Anonymity and Censorship Resistance in Unstructured Overlay Networks ⋆

Michael Backes[4] and Marek Hamerlik[1] and Alessandro Linari[2] and Matteo Maffei[1] and Christos Tryfonopoulos[3] and Gerhard Weikum[3]

[1] Saarland University {maffei,mhamerli}@cs.uni-sb.de
[2] Oxford Brookes University & Nominet UK alessandro@nominet.org.uk
[3] Max Planck Institut für Informatik {trifon,weikum}@mpi-inf.mpg.de
[4] Saarland University & MPI-SWS backes@cs.uni-sb.de

Abstract. This paper presents Clouds, a peer-to-peer protocol that guarantees both anonymity and censorship resistance in semantic overlay networks. The design of such a protocol needs to meet a number of challenging goals: enabling the exchange of encrypted messages without assuming previously shared secrets, avoiding centralised infrastructures, like trusted servers or gateways, and guaranteeing efficiency without establishing direct connections between peers. Anonymity is achieved by cloaking the identity of protocol participants behind groups of semantically close peers. Censorship resistance is guaranteed by a cryptographic protocol securing the anonymous communication between the querying peer and the resource provider. Although we instantiate our technique on semantic overlay networks to exploit their retrieval capabilities, our framework is general and can be applied to any unstructured overlay network. Experimental results demonstrate the security properties of Clouds under different attacks and show the message overhead and retrieval effectiveness of the protocol.

1 Introduction

Over the last years unstructured overlays have evolved as a natural decentralised way to share data and services among a network of loosely connected peers. The popularity of systems like Gnutella and Freenet [5], has propelled research in this field, while lately the proliferation of social networking has added another interesting dimension to the problem of searching for content in such networks. In unstructured overlays, peers typically connect to a small set of other peers, and queries are propagated along connections in the overlay network, using some query forwarding strategy that aims at finding peers with resources matching the issued query.

Semantic Overlay Networks (SONs) are an instance of unstructured networks that has lately been given considerable attention [6,18,2,20]. In a SON, peers that are semantically, thematically, or socially close (i.e., peers sharing similar interests or resources)

⋆ Work partially supported by the initiative for excellence and by Emmy Noether program of the German federal government and by Miur project SOFT: "*Security Oriented Formal Techniques*".

are organised into groups to exploit similarities at query time. This flexible organisation improves query performance while maintaining high peer autonomy, and has proved a useful technology not only for distributed Information Retrieval (IR) applications, but also as a natural distributed alternative to Web 2.0 application domains such as decentralised social networking in the spirit of Flickr or del.icio.us. Query processing is achieved by identifying which region in the network is better suited to answer the query and routing the query towards a peer in that region. This peer is then responsible for forwarding the query to his neighbours in the region.

In such an information exchange, not all information providers are willing to reveal their true identity: for instance, publishers may want to present their opinions anonymously to avoid associations with their race, ethnic background or other sensitive characteristics. Furthermore, people seeking for sensitive information may want to remain anonymous so as to avoid being stigmatised or even to avoid physical, financial or social detriment by suppressors. The freedom of information exchange is another important issue that got increasing attention in the last years. Some organisations, such as governments or private companies, may regard a discussion topic or a report as inconvenient or even harmful. They may thus try to censor the exchange of undesired information by either suppressing resource providers or, if these are protected by anonymity, by taking control of strategic regions of the network, such as gateways and proxies, in order to filter the communication.

Despite the importance of SONs as a building block of data management and social networking applications, security issues have not been investigated in this setting. To the best of our knowledge, there exist no studies that try to solve the anonymity and censorship resistance problems arising in SONs. This, together with the observation that SONs are actually vulnerable to a number of different attacks, varying from surrounding to man-in-the-middle attacks, shows the importance of enforcing security in such an environment.

In this work we present *Clouds*, a P2P search infrastructure for providing *anonymous* and *censorship resistant* search functionality in a SON. We exploit the inherently high connectivity among similar peers for guaranteeing anonymity by relying on a self-organisation of peers into groups that we call *clouds*. Message routing is modified to take place among clouds instead of peers, thus hiding the identity of both the resource provider and the querying peer, while cloud size is a tunable parameter that affects anonymity and efficiency. Censorship resistance at communication level is achieved by a cryptographic protocol that guarantees the secrecy of the resource, thus avoiding censorship based on the inspection of the messages circulating in the network. This protocol achieves a number of challenging goals: enabling the exchange of encrypted messages without assuming previously shared secrets, avoiding centralised infrastructures, such as trusted servers or gateways, and guaranteeing efficiency without establishing direct connections between peers. The contribution of this paper is two-fold:

- We present the first system to guarantee anonymity *and* censorship resistance in SONs. Although we instantiate our technique on SONs to leverage their retrieval capabilities and to support a rich data model and query language, our framework can be applied to any type of unstructured overlay network.

– We demonstrate the effectiveness of our architecture by analysing the anonymity and censorship-resistance properties provided by Clouds under different attack scenarios, namely surrounding, intersection, man-in-the-middle, and blocking attacks.

The rest of the paper is organised as follows. Section 2 discusses our data model and query language, and outlines the SON paradigm. Section 3 presents Clouds and its associated protocols. Section 4 introduces the attack scenarios addressed in this paper. Our experimental evaluation is given in Section 5, and related work is discussed in Section 6. Finally, Section 7 concludes the paper and gives directions for future research.

2 Background information

This section outlines our data model and query language, and describes the construction and properties of SONs. For more detail, we refer the interested reader to [6,18,2,20].

2.1 Data model and query language

We utilise the *Vector Space Model (VSM)* to represent documents, queries, peer and cloud descriptions. In our setting a *resource* is any piece of information that can be described by a set of *keywords*, such as a text document which is characterized by its terms, an image, which is often associated to a set of tags, or an mp3 file. We associate a *weight* to each keyword so as to represent the importance of the keyword as a description for the given resource. A query is a set of keywords, for which the weights are explicitly assigned by the user or implicitly by the system (e.g., through relevance feedback techniques [31]). A resource is characterized by a *resource description* r_i, that is a vector containing keywords and weights for these keywords. Similarly, the *profile* of a peer P, *profile*(P), is computed using the descriptions $r_1, \ldots, r_m$ of the resources stored at this peer.

A standard technique to decide which resource description r best matches a given query q, is to utilise a *similarity function* $sim(q,r)$, which assigns a numerical score to each pair (q,r). The scores corresponding to different resource descriptions are then compared to derive a relevance ranking with respect to query q. A common similarity measure in IR is the cosine of the angle formed by the vector representations of q and r. Notice that, in practice, any function that models the similarity between a resource and a query can be used. For example, in the case of a social network the similarity function could also contain a social component, that also considers the strength of relations among users.

2.2 Semantic overlay networks

In a SON each peer P performs a variable number of *random meetings* with other peers in the network, during which they exchange their profiles and compute their similarity. Based on the similarity function $sim()$, a peer P establishes two types of links:

– *Short-distance links* towards the k most similar peers in the network discovered through the random meeting process. The number of short-distance links k is usually small (e.g., $O(\log N)$, where N is the number of peers in the network).

– *Long-distance links* towards k' (typically $k' < k$) peers chosen at random from the rest of the network.

To maintain short-distance links up-to-date and ensure the clustering property of the SON, a periodic *rewiring procedure* [6,18,2,20] is executed by all peers, aiming at discovering new more similar peers, or refreshing links that have become outdated due to network dynamics. Long-distance links, usually updated using random walks, are necessary to avoid creating tightly clustered groups of peers that are disconnected from the rest of the network. Query answering in SONs benefits from the fact that peers containing related information are directly linked or at a short-hop distance from each other. Thus, the task of finding a peer that can answer a query q reduces to locating the appropriate cluster of peers. Once a peer in the appropriate cluster is reached (i.e., if $sim(profile(P), q)) \geq \beta$, where β is called *broadcast threshold*), the query goes through a limited broadcast using short-distance links aiming at reaching all neighbours of P. Due to the SON properties, these peers are able to answer q with high probability.

3 The Clouds protocols

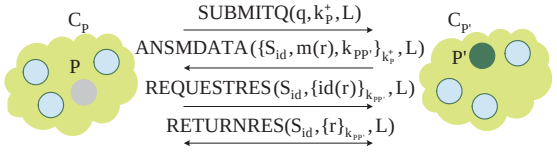
This section describes in detail the protocols that regulate the interactions between peers and allow them to anonymously share and retrieve resources available in the network.

3.1 Protocol overview

The key principle behind the anonymity mechanism of Clouds is to cloak both the querying peer and the resource provider behind a group of neighbouring peers, called cloud. Peers generate clouds at random, without necessarily using them, to minimise the correlation between the events of joining and using a cloud. Additionally, they non-deterministically decide to participate or not in clouds created by other peers. Clouds are created using the short-distance links of peers and are thus populated by peers in the neighbourhood of the cloud initiator. Communication takes place between clouds, and all peers in a cloud share the same probability of being involved in any communication which has this cloud as the start- or end-point (also known as *k-anonymity* [28]). To avoid correlation of roles in the protocol with specific actions, which would compromise anonymity, the protocol is designed so that the observable behaviour is the same for all peers, regardless of them being initiators of forwarders of a message.

The proposed cryptographic protocol aims at addressing the problem of censorship at the communication level, where a malicious party aims at filtering out any communication that contains unwanted content (either a query or a resource). The secrecy of the resource is protected by cryptography, making it hard for the attacker to censor the communication based on an inspection of the message content. The design of such a protocol is conceptually challenging since we do not assume previously shared secrets or centralised infrastructures.

The protocol is composed of four steps summarised in Figure 1. A querying peer chooses a cloud it participates in to issue a query. The query follows a random walk in the cloud to obscure the message initiator, leaves the cloud from multiple peers to



Require: $\mathcal{C}, p, i$
Ensure: *yes*/*no* (depending whether P joined or not $\mathcal{C}$)

1: $x \leftarrow$ random value in $[0,1]$
2: **if** $x > p$ **return** "no"
3: **if** $(\neg isParticipant(P,\mathcal{C}))$ **then**
4: **for all** P' neighbour of P **do**
5: $p' \leftarrow update(p)$
6: $status \leftarrow P'.joinCloud(\mathcal{C}, p', i+1)$
7: **if** *status* **then** $addParticipant(\mathcal{C}, P')$
8: **return** "yes"

Fig. 1. Communication protocol and algorithm *joinCloud*($\mathcal{C}$, p, i).

ensure higher resistance to censorship and is routed towards a region in the network that possibly contains matching resources. A *footprint list* is used to collect the list of traversed clouds and facilitates the routing of the subsequent messages. A responder to the query encrypts the answer with a public key received with the query message and routes it towards the cloud of the querying peer, as specified in the footprint list. All subsequent messages between the querying peer and the responder will have a cloud as a destination and, when this cloud is reached, the message will be broadcasted to reach the intended recipient. Finally, notice that the query message does not contain any session identifier which would connect the query to the subsequent messages. In the last two protocol steps, however, a session identifier is used to avoid costly decryption checks, since the message content is encrypted.

3.2 Cloud creation

The design of a cloud creation algorithm should satisfy some fundamental properties, such as randomness, tunability and locality, in order to reveal as little information as possible to potential attackers while maintaining the useful clustering properties of the underlying SON. According to our cloud creation algorithm, when a peer P generates a new cloud $\mathcal{C}$, it selects the participants among its neighbours utilising short-distance links. This guarantees that clouds are populated by semantically close peers (cloud *locality*). As shown in Section 4.1, this property is crucial to guarantee that cloud intersections have high cardinality, thus preventing *intersection attacks* that aim at breaking anonymity. Peers that join a cloud in turn select other neighbours, and the protocol is executed in a recursive way with decreasing probability to join $\mathcal{C}$.

The *joinCloud*() algorithm shown in Figure 1 shows this procedure from P's point of view, assuming that P has received a $joinCloud(\mathcal{C}, p, i)$ message that may have been generated either by itself or by any other peer. This message specifies the cloud $\mathcal{C}$, the probability p to join it, and the step i. With probability $1-p$, P replies negatively (line 2) to the request, otherwise it accepts to join $\mathcal{C}$ (line 8). In this case, if P is not already in $\mathcal{C}$, it triggers a recursive procedure in its neighbours (line 3) by sending a $joinCloud(\mathcal{C}, p', i+1)$ message to each of them (lines 4-7) with join probability p' (line 5). Finally, the peers joining $\mathcal{C}$ are marked by P as neighbours in $\mathcal{C}$ (lines 6-7).

The $update(p)$ function used to obtain p' is the means to control cloud population and to offer tunability between anonymity and efficiency. For simplicity, in the experi-

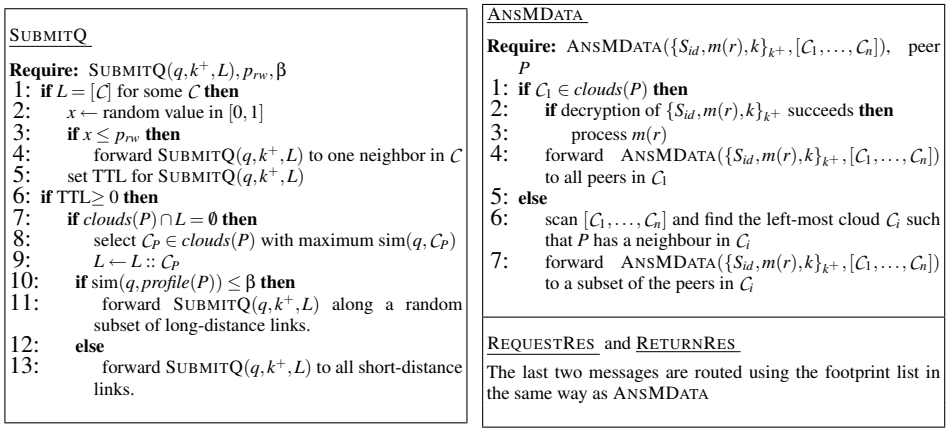


Fig. 2. Behavior of peer P in the different protocol steps.

ments we consider the same function throughout the network, but in practice each peer may use its own *update*() function. Notice that peers only know the neighbours certainly belonging to their clouds (i.e., the neighbours that have sent or have positively answered to a *joinCloud* request). No information can be derived from a negative answer, since a peer already belonging to the cloud answers negatively with probability $1-p$. This helps to avoid statistical attacks on cloud membership.

3.3 Query routing

In this section, we present the routing algorithm used to route a query q from a peer P to a resource provider P'. Figure 2 gives the pseudocode for the query routing procedure followed by any peer P. Notice that the protocol is the same regardless of P being the initiator or the forwarder of the message, in order to avoid breaches in anonymity.

When P wants to issue a query q, it constructs a message $msg = \text{SUBMITQ}(q,k_P^+,L=[\mathcal{C}_P])$, where k_P^+ is a public key generated by P especially for this session, and L is the *footprint list* that will be used to collect the list of clouds *msg* will traverse during the routing. P initialises this list with one of its clouds. The collection of clouds in the footprint list is performed as follows. A peer P_i that receives *msg* checks whether one of the clouds it participates in is already listed in L. If not, it chooses a cloud $\mathcal{C}_{P_i}$ and appends it to L (i.e., $L \leftarrow L :: \mathcal{C}_{P_i}$). This information will be exploited by the successive phases of the communication protocol to optimise routing between P and the resource provider.

The routing algorithm for a SUBMITQ message is reported in Figure 2. The algorithm consists of two steps: an *intra-cloud* routing, during which the message *msg* performs a random walk in $\mathcal{C}$, and an *inter-cloud* routing, during which *msg* is delivered to a peer P' not participating in $\mathcal{C}$. Each peer receiving (or creating) *msg*, forwards it to a random peer participating in $\mathcal{C}$ with probability p_{rw} and also to a peer that subsequently enters the inter-cloud phase. The intra-cloud routing phase is necessary to avoid revealing the identity of the query initiator to a malicious long-distance neighbour that exploits the existence of a single cloud $\mathcal{C}$ in L. By adding the random walk phase within

C, an attacker cannot know whether P is the initiator of *msg* or simply a forwarder entering into the inter-cloud phase.

Inter-cloud routing is based on the fireworks query routing algorithm [17]. A peer P receiving message *msg* computes the similarity $sim(q, profile(P))$ between q and its profile. If $sim(q, profile(P)) \leq \beta$, where β is the broadcast threshold, this means that neither P, nor P's neighbours are suitable to answer q. Thus, *msg* is forwarded to a (small) subset of P's long-distance links. If $sim(q, profile(P)) > \beta$, the query has reached a neighbourhood of peers that are likely to have relevant resources and *msg* goes through a limited broadcast using short-distance links.

In order to limit the network traffic, each message entering the inter-cloud phase is associated to a time-to-live (TTL), which is updated at every hop. Message forwarding is stopped when TTL reaches zero. Finally, all peers maintain a message history and use it to discard already processed messages.

3.4 Answer collection

When a peer P' receives the message $\text{SUBMITQ}(q, k_P^+, L)$, it searches its local collection and retrieves the list $R = \{r_1, r_2, \ldots, r_n\}$ of resources matching q. Then P' constructs the reply message $msg = \text{ANSMDATA}(\{S_{id}, M, k_{PP'}\}_{k_P^+}, L)$, i.e., it encrypts with k_P^+ the list $M = \{m(r_1), m(r_2), \ldots, m(r_n)\}$ of metadata for the local result list R, a unique session identifier S_{id}, and a symmetric key $k_{PP'}$. S_{id} will be used by P' in the subsequent protocol steps to identify *msg* as an open transaction, while $k_{PP'}$ will be used to encrypt the remaining messages between P and P' and to avoid computationally expensive public key cryptography.

The routing algorithm for *msg* is based on the footprint list L, as described in Figure 2. A peer receiving *msg* and not belonging to the destination cloud C_P, forwards *msg* to a (small) subset of its neighbours that participate in the left-most cloud of L.

Finally, when a peer P participating in the destination cloud C_P receives *msg*, it broadcasts it to all its neighbours in C_P. Subsequently, it tries to decrypt the message using its session private key k_P^-, to discover if it is the intended recipient of *msg*. During the broadcast in C_P, the message history of each peer is used to discard already processed messages. Note that even the intended recipient P forwards *msg* in C_P, in order to avoid detection by malicious neighbours. After a predefined timeout or a large enough answer set, P chooses the resources to be retrieved and enters the last two phases of the protocol with the peers responsible for them.

Assume that P is interested in the resource corresponding to the metadata $m(r)$ and stored at P'. It creates the message $\text{REQUESTRES}(S_{id}, \{id(r)\}_{k_{PP'}}, L)$, where the footprint list L, the session identifier S_{id}, and the symmetric key $k_{PP'}$ have been delivered with ANSMDATA. Here and in the remainder of the protocol, S_{id} can be used in clear, since it cannot be associated with any of the previous transactions. Its usage reduces the amount of data that needs to be encrypted/decrypted at each step and allows the peers in the destination cloud to quickly discard messages that are not addressed to them.

Routing of REQUESTRES and RETURNRES messages is the same as ANSMDATA; it utilises the footprint list L to reach the destination cloud, and then a cloud broadcast to reach the intended recipient.

Note that the cloud-based communication protocol is largely independent on the underlying network: the only connection is given by the strategy utilised to route the query, while the rest of the messages is routed according to the footprint list. This makes it possible to apply our framework to any kind of unstructured overlay network, choosing an arbitrary routing strategy for the SUBMITQ message.

4 Attack scenarios

In this section, we introduce the attacks that might in principle break anonymity and censorship-resistance in our framework. We qualitatively reason about the resistance of our framework against such attacks, referring to Section 5 for experimental evaluations.

4.1 Attacks on anonymity

Surrounding Attack. Assume that a malicious peer P_{Adv} generates a fake cloud $\mathcal{C}_{Adv}$ and sends a message $JoinCloud(\mathcal{C}_{Adv}, p, i)$ to P, and assume that P accepts to join cloud $\mathcal{C}_{Adv}$. If i is the last step of the join algorithm, P does not have any other neighbours in $\mathcal{C}_{Adv}$. The anonymity of P is thus compromised since P_{Adv} can monitor all P's activities in $\mathcal{C}_{Adv}$. This attack can be generalized by considering a population of colluding malicious peers trying to surround P. They block all the *JoinCloud* messages received from honest peers and instead send P messages of the form $JoinCloud(\mathcal{C}_{Adv}, p, i)$, where i is one of the last steps of the joining algorithm. Notice that the risk of incurring in surrounding not only depends on the topology of the network but also on its physical implementation. In a wireless scenario, for example, it is very unlikely that a malicious peer is able to gain exclusive control of the communication channel of another peer, which is constituted by its surrounding atmosphere.

To mitigate the threat of this attack, Section 5.1 presents an $update()$ function that keeps the number of peers joining the cloud in the first steps small, letting the majority of the peers in the region join the cloud in the last steps. This guarantees that the number of peers joining the cloud after the invitation of the peer under attack is relatively high in the first steps of the joining algorithm, and this number represents the anonymity guarantee of P. In fact, each peer has a significant number of clouds which he joined in the first steps of the joining algorithm, and these clouds can be used for obtaining strong anonymity guarantees. The experimental evaluation in Section 5.2 shows that the surrounding attack is effective only if the adversary controls the majority (at least 50%) of the peers around the peer under attack.

Intersection Attack. Since peers participate in different clouds, a malicious peer might try to "guess" the identity of the querying peer based on the (even partial) information that it has available about the intersection of two clouds. Notice that computing cloud intersections is difficult because cloud topology is not known in general and a malicious peer only knows its neighbors.

Remember that clouds are generated using only short-distance links and that clouds are thus confined in a small region of the network (locality property). As confirmed by the experiments in Section 5.3, this guarantees that the intersection of the clouds that a peer participates in has high cardinality, demonstrating that our framework is resistant to intersection attacks.

4.2 Attacks on censorship resistance

Blocking Attack. The blocking attack aims at blocking (instead of monitoring) all SUBMITQ messages containing undesired queries and, tracking the cloud $\mathcal{C}$ of the querying peer, at subsequently blocking all ANSMDATA messages directed to $\mathcal{C}$. Note that our query routing mechanism (see Section 3.3) guarantees that SUBMITQ is replicated and routed through different paths: this redundancy helps to overcome the blocking attack, since it requires the attackers to be located either in *all* the paths followed by SUBMITQ message or in one of the paths followed by SUBMITQ message and *all* the paths followed by ANSMDATA message. As confirmed by the experiments in Section 5.4, the blocking attack is effective only if the adversary has a pervasive control (at least 50%) of the region to surround.

In the following we present a theoretical characterisation of the resistance to blocking attacks. This is formalised as the *probability that a communication which an adversary is willing to censor will not be blocked*.

The probabilities that all SUBMITQ and all ANSMDATA messages are blocked is given by:

$$\Pr\{block(\text{SUBMITQ})\} = [\Pr\{A_1\}]^{p_1} \tag{1}$$

$$\Pr\{block(\text{ANSMDATA})\} = \Pr\{A_1\} \times [\Pr\{A_2\}]^{p_2} \tag{2}$$

where p_1 (resp. p_2) is the number of distinct paths followed by the first (resp. second) message and A_1 (resp. A_2) denotes the event that *at least one attacker resides in one of the paths of* SUBMITQ *(resp.* ANSMDATA*)*. For simplicity, we have assumed these events to be independent from each other and from the specific path followed by the messages. If the attackers are randomly distributed in the network, the probability associated to events A_1 and A_2 is:

$$\Pr\{A_1\} = \Pr\{A_2\} = 1 - \binom{N-k}{m} / \binom{N}{m} \tag{3}$$

where N is the number of peers in the network, k is the number of adversaries, and m is the average number of peers in a path connecting P to P' (where P and P' are not counted). The fraction in Equation 3 represents the number of *safe* (i.e., not including any attacker) paths divided by the total number of paths.

Finally, the degree of censorship resistance can be computed as

$$1 - \Pr\{block(\text{SUBMITQ}) \vee block(\text{ANSMDATA})\} \tag{4}$$

which can be obtained as the combination of the two non-independent events.

Man-in-the-middle Attack. In the man-in-the-middle attack (MITM), the attacker intercepts the $\text{SUBMITQ}(q, k_P^+, [C_P, \ldots])$ message sent by P and replaces it by a freshly generated message $\text{SUBMITQ}(q, k_{Adv}^+, [C_{Adv}])$, where k_{Adv}^+ and C_{Adv} are the adversary's public key and cloud, respectively. The adversary then runs two sessions of the protocol, one with the querying peer P and one with provider R, pretending to be the query responder and initiator respectively. This allows the attacker to filter and possibly interrupt the communication between P and R. Note that the attackers need to be located in *all* the paths followed by SUBMITQ, and hence the MITM is just a special case of the blocking attack.

5 Experimental evaluation

In this section, we present our experimental evaluation, which is designed to demonstrate the anonymity and censorship resistance properties of our framework and the message overhead imposed by the protocol.

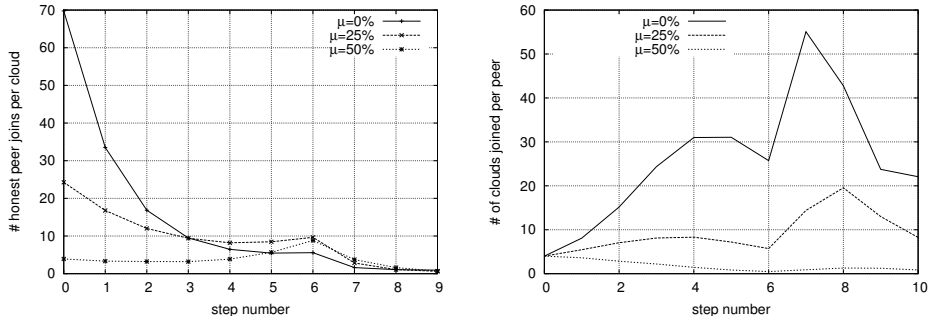
5.1 Experimental setup

For our experiments, we used a subset of the OHSUMED medical corpus [16] with 32000 documents and 100 point queries. Documents were clustered using the incremental k-means algorithm and each cluster was assigned to a single peer to simulate different peer interests. This peer specialisation assumption is common in SONs [6,18,2], and does not restrict our setting since a peer with multiple interests might cluster its documents locally and create one identity for each interest in the spirit of [20]. These interests are then used to build a semantic overlay network according to standard techniques [6,18,2]. The resulting SON is composed of 2000 peers and the description of a peer's interest is represented by the centroid vector of its documents. Each peer maintains a routing index with 10 links to other peers, 20% of which are long-distance links. On top of the created SON we invoke the cloud creation process and implement the cryptographic cloud-based protocol presented in Section 3. The *update*() function used in our setting has values 0.25 for the first six steps of the cloud creation and 1.0 for the remaining 4 steps. Therefore the number of peers joining the cloud in the first steps is small, and the majority of the peers in the region joins the cloud in the last steps. As shown in Section 5.2, this choice of the *update*() function mitigates the threat of surrounding attacks. This *update*() function produces clouds with average size of about 70 peers, which is the baseline value for our experiments.

We have explored different values for the *update*() function, and the values considered in this section returned the best results in terms of anonymity and cloud locality. Due to space constraints, we do not further discuss other options.

Our simulator is written in Java, and our experimental results were averaged over 64 runs to eliminate fluctuations in the measurements. In the experiments presented below we consider the following system parameters that are varied appropriately to demonstrate Clouds' performance.

- κ: The average number of clouds created per peer. In our experiments the baseline value for κ is one, unless otherwise stated.
- μ: In our experimental setting, we consider colluding attackers that aim at attacking a region of the network (either for censoring or for breaking anonymity). Parameter μ represents the percentage of malicious peers in the region under attack, where the region size may vary depending on the attack type.
- β: The broadcast threshold is used to measure the similarity between a query and a peer profile, and assists a peer in deciding whether a query may be effectively answered by it and its neighbours. Decreasing β results in more broadcasts and thus higher message traffic and also higher recall. The baseline value used in our experiments is 0.6.



(a) Peers joining a cloud after invitation at step i (b) Clouds joined by each peer at step i

Fig. 3. Surrounding attack when varying the percentage of malicious peers.

5.2 Surrounding Attack

The experiments on the surrounding attack are conducted by selecting 100 peers with profiles closest to the peer under attack and, from this pool of peers, by selecting a percentage μ varying from 0% to 50% to be malicious. This surrounding scenario corresponds to the best strategy for the attackers to surround a peer, since in our system each peer randomly selects his neighbours among the peers with close profile. In other words, trying to surround a peer by positioning malicious peers as close as possible, so as to form a sort of barrier, is not the best strategy since that peer would not necessarily select them as neighbours. Notice that if the neighbours of a peer were selected to be the peers with the closest profiles, as in a typical SON, then the attackers might generate profiles extremely close to the peer under attack in order to occupy all its short-distance links and effectively isolate it from the rest of the network. We consider this architectural choice of paramount importance to enforce anonymity in SONs.

Figure 3(a) shows the average number of honest participants joining a cloud $\mathcal{C}$ after an invitation from peer P or from one of its descendants in $\mathcal{C}$, depending on the step i in which P has joined $\mathcal{C}$. This number represents the anonymity degree of P in $\mathcal{C}$, assuming that the adversary knows i. In this experiment, malicious peers intercept and block all $JoinCloud(\mathcal{C}, p', i')$ messages received from honest participants. The graph shows that under a scenario where the 25% of the peers in a region attack a single honest peer ($\mu = 25\%$), Clouds offers good anonymity guarantees, and at least 10 honest peers join the cloud after P in all the first six steps of the joining algorithm. Even under a scenario where $\mu = 50\%$, the peer under attack is not completely surrounded and in the first six steps a number between five and ten peers joins the cloud after P.

Observe that the number of peers joining the cloud decreases over the number of steps in the scenario without malicious peers, as expected, but tends to increase until the sixth step when half of the peers is set to be malicious. This positive and seemingly surprising result can be explained by the cloud shape induced by the *update* function. In the first phase of the joining algorithm (steps 0-5), the probability p that P's neighbours join the cloud is low and, since many of them are malicious, the blocking strategy of

the adversary is effective and the number of peers joining the cloud after P is small. At the beginning of the second phase (steps 6-7), however, the joining probability p is high, the blocking strategy of the adversary is less effective, and so the number of P's neighbours joining the cloud is high. Finally, after the seventh step, most of the honest peers in the region have already joined cloud $\mathcal{C}$, and so the number of peers joining after P in the last steps is very small. Notice that the numbers of peers joining in the last steps of the protocol is higher when under attack. This is easily explained since clouds are local and when all the peers in a region are honest, most of them have already joined the cloud in the first steps. Under attack, most of the honest peers in a region do not join the cloud in the first steps since malicious peers block most of the *JoinCloud* messages. In the last steps, however, the joining probability increases, the blocking activity of the attackers is less effective, and, since most of the honest peers have not joined the cloud yet, they do it in the last steps.

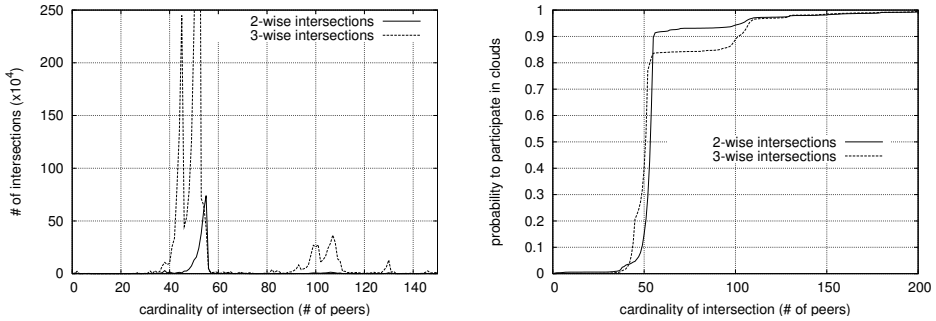
Figure 3(b) shows the number of clouds joined by each peer depending on the step number and the percentage of malicious peers. In this experiment, we consider a scenario where each peer generates four clouds on average. As we can see, even under attack, each peer has at disposal a number of clouds to join at each step, and in particular at the first steps which give in general better anonymity guarantees.

5.3 Intersection Attack

In Figure 4(a), we measured the average cardinality of all 2-wise and 3-wise intersections of the clouds which a peer participates in. Notice that the majority of the intersections has cardinality between 40 and 50 peers, and there exist very few cloud intersections with cardinality less than 30 peers. This result was expected due to the cloud creation process that exploits locality and creates clouds with high overlap, populated by semantically close peers. Figure 4(b) shows the probability for a peer to participate in clouds with 2-wise and 3-wise intersection cardinality lower than the value indicated in the x-axis. Notice that the probability to participate in clouds with intersection smaller than 40 is negligible, and that this probability does not change when moving from 2-wise to 3-wise intersections, demonstrating the resistance of Clouds to intersection attacks.

5.4 Blocking Attack

This set of experiments shows the censorship resistance properties of our system under the strongest attack, i.e., the blocking attack. In this experiment we assume a set of colluding attackers that try to censor some specific topic; notice however that the attackers cannot have any precise information about which peer(s) store documents about this topic. Thus, the attackers aim at surrounding a region in the SON and blocking all non-qualifying information that is exchanged. To perform this attack we assumed a region of 200 peers that are the closest to the topic to be censored and varied the percentage μ of malicious peers in this region. Figure 5 shows the percentage of malicious peers in this region that is needed to censor a single topic. The most interesting observation emerging from this graph is that the attackers need to occupy at least 50% of the region (i.e., 100 peers or 5% of the network in our setting) to make the attack effective. Notice



(a) Number of intersections with cardinality of size x

(b) Probability to participate in clouds with intersection cardinality lower than x

Fig. 4. Cardinality of intersections as a measure for intersection attacks.

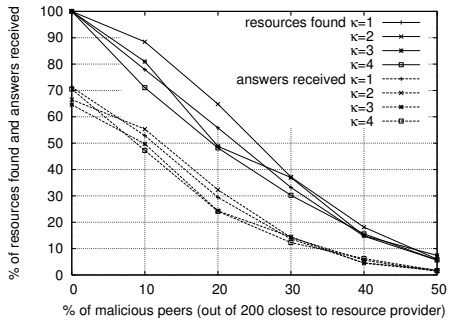
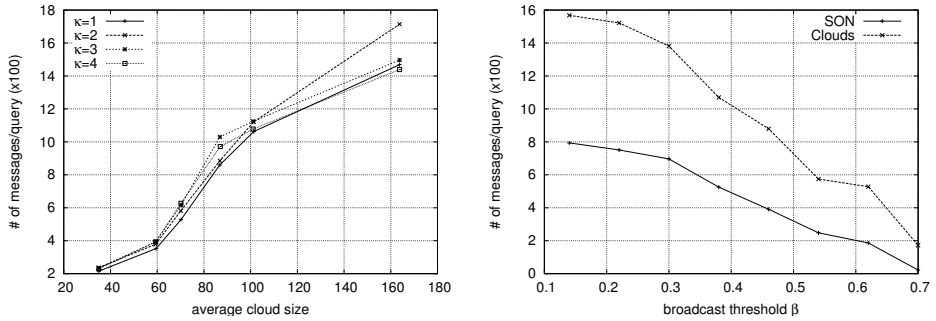


Fig. 5. Censorship resistance for colluding attackers surrounding a resource provider.

that this corresponds to censoring only a single topic, and an attacker should occupy a significant part of the network to mount an effective attack against multiple topics. Another interesting observation is that due to our fuzzy routing mechanism, some answers are not returned to the requesters even without malicious peers (leftmost point in the x-axis). Finally, observe that the number of clouds created per peer (κ) does not affect the censorship resistance properties of our approach. This is particularly interesting since, as stated before, anonymity and censorship resistance are often conflicting goals.

5.5 Message Traffic

Figure 6(a) shows how the message traffic in Clouds is affected by the size and number of the clouds in the system. To control the former parameter, we modify the *update*() function described in Section 5.1, while for the latter we modify the average number of clouds created per peer through parameter κ. An important observation derived from this figure is that message traffic is highly sensitive to the cloud size. This was expected since our routing mechanism, that is based on the fireworks technique, depends



(a) Message cost per query for different cloud sizes and κ

(b) Message overhead of Clouds compared to a SON without security features

Fig. 6. Message overhead of Clouds.

on message broadcasting inside clouds in all four phases of the protocol. Therefore, even a small increase in the average cloud size results in a significant increase in message traffic, since all cloud members have to receive all messages that have their cloud as a destination. Another interesting conclusion drawn from this experiment, is the insensitivity of message traffic to the number of clouds each peer creates. In fact, the increase in the number of clouds a peer participates in is dampened by the decrease of the probability to find one of these clouds in L (due to the increase in the domain of L). Finally, notice that the message traffic imposed on the network is heavily dominated by the broadcasting of messages within the clouds, rather than by the routing through them. This causes the routing of messages to contribute only for a small fraction to the observed traffic, thus resulting in a minimal effect in message overhead.

Figure 6(b) shows the overhead introduced by Clouds to achieve both anonymity and censorship resistance in a SON environment. The important observation in this graph is that both protocols are affected by the increase of the broadcast threshold in the same way, since both graphs present the same behaviour. Intuitively, increasing β results in less peers broadcasting the message to their neighbourhood, which eventually decreases retrieval effectiveness, but also heavily affects message traffic as also shown in the graph. The decrease in messages observed when increasing β is mainly due to less broadcasts taking place during a SUBMITQ message. The extra message traffic imposed by Clouds is mainly due to the broadcasting of messages inside clouds, while the main volume of traffic for the SON is the result of the resource discovery protocol. As already mentioned, this extra message traffic is the price we have to pay in order to avoid previous failure-prone and attack-prone solutions like static cloud gateways [22,26], or static paths between information providers and consumers [5,8].

5.6 Summary of Results

We have demonstrated the effectiveness of our architecture under different attack scenarios and we have shown that an attacker must control a large fraction of a network

region to effectively cast any type of attack. Although adversaries can utilise the SON to place themselves near target peers, they need to control over 50% of a region to compromise a peer's anonymity or effectively censor a topic. Intersection attacks are also difficult to perform, even under the assumption that an adversary knows the cloud topology and all the participants in these clouds. Both 2-wise and 3-wise cloud intersections have a cardinality higher than 40, and the probability that a peer participates in two or three clouds with small intersection cardinality is below 10^{-4}.

Cloud size affects message traffic, due to the broadcasts performed at destination clouds to locate the intended recipient of a message. Contrary, our architecture is insensitive to the number of clouds generated by each peer, as shown in Figures 5 and 6(a), thus fitting dynamic scenarios where κ is neither predictable nor enforceable. Finally, as Figure 5 suggests, Clouds is able to retrieve about 70% of the answers created by the resource providers, even when not under attack. This is due to our cloud-based routing mechanism, and creates a decrease of about 30% in retrieval performance when compared to a SON without security features. Notice, however, that Clouds is not meant to be a full-fledged information retrieval system operating under ideal conditions, but is rather designed to deliver results even under challenging attack scenarios.

6 Related work

In this section, we briefly discuss the papers that are related to our approach and overview proposals that support anonymity and censorship resistance in a P2P setting.

Systems for anonymity. In the recent past, several techniques have been presented for preserving anonymity in a P2P environment. One strand of research, pioneered by Freenet [5], tried to protect the communication channel, while other approaches tried to anonymize the communication parties by hiding them in groups.

Onion routing [14,15] belongs to the former family and is adopted by TOR [8], one of the most known systems for anonymity. An onion is a recursively layered data structure, where each layer is encrypted with the public-key of the peers it was routed through. Encrypting each layer with a different key preserves the secrecy of the path through which the onions is routed. Contrary to our approach, onion routing relies on static paths which makes the approach vulnerable to peer fails. A similar approach is followed by Tarzan [13] and MorphMix [23], where layered encryption and multi-hop routing is utilised. With the proliferation of DHTs, approaches exploiting the anonymity properties of onion routing in structured overlays have also been proposed [29].

Crowds [22] hides peers in groups, and all intra-group communication is orchestrated by a trusted server that knows all peers in its group, while Hordes [26] utilises multicast groups as reply addresses from the contacted server. Contrary to our approach, both systems rely on static groups and trusted servers, which introduce vulnerability (e.g., if servers are compromised) and central points of failure. The notion of groups (without a server component) as cloaks of peer identity was first introduced in Agyaat [1] for a DHT environment. We extend this approach to unstructured networks, and utilise a richer query language, provide cryptography-based censorship resistance, and support dynamic cloud creation and maintenance. Other approaches like DC-nets [4] and XOR-trees [9] guarantee anonymity by allowing at exactly one client

to transmit a message at a given time slot, making it a restrictive model for large-scale applications. Finally, P^5 [25] provides an interesting architecture that enables tuning between efficiency and anonymity degree, a feature that we also support by the control of cloud population.

Systems for censorship resistance. Censorship at storage level consists in performing a selective filtering on the content of a peer and aims at blocking the resource as soon as its presence is detected. To avoid this type of attack, techniques such as replication of resources [5,11] and encryption of file chunks [19,30] have been proposed.

Censorship at communication level aims at corrupting the channel between communication participants. Approaches like [12,7,21] address the problem of an adversary which is able to take control of a large fraction of peers and to render unwanted resources unreachable. Similarly to our approach, these solutions are based on limited broadcasts of the query and the result, but can only be applied to structured overlays and do not address anonymity.

7 Conclusion and Future Work

We have presented Clouds, a novel P2P search infrastructure for proving anonymous and censorship-resistant search functionality in SONs. We have demonstrated that our system is resistant against a number of attacks. Although we instantiate our technique on SONs to leverage their retrieval capabilities and to support a rich data model and query language, the anonymity and censorship resistance mechanisms presented in this paper are general and can in principle be applied to any kind of unstructured overlay. The reason is that the cloud-based communication protocol is largely independent on the underlying network: the only connection is given by the strategy utilized to route the query, while the rest of the messages is routed according to the footprint list. This makes it possible to apply our framework to any kind of unstructured overlay network, by choosing an arbitrary routing strategy for the SUBMITQ message. For instance, our approach could be applied to Gnutella-style networks by adopting flooding as the routing for the SUBMITQ message, or small-world networks [24,20] by adopting interest-based query routing, while leaving the rest of the underlying protocols (e.g., rewiring in the case of small world networks) intact.

We emphasize that SONs represent a stress-test for our architecture, due to the number of attacks that adversaries can mount by exploiting the semantic correlation among peers. Applying our architecture to other types of unstructured overlay networks would enhance the security guarantees offered by our framework. For instance, a Gnutella-style overlay network, that would lack the semantic correlation between peers, would offer more security guarantees due to the randomness of peer connections. Additionally, in social networks peers create connections based upon trust relationships from real-life (e.g., do not connect to people that either you or your friends do not know). This trust mechanism would enhance the security properties of our architecture since mounting the surrounding and blocking attacks would be harder for adversaries.

As future work, we plan to extend our analysis to other sophisticated attack scenarios like the *Sybil attack* [10], where a malicious peer obtains multiple fake identities and pretends to be multiple distinct peers in the system and the more general *Eclipse attack*

[3], where multiple identities are used to isolate a peer from the rest of the network. Syngh et al. [27] present a defense against Sybil attacks in structured overlay networks: Since a peer mounting an Eclipse attack must have a higher than average peer degree (i.e., incoming and outgoing links), the idea is to enforce a peer degree limit by letting peers anonymously *audit* each other's connectivity. This conceptually elegant solution needs the introduction of a central authority to certificate peer identifiers and the presence of a secure routing primitive using a constrained routing table [3]. It is also possible to exploit specific features of the network in order to prevent or mitigate the threat of Sybil attacks: for instance, Sybilguard [32] is a recently proposed protocol that exploits *trust relationships* among peers to limit the corruptive influences of Sybil attacks in social networks. It would be interesting to investigate a combination of these techniques in our system; in particular, trust relationships for choosing the bootstrap peer and for guiding the joining procedure and auditing-based Sybil identification for reducing the threat of Eclipse attacks against peers already present in the network.

Finally, given the importance of social networking in the Web 2.0 framework, an interesting extension of Clouds would be to exploit links among friends in a social network to determine the security and efficiency guarantees of our protocols against different types of attacks.

References

1. A. Singh and B. Gedik and L. Liu. Agyaat: Mutual Anonymity over Structured P2P Networks. *Emerald Internet Research Journal*, 2006.
2. K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2004.
3. M. Castro, P. Druschel, A. J. Ganesh, A. I. T. Rowstron, and D. S. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.
4. D. L. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1988.
5. I. Clarke, S. Miller, T. Hong, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 2002.
6. A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. In *Proceedings of the International Workshop on Agents and Peer-to-Peer Computing (AP2PC)*, 2004.
7. M. Datar. Butterflies and Peer-to-Peer Networks. In *Proceedings of the European Symposium on Algorithms (ESA)*, 2002.
8. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the USENIX Security Symposium*, 2004.
9. S. Dolev and R. Ostrobsky. Xor-Trees for Efficient Anonymous Multicast and Reception. *ACM Transactions on Information and System Security (TISSEC)*, 2000.
10. J. Douceur. The Sybil Attack. In *Proceedings of the International Workshop on Peer-To-Peer Systems (IPTPS)*, 2002.
11. R. Endsuleit and T. Mie. Censorship-Resistant and Anonymous P2P Filesharing. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, 2006.
12. A. Fiat and J. Saia. Censorship Resistant Peer-to-Peer Content Addressable Networks. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

13. M. Freedman, E. Sit, J. Cates, and R. Morris. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the International Workshop on Peer-To-Peer Systems (IPTPS)*, 2002.
14. D. Goldschlag, M. Reed, and P. Syverson. Onion Routing. *Communications of the ACM (CACM)*, 1999.
15. J. Han, Y. Liu, L. Xiao, and L. Ni. A Mutual Anonymous Peer-to-peer Protocol Design. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2005.
16. W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the Annual International ACM SIGIR Conference*, 1994.
17. I. King, C. H. Ng, and K. C. Sia. Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transactions on Information Systems*, 2002.
18. A. Loser, M. Wolpers, W. Siberski, and W. Nejdl. Semantic Overlay Clusters within Super-Peer Networks. In *Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2003.
19. A. R. M. Waldman and L. Cranor. Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System. In *Proceedings of the USENIX Security Symposium*, 2000.
20. P. Raftopoulou and E. Petrakis. iCluster: a Self-Organising Overlay Network for P2P Information Retrieval. In *ECIR*, 2008.
21. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM Special Interest Group on Data Communications (SIGCOMM)*, 2001.
22. M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1998.
23. M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer Based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the International Workshop on Privacy in the Electronic Society (WPES)*, 2002.
24. C. Schmitz. Self-Organization of a Small World by Topic. In *Proceedings of the International Workshop on Peer-to-Peer Knowledge Management (P2PKM)*, 2004.
25. R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P^5: A Protocol for Scalable Anonymous Communication. In *IEEE Security and Privacy*, 2002.
26. C. Shields and B. Levine. A Protocol for Anonymous Communication over the Internet. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2000.
27. A. Singh, T. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–12, 2006.
28. L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 2002.
29. H. Tsai and A. Harwood. A scalable anonymous server overlay network. In *International Conference on Advanced Information Networking and Applications (AINA)*, 2006.
30. M. Waldman and D. Mazières. Tangler: a Censorship-Resistant Publishing System Based on Document Entanglements. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2001.
31. L. Wu, C. Faloutsos, K. Sycara, and T. Payne. FALCON: Feedback Adaptive Loop for Content-Based Retrieval. In *Proceedings of the VLDB Conference*, 2000.
32. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 267–278, 2006.