2013 IEEE 33rd International Conference on Distributed Computing Systems

RAC: a Freerider-resilient, Scalable, Anonymous Communication Protocol

Sonia Ben Mokhtar
CNRS
Lyon

Gautier Berthou
Grenoble University
Grenoble

Amadou Diarra
Grenoble University
Grenoble

Vivien Quéma
Grenoble INP
Grenoble

Ali Shoker
CNRS
Lyon

***Abstract*—Enabling anonymous communication over the Internet is crucial. The first protocols that have been devised for anonymous communication are subject to freeriding. Recent protocols have thus been proposed to deal with this issue. However, these protocols do not scale to large systems, and some of them further assume the existence of trusted servers. In this paper, we present *RAC*, the first anonymous communication protocol that tolerates freeriders and that scales to large systems. Scalability comes from the fact that the complexity of *RAC* in terms of the number of message exchanges is independent from the number of nodes in the system. Another important aspect of *RAC* is that it does not rely on any trusted third party. We theoretically prove, using game theory, that our protocol is a Nash equilibrium, i.e, that freeriders have no interest in deviating from the protocol. Further, we experimentally evaluate *RAC* using simulations. Our evaluation shows that, whatever the size of the system (up to 100.000 nodes), the nodes participating in the system observe the same throughput.**

I. Introduction

Anonymous communication protocols are important for they allow the dissemination of sensitive content over the Internet. The first protocols that have been proposed in the literature to enable anonymous communication are DC-Net [1] and onion routing [2]. These two protocols have focused on enabling the strongest possible anonymity for the former, and on providing practical performance for the latter.

An issue shared by the two above protocols, as well as by other protocols devised using the same principles (e.g., [3], [4]) is that they take the participation of nodes for granted. This assumption is unrealistic in collaborative systems, which are well-known to be perfect playgrounds for freeriders. A freerider is a node that benefits from the system, while trying to minimize its contribution to it, in order to save resources. Research has thus moved towards the design of freerider-resilient anonymous communication protocols.

The first protocol that has been devised to deal with freeriders is called Dissent v1 [5]. This protocol forces nodes to participate in the protocol. Specifically, for each message sent anonymously, Dissent v1 forces each node in the system to send messages to all the other nodes. As such, it is easy to detect whether a node contributed its fair share to the system. Unfortunately, this approach yields very poor performance. Indeed, this protocol becomes unpractical for systems involving as little as 40 or 50 nodes.

A second protocol, called Dissent v2 [6], has been very recently proposed with the aim to improve the performance achieved by Dissent v1. The key idea of this protocol is to rely on a set of trusted nodes to avoid involving all the nodes in the system for every anonymous communication. In this protocol, each trusted node receives anonymous communication requests from untrusted nodes and runs a protocol involving all-to-all communications between trusted nodes. This considerably reduces the number of messages exchanged for every anonymous communication, which allows Dissent v2 to exhibit better performance than Dissent v1. However, as we show in Section III, this protocol is still not scalable: trusted nodes are involved in all communications and the throughput achieved by the protocol drastically decreases when the number of nodes in the system increases, until it reaches zero (for a system with 100.000 nodes). Besides, this protocol relies on trusted nodes, which is something that people sending anonymous data usually prefer to avoid.

Our contribution in this article is *RAC*, a freerider-resilient anonymous communication protocol that scales to large systems. Regarding the resilience to freeriders, we theoretically prove, using game theory, that our protocol is a Nash equilibrium, i.e, that freeriders have no interest in deviating from the protocol. Regarding scalability, both the number of broadcast messages and the size of the groups in which the broadcast messages need to be sent, are *independent* from the number of nodes in the system. Specifically, in our protocol, these parameters are exclusively dependent on constants that are associated with the degree of anonymity our system guarantees. In other words, *RAC* is scalable and it exhibits a clear tradeoff between anonymity and performance. Finally, *RAC* does not assume any trusted server.

We experimentally evaluate RAC using simulations. Our evaluation shows that RAC achieves a very good level of anonymity. Moreover it shows that, contrary to Dissent v1 and Dissent v2 that exhibit a drop of throughput when the number of nodes grows, the throughput of *RAC* remains constant when increasing the size of the system.

The remaining of the paper is structured as follows. We first introduce some definitions and analyze the related work in Section II. We then discuss the need of a new protocol in Section III, before presenting our protocol in Section IV. We further present the proofs of *RAC* freerider resiliency, as well as the proofs of *RAC* anonymity guarantees in Section V. We finally present the experimental evaluation of *RAC* in Section VI, and our concluding remarks in Section VII.

1063-6927/13 $26.00 © 2013 IEEE
DOI 10.1109/ICDCS.2013.52

II. Definitions and related work

We start this section by the definition of anonymity properties guaranteed by anonymous communication protocols. We then review existing anonymous communication protocols.

A. Definitions

We rely in this paper on the definitions introduced by Pfitzmann and Hansen [7]. We consider a system composed of nodes that communicate with each other via communication channels. In such a system, a node is acting *anonymously* if it is impossible for an observer to distinguish this node from the other nodes present in the system. Specifically, researchers distinguish the following three anonymity properties. The first one is *sender anonymity*. This property holds if it is not possible to identify the sender of any given message. The second property is *receiver anonymity*. This property holds if it is not possible to identify the destination of any given message. The third property is *unlinkability*. This property holds if an observer is not able to identify a pair of nodes as communicating with each other.

The goal of an anonymous communication protocol is to guarantee one or more (preferably all) of these anonymity properties in presence of an opponent, i.e., a malicious entity trying to break these properties. All existing protocols, as well as the protocol we present in this paper, consider the strongest possible opponent, called the *global* and *active* opponent. *Global* means that the opponent can monitor and record the traffic on all the network links. *Active* means that the opponent can control some nodes in the system and make them deviate from the protocol in order to reduce the anonymity of other nodes. The higher the number of nodes that the opponent must control to break a protocol, the stronger the anonymity guaranteed by this protocol. The only limitation of the global and active opponent is that it is not able to invert encryption.

B. Related work

The two pioneering protocols for anonymous communication are the DC-Net [1] and Onion routing [2] protocols.

The DC-Net protocol exhibits strong anonymity guarantees. Specifically, it is not possible for an opponent to break anonymity without controlling all the nodes executing the protocol. DC-Net reaches this objective by relying on the principle of secret-sharing. Specifically, nodes in DC-Net proceed in rounds. During one round, only one node is allowed to send a message. If two nodes send a message during the same round, there is a *collision* and none of the messages is correctly delivered. To avoid collisions, there exist mechanisms for reserving sending slots [8], [9]. To guarantee anonymity, DC-Net organizes nodes in a structured network and requires nodes to forward encrypted messages received from their neighbors. Nodes then rely on a XOR-based mechanism (that they apply on messages they receive from their different neighbors) to decrypt messages. The major limitation of DC-Net is that it yields a considerable overhead. Indeed, at every round, every pair of nodes in the system needs to exchange messages. This is why other protocols that aim at reducing the cost of DC-Net have been devised. The Herbivore protocol [4] is one such protocol. In this protocol, nodes are organized in groups, which limits the exchange of messages inside the groups. Despite this optimization, this protocol is considered as unusable in practice as soon as the system grows to more than (approximately) 50 nodes, as analyzed in [10].

The onion routing protocol guarantees a lower degree of anonymity than DC-Net. It is nevertheless considered secure enough to be widely used in practice [11]. Furthermore, it exhibits way better performance. This protocol works as follows. A node wanting to send a message randomly selects a list of nodes (called relays) and encrypts the message into multiple layers of encryption: one layer for every relay. The resulting encrypted message is called an onion. The node then sends the onion to the first relay that deciphers the first layer. This layer contains the address of the second relay and an inner onion, encrypted with the key of the second relay. The first relay sends the inner onion to the second relay. This process is repeated until the message reaches its destination. Note that because of its onion structure, a forwarded message changes at each relay. Various variants of this protocol have been proposed in the literature, such as crowds [3], cashmere [12], Tarzan [13], and TOR [11]. The three first protocols aim at better tolerating churn. The last paper describes a popular implementation of the onion routing protocol.

The major limitation of these protocols is that they assume that nodes participating in the system are *altruistic*, i.e., they follow the protocol. However, anonymous communication protocols are subject to nodes that freeride in order to benefit from the system, without contributing their fair share to it. This explains why a recent attention has been given to the design of freerider-resilient protocols, in which all nodes are obliged to participate, otherwise risking eviction.

The first protocol that follows this direction is the Dissent v1 protocol [5]. This protocol relies on the principles of DC-Net to which it adds a double encryption system. This system allows stopping the execution of a round whenever a node detects the misbehavior of another node (including freeriding). It then allows exposing the misbehaving node without breaking anonymity. While this protocol is resilient to freeriding, it suffers from the same performance limitations as DC-Net, as further analyzed in Section III. A second protocol, i.e., Dissent v2 [6], has very recently been proposed to address this performance issue. The key idea behind this protocol is to run Dissent v1 on a small number of trusted servers. These servers are then used by a large number of nodes (called clients in [6]) to enforce the anonymity of their communications. To reach this objective, nodes need to trust the servers they use to exchange messages, which is a strong assumption. Performance wise, as analyzed in the following section, even though Dissent v2 exhibits better performance that Dissent v1, it still suffers from a scalability issue.

III. The Case for a New Protocol

In this section, we motivate the need for a new anonymous communication protocol. We show, using simulations, that

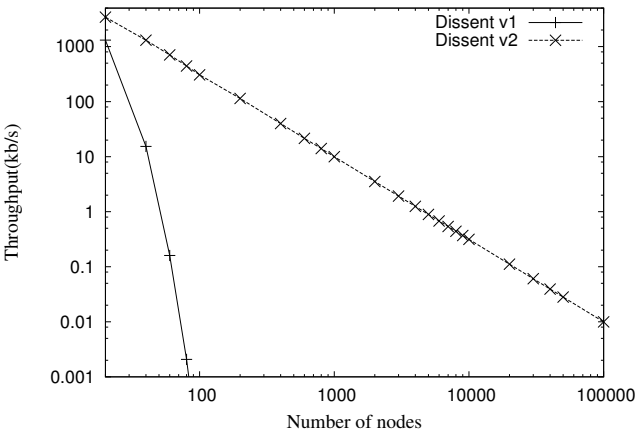

Fig. 1. Throughput as a function of the number of nodes for Dissent v1 and Dissent v2.

the two existing freerider-resilient anonymous communication protocols achieve poor performance in large-scale systems. We simulate (using the Omnet++ simulator), a network of nodes connected via 1Gb/s links[1]. In our simulation, each node chooses a random node to which it sends anonymous messages of a fixed size equal to 10kB, at the highest possible throughput it can sustain. We run this experiment several times with an increasing number of nodes, and we measure the average throughput at which nodes receive anonymous messages.

We configure Dissent v2 with the optimal number of trusted servers for each network size. Specifically, increasing the number of trusted servers allows reducing the number of messages processed by each trusted server. However, it also increases the number of broadcast messages exchanged between trusted servers, and the size of these broadcasts. For each different number of nodes in the system, there is thus an optimal number of servers to be used, which maximizes the throughput of the protocol. Moreover, in order to balance the load, we equally distribute the number of nodes between trusted servers.

The throughput as a function of the number of nodes in the system for both Dissent v1 and Dissent v2 is depicted in Figure 1. From these results, we can first observe that the throughput of nodes in Dissent v1 drops down to (almost) zero when the number of nodes is higher than 50. The reason is that for each anonymous communication, every node needs to send a message to all the other nodes in the system. This requires the sending of what is equivalent to N broadcast messages involving all the nodes, where N is the number of nodes in the system. Let us note the cost of Dissent v1 as $N * Bcast(N)$[2], which refers to the fact that for each anonymous communication, N broadcast messages are sent in a group of N nodes[3]. It becomes thus obvious why Dissent v1 exhibits such poor performance when the number of nodes in the system increases. Indeed, both the number of broadcast messages sent in the network, and the size of the broadcast groups depends on the system size.

We can also observe from Figure 1 that the throughput reached by nodes in Dissent v2 is higher than the throughput reached by nodes in Dissent v1, which was the key objective of this protocol. Nevertheless, the throughput of Dissent v2 decreases when the number of nodes in the system increases. The reason of this decrease is that, for each anonymous communication, Dissent v2 requires the sending of S broadcasts (where S is the number of trusted servers) in a group of size S servers, plus one broadcast in a group of size $\frac{N}{S}$. Hence, the cost of Dissent v2 is equal to $Bcast(\frac{N}{S}) + S * Bcast(S)$, which is also dependent on N.

We conclude from that study that existing freerider-resilient anonymous communication protocols are not scalable. This motivates the development of a new protocol that scales, i.e. a protocol that has a cost independent from the number of nodes in the system.

IV. The *RAC* protocol

In this section, we present *RAC*, a freerider-resilient, anonymous communication protocol that scales to large systems. Scalability stems from two key ideas: (1) the reduction of the number of messages broadcast in the system, and (2) the reduction of the size of the broadcast groups. We first explain how we achieve these two key ideas. We then provide a detailed description of the protocol.

A. Key idea #1: reducing the number of broadcasts

In order to be more efficient than Dissent v1 and Dissent v2, we decided to start the design of our protocol from the principle of the onion routing protocol. Indeed, this is the most efficient available protocol existing today. As described earlier, in onion routing, the sender of a message selects a subset of nodes to act as relays. Although the anonymity of onion routing, which is a function of the number of relays, is lower than the one of protocols such as DC-Net, it is strong enough to be widely used in practice. The problem with existing onion-based protocols is that freeriders have no interest in acting as relays; they will thus drop the messages they are supposed to relay whenever they can (i.e. when it does not endanger their anonymity). It is thus necessary to find a way to monitor the behavior of relays. The only node that knows all the relays on the path of an onion and that can identify all the layers of the onion is the sender of the message. The question is: *how to make the sender monitor the relays without disclosing its identity?* Indeed, if a relay knows that a node is using it as a relay, it can break the sender anonymity. Our solution is to have senders and relays broadcast the messages they send or relay. Broadcasting messages allows senders to receive the messages broadcast by relays and to check that relays forward

[1]This ideal network configuration allows us to measure the maximum throughput each protocol can reach.

[2]In the remaining of the paper, we adopt the following notation: we write "the protocol P has a cost of $x * Bcast(y)$" to refer to the fact that for each anonymous communication done using P, x broadcast messages are sent in a group of y nodes.

[3]For the sake of simplicity we assume that all broadcast messages have approximately the same size.

correctly, without disclosing their identity (every node receives the messages).

The broadcast exchanges must be reliable, despite the presence of freeriders and opponent nodes. In order to force nodes to forward messages, the broadcast protocol we propose relies on a structured network similar to the one used in the Fireflies group membership protocol [14]. Specifically, nodes are placed on several virtual rings using a hash function. On each ring, a node has a predecessor node and a successor node. The broadcasting protocol works as follows: each time a node receives a message from one of its predecessors, it forwards it to all its successors. A node thus expects to receive each given message from all its other predecessors. If a node does not receive a given message from one of its predecessors, it considers the latter as a freerider. This forces nodes to forward all the messages they receive. Note that a node also verifies that its predecessors are broadcasting at a constant rate, because this is required to ensure anonymity, as explained in the detailed protocol description.

Figure 2 illustrates the dissemination of a message in *RAC*. A node A that wants to send a message to a node D builds an onion (containing two relays B and C in that case). The node then broadcasts the onion using the multiple ring structure. Upon the reception of the onion, each node first forwards the onion to its successors. Each node then tries to decipher the onion. If a node manages to decipher the onion and it obtains a new onion, this means that the node is a relay and it will also broadcast the new onion (this is the case of node B and then of node C). If a *node* obtains a clear message, this means that it is the destination (this is the case of node D). If a node does not manage to decipher the onion, it does not do anything else than forwarding the onion to its successors.

The resulting protocol has a cost of: $L * R * Bcast(N)$, where $L << N$ is the number of relays employed in the onion path, and $R << N$ is the number of rings used in the broadcast protocol. In the experiments presented in Section V, we use $L = 5$ and $R = 7$. As we show, these values are enough to ensure a high level of anonymity: receiver anonymity and unlinkability are optimal, and an opponent only has a probability of $9.9 * 10^{-7}$ to break the sender anonymity.

With the protocol presented in this section, we have reached the first objective: making the number of messages that are broadcast in the system independent of N. In the next section, we explain how we reach our second objective: making the size of groups in which messages are broadcast independent of N.

B. Key idea #2: reducing the size of broadcast groups

In order to make the size of groups in which messages are broadcast independent of N, we propose to cluster nodes into groups of approximately the same size, say G[4]. The identifier of the group to which a node should belong is computed deterministically by the node when it first joins the system (e.g., using a hash of its public key modulo the number of groups). If two nodes that belong to the same group want to communicate they use the protocol we saw in the previous section, inside the group they belong to (of size G) instead of running it in the whole system. The cost is thus reduced to $L * R * Bcast(G)$. The remaining question is: *How do we enable the communication between two nodes that do not belong to the same group?* A straightforward solution would be to run the protocol presented in the previous section in a supergroup composed of the union of the two groups to which these nodes belong. The resulting cost would thus be $L*R*Bcast(2*G)$. We adopt in our protocol a more optimized solution described below.

The sender sends the anonymous message by running the protocol in his group, as if the destination was part of the same group. It nevertheless sets a marker in the innermost onion for the last relay, to inform him about the group Id of the destination node. This allows sending most broadcasts inside a group of size G. Upon receiving the innermost onion, the last relay needs to perform two actions. First, it needs to forward the message to the destination node, which belongs to another group. Second, it needs to inform the sender that it effectively forwarded the message. To perform these two actions at once, the last relay broadcasts the message in a super group constituted of the union of the two groups, i.e., its group and the group of the destination. This super group is what we call a *channel* in the remaining of the paper. As such, the overall cost of the protocol is equal to $(L-1)*R*Bcast(G)$ for the first part of the protocol, which executes inside the sender's group plus $R * Bcast(2G)$ for the broadcast that the last relay executes inside the channel. As $Bcast(2G) = 2 * Bcast(G)$, the overall cost of our protocol is equal to $(L + 1) * R * Bcast(G)$, which is lower than the original $L * R * Bcast(2 * G)$ because for the commonly chosen values of L, $L + 1 < 2 * L$.

C. Detailed description

In this section, we provide a detailed description of the protocol.

Joining the system. Each node in the system has an identifier (ID), a view containing the list of the nodes present in the system, and two private/public key pairs. The first pair of keys is linked to the node ID. We call these keys the ID keys. The second pair of keys is used to encrypt messages for their destination (using the key of the destination). This pair of keys cannot be linked to the node ID. We call these keys the pseudonym keys. The way nodes learn about pseudonym keys is application-dependent. For instance, in an anonymous publish-subscribe system, nodes would subscribe to a given topic using their public pseudonym key.

To join the system, a node n sends a JOIN request to a node x that is already part of the system. This JOIN request contains the ID public key of n, denoted by K, and ID, the identifier

[4]Note that using smaller groups (size G instead of N) gives an opponent that captures a message more information about the possible senders and receivers of this message. Specifically, the opponent knows that the sender and the receiver of the given message is one among the G nodes of the group, instead of one among the N nodes of the system. This is nevertheless not an issue if G is big enough. In our experiments, we use $G = 1000$.

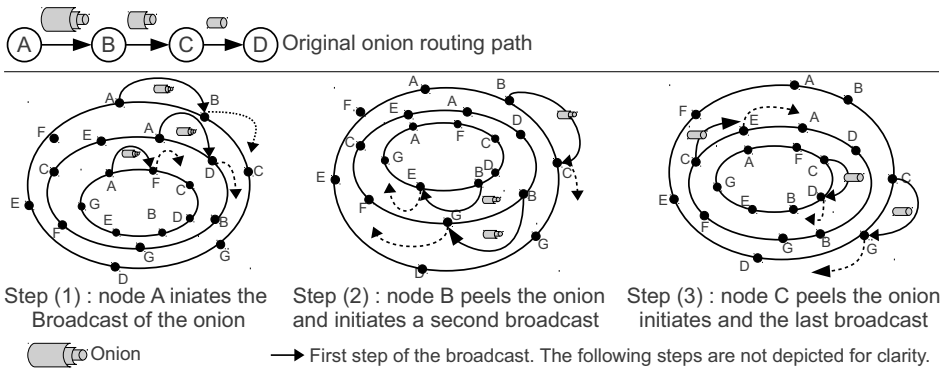

Fig. 2. Illustration of the *RAC* protocol.

of n, which determines the group that n will join. To compute this identifier, we use a solution inspired by the Herbivore system [4]. Specifically, let f and g be two one-way functions. The new-coming node has to generate random vectors until it finds a vector $y \neq K$ such that the least significant mk bits of $f(K)$ are equal to those of $f(y)$. The value $g(K, y)$ gives n the value of its ID. An opponent node may want to circumvent the system to join a particular group. The solution we use is robust to this attack, because it is difficult for a node to obtain the values of K and y that are necessary to join a given group, provided that the functions g and f are one-way functions.

Once x has received the JOIN request from n, it computes which group n should join (for example, the group containing the node with the nearest ID). Then, x anonymously broadcasts the JOIN request to this group. Upon receiving this request, all nodes of the group verify that the ID of n is correct. If the ID is not correct, the request is ignored; otherwise, the nodes add n to their view and compute the positions of n in the various rings used to broadcast messages. This is done as in the Fireflies protocol [14]: the position of a node on the i^{th} ring is determined by the hash of the couple (ID, i). The number of rings to create depends on the size of the system, as well as of the percentage of opponent nodes that is assumed in the system. More precisely, the successor set of each node (which is, for a given node, a set comprising the successor of this node on the various rings) should contain a majority of non-opponent nodes, and this majority should be large-enough to ensure reliable dissemination of broadcast messages[5]. Concretely, in a system comprising 1000 nodes and assuming that 10% of the nodes are opponent nodes, it is enough to use 7 rings to ensure that the successor sets will contain less than 3 opponent nodes with a probability $0,999$.

After a period T, which corresponds to the maximum time necessary for a message to reach all nodes in the group, x sends a READY message to n, signaling that all nodes have been informed of its arrival in the system. When n receives the READY message, it sends a message to its followers and predecessor to indicate them that they can use it as their new follower or predecessor. Other nodes than n's predecessors and followers need to wait a period equal to $2T$ from receiving n JOIN request before using n as a relay (to make sure that n completed the join procedure). Once n has joined a group, the nodes of this group broadcast the JOIN messages in the channels they belong to. This way, all nodes in the system are informed of n's arrival and n can find its position in each channel.

Managing groups. To guarantee a lower bound on anonymity and an upper bound on the cost of the protocol, groups must have a size bigger than s_{min}, and smaller than s_{max}, which are system parameters. When the size of a group becomes lower than s_{min}, nodes belonging to the group broadcast a message indicating that the group should be dissolved. They then rejoin the system in order to be assigned to another group. When nodes join the system and a group becomes larger than s_{max}, nodes of this group broadcast a message indicating that the group should be split in two. Nodes then compute which of the two new groups they should join. Specifically, nodes with the lower IDs go in the first group, and nodes with the higher IDs go in the second group. Finally, each node computes its position in the various rings of the group and connect to its new followers and predecessors.

Sending a message. To send a message, a node ciphers the message with the public pseudonym key of the destination node. It then randomly chooses L public ID keys of nodes of its group and ciphers the message in successive layers (called onions). Each layer contains a flag that allows a node to know whether it successfully deciphered it. If the message destination is in another group, the innermost layer also contains a marker indicating the channel to which it should be broadcast. Once the L layers have been created, the sender pads the message to reach a defined size, and initiates a broadcast by sending the message to its direct successors on the different rings of its group. Messages are padded because it makes it impossible for opponent nodes to use the size of network packets to track the path followed by a given message. Note that, similarly as in the onion routing protocol, to preserve the anonymity of the sender, nodes must send or forward messages at a constant rate, defined by the system.

[5]To ensure reliable dissemination in a system with N nodes, each node should have at least $log(N) + c$ non-opponent nodes in its successor set, where c is a constant [15].

If a node does not have messages to send or forward, it must create fake messages (called *noise* messages) and send them in the system, following the above-described procedure.

Receiving a message. Upon reception of a message, a node checks if it has already received the message. If that is not the case, the node forwards the message to its direct successors on the different rings of the group or channel in which the message is broadcast. The node then checks whether it can decipher the message using its private ID key. In case of success (the node is able to read the flag put by the sending node), this means that the node should act as a relay for this message. Consequently, it checks if there is a marker indicating that the message should be broadcast in a channel. It then pads the deciphered message and sends it to its direct successors on the different rings of the appropriate group or channel. Otherwise, the node checks whether the message is intended to it by trying to decipher it using its private pseudonym key. If that is the case, it delivers the message. Note that only innermost layers are broadcast in the channels, so as when they receive a message on a channel, nodes only have to check if they can decipher it with their private pseudonym key.

Checking the misbehavior of nodes. In order to discourage freeriding, nodes check that (1) the relays they use to send their own messages correctly forward messages, (2) the nodes that directly precede them in the different rings of channels and group correctly forward messages (i.e. once and only once), and (3) the nodes that directly precede them in the different rings of their group send messages at a constant rate. Every time a node discovers a misbehavior, it locally blacklists the corresponding node. Each node maintains several blacklists: a blacklist per channel for suspected predecessors, a blacklist for their group for suspected predecessors, and a blacklist for suspected relays. We explain later how these blacklists are used to evict nodes from the system.

The first check is performed as follows. When a node sends one of its own messages, it keeps a copy of the various layers of the message, together with the public ID keys they have been ciphered with. It then expects to receive the messages corresponding to the different layers before the expiration of a timer (recall that all messages are broadcast in the system). The first relay, if any, that does not correctly decipher and forward the message, is suspected and added by the sender to the relays blacklist and it is not used anymore by this node.

The second check is performed as follows. Provided that messages are broadcast, for each message, a node expects to receive a copy from each of its direct predecessors in the channel or group in which the message is broadcast. If a predecessor does not send a copy of a given message within a bounded time[6], of if it sends a given message twice[7], it will be suspected by its successors, who will add it to the appropriate predecessors blacklist.

The third check is performed as follows. Each node checks that it receives messages from its direct predecessors on the different rings of its group at a constant rate. Whenever a node detects a misbehaving predecessor, it adds it to the predecessors blacklist.

Evicting nodes. As described above, nodes maintain two kinds of blacklists: a blacklist for suspected relays, and several blacklists for suspected predecessors. Nodes disseminate these blacklists as follows. The relays blacklist is disseminated periodically to the node's group. Because it can disclose information about the identity of a message sender, this list has to be disseminated anonymously. To avoid an attack where malicious nodes send more than one blacklist at each round, we use the shuffle protocol of Dissent v1 which allows permuting a set of fixed-length messages and broadcasting the set to all members with cryptographically strong anonymity. The predecessors blacklists are disseminated as clear messages in the channels or group to which they correspond.

A node A removes a node B from its view as soon as it collects evidence that: (1) B belongs to the predecessors blacklist of $(t+1)$ of B's followers in one channel or group, with t the maximum number of opponent followers that node B can have (as defined by Fireflies [14]), or (2) B belongs to the relays blacklist of $(f+1)$ nodes of B's group, with f the maximum number of opponent nodes in a group. Further, if B is one of A's predecessors or followers, it replaces it with a new predecessor or successor deterministically computed from the view updated after the eviction of B.

When a node is evicted from a group or a channel, the nodes of its group must broadcast messages to all the channels it belonged to. This message informs the nodes of the channels that the node was evicted. Nodes that fail at sending this message are suspected by the nodes of their group. Nodes in the channels wait to received $(f+1)$ eviction notifications to take this eviction into account.

V. Proofs

In this section, we first prove that *RAC* ensures anonymity. Then we prove that *RAC* is freerider-resilient.

A. Anonymity proof

We prove that *RAC* ensures sender anonymity, receiver anonymity, and unlinkability. We use the following notations: the network size is denoted by N, the groups size is denoted by G, the fraction of opponent nodes is denoted by f, and the number of relays used to disseminate each message is denoted by L. We explained in section IV-B that dividing nodes into groups reduce nodes anonymity from one among N to one among G. We now show how this anonymity of one among G is protected. We first assume that opponent nodes are passive, which means that they follow the protocol and can only gather information by monitoring the network. We then consider the case of active opponents nodes that are willing to deviate from the protocol in order to break anonymity.

[6]Our implementation uses TCP, which ensures reliable delivery between pairs of nodes.

[7]The node could be performing a replay attack [16].

1) Passive opponent nodes:

a) Sender anonymity: As nodes in *RAC* broadcast messages at a constant rate, it is not possible to know if they are forwarding messages, sending messages or sending noise. Thus, an opponent node receiving a message can only discover the sender of this message when it colludes with all the relays of that message. This is quite hard to achieve. In fact, the probability for a correct node to build a path only containing opponent nodes is equal to $\prod_{i=0}^{L} \frac{X-i}{G-i}$, with X the number of opponent nodes in its group. Thus, an opponent that tries to break sender anonymity should control a number X of nodes as large as possible in the targeted node's group. But, as nodes are randomly spread among the groups, the probability that the opponent control X nodes in a given group is equal to $\prod_{i=0}^{X-1} \frac{fN-i}{N-i}$. As a result, the probability that the opponent break the sender anonymity of a given node is equal to $\max_X(\prod_{i=0}^{L} \frac{X-i}{G-i-1} * \prod_{i=0}^{X-1} \frac{fN-i}{N-i})$. With $N = 100.000$, $G = 1000$, $f = 5\%$, and $L = 5$ this probability is equal to $5.7 * 10^{-25}$, which is extremely low.

b) Receiver anonymity: When a node sends a message m to a destination node n_D, it ciphers m using n_D's public pseudonym key, and broadcasts it using a set of relays. Note first that, as explained in Section IV, the public pseudonym key of n_D cannot be linked to n_D. The only node that is able to decipher the message m is node n_D, using its private pseudonym key. Nodes cannot distinguish n_D from other nodes (i.e. they have no way to know that it was able to decipher the message). For an external observer, n_D behaves like every other nodes in the system: it forwards the message m once and only once to all its direct successors in the different rings. Consequently, no node is able to detect that n_D is the destination of message m. This is optimal: to break receiver anonymity the opponent must control all the nodes of the group but one.

c) Unlinkability: As we have seen before, *RAC* ensures optimal receiver anonymity in the groups. Consequently, it is impossible for a node to determine the destination of a given message m within a group. It is thus impossible for an external observer to know whether two nodes are communicating or not. At best, an opponent can know that two nodes are communicating using a channel. But it cannot know which nodes are communicating. Hence, the protocol ensures unlinkability.

2) Active opponent nodes:

a) Sender anonymity: An active opponent can try to break sender anonymity: (1) by trying to force nodes to build relay paths only containing opponent nodes, or (2) by trying to evict some nodes from the system. Evicting nodes can be used to reduce the number of non-malicious nodes in the system, or to render the system prone to intersection attacks [17] by comparing sent messages before and after the eviction of some nodes.

Case 1: an opponent node acting as relay can drop the messages it is supposed to broadcast. This forces the sending node to build new paths and, thus, increases the probability that the node build a path that is totally composed of opponent nodes. Nevertheless, this is not an easy task since, if an opponent node drops a message, it will be blacklisted by the sender that will not use it as a relay anymore. Consequently, provided there is a fraction f of opponent nodes, if they coordinate their actions, and if each time a new path is created, it contains an opponent node, then they will be able to force at most fN new paths to be created. This makes the probability of building a complete path of opponent nodes very low. For instance, in a system with $N = 100.000$, $G = 1000$, $f = 5\%$ and $L = 5$, the probability that opponent nodes manage to force one single node to build a path only composed of opponent nodes is at most $2.8 * 10^{-23}$. This is extremely low.

Case 2: an opponent node can try to break sender anonymity by evicting nodes from the system (e.g. to allow intersection attacks [17] to be run). In *RAC*, a node n is evicted if: (1) $fG + 1$ nodes notify their group that the node n misbehaves, or (2) a majority of the direct successors of node n on the different rings notify their group or channel that node n misbehaves. Let us consider the first case. There are fG opponent nodes in the network. Consequently, opponent nodes alone cannot force the eviction of node n by sending a wrong notification to the group. They need to behave in such a way that at least one non-opponent node will also send a notification regarding node n. The only thing they can do is to try to stop the dissemination of some messages, so that node n will not be reached and will possibly be blacklisted (e.g. if n was a relay for that message). Nevertheless, as shown in [14], it is possible to increase the reliability of the broadcast by increasing the number of rings. It is thus possible to use a number of rings guaranteeing that opponent nodes will have an arbitrarily low probability to succeed in evicting node n by making a non-opponent node send a notification against it. Let us now consider the second case. As explained in Section IV, the higher the number of rings, the lower the probability that node n will have a majority of opponent nodes in its set of direct successors. Consequently, it is possible to use a number of rings guaranteeing that opponent nodes will have an arbitrarily low probability to succeed in evicting node n due to a majority of opponent nodes in its set of direct successors. For exemple with $f = 5\%$, a number of rings equal to 7 guarantees that each node has a probability lower than $6.0 * 10^{-6}$ to have a majority of opponent nodes in its set of direct successors.

b) Receiver anonymity: Opponent nodes can try to break receiver anonymity using the same attacks as the ones described in the proof of the "sender anonymity" property. As we have seen, these attacks do not succeed. Consequently, receiver anonymity is not impacted by the presence of active opponent nodes.

c) Unlinkability: The proof is similar to the one made for passive opponent nodes.

B. Freerider-resiliency proof

In this section, we prove that *RAC* tolerates freeriders, i.e. that freeriders do not have any interest in deviating from the protocol. To prove this, we prove that *RAC* provides a *Nash equilibrium* [18]. In a Nash equilibrium, no node has an incentive to unilaterally deviate from the equilibrium, assuming every other node follows the protocol. Note that the proof presented in this paper makes similar assumptions as those presented in previous works related to freeriders, e.g., [19], [20], [21].

1) Assumptions on freeriders: To reason about freeriders, it is necessary to formalize their behavior. In the case of *RAC*, the benefit of a node n depends of the following parameters:

- (A) reducing the risks of compromising its anonymity.
- (T) succeeding in sending its own messages.
- (R) receiving messages that are intended to it.
- (F) forwarding as few messages as possible.
- (C) ciphering as few messages as possible.
- (D) deciphering as few messages as possible.

We can define the overall benefit of a node n as $B = \alpha A + \beta T + \gamma R + \delta F + \omega C + \phi D$, where $\alpha \approx \beta \approx \gamma \gg \delta \approx \omega \approx \phi$. Intuitively, this means that nodes do not want to trade-off their anonymity and the reliable transmission of their own messages against a lower bandwidth and CPU consumption. Moreover, freeriders are assumed not to collude; freeriders expect opponent nodes to try to decrease their benefit as much as possible; freeriders expect other nodes to follow the protocol.

2) Nash equilibrium proof:

Theorem 1: The *RAC* protocol provides a Nash equilibrium.

We prove the above theorem following the approach adopted by related work [19], [20]: we decompose the theorem into a set of lemmas representing the protocol steps, and we explain why it is in the best interest of a freerider to follow the protocol at each given step. The proof of Theorem 1 directly follows from the proofs of the different lemmas.

Lemma 1: A freerider always sends messages to all its direct successors in the different rings of a channel or group.

Proof: A freerider knows that up to half of its direct successors but one can be opponent nodes and can thus send wrong notifications to the group or channel. Consequently, a freerider knows that it risks eviction if at least one correct successor sends a notification to the group or channel. Consequently, a freerider sends messages to all its direct successors in the different rings. ■

Lemma 2: A freerider always correctly forwards the messages it acts as a relay for.

Proof: A freerider n knows that the sender of a message will notify the group if it does not receive the message that node n was supposed to broadcast as a relay. Provided that there are fG opponent nodes in the group that can send wrong notifications to the group, node n knows that it is enough to be suspected by one non-opponent node to be evicted from the system. Consequently, node n correctly forwards the messages it acts as a relay for. ■

Lemma 3: A freerider always checks that its predecessors are sending every message once and only once.

Proof: A direct predecessor that does not send a given message can be an opponent node trying to run an *(N-1)-attack* [22] on the node. As a result, freeriders always checks if all direct predecessors send them all messages they are supposed to. A direct predecessor that sends a given message twice can be an opponent node trying to run a *replay attack* [16] on the node. As a result, freeriders always checks that their direct predecessors do not send the same message twice. ■

Lemma 4: A freerider sends the list of nodes it suspects during the periodic anonymous blacklist broadcasting.

Proof: As shown in [5], the anonymous blacklist broadcasting protocol we rely on is accountable. Consequently, a freerider participates in it to avoid eviction by the network. Moreover, messages sent in this protocol have a fixed-size. Consequently, a freerider does not gain bandwidth or CPU by sending a message containing wrong information. A freerider thus sends the exact list of nodes that it suspects. ■

Lemma 5: A freerider broadcasts the join requests it receives from nodes that want to enter in the system.

Proof: A freerider has interest in helping nodes to enter the system because that increases its anonymity. In fact, the more nodes in the system, the more difficult for an opponent to control a significant of the network. Moreover, helping nodes to join the system ensures that opponent nodes are not the only ones to control who can join the system. If only opponent nodes control who can join the system, they may only accept opponent nodes, and this way, increase their chances of breaking anonymity. Finally, freeriders have interest in helping nodes they want to communicate with to join the system. ■

Lemma 6: A freerider always sends new messages (possibly noise) at the rate required by the protocol.

Proof: A freerider n knows that if it omits to send new messages (possibly noise) at the rate defined by the protocol to some of its direct successors, the latter will accuse it. Moreover, node n knows that up to half of its direct successors but one can be opponent nodes and can thus produce wrong accusations. Consequently, node n knows that it risks being evicted if at least one correct successor sent a notification to the group. Consequently, node n sends messages at the required rate. ■

Lemma 7: A freerider always checks if its predecessors send new messages at the rate defined by the protocol.

Proof: A freerider notifies the group or channel if one of its direct predecessors sends a message at a higher rate than that defined by the protocol. Indeed, such a behavior increases its bandwidth and CPU consumption. Moreover, a freerider notifies the group if one of its direct predecessors sends messages at a lower rate than required as this can be the sign of an opponent running an attack. ■

VI. Evaluation

In this section, we evaluate both the performance and the anonymity of *RAC*. We evaluate the former using simulations and the latter using the formulas presented in Section V. We use simulations in order to be able to evaluate the performance of our protocol in configurations comprising up to 100.000 nodes. The objective of this evaluation is to answer the following two questions:

- What is the throughput achieved by *RAC*, Dissent v1, and Dissent v2?
- What are the anonymity guarantees ensured by *RAC*, Dissent v1, and Dissent v2?

We start by a presentation of the simulation settings and the configuration parameters used for the various protocols. We then reply to the two questions mentioned above.

A. Simulation settings

We performed simulations using Omnet++ [23], a discrete event simulator written in C++. Using Omnet++, we simulate a network of nodes interconnected by a router. Nodes are connected to the router using 1Gb/s links. We use this ideal network configuration as it allows evaluating the maximum throughput that each protocol can achieve. We plan to evaluate the complexity of *RAC* and its performance in a real setting as part of our future work.

B. Protocol configuration parameters

In all the evaluations, we consider two configurations for *RAC*: a configuration without group (i.e. all nodes actually belong to a single group), and a configuration with groups comprising 1000 nodes each. We refer to these two configurations as *RAC*-NoGroup and *RAC*-1000, respectively. In both configurations, we used seven rings for the broadcast protocol, i.e., $R = 7$ and five relays for the onion paths, i.e., $L = 5$.

For each system size, we configure Dissent v2 with the optimal number of trusted servers (as explained in Section III).

C. Throughput

In order to assess the throughput of the various protocols, we run the following experiment. We consider a system comprising N nodes. Each node randomly selects a destination node and sends anonymous messages to this node at the maximum throughput it can sustain. Messages have a fixed size of 10kB. We measure, for each protocol, the average throughput at which the N nodes receive anonymous messages. Figure 3 shows the throughput measured for *RAC*-NoGroup, *RAC*-1000, Dissent v1, and Dissent v2 as a function of N. Note that in this situation, with an onion path length of 5, the throughput provided by onion routing is 200Mb/s.

We first observe that both *RAC* configurations achieve a better throughput than the two versions of Dissent when there are more than 1000 nodes in the system. For instance, when the system contains 100.000 nodes, the throughput of *RAC*-NoGroup (resp. *RAC*-1000) is 15 times (resp. 1300 times) higher than that of Dissent v2. With this system size, Dissent v1 is so slow that we were not even able to observe a single message delivery.

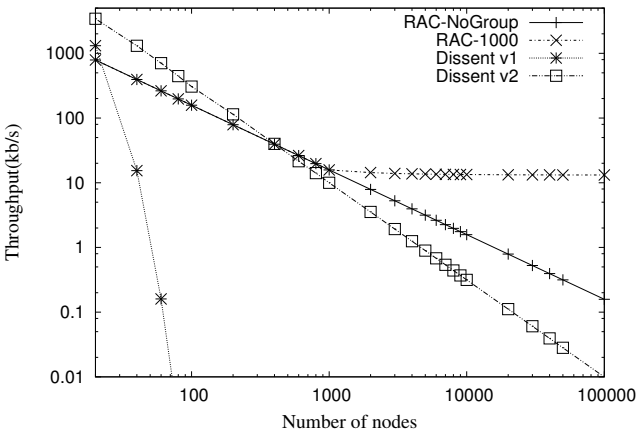

Fig. 3. Throughput as a function of the number of nodes in the system for Dissent v1, Dissent v2, *RAC*-NoGroup and *RAC*-1000.

We also observe that *RAC*-1000 scales: in systems comprising more than 1000 nodes, its throughput is not impacted by the size of the system. This was expected. Indeed, adding nodes in the system does not increase the number of broadcasts required for each message that is anonymously sent, nor the size of the broadcast groups that are used.

Finally, we observe that when N is smaller than 1000, both *RAC* configurations achieve the same throughput. This comes from the fact that when N is smaller than 1000, *RAC*-1000 only uses one group, and does thus execute the exact same protocol as *RAC*-NoGroup.

To summarize, unlike other protocols, *RAC*-1000 scales: performance do not decrease when increasing the system size.

D. Anonymity guarantees

We compare in this section the anonymity guaranteed by the two configurations of *RAC* against Dissent v1, Dissent v2 and the onion routing protocol in the case of a passive opponent. Results are depicted in Table I. The numbers depicted in this table result from the instantiation of the formulas characterizing the anonymity guarantees of each protocol.

The first line of the table represents the size of the set to which the sender or the receiver of a given message belong. The higher the size, the better. This value should not be too small as it can disclose a lot of information regarding the identities of the sender and receiver of a message. For instance, if this value is equal to 10, this means that the probability that a given node be the sender or the receiver of a given message is $\frac{1}{10}$. From the table, we can observe that this value is the highest possible for all protocols but *RAC*-1000, for which it is equal to 1000, i.e., the size of the groups. We believe that 1000 is a big enough value in most contexts. This value can be increased if required by *RAC* users (using the s_{min} parameter presented in the detailed protocol description).

The table is then structured in three subparts, corresponding to the proportion of nodes controlled by the opponent (P in the table). We consider three values of P: 10%, 50%, and

System with 100.000 nodes			Dissent v1	Dissent v2	Onion	*RAC*-NoGroup	*RAC*-1000
Anonymity, the sender/receiver is one among			100.000	100.000	100.000	100.000	1000
Probability to break a given node anonymity of type T when controlling P% of the nodes	% of opponent nodes (P)	Anonymity type (T)					
	90%	Sender	0	0	0.53	0.53	$7.1 * 10^{-11}$
		Receiver	0	0	0.53	0	$1.1 * 10^{-46}$
		Unlinkability	0	0	0.53	0	$1.1 * 10^{-46}$
	50%	Sender	0	0	$1.5 * 10^{-2}$	$1.5 * 10^{-2}$	$1.8 * 10^{-16}$
		Receiver	0	0	$1.5 * 10^{-2}$	0	$1.2 * 10^{-303}$
		Unlinkability	0	0	$1.5 * 10^{-2}$	0	$1.2 * 10^{-303}$
	10%	Sender	0	0	$9.9 * 10^{-7}$	$9.9 * 10^{-7}$	$7.3 * 10^{-22}$
		Receiver	0	0	$9.9 * 10^{-7}$	0	$5.8 * 10^{-1020}$
		Unlinkbility	0	0	$9.9 * 10^{-7}$	0	$5.8 * 10^{-1020}$

TABLE I
ANONYMITY GUARANTEES OF THE VARIOUS PROTOCOLS IN A SYSTEM OF 100.000 NODES.

90%. For each value of P, we compute the probability that the opponent be able to disclose the identity of the sender, the receiver or to link the sender and the receiver of a given message. Results show that these probabilities are equal to zero for both Dissent v1 and Dissent v2. Indeed, in these protocols, the opponent must control all the nodes in the system (or all the trusted servers in Dissent v2) to break anonymity. We also observe that in both configurations of *RAC*, these probabilities, although not (always) null, are extremely low, making *RAC* an extremely robust protocol.

We also observe that, counter-intuitively, the probability that an opponent break anonymity in *RAC*-1000 is lower than in *RAC*-NoGroup. This is due to the fact that in *RAC*-1000, a node cannot choose the group to which it belongs. Consequently, an opponent needs to control almost all the nodes in the system for having enough nodes in the same group in order to break anonymity within that group.

Finally, we observe that *RAC*-1000 provides stronger anonymity guarantees than onion routing in all cases. This is due to two reasons. First, *RAC*-1000 has a better sender anonymity than onion routing due to the fact that it uses groups (see explanation above for *RAC*-NoGroup vs *RAC*-1000). Second, to break the receiver anonymity in onion routing, the opponent must only control the relays of the onion path. Instead, in our protocol, the opponent must control all the nodes of the destination group, which is less likely to happen. Better unlinkability follows from better receiver anonymity.

VII. CONCLUSION

Two protocols for anonymous communication in the presence of freeriders have been recently proposed. Unfortunately, they do not scale: their performance decrease when increasing the number of nodes in the system. In this paper, we present *RAC*, a freerider-resilient anonymous communication protocol that scales, while providing strong anonymity guarantees.

ACKNOWLEDGMENTS

This work was partially funded by European FP7 PLAY project and the French SocEDA project.

REFERENCES

[1] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, 1988.
[2] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Commun. ACM*, vol. 42, no. 2, 1999.
[3] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, 1998.
[4] S. Goel *et al.*, "Herbivore: A scalable and efficient protocol for anonymous communication. tech rep." Cornell University, Tech. Rep., 2003.
[5] H. Corrigan-Gibbs and B. Ford, "Dissent: accountable anonymous group messaging," in *Proceedings of CCS*, 2010.
[6] D. I. Wolinsky, H. Corrigan-Gibbs, and B. Ford, "Dissent in numbers: Making strong anonymity scale," in *Proceedings of OSDI*, 2012.
[7] A. Pfitzmann and M. Hansen, "Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology," 2008.
[8] M. Waidner and B. Pfitzmann, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability," in *Proceedings of EUROCRYPT*, 1989.
[9] P. Golle and A. Juels, "Dining cryptographers revisited," in *Proceedings of EUROCRYPT*, 2004.
[10] M. Edman and B. Yener, "On anonymity in an electronic society: A survey of anonymous communication systems," *ACM Comput. Surv.*, vol. 42, no. 1, 2009.
[11] R. Dingledine *et al.*, "Tor: the second-generation onion router," in *Proceedings of USENIX Security Symposium*, 2004.
[12] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron, "Cashmere: resilient anonymous routing," in *Proceedings of NSDI*, 2005.
[13] M. J. Freedman and R. Morris, "Tarzan: a peer-to-peer anonymizing network layer," in *Proceedings of the CCS*, 2002.
[14] H. Johansen *et al.*, "Fireflies: scalable support for intrusion-tolerant network overlays," in *Proceedings of EuroSys*, 2006.
[15] A. M. Kermarrec, L. Massoulie, and A. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *IEEE TPDS*, vol. 14, no. 3, 2003.
[16] R. Pries, W. Yu, X. Fu, and W. Zhao, "A new replay attack against anonymous communication networks," in *Proceedings of ICC*, 2008.
[17] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies*. Springer, 2001, vol. 2009.
[18] J. Nash, "Non-Cooperative Games," *Annals of Mathematics*, vol. 54, no. 2, 1951.
[19] A. Aiyer *et al.*, "Bar fault tolerance for cooperative services," in *Proceedings of SOSP*, 2005.
[20] H. Li *et al.*, "Bar gossip," in *Proceedings of OSDI*, 2006.
[21] S. Ben Mokhtar *et al.*, "Firespam: Spam resilient gossiping in the bar model," in *Proceedings of SRDS*, 2010.
[22] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Information Hiding*, ser. Lecture Notes in Computer Science, F. Petitcolas, Ed. Springer, 2003, vol. 2578.
[23] "The omnet++ simulation environment," 2013, http://www.omnetpp.org/.