Detection and Mitigation of the Eclipse Attack in Chord Overlays

Abstract

Distributed Hash Table-based overlays are widely used to support efficient information routing and storage in structured peer-to-peer networks, but they are also subject to numerous attacks aimed at disrupting their correct functioning. In this paper we analyze the impact of the Eclipse attack on a Chord-based overlay in terms of number of key lookups intercepted by a collusion of malicious nodes. We propose a detection algorithm for the individuation of ongoing attacks to the Chord network, relying on features that can be independently estimated by each network peer, which are given as input to a C4.5-based binary classifier. Moreover, we propose some modifications to the Chord routing protocol in order to mitigate the effects of such attacks. The countermeasures can operate in a distributed fashion or assume the presence of a centralized trusted entity and introduce a limited traffic overhead. The effectiveness of the proposed mitigation techniques has been shown through numerical results.

Index Terms

Distributed Hash Tables; Chord; Eclipse Attack;

I. Introduction

Self-organizing Distributed Hash Table-based (DHT) overlays such as Chord [1], CAN [2], Tapestry [3], and Pastry [4] enable scalable information routing in peer-to-peer (P2P) distributed networks, supporting a wide range of applications, from content distribution and file-sharing to decentralized network services.

Unfortunately, given the relative ease for a node in obtaining a membership to the overlays, such networks are prone to a variety of attacks which can be performed by a collusion of malicious nodes controlled by a single adversary, aimed at altering the routing and/or the content of the messages. Among those, the *Sybil* [5] and *Eclipse* [6] attacks have received particular attention by the research community, due to their disruptive effects.

The *Sybil* attack exploits the fact that the nodes joining the network are not required to provide any reputation guarantees and the security checks on their identities are usually very loose or even absent. Therefore, a single physical entity can easily obtain multiple logical identities and thus create a set of colluded nodes which control a considerable fraction of the keys stored in the network.

Though the *Sybil* attack is not specific to DHTs, it has been widely investigated in the last decade, since it can be used to subvert the protocol and by mounting other categories of security attacks, including the *Eclipse* attack. In this attack, the collusion of corrupted nodes performs routing table poisoning by providing to their neighbors only malicious references. If most of the entries in the routing table of an honest node are malicious, then almost all the communication flows originating from that node can be intercepted by the collusion of dishonest nodes, thus "eclipsing" the rest of the network.

The *Eclipse* attack can be in turn exploited to amplify the effect of other attacks, e.g. routing and storage attacks [6].

Numerous countermeasures to mitigate the effect of such attacks have been proposed in the recent literature (see [7] for an overview), but most of them have the drawback of limiting the scalability of the system or of introducing centralized certification/authentication mechanisms, thus distorting the intrinsically distributed nature of P2P networks.

In this paper, we analyze the impact of the *Eclipse* attack on a Chord- based DHT overlay and propose a a detection algorithm which relies on the properties of consistent hashing in order to individuate whether the network is experiencing a tentative of attack. Moreover, we discuss two mitigation techniques which can be triggered by the detection algorithm to limit the effects of *Eclipse* and ensure the correct functioning of the under-attack overlay, though with reduced performance.

The remainder of the paper is structured as follows: Section II provides an overall view about cyber-attacks to P2P overlays, while Section III recalls some basic notions about the Chord protocol. A detailed description of the *Eclipse* attacker model in Chord overlays is provided in Section IV. Section V presents our proposed detection technique, while the set of countermeasures is discussed in Section VI. The effectiveness of our detection and mitigation methods is evaluated in Section VII through numerical results. The paper is concluded in Section VIII.

II. Related Work

Various methods to counteract the effects of the *Sybil* and *Eclipse* attacks to DHT based overlays have been proposed in the recent literature: for a comprehensive overview, the reader is referred to [7]. For what concerns the *Sybil* attack, its identification has gained particular attention in several contexts, ranging from P2PSIP RELOAD protocol [8] to mobile ad-hoc sensor networks: the authors of [9] define an analytical model to compute the number of random node-IDs that an attacker would need to generate in order to achieve a given probability of success in preventing a targeted resource from being accessed by any user of a Chord DHT, by replacing all the replica nodes that store it with sybils. Paper [10] proposes a taxonomy

of the various forms of the *Sybil* attack according to different kinds of communication and identity theft methods chosen by the attacker, as well as according to the temporal evolution and the goals of the attack. That work also proposes two techniques to validate the identities of the overlay peers, relying on radio resource testing and on random key pre-distribution respectively, which require the collaboration of multiple nodes. Conversely, our proposed detection and mitigation solutions can be independently applied by each network node.

Similarly, paper [11] proposes to secure the assignment of the node identifiers by relying on centralized certification authorities, which ensure that the node IDs are randomly assigned with uniform distribution on the ID space and bounded to the IP address of the nodes. This prevents an attacker from impersonating multiple identities strategically placed along the ring, in order to gain control on a wide fraction of the network, and avoids ID swapping among the malicious nodes. The drawback of this solution is that it limits the scalability of the system. Therefore, the authors propose an alternative distributed technique aimed at limiting the rate at which the identifiers can be obtained by the nodes joining the Chord ring: in order to obtain an identifier, the node must solve a computationally demanding crypto-puzzle. Variations on the crypto-puzzle approach have been investigated also by [12] and [13].

A distributed registration procedure has been proposed in [14], which relies on registration nodes that list the node IDs associated to the same IP address and reject new joins if it the number of IDs per IP address exceeds a threshold. The authors also propose some metrics to quantify the impact of the *Sybil* attack.

Other countermeasures rely on trusted third parties [15], network coordinates (such as the hop-count distance) [16], or on social information obtained through social networks [17], [18] to individuate and exclude malicious nodes. Our proposed mitigation techniques rely on the assumption that the Chord IDs are randomly assigned, though assuming that a single attacker can control a collusion of multiple malicious nodes.

Paper [19] describes a Passive Ad-Hoc Sybil Identity Detection (PASID) mechanism in the context of mobile networks, which exploits proximity information among the nodes. The proposed algorithm requires only passive tests autonomously performed by each peer by monitoring the received messages over a certain time interval. Our detection solution is based on the same design approach.

For what concerns the detection of the *Eclipse* attack, the authors of [20] present a set of heuristics aimed at evaluating the susceptibility to localized *Eclipse* attacks of various DHT overlays based on their topology characteristics and routing mechanisms.

Among the possible defenses against *Eclipse*, [11] suggests to combine the countermeasures to the *Sybil* attack with replicating the keys on multiple nodes, thus adding redundancy to the system and allowing the usage of multiple routes to reach the key requested with the query routine, as well as routing failure tests to identify which routes have been altered by malicious nodes. The usage of redundant routing tables has been proposed also by [21] and associated to periodic churn induction [22]. However, this approach introduces a significant communication overhead: our proposed countermeasures do not require any additional message exchange or introduce a limited message overhead.

The crypto-puzzle approach is used also in [23] as countermeasure to the *Eclipse* attack: the proposed ID generation method does not require a centralized certification server and induces a limited churn.

Paper [24] proposes the combination of distributed anonymous auditing of the connectivity of the neighboring nodes with a degree bounding mechanism, in order to individuate nodes having excessively high in-degree and to remove them from the routing tables. We also propose a distributed method to identify and remove polluted entries from the nodes' successor lists.

III. Background

In this Section we provide a short overview on the Chord protocol, which provides efficient location of data items in a distributed network by identifying the items through a key and assigning the keys to one of the nodes using consistent hashing. More in detail, each node and key is associated to an m-bit identifier through a hash function such as SHA-1 [25], which takes as input the node's IP address and the key itself, respectively. The identifiers are ordered along a circle of numbers modulo 2^m and the k-th key is assigned to the first node whose identifier is equal or follows the identifier of k along the circle. Such node is referred to as the *successor node* of key k. Every time a new node n joins the network, some of the keys previously assigned to n's successor are reassigned to n, according to the values of their identifiers. Conversely, when a node leaves the network, all its keys are reassigned to its successor.

The key lookup procedure can be implemented in a distributed fashion by relying on a routing table named *finger table* and maintained by each node n, where the i-th entry contains the identifier of the successor of $ID(n) + 2^{i-1}$. A node looking for key k consults its finger table and contacts the node whose identifier most closely precedes the identifier of k by means of Algorithms 1 and 2, defined in [1]. In turn, the targeted node repeats the same operation, thus creating a recursive mechanism, until the node responsible for the key k is reached. Paper [1] proves that the lookup procedure involves $O(\log N)$ nodes, where N is the total number of nodes of the Chord ring.

To increase robustness to node failures, each node also maintains a *successor list*, where it stores the identifiers of its l nearest successors on the ring (where l is a system parameter): in case the node notices that its successor has failed, it replaces the failed node with the first live entry in the list.

Algorithm 1 n.find_successor(ID)

1: **if** $ID \in (n, successor]$ **then**
2: **return** $successor$
3: **else**
4: $n' = closest\_preceeding\_node(ID)$
5: **end if**
6: **return** $n'.find\_successor(ID)$

Algorithm 2 n.closest_preceding_node(ID)

1: **for** $i = m$ **downto** 1 **do**
2: **if** $finger[i] \in (n, ID)$ **then**
3: **return** $finger[i]$
4: **end if**
5: **end for**
6: **return** n

In the remainder of the paper, the set of nodes whose identifiers are stored in the finger table and successor list of the n-th node will be referred to as the *neighborhood* of node n.

In order to ensure the correctness of the information contained in the finger tables and successor lists, Chord supports stabilization and fix finger procedures, which periodically update the entries adapting to the actual joining nodes and network structure. For further details, the reader is referred to [1].

IV. Attacker Model and Security Definition

A. Attacker Model

In this paper we consider the *Eclipse* [6] attacker model, assuming that:

- the attacker controls a fraction f of the network peers;
- the attacker communicates out-of-band with all the colluded nodes, possibly providing them with auxiliary information;
- Chord IDs are obtained from node network addresses by using a cryptographically secure hash function, therefore the attacker has no control on the choice of the Chord IDs assigned to the corrupted nodes. Further, the Chord IDs of the malicious nodes can be assumed to be uniformly distributed along the ID space;
- malicious nodes cannot send messages with a spoofed source address (and thus with a forged Chord ID).

The malicious nodes behave according to a *dishonest* model, i.e. they do not always adhere to the protocol specifications but introduce modifications in the execution of the protocol routines. In particular, the `find_successor(x)` algorithm always returns the ID of the first malicious node following ID x in the Chord ring. What the malicious node does when it receives a query to obtain the data item associated to ID x depends on the application and on the specific attack: the malicious node could for example ignore the query request or provide altered data. Since our mitigation techniques are agnostic with respect to the specifications of the application layer, we do not make any assumption regarding the behavioral model of the attacker during the data retrieval phase. In the rest of the paper, the probability that a query is captured by a malicious node will be considered a key indicator of the effectiveness of the attack or of the countermeasure.

B. Security Definition

In the remainder of the paper, we assume that the security level achieved by the Chord-based overlay is quantified by means of the percentage of keys K_c which are captured by a collusion of malicious nodes. Therefore, the effectiveness of the countermeasures presented in Section VI will be evaluated in Section VII by means of such metric.

V. The Proposed Detection Technique

This section describes a distributed detection technique for the *Eclipse* attack. The underlying principle is that an honest node should be able to detect the attack by monitoring a set of parameters extracted from its finger table and successor list.

A. The Detection Algorithm

We assume that time is divided in intervals of finite duration T. At the end of every interval r, each honest node independently evaluates a set of input parameters (which will be referred to as *features* and enumerated in section V-B) and updates a feature vector, which contains a moving average of each parameter, computed over the last W intervals. Such sliding window mechanism is aimed at smoothing the impact of abrupt fluctuations in the trend of the considered parameters. The feature vector is given

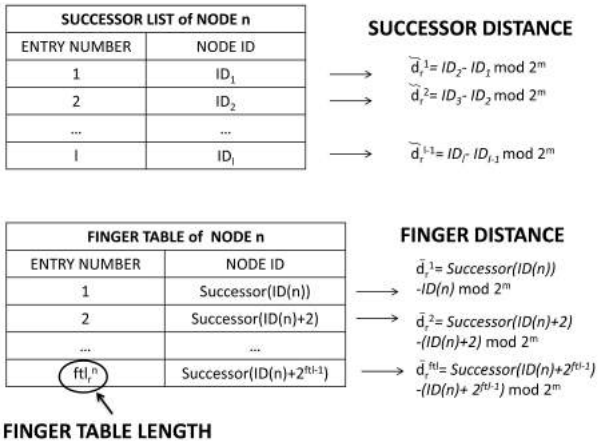


Fig. 1: The Finger Table Length, Finger Distance and Successor Distance features

as input to the detection algorithm, which returns a binary output (attack / no attack). The detection is realized by means of a supervised machine learning algorithm: the classifier models the binary classification problem using a set of training samples of features obtained by simulating a network both in presence and absence of attack. For the purpose of this paper, we use the machine learning C4.5 algorithm [26], which performs classification by building a decisional tree based on the training set. In case an attempt of attack is identified, the countermeasures described in Section VI are triggered.

B. Features Definition

We now list the set of identified features: ideally, such parameters should be sensitive to the presence of malicious nodes in the overlay, while showing a negligible dependency on the total number of nodes in the overlay, N. In Section VII, we will show through numerical results that the proposed features exhibit a significantly different trend in presence or absence of the *Eclipse* attack and a mild dependency on N.

a) Average Response Distance: The *Average Response Distance* feature evaluated at the r-th interval is defined as:

$$RD^n(r) = \frac{1}{|\mathcal{K}_r^n|} \sum_{k \in \mathcal{K}_r^n} d_r^k \mod 2^m$$

where $\mathcal{K}_r^n$ is the set of key lookup answers received by the n-th node during the r-th interval and the *Responce Distance*

$$d_r^k = successor(ID(k)) - ID(k) \mod 2^m$$

is the difference modulo 2^m of the IDs of the successor of the requested key and of the key itself, respectively. It has been shown in [27] that if the identifiers of the nodes are uniformly distributed along the Chord ring, the distance between any two nodes can be modeled as a random variable X with geometric distribution and mean $\mu = 2^m/N$. Let D_r^k be a random variable indicating the *Response Distance* attribute: thanks to the memoryless property of X, the same considerations hold for the distance between a randomly chosen point $ID(k)$ along the Chord ring and its successor $successor(ID(k))$ [27]. Therefore, in normal network conditions, $E[D_r^k] = \mu$. Since the RD feature is the arithmetic mean of $|\mathcal{K}_r^n|$ i.i.d. random variables distributed as X, $E[RD^n(r)] = E[D_r^k] = \mu$. Conversely, in presence of the *Eclipse* attack, assuming that the malicious nodes are also uniformly distributed along the ring, the fraction of keys captured by malicious nodes exhibits a higher *Response Distance*, therefore $E[RD^n(r)] \geq \mu$.

b) Finger Table Length: This feature is indicated as ftl_r^n and defined as the number of distinct entries of the finger table of node n (see Fig. 1), evaluated at the r-th time interval. In normal network conditions, ftl_r^n shows a $O(\log N)$ dependency on the number of peers [1], but it is expected to decrease in presence of the *Eclipse* attack, since a polluted finger table contains the identifiers of a subset of the malicious nodes, which usually has a low cardinality.

c) Average Finger Distance: This feature computes the distance between the start values of the finger list and their respective successors and is defined as:

$$FD^n(r) = \frac{1}{\mathrm{ftl}_r^n} \sum_{i \in \mathcal{E}_r^n} \overline{d}_r^i \mod 2^m$$

where, as shown in Fig. 1, the *Finger Distance*

$$\overline{d}_r^i = successor(ID(n) + 2^{i-1}) - (ID(n) + 2^{i-1}) \mod 2^m$$

is the distance modulo 2^m between the i-th start value of the finger table (defined as $ID(n) + 2^{i-1} \mod 2^m$) and its respective successor at interval r, and

$$\mathcal{E}_r^n = \{1\} \cup \{j \in \{2, ..., m\} \text{ such that } successor(ID(n) + 2^{j-1}) \neq successor(ID(n) + 2^{i-2})\}$$

Note that $|\mathcal{E}_r^n| = \text{ftl}_r^n$. Analogously to the RD feature, in normal network conditions $E[FD^n(r)] = \mu$.

d) Average Hop Count: This feature is defined as:

$$HC^n(r) = \frac{1}{Q_r^n} \sum_{i=1}^{Q_r^n} h_r^i$$

where Q_r^n is the number of query messages received by the n-th node during the r-th interval and the *Hop Count* h_r^i is the path length (expressed in number of hops) for th i-th query message at interval r. We assume that the value of h_r^i is included in the Chord query message. The number of hops between a node originating a query for the k-th key and the successor of k normally exhibits a logarithmic dependency on N: the authors of [1] show through numerical results that the average path length between two randomly chosen nodes is about $\tilde{h} = 1/2 \log_2 N$. During the *Eclipse* attack, the hop count experiences a decrease due to the dishonest behavior of the malicious nodes, which impersonate the successor of every key query they receive.

e) Average Successor Distance: This feature evaluates the average distance among two consecutive entries of the nodes' successor list and is defined as:

$$SD^n(r) = \frac{1}{l} \sum_{i=1}^{l-1} \tilde{d}_r^i \mod 2^m$$

where l is the number of entries of the successor list and, as shown in Fig. 1, the *Successor Distance*

$$\tilde{d}_r^i = successor\_list[i+1] - successor\_list[i] \mod 2^m$$

is the distance modulo 2^m between the i-th and $(i+1)$-th entries of the successor list. Similarly to the *RD* feature, such distance is modeled by the aforementioned random variable X and is expected to increase in presence of malicious nodes. Therefore, if no attack takes place, $E[SD^n(r)] = \mu$, and $E[SD^n(r)] \geq \mu$ otherwise.

VI. Countermeasures

In this Section we propose two countermeasures to mitigate the effects of the *Eclipse* attack. Such countermeasures operate in two different phases of the Chord protocol, namely the construction procedure of the nodes' finger table and successor list and the message forwarding during the key lookup routine. Both techniques are aimed at improving the knowledge of the honest nodes about the network topology according to the following approaches:

- building an auxiliary local list of node identifiers which can integrate the node's finger table and successor list;
- purging the node's successor list form the entries sent by the malicious nodes, in order to isolate them during the execution of the protocol.

Moreover, our proposed techniques can be categorized as follows:

- countermeasures which do not require any modification of the standard Chord protocol;
- countermeasures which modify the standard protocol by introducing additional information exchanges.

A. Auxiliary Node List

The basic idea is to provide to each honest node an auxiliary list of IDs of other peers, which can be used to integrate both the local finger table and successor list, as shown in Algorithm 3. The Algorithm works assuming that the IDs in the auxiliary list are stored in ascending order and supposes that the successor list lookup (line 4) and the finger table lookup (lines 5-7) are implemented as within the OMNET++/OverSim framework [28], [29]. Moreover, the Algorithm introduces the auxiliary node list lookup (lines 8-10). If at least a certain amount of entries of the auxiliary list are IDs of honest nodes, the effects of the routing pollution performed by the malicious nodes, which always provide false routing information when contacted by the honest nodes during the find successor process, can be limited. We propose the following three approaches to generate the auxiliary node list.

1) Centralized approach: It assumes the presence of an external trusted node which has full knowledge of the network topology and periodically communicates to each node w peers' identifiers, obtained by random sampling of the whole list of identifiers of the nodes currently joining the Chord ring. Neglecting the communication overhead due to the notifications of joins and leaves to the external node, the size of a message containing the auxiliary list destined to the peers is $w \cdot m$ bits.

The drawback of this approach is that it requires a centralized infrastructure, which might limit the scalability of the P2P network.

Algorithm 3 n.modified_find_successor(ID)

```
if ID ∈ (n, successor] then
    return successor
else
    n' = successor_list.closest_preceeding_node(ID)    // successor list lookup
    if finger_table.closest_preceeding_node(ID) ∈ (n', ID) then
        n' = finger_table.closest_preceeding_node(ID)    // finger table lookup
    end if
    if auxiliary_table.closest_preceeding_node(ID) ∈ (n', ID) then
        n' = auxiliary_table.closest_preceeding_node(ID) // auxiliary table lookup
    end if
end if
return n'.find_successor(ID)
```

Algorithm 4 n.estimate_μ

```
SD_1^n(r) ← successor(ID(n)) − ID(n)
for j = 2 to l do
    d̃_r^j ← successor_list[j] − successor_list[j − 1]
    if d̃_r^j > z · SD_{j−1}^n(r) then
        return SD_{j−1}^n(r)
    end if
    SD_j^n(r) ← ((j−1)/j) SD_{j−1}^n(r) + (1/j) d̃_r^j
end for
return SD_l^n(r)
```

2) Distributed non interactive approach: The auxiliary table can be built relying exclusively on local information as follows. Every honest node n independently stores in the auxiliary list the identifier of the source node of each key lookup message it receives.

This solution has the advantage of not requiring any additional information exchange among the nodes other than the standard Chord procedures.

3) Distributed interactive approach: Periodically, the n-th honest node sends to each node belonging to its neighborhood a *neighborhood request* message. If the contacted node is honest, it answers sending the list of identifiers of its own neighbors, which node n uses to fill its auxiliary table.

Since a node neighborhood comprehends the finger table entries (which are on average $\log N$) and the l successor list entries, the average size of a neighborhood response message can be computed as $(\log N + l) \cdot m$ bits.

Note that the neighbor request/response messages are not included in the standard Chord protocol, thus the corresponding routines must be defined accordingly. The drawback of this solution is that the neighborhood exchange among the nodes introduces an additional traffic overhead.

B. Elimination of Far Successors

Differently from the previous countermeasure, this technique is aimed at identifying and eliminating the malicious entries contained in the local successor list, without introducing any communication overhead.

The main idea is than malicious nodes exclusively distribute routing information about other malicious nodes, therefore the distance among two consecutive malicious entries of the successor list is expected to be higher than in the case of honest nodes. It follows that a large distance between entries in the successor list is a strong indication that these successors may be malicious.

The countermeasure operates as follows: every honest node independently computes the distance between every pair of subsequent nodes i and $i+1$ in the successor list. When such a distance is larger than $h\mu$, where μ is the average distance and h is an empirical factor, the $(i+1)$-th node is classified as malicious and the corresponding entry is eliminated from the successor list. For a discussion about the tuning of h, the reader is referred to Section VII.

We observe that $\mu = 2^m/N$ depends on the total number of peers N, which is not known to the network nodes unless it is provided by a centralized entity. Therefore, in a distributed scenario, an estimate $\hat{\mu}^n$ must be autonomously calculated by each peer. To do so, the n-th node executes Algorithm 4. The node relies on its successor list and evaluates the *Successor Distance* $\tilde{d}_r^i$ at every time interval r for each pair of consecutive IDs stored in the list, starting from the first entry ($j = 1$),

TABLE I: Simulation parameters

Total simulation time	5500 s
Query generation rate	0.2 queries/s
Successor list size	16 entries
Fix finger interval	100 s
Stabilize interval	20 s

which corresponds to $successor(ID(n))$ and calculating the average distance between the first j successors as:

$$SD_j^n(r) = \frac{1}{j}(successor\_list[j] - ID(n)).$$

If the network is not subject to an attack, then the estimate $SD_l^n(r)$ calculated over all the nodes in the successor list is a good estimator of the average distance between subsequent nodes. However, the successor list might be polluted by malicious entries. In that case, all the entries following a malicious one are also likely to be malicious, since a colluded node provides routing information pointing exclusively to other colluded nodes. Under the assumption that the malicious nodes are uniformly distributed along the Chord ring, the average distance among two consecutive malicious nodes will be higher than μ. We consider two cases. If for all nodes $2 \leq j \leq l$, the inequality $\tilde{d}_r^j \leq z \cdot SD_{j-1}^n(r)$ holds, then an estimate for $\hat{\mu}^n(r)$ is $SD_l^n(r)$. Otherwise, calling $\hat{j}$ the first successor for which $\tilde{d}_r^j > z \cdot SD_{j-1}^n(r)$, the estimation procedure yields $\hat{\mu}^n(r) = SD_{\hat{j}-1}^n(r)$. The parameter z is a tuning factor which defines the sensitivity of the estimation mechanism. In order to smooth out fluctuations in the estimate due to churn and stabilization events, the running estimate $\hat{\mu}^n$ is calculated by averaging the estimations for the W previous rounds, exploiting the same sliding window mechanism discussed in Section V. An evaluation of the effectiveness of this algorithm is provided in Section VII.

VII. Performance Assessment

In this section, we evaluate the impact of the *Eclipse* attack on the Chord protocol and the performance of our proposed detection and mitigation techniques. For this purpose, the C4.5-based detection algorithm has been implemented relying on the WEKA data mining software [30], while the attack and the countermeasures have been implemented within the *OMNET++/OverSim* framework with the parameters reported in Table I under the following assumptions regarding the behavioral model of the malicious nodes:

- a key lookup procedure is never initiated by a malicious node;
- malicious nodes ignore all the key lookup messages they receive.

A. Detection Phase

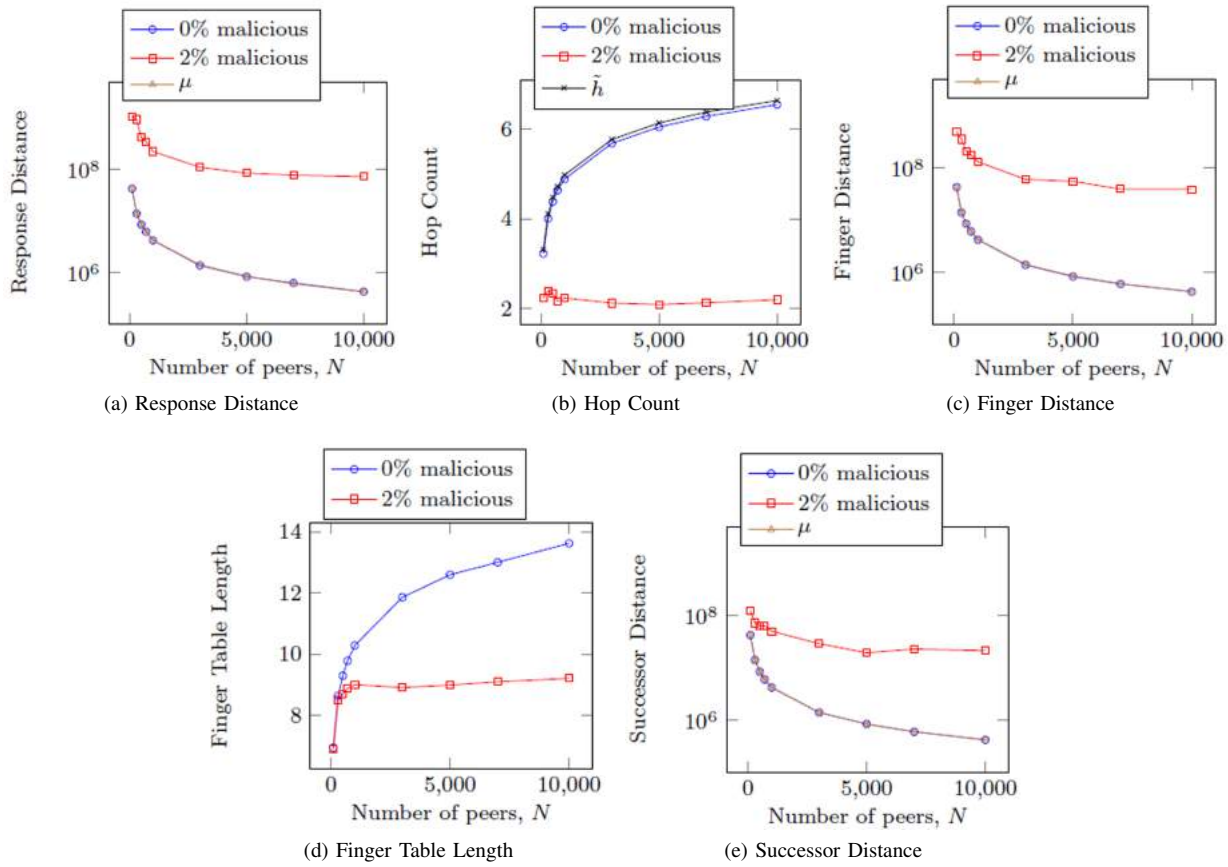
Figure 2 depicts the trend of the averaged features described in Section V for different amounts of Chord peers, assuming $W = 10$ and $T = 200s$, and compares them with their analytic estimations. In all the considered scenarios, the difference in the values of the features between in presence/absence of attack is significant and, especially for medium to high values of N, their dependence on the number of network peers is mild. Moreover, in case of normal network conditions, results perfectly match the analytic estimations.

The C4.5-based detection algorithm has been evaluated over a data trace of almost 50,000 instances (with N ranging from 100 to 10,000, which we believe includes the sizes of most of real DHT overlay deployments, and $0 \leq f \leq 0.05$) by means of a 10-fold cross-validation. Results provided a True Positive Rate of 99.78% (defined as the percentage of instances with ongoing attack correctly identified), a True Negative Rate of 99.77% (defined as the percentage of instances with normal network conditions correctly identified) a False Discovery Rate of 0.25% (defined as the percentage of incorrectly classified instances, among all the instances with ongoing attacks), and an overall Accuracy of 99.775% (defined as the overall percentage of instances correctly classified) showing that the selected features make it possible to correctly individuate an ongoing *Eclipse* attack with very high probability.

B. Mitigation Phase

We now discuss the performance of our proposed countermeasures. Figure 3 plots the trend of the percentage of captured keys, K_c, as a function of the fraction of colluded nodes, f, for various values of the overall number of nodes, N, ranging from 100 to 1000. Results show that such percentage increases superlinearly with f: even with a small fraction of malicious nodes, K_c is very high and closely approaches 100% in case of large networks.

Figure 4 shows the impact of the length of the auxiliary node list, w (ranging from 0 to a fraction of 1/5 of the total number of nodes N), on the effectiveness of the *Auxiliary Node List* defense: even if the number of entries of such tables is limited



(a) Response Distance (b) Hop Count (c) Finger Distance

(d) Finger Table Length (e) Successor Distance

Fig. 2: Dependence of the detection features on the number of peers, N, in absence/presence of malicious nodes. μ indicates the analytic average distance among two consecutive nodes and $\tilde{h}$ the analytic average hop count of a query, both evaluated in normal network conditions

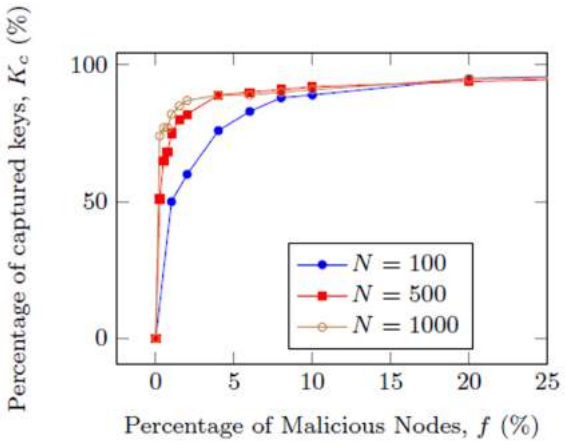


Fig. 3: Percentage of captured keys, K_c, versus the fraction of malicious nodes, f, for various numbers of peers.

(e.g. $w = N/50$, which we believe does not hinder scalability even in large overlays), K_c drops significantly, especially for low f.

For the *Elimination of Far Successors* countermeasure, Figure 5 plots the distribution of the relative error in the estimation of μ, defined as $e = (\widehat{\mu} - \mu)/\mu$, over more than 10,000 instances in three different attack scenarios, for $z = 5$. The right long tail visible in all distributions is due to the effect of pollution caused by the malicious nodes. Further, the tuning of the parameter h has been performed empirically, by evaluating K_c for different values of N and h, concluding that the effectiveness of the countermeasure is only marginally affected by the value of h, which can be chosen without need of a fine grained tuning. In

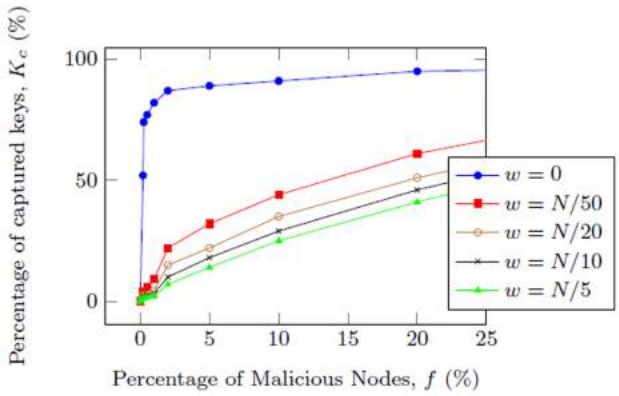


Fig. 4: Percentage of captured keys, K_c, versus the fraction of malicious nodes, f, for various sizes of the auxiliary node list, w, assuming $N = 1000$.

the remainder of the Section, if not differently stated, we set $h = 1.2$.

Finally, Figure 6 compares the percentage of captured keys in presence of the *Eclipse* attack in case no countermeasure is used to the values of K_c obtained by applying each of the three distributed mitigation techniques independently, and then by combining all of them (the *Auxiliary Node List* centralized approach is considered as benchmark), for different values of N. The size of the auxiliary node list has been set to $w = N/50$. The Figure shows that the reduction of K_c obtained through each single distributed countermeasure is mild, but combining all the distributed approaches leads to a considerable decrease of the percentage of captured keys, which results to be only slightly above the performance of the benchmark centralized approach (which obtains the best performance due to the omniscient knowledge of the overlay topology of the external trusted node). Therefore, we conclude that the distributed mitigation techniques provide comparable effectiveness with respect to the centralized approach, avoiding the introduction of scalability issues of the P2P overlay.

VIII. Conclusions

This paper discusses the impact of the *Eclipse* attack to Chord-based structured peer-to-peer overlays, proposing a detection technique to individuate its presence and a set of countermeasures to mitigate its effects. The attack detection can be independently performed by each node by relying exclusively on passively monitored classification features, which are given as input to a C4.5 classifier. In case an ongoing attack is individuated, the classification output triggers the defense mechanisms, which operate on two different phases of the protocol: the construction procedure of the nodes' finger table and successor list and the message forwarding during the key lookup routine. For the former phase, we propose both a centralized and a distributed defense approach, the second being declined in an interactive and non interactive variant, while the countermeasure operating during the message forwarding phase can be non-interactively performed by each peer. Numerical results show the effectiveness of the proposed identification and mitigation algorithms.

References

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *Networking, IEEE/ACM Trans. on*, vol. 11, no. 1, Feb. 2003.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, Aug. 2001. [Online]. Available: http://doi.acm.org/10.1145/964723.383072

[3] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and," University of California at Berkeley, Berkeley, CA, USA, Tech. Rep., 2001.

[4] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware '01. London, UK, UK: Springer-Verlag, 2001, pp. 329–350. [Online]. Available: http://dl.acm.org/citation.cfm?id=646591.697650

[5] J. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Springer Berlin / Heidelberg, 2002, vol. 2429, pp. 251–260.

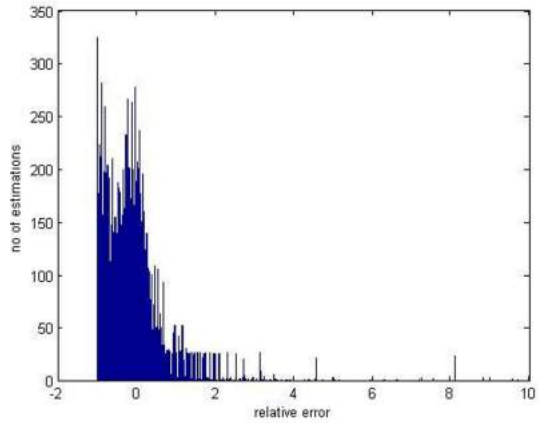
[6] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 261–269. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687810

[7] G. Urdaneta, G. Pierre, and M. van Steen, "A survey of DHT security techniques," *ACM Computing Surveys*, vol. 43, no. 2, Jan. 2011. [Online]. Available: http://www.globule.org/publi/SDST_acmcs2009.html

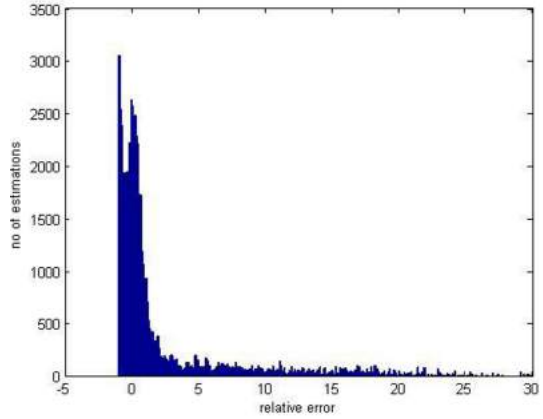
[8] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "REsource LOcation and discovery (RELOAD) base protocol ¡draftietf- p2psip-base-22¿," Internet Draft, Tech. Rep., July 2012.

[9] M. Uruena, R. Cuevas, A. Cuevas, and A. Banchs, "A model to quantify the success of a sybil attack targeting reload/chord resources," *Communications Letters, IEEE*, vol. 17, no. 2, pp. 428–431, February 2013.

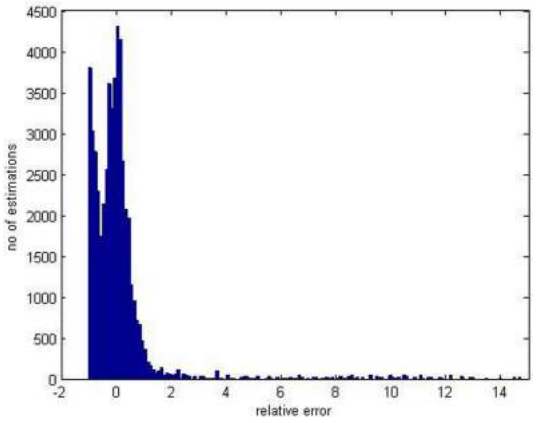
[10] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, ser. IPSN '04. New York, NY, USA: ACM, 2004, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/984622.984660



(a) $N = 100, f = 0.05$



(b) $N = 500, f = 0.05$



(c) $N = 1000, f = 0.02$

Fig. 5: Distribution of the relative error e in the estimation of μ, for different values of N and f, assuming $z = 5$.

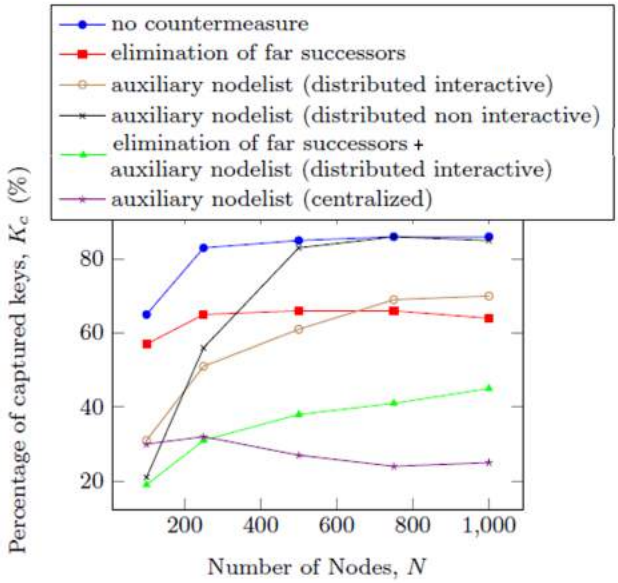


Fig. 6: Percentage of captured keys, K_c, versus the number of nodes, N, for the proposed countermeasures, assuming $f = 0.03$.

[11] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 299–314, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844156

[12] N. Borisov, "Computational puzzles as sybil defenses," in *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, ser. P2P '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 171–176. [Online]. Available: http://dx.doi.org/10.1109/P2P.2006.10

[13] H. Rowaihy, W. Enck, P. Mcdaniel, and T. L. Porta, "Limiting sybil attacks in structured peer-to-peer networks," Tech. Rep., 2005.

[14] J. Dinger and H. Hartenstein, "Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration," in *IN ARES 06: proceedings of the first international conference on Availability, Reliability and Security*, 2006, pp. 756–763.

[15] J. Caubet, O. Esparza, J. L. Muoz, J. Alins, and J. Mata-Daz, "RIAPPA: a robust identity assignment protocol for p2p overlays," *Security and Communication Networks*, pp. n/a–n/a, 2014. [Online]. Available: http://dx.doi.org/10.1002/sec.956

[16] R. Bazzi, Y.-r. Choi, and M. Gouda, "Hop chains: Secure routing and the establishment of distinct identities," in *Principles of Distributed Systems*, ser. Lecture Notes in Computer Science, M. Shvartsman, Ed. Springer Berlin Heidelberg, 2006, vol. 4305, pp. 365–379. [Online]. Available: http://dx.doi.org/10.1007/11945529_26

[17] G. Danezis, C. Lesniewski-laas, M. F. Kaashoek, and R. Anderson, "Sybil-resistant dht routing," in *In ESORICS*. Springer, 2005, pp. 305–318.

[18] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 267–278, Aug. 2006. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159945

[19] C. Piro, C. Shields, and B. Levine, "Detecting the sybil attack in mobile ad hoc networks," in *Securecomm and Workshops, 2006*, 2006, pp. 1–11.

[20] D. Germanus, R. Langenberg, A. Khelil, and N. Suri, "Susceptibility analysis of structured p2p systems to localized eclipse attacks," in *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, Oct 2012, pp. 11–20.

[21] B. Awerbuch and C. Scheideler, "Towards a scalable and robust dht," in *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, ser. SPAA '06. New York, NY, USA: ACM, 2006, pp. 318–327. [Online]. Available: http://doi.acm.org/10.1145/1148109.1148163

[22] K. Hildrum and J. Kubiatowicz, "Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks," in *in proceedings of the 17th international symposium on distributed computing*, 2003, pp. 321–336.

[23] R. Zhang, J. Zhang, Y. Chen, N. Qin, B. Liu, and Y. Zhang, "Making eclipse attacks computationally infeasible in large-scale dhts," in *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, nov. 2011, pp. 1 –8.

[24] A. Singh, T. Ngan, P. Druschel, and D. Wallach, "Eclipse Attacks on Overlay Networks: Threats and Defenses," in *Proc IEEE INFOCOM*, Barcelona, Spain, April 2006.

[25] D. Eastlake, 3rd and P. Jones, "Us secure hash algorithm 1 (sha1)," United States, 2001.

[26] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[27] A. Binzenhöfer, D. Staehle, and R. Henjes, "On the fly estimation of the peer population in a chord-based p2p system," in *19th International Teletraffic Congress (ITC19)*, Beijing, China, 9 2005.

[28] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Simutools '08: Proceedings of the 1st International Conference on Simulation tools and techniques for Communications, Networks and Systems Workshops*, 2008.

[29] "OverSim: The Overlay Simulation Framework," http://www.oversim.org/.

[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," vol. 11, no. 1, 2009.