Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 1115 – 1120

Procedia Computer Science

International Workshop on Secure Peer-to-Peer Intelligent Networks Systems (SPINS-2014)

Mitigation of the Eclipse Attack in Chord Overlays

Cristina Rottondi[a,*], Alessandro Panzeri[a], Constantin Yagne[a], Giacomo Verticale[a]

[a]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, P.zza Leonardo da Vinci 32, 20133 Milano, Italy*

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. Moreover, we propose some modifications to the Chord routing protocol in order to mitigate its effects. Such 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.

Keywords: Distributed Hash Tables; Chord; Eclipse Attack;

1. Introduction

Self-organizing Distributed Hash Table-based (DHT) overlays such as Chord[1] 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, *Sybil*[2] and *Eclipse*[3] 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 perform 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

* Corresponding author. Tel.: +39-02-2399-3691 ; fax: +39-02-2399-3413.
E-mail address: cristinaemma.rottondi@polimi.it

doi:10.1016/j.procs.2014.05.541

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[3].

Numerous countermeasures to mitigate the effect of such attacks have been proposed in the recent literature (see Urdaneta *et al.*[4] 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 *Eclipse* on a Chord- based DHT overlay and propose two mitigation techniques to limit its effects and ensure the correct functioning of the under-attack overlay, though with reduced performance. The remainder of the paper is structured as follows: Section 2 provides an overview about cyber-attacks to P2P overlays, while Section 3 recalls basic notions about the Chord protocol. A detailed description of the *Eclipse* attacker model in Chord overlays is provided in Section 4. Our proposed countermeasures are discussed in Section 5 and their effectiveness is evaluated in Section 6 through numerical results. The paper is concluded in Section 7.

2. 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 Urdaneta *et al.*[4]. For what concerns the *Sybil* attack, Castro *et al.*[5] propose 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 were investigated by Borisov *et al.*[6] and Rowaihy *et al.*[7].

A distributed registration procedure has been proposed by Dinger *et al.*[8], 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 network coordinates (such as the hop-count distance), as proposed by Bazzi *et al.*[9], or on social information obtained through social networks, as discussed by Danezis *et al.*[10] and Yu *et al.*[11] 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.

Among the possible defenses against the *Eclipse* attack, Castro *et al.*[5] suggest 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 Awerbuch *et al.*[12] and associated to periodic churn induction by Hildrum *et al.*[13]. However, this approach introduces a significant communication overhead: our proposed countermeasures do not require any additional message exchange or introduce a limited message overhead. Singh *et al.*[14] propose the combination of distributed anonymous auditing of the connectivity of the neighbouring 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.

3. Background on the Chord Protocol

The Chord protocol 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, 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

Algorithm 1 n.find_successor(ID)

```
1: if ID ∈ (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] ∈ (n, ID) then
3:         return finger[i]
4:     end if
5: end for
6: return n
```

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 $n + 2^{i-1}$. 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 r nearest successors on the ring (where r 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. 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].

4. Attacker Model

We consider two attack scenarios: the *Sybil* and the *Eclipse* attacks. In both cases, we assume that the attacker controls a fraction f of the network peer nodes and communicates out-of-band with all of the colluded nodes, possibly providing them with auxiliary information. Moreover, we assume that 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. Finally, we assume that a malicious nodes cannot send messages with a spoofed source address, (and thus with a forged Chord ID).

In the *Sybil* attack, the malicious nodes follow the *honest-but-curious* behavioral model, i.e. they honestly execute the protocol but they keep all the information they obtain by participating to the Chord query and routing procedures.

Differently, in the *Eclipse* attack 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.

5. 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

Algorithm 3 n.modified_find_successor(ID)

```
1: if ID ∈ (n, successor] then
2:     return successor
3: else
4:     n' = successor_list.closest_preceeding_node(ID) // successor list lookup
5:     if finger_table.closest_preceeding_node(ID) ∈ (n', ID) then
6:         n' = finger_table.closest_preceeding_node(ID) // finger table lookup
7:     end if
8:     if auxiliary_table.closest_preceeding_node(ID) ∈ (n', ID) then
9:         n' = auxiliary_table.closest_preceeding_node(ID) // auxiliary table lookup
10:    end if
11: end if
12: return n'.find_successor(ID)
```

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: *i)* building an auxiliary local list of node identifiers which can integrate the node's finger table and successor list; *ii)* 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: *i)* countermeasures which do not require any modification of the standard Chord protocol; *ii)* countermeasures which modify the standard protocol by introducing additional information exchanges.

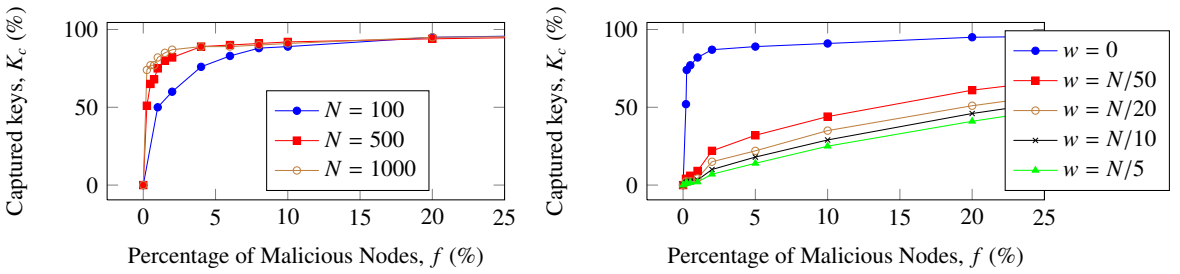
5.1. Auxiliary Node List

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 implements the successor list lookup (line 4) and the finger table lookup (lines 5-7) as implemented within the *OMNET++/OverSim* framework [15,16]. 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.

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

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.

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 r successor list entries, the average size of a neighborhood response message can be computed as $(\log N + r) \cdot m$. Note that the neighbor request/response messages are not included in the standard Chord protocol, thus the 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.



(a) Dependence on the numbers of peers, N

(b) Dependence on the size of the auxiliary node list, w, assuming $N = 1000$.

Fig. 1: Percentage of captured keys, K_c, versus the fraction of malicious nodes, f

5.2. *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. It has been shown in [17] that if the identifiers of the malicious nodes are uniformly distributed along the Chord ring, the distance (expressed as the difference of two node IDs modulo m) between two consecutive malicious nodes can be modelled as a random variable with geometric distribution and mean $\mu_f = \frac{2^m}{fN}$, while the distance among any two consecutive nodes is a random variable with geometric distribution and mean $\mu = \frac{2^m}{N} \leq \mu_f$. Since the successor list of a malicious nodes exclusively contains the identifiers of other malicious nodes, the distance among two consecutive entries of the list is expected to be higher with respect to the case of honest nodes. The countermeasure operates as follows: every honest node independently computes the distance between the i-th and $(i+1)$-th entry of the successor list ($1 \leq i < r$). If such distance exceeds $h \cdot \mu$, where h is a system parameter, 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 6.

6. Performance Assessment

In this section, we evaluate the impact of the *Eclipse* attack on the Chord protocol and the performance of our proposed mitigation techniques. For this purpose, the attack and the countermeasures have been implemented within the *OMNET++/OverSim* framework under the assumptions regarding the behavioral model of the malicious nodes that a key lookup procedure is never initiated by a malicious node, and that malicious nodes ignore all the key lookup messages they receive.

Figure 1a 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. 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 1b shows the impact of the length of the auxiliary node list, w, on the effectiveness of the *Auxiliary Node List* defense: even if the number of entries of such tables is limited (e.g. $w = N/50$), K_c drops signifcantly, especially for low f. For the *Elimination of Far Successors* countermeasure, the tuning of the parameter h has been performed empirically, by evaluating K_c for different values of N and h: numerical results (not reported for the sake of conciseness) show that the effectiveness of the countermeasure is not significantly affected by the value of h, which can be chosen without need of a fine grained tuning.

Finally, Figure 2 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 parameter h for the *Elimination of Far Successors* countermeasure has been set to $h = 1.2$, while the size of the auxiliary node list has been set to $w = N/50$. Results show 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

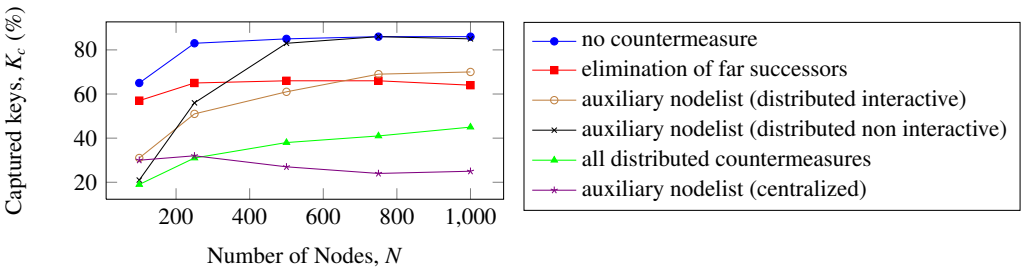


Fig. 2: Percentage of captured keys, K_c, versus the number of nodes, N, for the proposed countermeasures, assuming $f = 0.03$.

centralized approach. 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.

7. Conclusions

This paper discusses the impact of the *Eclipse* attack to Chord-based structured peer-to-peer overlays and proposes a set of countermeasures to mitigate its effects. Numerical results show the effectiveness of the proposed mitigation techniques. Future research activities will be devoted to the design of detection algorithms aimed at identifying a possible ongoing attack, in order to trigger adequate defense mechamisms.

References

1. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M., Dabek, F., et al. Chord: a scalable peer-to-peer lookup protocol for internet applications. *Networking, IEEE/ACM Trans on* 2003;**11**(1).
2. Douceur, J.. The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A., editors. *Peer-to-Peer Systems*; vol. 2429 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg; 2002, p. 251–260.
3. Sit, E., Morris, R.. Security considerations for peer-to-peer distributed hash tables. In: *Revised Papers from the First International Workshop on Peer-to-Peer Systems*; IPTPS '01. London, UK, UK: Springer-Verlag. ISBN 3-540-44179-4; 2002, p. 261–269.
4. Urdaneta, G., Pierre, G., van Steen, M.. A survey of DHT security techniques. *ACM Computing Surveys* 2011;**43**(2).
5. Castro, M., Druschel, P., Ganesh, A., Rowstron, A., Wallach, D.S.. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper Syst Rev* 2002;**36**(SI):299–314.
6. Borisov, N.. Computational puzzles as sybil defenses. In: *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*; P2P '06. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-2679-9; 2006, p. 171–176.
7. Rowaihy, H., Enck, W., Mcdaniel, P., Porta, T.L.. Limiting sybil attacks in structured peer-to-peer networks. Tech. Rep.; 2005.
8. Dinger, J., Hartenstein, H.. 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, p. 756–763.
9. Bazzi, R., Choi, Y.r., Gouda, M.. Hop chains: Secure routing and the establishment of distinct identities. In: Shvartsman, M., editor. *Principles of Distributed Systems*; vol. 4305 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-49990-9; 2006, p. 365–379.
10. Danezis, G., Lesniewski-laas, C., Kaashoek, M.F., Anderson, R.. Sybil-resistant dht routing. In: *In ESORICS*. Springer; 2005, p. 305–318.
11. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput Commun Rev* 2006;**36**(4):267–278.
12. Awerbuch, B., Scheideler, C.. Towards a scalable and robust dht. In: *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*; SPAA '06. New York, NY, USA: ACM. ISBN 1-59593-452-9; 2006, p. 318–327.
13. Hildrum, K., Kubiatowicz, J.. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In: *in proceedings of the 17th international symposium on distributed computing*. 2003, p. 321–336.
14. Singh, A., Ngan, T., Druschel, P., Wallach, D.. Eclipse Attacks on Overlay Networks: Threats and Defenses. In: *Proc IEEE INFOCOM*. Barcelona, Spain; 2006, .
15. Varga, A., Hornig, R.. 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, .
16. Baumgart, I., Heep, B., Krause, S.. OverSim: A flexible overlay network simulation framework. In: *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*. 2007, p. 79–84.
17. Binzenhöfer, A., Staehle, D., Henjes, R.. On the fly estimation of the peer population in a chord-based p2p system. In: *19th International Teletraffic Congress (ITC19)*. Beijing, China; 2005, .