers we obtain security against message tampering and ability to establish end-to-end encrypted channels.

The remainder of this paper is organised as follows. Section II presents previous work related to the problem. In Section III we discuss in more details general requirements on our overlay, with elaboration on the actual overlay design in Section IV. Section V discusses how the proposed overlay behaves when it is under different kinds of attacks. An example of a real-world application being built on top of our work is presented in Section VI. Section VII provides final conclusions.

II. Related Work

There are several different approaches to anonymous P2P overlays. One common approach is to form groups of peers within which messages are delivered via broadcast to achieve anonymity of their receiver, as in P^5 [3] or Hordes [4]. Similarly, for anonymity of a sender a probabilistic forwarding is used in Crowds [5]. In every step a forwarding peer randomly decides with a given probability whether to forward the message to another randomly-chosen forwarding peer or whether to deliver it to its final destination. AP3 [6] further uses this method to build anonymous channels for accessing Pastry DHT [7].

A similar approach to broadcasting is the usage of epidemic (or gossip) protocols. MuON [8] achieves higher bandwidth efficiency while preserving the level of anonymity of broadcasting.

Another extensively used approach is onion routing [9].

A-Kad [10] adds anonymity to the Kademlia overlay [11]. In A-Kad a peer sets up anonymous channels using onion routing over chosen sets of onion routers. Subsequently, they uses these channels for publishing information into DHT. A peer who wants to obtain a resource they found via DHT then communicates with the resource provider over these anonymous channels, thence neither of them knows the other peer's identity.

Pecan [12] further improves onion routing by relaxing a requirement to establish circuits before a communication can take place. A sender chooses for every request two sets of routers—one for delivering the request itself and one for anonymous routing of the response. The responding peer therefore receives with the request also information necessary for delivering the response.

The issue with onion routing is a requirement for a peer to know routers in the network in order to either establish circuits or to select path for their packets (in the case of Pecan). Not only there is reliance on central directory server, but the communication is also sensitive to churn. To improve these properties while avoiding overhead of broadcast in a mutually anonymous 2-party communication we employ features of ad-hoc routing algorithms DSDV [2] and Babel [13].

DSDV is a simple loop-free routing algorithm. Although not designed for anonymous communication, it well handles communication between any two nodes by maintaining information what is a next hop on the shortest path towards each other node. Babel enhances DSDV and other ad-hoc routing protocols and for usage in overlay network it replaces TTL-based routing metric with a delay-based routing metric [14]. Aside from anonymity features, we improve in our routing algorithm the usage of inherently present redundancy and try to minimise communication overhead.

III. Our Requirements

Motivated by our application, a cryptocurrency exchange, we impose several requirements we want the P2P overlay to accomplish. First, a connection between a peer's identifier and their IP address must not be made known—not even to peer's neighbours if possible. Thereby users will experience up to the same level of privacy as in cryptocurrencies themselves. Second, even under this condition, we still need a way for unicast-like communication between any two peers with any of them being able to initiate the communication. Third, we want to be as efficient in message delivery as possible, therefore eliminating a highly anonymous, but also very inefficient broadcasting mechanism. Fourth, for Coincer it is important that any user is able to participate in the network, even if they is behind a NAT. Fifth, a P2P network witnesses continuous churn. On the other hand, its topology is implicitly redundant which is desirable to use for improving reliability. And finally, as a network of unknown nodes forms an inherently untrustworthy environment, we need reasonable ways for operating in such an environment.

In this section we elaborate on these requirements.

A. Anonymity of Peers

Disconnecting peer's identifier from their IP address servers two basic goals. First, it is unlinkability of information or actions to their real originator, which includes resource possession, its provision or retrieval. It provides privacy to the user. Second, it hinders situation of an attacker, because they does not know who is who in the network and where. Beyond simple network attacks, a user of Coincer might be a target of a money heist which is significantly more difficult to execute when the attacker does not know victim's IP address.

In cryptocurrency networks it is usual that a user who sends a transaction is indistinguishable from a user who only forwards this transaction to other peers. This property ensures that not even a direct connection of an attacker to their intended victim gives them the information that this is actually the user to be attacked. Therefore, we strengthen our former requirement of anonymity in the network also to anonymity towards neighbours.

B. Efficient Communication Between Peers

Generally, communication in a P2P network can be divided into two categories: globally-relevant and private. Messages belonging to the globally-relevant category define and maintain the state of the network (e.g., market orders in Coincer), while private messages are part of a communication between two specific peers, which includes transmission of a file in a file sharing application or an execution of a cryptocurrency trade in Coincer. Based on this division we also distinguish between ways each of these two categories is delivered over the network: while globally-relevant can be broadcast (or, more efficiently, disseminated by a gossip protocol), flooding the network with private messages addressed to a single peer is not only very inefficient (although it is indeed very anonymous), but also forms a scalability bottleneck for the network. Therefore, we seek for a more ingenious routing mechanism for the purpose of direct communication between peers.

C. Possibility of Full Participation

P2P overlays often rely on the ability of peers to receive incoming connections from the Internet, otherwise they cannot fully participate in the overlay and their resource utilisation is limited (e.g., in file sharing overlays). However, in real world there are many users for whom this assumption is not true, be it for any possible reason—a NAT due to a lack of IPv4 addresses, a restrictive firewall, or other potential cause.

Our goal is therefore to ensure that any peer can initiate communication with any other peer without limitations or a need for explicit path-building. This requirement also includes all peers connecting to the network via Tor or other proxy networks.

D. Redundancy

Churn belongs to natural phenomena of P2P networks. Peers join and leave the network or just change their location within it. While graceful departures are usually preceded by notices and the network can adjust its paths, sudden disconnections of peers, e.g., because of a failure of the underlying network infrastructure, take time to detect and recover from, causing broken routes and local disruptions of the overlay operation.

P2P networks contain peers with multiple connections, forming a mesh. This inherent redundancy can be leveraged for better routing resilience and fault-tolerant overlay connectivity, therefore providing a quick recovery of overlay operation in case of any failure.

IV. Design of the Overlay

Unstructured P2P overlays resemble ad-hoc networks—peers connect to the overlay at random points and there is lack of any inner structure in the network, all peers are equivalent. In order to avoid reliance on the ability to receive

new connections from other peers, we carry all communication over existing connections among peers. Therefore, utilising an ad-hoc routing algorithm, messages are delivered on hop-by-hop basis, similarly to IP packets. For redundancy, each peer maintains a set of connections to the network.

Some approaches equip messages with a time-to-live (TTL) field to limit potential flooding of the network. However, we drop it, because this overlay uses either one-hop messages, messages supposed to reach all peers or messages routed towards a single destination. Moreover, TTL is easy to spoof, it creates problems with maintaining anonymity [15] and we already employ other ways for detecting repeated messages.

All messages routed through the overlay carry an identifier of their sender. However, any message sent only to a neighbour (whose identifier is never known) does not contain neither the identifier of sender, nor the identifier of receiver. Hence, when a peer sends a message with their identifier, no neighbour can tell whether the peer is the sender or whether they just forwards the message.

A. Joining

Before a node connects to the network, they generates a key pair. The public key is used as their identifier in the network. We recommend a new identifier to be used every time a node joins the network. However, if the application establishes long-lasting interactions, the previous identifier may be required for finishing open interactions. In such cases more identifiers may be used at the same time, nonetheless only the newest is allowed to establish new interactions.

Having an identifier, a node starts the following joining procedure:

1) They establishes a connection to a node known to be in the network.
2) The node asks this new neighbour for a list of addresses of known nodes in order to update their database of possible peers to connect to.
3) They repeats the procedure until it has enough of outgoing connections.

Initial seed addresses, used by nodes that have never before connected to the network, can be for instance hard coded in the application, provided manually by the user, served over HTTP or obtained from DNS. If DNS is used, served records need to have sufficiently low TTL (of the order of minutes) so that if none of provided nodes is alive, the DNS can be queried again after a while for a different set of records without hitting a cache.

B. Routing

Ad-hoc networks use various routing algorithms, of which we select as a base a proactive DSDV algorithm [2] as it best accomplishes our requirements and fits in with achieving anonymity of peers. Algorithm with our modifications follows.

When a peer wants to start actively participating in the network, they needs to let the network know how to reach them. Until this moment they is only a passive peer gathering information about resources offered by other peers. If their participation is to be initiated by broadcasting a message, this message also serves the purpose of establishing routes. However, for unicast-like communication, an announcement of presence needs to be broadcast first, otherwise there will be only a single path back.

Each peer tracks announcements of presence (or first broadcast messages) for each other peer. When a peer receives such a message, they saves the information which neighbour forwarded it to them. Neighbours are sorted by the order in which the message is received from them. This order also specifies a priority for selection of a route when delivering a message to this peer. This way each peer in the network knows where to forward messages for any other active peer in the network, although they does not know neither the path towards this peer, nor whether any of their neighbours is that corresponding peer.

C. Loop-free Routing

Under normal circumstances the routing protocol does not form loops. However, it is possible that the optimal route breaks (e.g., some peer on the path leaves the network) and the rest of routing information in the network causes messages on this route to loop.

Solutions of other algorithms vary. DSDV [2] broadcasts information about broken route on every node departure, which ignores available redundancy in the network and might generate large amount of traffic on high churn. In contrast to that, Babel [13] uses short intervals between consecutive announcements of presence as a preventive measure against both loops and general disruption of routing. Nonetheless, incurred overhead may become nonnegligible when number of peers in the network is high.

Our solution is based on loop detection and last resort reaction. In order to detect a loop, each peer maintains a set of last n hashes of messages they forwarded in the last minute with information about which neighbour forwarded it to them. If a peer detects a message repetition from a neighbour different from the first forwarder, they detected a loop. To remove the loop, the peer deletes their best route for given destination (which now contains a loop) and forwards the message via the second best (which is now the first best) route. When a peer has no route left, they broadcast a request for announcement of presence. Therefore, if the peer is still connected to the network, they will receive the request and respond to it (again via broadcast), which refreshes routing information among all peers.

Note that if a message comes for a second time from the same neighbour, it only says that the neighbour sent it again (and the message is ignored). If there was to be a loop, this neighbour is supposed to detect it first.

D. Routing Table Updates

There is no mechanism in the proposed overlay that could in a short time detect unannounced departures of peers. Not even neighbours are able to detect it as the network cannot establish connection between nodes and their identifiers. Therefore, all

routing table entries are assigned an expiration time. Hence, if a peer wants to preserve routability of messages towards them, they has to periodically rebroadcast their announcement of presence. This must be one ahead of the expiration time, otherwise the peer risks becoming unreachable.

Expiration ensures that peers do not store stale records and utilise their resources in an efficient manner.

E. Graceful Departure

Disconnection of passive peers is trivial and no other peer in the network needs to take any action. On the other hand, active peers who propagate their presence, leave information in other peers' routing tables and possibly also in application-specific data structures. However, sending an announcement about departure and consequently disconnecting from neighbours may lead to backward anonymity breach and exposure of relation between peer's identifier and IP address.

If the application conditions allow it, we suggest abandoning the conception of graceful departure in favour of increased anonymity of peers. Nonetheless, in applications that could greatly benefit from early knowledge of peers' departures, especially in terms of resource saving, a peer broadcasts this announcement in advance of their actual departure, therefore making the possibility of raising correlations less likely.

F. Peers Exchange

Usually, peers exchange IP addresses either of their current neighbours or of a subset of all nodes they know. Giving away a list of peer's neighbours is sensitive and potentially exploitable information. Therefore, the list must not in any way indicate which addresses are possibly the node's neighbours.

Similarly, a receiver of such a list needs to exercise caution: to randomly reorder addresses on the list and combine data from several neighbours.

G. Cryptography

An anonymous overlay forms a very untrusted environment. For safe and reliable communication over it we include several cryptographic elements. First of all, we use public keys in place of peers' identifiers. Consequently, as every message to be forwarded has to contain an identifier of its sender (note that messages for neighbours do not), public keys of all active peers are implicitly distributed over the network to all peers.

With key distribution secured, we require all messages carrying the sender's identifier to be signed by their key corresponding to their identifier. Consequently, all peers are required to perform signature verification of each received message before its processing. If the verification fails, integrity of the message has been corrupted and the message is therefore silently discarded. Only messages with valid signatures are allowed to be forwarded through the network.

For any unicast-like communication of two arbitrary peers we require it to be encrypted in end-to-end fashion. To avoid any complicated shared key agreement protocol we suggest usage of elliptic curve cryptography (ECC) that is able to make use of a very elegant ECDH algorithm for establishing a shared key between respective peers without any extra communication overhead [16]. Besides, keys of ECC schemes have the benefit of being small in size.

V. Attack Vectors

In this section we present an analysis of attacks deployable against proposed overlay.

A. Man-in-the-Middle Attack

This is the most obvious attack vector as messages to distant peers are always sent via other peers and certain messages are even broadcast to everyone. Any peer could therefore monitor contents of communication between particular peers or even tamper with them. To prevent deliberate modifications or random corruption of messages, all messages must include digital signatures. To provide confidentiality, end-to-end encryption of contents is used. In order to bypass complicated key exchange procedures, peers use their public keys as their identifiers. Therefore, any peer is able to address any other peer and to start sending them encrypted messages right away, and everyone is also able to verify signature of every message.

To save bandwidth and processing power, corrupted messages are not forwarded, but instead immediately silently discarded.

Depending on a particular application, a peer acting as a counterparty, but in fact carrying out this attack, could be easily detected. Real parties can produce a proof using their unique knowledge or possession which is very hard or impossible for the attacker to forge. In our application, a cryptocurrency exchange, this proof consists of a signature of both parties' identifiers by respective party's "coins" (these are identified by public keys in cryptocurrencies). Hence, the attacker is quickly uncovered and the attack thwarted.

B. Replay Attack

While modifying messages is prevented by digital signatures, attackers could still keep forwarded messages and send them later to the network again. This might easily create false state of affairs in the network, with severity varying depending on a particular application.

To eliminate potential replay attacks, every message sent within the network carries a *nonce*. The nonce must be a positive ever-increasing number. This nonce is unique within messages sent by one specific peer. If a peer reuses their identifier across restarts, they also must remember the value of their last used nonce. Otherwise, it might occur that if a peer uses some nonces again then (a) their messages might be discarded by some peers (possibly even by all of them), or (b) an attacker might carry out a replay attack against this peer. For peers with new identifiers this problem does not arise.

We distinct two types of nonces with regards to importance of messages that carry them:

1) Message for Announcement of Presence: This is a very special message, because it ensures routability within the network and reachability of peers. Therefore, its nonce needs to be stored separately so that it is not accidentally pruned with stale nonces of other messages.

Repetition of this message from a neighbour that has already sent it is ignored. Delaying of the announcement by some neighbours only causes these neighbours to be assigned the lowest priority for route selection.

2) Other Messages: Both kinds of messages—those that are broadcast to every peer and those belonging to a communication with a specific peer only—may arrive out of order, i.e., in different order than they were originally sent. Therefore, it is not sufficient to remember only the last nonce, because that could easily cause discarding legitimate messages that arrived later than expected.

Proposed solution is to store nonces of messages received in recent moments (in Coincer we use one minute), which is sufficient for legitimate traffic and still keeps associated storage overhead in reasonable bounds. If a newly arrived message has a nonce lower than the oldest nonce in the list, it is discarded. If it is higher than the oldest nonce, but still lower than the last one, the message is accepted, but its nonce is not stored at the end of the list as normally, but—in order to maintain the ever-increasing property of the nonce list—in the position it should be stored so that the list remains sorted. For its storage is then not used the real time of its reception, but the same timestamp as of the nonce preceding it in its position in the list. Stale nonces are always removed from the list as soon as possible, except for the last one—at least one nonce must always be remembered (only for newly connected peers this list may be empty).

C. Sybil Attack

Under a Sybil attack, an attacker introduces into the network many peers under their control [17]. The more peers, the higher is the probability that a random new peer connects solely to these malicious peers. Hence, the more outgoing connections a peer maintains, the lower is the probability it becomes a victim of this attack.

Sensitivity of each application to Sybil attacks needs to be taken into account when considering a recommended minimum number of outgoing connections. While possible damage in a case of some applications might be negligible, for others such an attack may pose a serious threat. Coincer is an example of the first group; a representative of the second group in general, not for this specific overlay, is Bitcoin.

In Bitcoin a user is provided with the transaction history of the whole network. Therefore, if they is connected only to attacker's nodes, they can be tricked into believing in incorrect state of the history, e.g., that they received some transactions, although they did not in the real history. On the other hand, it is not possible to cause harm to a user of Coincer, because the underlying cryptocurrency protocol for trading ensures that the attacker would have to successfully carry out a Sybil attack against the user on at least one cryptocurrency (with the consequences stated above). Nonetheless, at that moment the attacker gains very little from carrying out a Sybil attack also against Coincer itself.

D. Denial of Service Attack

A Denial of Service (DoS) attack can have many forms. Generally, its goal is to prevent user(s) from achieving the purpose of the application or network. We cover only a generic attack scenario from the overlay perspective—flooding the network—, i.e., leaving out any application-specific DoS variations.

Peers injecting excessive amount of messages into the network are consuming available resources in bad faith and need to be expelled out of the network. If a peer receives too many messages from a particular neighbour, first they should either limit forwarding of these messages or even filter them based on their importance for the network. If the neighbour continues flooding, the peer closes its connection to this neighbour.

VI. Application

The application described in this section has been our original motivation for development of the P2P overlay described in this paper. Our use case is a decentralised exchange for cryptocurrencies—Coincer [1]. It leverages cryptographic features of cryptocurrencies to allow users to safely exchange different cryptocurrencies directly between themselves without any third party. Cryptocurrency applications are generally privacy-sensitive and attacks-attracting, therefore we strove to achieve similar level of anonymity as cryptocurrencies usually provide.

Purpose of Coincer, from the perspective of this paper, is to globally maintain a decentralised market of users' sell/buy offers on top of the overlay and to provide an appropriate communication channel for execution of trades between users. Therefore, it utilises both available communication patterns—market messages for distribution of offers are broadcast, while communication related to particular trade between some two peers is carried out in unicast fashion. The latter also makes use of implicitly available means for encryption of messages, which is important for privacy. Also very useful is the feature of integrity assurance of all messages through digital signatures, because it precludes the possibility of impersonating honest users by an attacker.

Coincer itself does not extend the list of possible attacks on the overlay. It rather makes several of the described attacks harder to execute, namely the man-in-the-middle attack and the Sybil attack. In the case of the Sybil attack, an attacker has to also carry out a successful Sybil attack in the cryptocurrency network where the victim would like to buy coins from the attacker. However, under such conditions it is no longer necessary to execute a Sybil attack in Coincer's network, because it does not influence in any way the primary attack itself.

The proposed overlay enables Coincer to accomplish its goals, while providing good protection and level of anonymity to the users with a reasonably low overhead.

Coincer is published as a free software and its development version is already publicly available.

VII. Conclusion

We addressed a problem of safe mutually anonymous communication in a P2P overlay. We leveraged similarities between an ad-hoc network and an unstructured P2P overlay in the design of our overlay in order to provide desired anonymity properties. As a result, any peer in the overlay is able to initiate communication with another peer, without knowing their real IP address—using only their identifier within the overlay. Thanks to establishing peers' public keys as identifiers, every message is digitally signed and the usage of ECC provides an instant access to end-to-end encryption without any need for negotiation of a shared key. Usefulness of our proposal for privacy-sensitive applications is demonstrated on a decentralised cryptocurrency exchange Coincer.

Acknowledgement

The authors would like to thank all reviewers for their valuable feedback which helped improve the paper.

References

[1] M. Zima, "Coincer." [Online]. Available: https://www.coincer.org/

[2] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM computer communication review*, vol. 24. ACM, 1994, pp. 234–244.

[3] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A protocol for scalable anonymous communication," *Journal of Computer Security*, vol. 13, no. 6, pp. 839–876, 2005. [Online]. Available: http://content.iospress.com/articles/journal-of-computer-security/jcs245

[4] B. N. Levine and C. Shields, "Hordes — A Multicast Based Protocol for Anonymity," *Journal of Computer Security*, vol. 10, no. 3, pp. 213–240, 2002.

[5] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=290168

[6] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: Cooperative, decentralized anonymous communication," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. ACM, 2004, p. 30. [Online]. Available: http://dl.acm.org/citation.cfm?id=1133578

[7] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Springer, 2001, pp. 329–350. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-45518-3_18

[8] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo, "MuON: Epidemic based mutual anonymity in unstructured P2P networks," *Computer Networks*, vol. 52, no. 5, pp. 915–934, Apr. 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128607003349

[9] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, May 1998. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=668972

[10] Y. Ni, D. Nyang, and X. Wang, "A-Kad: an anonymous P2P protocol based on Kad network," in *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*. IEEE, 2009, pp. 747–752. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5336924

[11] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems*. Springer, 2002, pp. 53–65. [Online]. Available: http://link.springer.com/10.1007%2F3-540-45748-8_5

[12] G. Xu, L. Aguilera, and Y. Guan, "Pecan: A circuit-less p2p design for anonymity," in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 820–825. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6503214

[13] J. Chroboczek, "RFC 6126: The Babel routing protocol," 2011. [Online]. Available: http://tools.ietf.org/html/rfc6126

[14] B. Jonglez, M. Boutier, and J. Chroboczek, "A delay-based routing metric," *arXiv preprint arXiv:1403.3488*, 2014. [Online]. Available: http://arxiv.org/abs/1403.3488

[15] R.-Y. Xiao, "Survey on anonymity in unstructured peer-to-peer systems," *Journal of Computer Science and Technology*, vol. 23, no. 4, pp. 660–671, 2008. [Online]. Available: http://link.springer.com/article/10.1007/s11390-008-9162-7

[16] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement," Designs, Codes and Cryptography, Tech. Rep., 1998.

[17] J. R. Douceur, "The Sybil Attack," in *Peer-to-peer Systems*. Springer, 2002, pp. 251–260. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-45748-8_24