Computer Networks 94 (2016) 205–218

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

S-Kademlia: A trust and reputation method to mitigate a Sybil attack in Kademlia

Riccardo Pecori*

University of Parma, Department of Information Engineering, V.le G.P. Usberti 181/A, 43124 Parma PR, Italy

ARTICLE INFO

Article history:
Received 11 January 2015
Revised 14 October 2015
Accepted 6 November 2015
Available online 12 November 2015

Keywords:
Peer-to-peer
Sybil attack
Incorrect routing
Kademlia
Trust and reputation
Storage and retrieval attacks

ABSTRACT

Peer-to-peer architectures have become very popular in the last years for a variety of services and applications such as collaborative computing, streaming and VoIP applications. The security and integrity of the overlay involved in such networks is a fundamental prerequisite for deploying such a technology. Withstanding multiple false identities in the overlay, also known as a Sybil attack, is one of the main challenges in securing structured peer-to-peer networks. Poisoning routing tables through these identities may make the routing and storage and retrieval processes extremely difficult and time consuming. In this paper we investigate possible countermeasures and propose a novel method for making the routing and the storage and retrieval of resources in a Kademlia network more secure through the use of a combined trust-based algorithm exploiting reputation techniques. Our solution provides a balanced mixing of standard Kademlia algorithms and trust-based algorithms showing promising results in thwarting a Sybil attack in a Kademlia network, in comparison with similar methods as well.

1. Introduction

In the current structure of the Internet, peer-to-peer (P2P) networks have conquered a significant part in the whole traffic distribution and they have become very popular for their scalability and the great amount of services they can support. Upon their inception, they were mainly deployed as a simple, decentralized and scalable way to exchange files, but they are now used for several different services as well. Among these we can recall P2P Voice-over-IP (VoIP) communications [1,2], streaming applications, sharing bandwidth and computing power, storage capacity and many others.

During the last years, in particular, the role of structured P2P networks has been becoming preponderant; this is mainly due to their ability of both organizing the network, through their own addressing schemes, into a so called "overlay network" and of exploiting particular algorithms for fast and efficient lookup and storage and retrieval operations. Some examples of these algorithms are Chord [3], Kademlia [4], Pastry [5], CAN [6], and the like. They use Distributed Hash Tables (DHTs) to allow for efficient lookup of identifiers and routing to the corresponding nodes. This is achieved by imposing a strict structure on the routing tables of nodes, warranting quick convergence to a target. This firm framework makes DHTs, on one side, efficient and simple to use, on the other side, subject to be easily attacked and broken by a set of malicious nodes that return not useful information instead of helping in the routing. As a matter of fact, a very large number of fake identities (IDs) in the overlay can poison honest nodes routing tables and disrupt or degrade the DHT performances. This is the so-called "Sybil attack" [7].

Therefore some security issues must still be fixed, and this is the purpose of this paper that takes place in this scenario where we studied possible solutions for avoiding or better mitigating the misbehavior caused by sybil nodes. Particularly we considered an application context in which the P2P routing scheme is based on the classic routing structure of

* Tel.: +39 3920989482.
E-mail address: riccardo.pecori@unipr.it, riccardo.pecori@gmail.com

http://dx.doi.org/10.1016/j.comnet.2015.11.010
1389-1286/© 2015 Elsevier B.V. All rights reserved.

Kademlia [4], one of the most widespread DHTs (so much that it is used, for example, by eMule, BitTorrent, etc.).

Moreover, we did not try to prevent the creation of sybil nodes, rather, we supposed the environment to analyze is inherently infected by sybils, i.e., there is a subset of nodes that can present multiple identities. So, like in [8], we are not interested in a "clean network" but in a "trusted network", in which only the most trustworthy nodes should be used for both routing and storage and retrieval operations. A node routing table may and even needs to contain nodes it does not trust. The requesting node shall be able to decide on its own which nodes to trust in order to store its data or to retrieve data from, on the basis of a quality rating rooted in a trust score. Moreover, we allowed the decision, about which nodes are trustworthy and which are not, to vary between nodes, so we considered a local trust.

So, following this more conservative and robust approach, we re-adapted Kademlia standard procedures (both routing and storing) in order to take into account trust and reputation as well. This allows the reordering of the temporary search list and the final storing list in such a way that the sybils, if they are a limited percentage of all nodes, cannot completely degrade performances. Our solution is based on security measures that have proven themselves effective in other contexts and we combined these measures in a new way, applying them to a structured Kademlia network and making opportune adaptations. With our proposal we got a more reliable and secure Kademlia network, called S-Kademlia, which, even if not the optimum, makes the routing and storing mechanisms stronger against the presence of sybil nodes in the network.

To summarize, the main contributions of our work concern:

1. the evaluation, through a detailed survey, of both current and past anti-Sybil solutions,
2. the assessment of the degradation of Kademlia standard algorithm in presence of a growing number of sybil nodes with different malicious behaviors,
3. the proposal and evaluation of a new algorithm mixing the standard Kademlia routing and storage/retrieval procedures with a proper trust scheme, and
4. the comparison of this new algorithm with one already presented in the literature and based only on trust.

The rest of the paper is organized as follows: in Section 2 the background about the Sybil attack and possible countermeasures is described, in Section 3 there is a brief summary of the classic Kademlia routing and storage and retrieval procedures along with an analysis of sybils effects on them, in Section 4 we accurately present our integrated trust-based algorithm besides the trust management, Section 5 presents in details simulation results, whereas Section 6, in the end, seals up the work with some conclusions.

2. Sybil attack: rationale and countermeasures

Peer-to-peer networks usually rely on the existence of multiple, independent, and remote entities to mitigate the threat of bogus peers. The Sybil attack [7] takes advantage

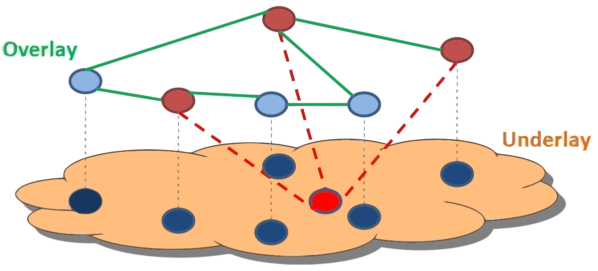


Fig. 1. Sybil attack.

of this feature, commonly present into P2P systems, as it is normally started by a faulty entity masquerading with multiple identities in the overlay network. The key idea behind the Sybil attack is to insert in the overlay malicious identities, the *sybils*, which are all controlled by one single physical entity (Fig. 1). This allows, provided that the sybils are positioned into strategic places, to gain control over a part of the P2P network or even over the whole network, to monitor the traffic or to abuse of the DHT protocol in other ways. This can lead to other types of attacks such as the eclipse attack [9]. As already stated in the previous section, we assume our scenario is inherently affected by sybils, so threats coming from a malicious behavior of these nodes are very probable.

The threats they can generate may be roughly divided into the following three categories [10]:

- routing attacks,
- storage and retrieval attacks,
- miscellaneous attacks.

The first group of attacks can take various forms such as incorrect lookup routing, incorrect routing updates for the routing tables and partition into an incorrect network. The second type of attacks can be represented by the following behaviors: denying actually stored data or pretending to have resources not owned, for example by providing fake resource data. The third kind is made up of different malicious attitudes that cannot be classified in the previous two points: inconsistent behavior, overload of target nodes, churning or sending of unsolicited messages are examples of these kinds of threats. In our work we only focused on incorrect routing attacks, partly studied in [11], and storage and retrieval attacks, leaving other attacks out of the scope of this paper for future work.

Currently no one has found a general and effective distributed solution to the Sybil attack despite many general approaches presented in the literature. Our purpose was, after analyzing the effectiveness of various proposed solutions, to devise an effective scheme applicable to a particular structured P2P network, Kademlia. So, first of all, we present in the following subsection a complete analysis of current known countermeasures to different misbehaviors that can be caused by a Sybil attack and then, in the following sections, we highlight a possible complete and effective countermeasure for Kademlia based upon trust and reputation.

2.1. Countermeasures to the Sybil attack

The current solutions to a Sybil attack, which can be found in the literature, can be roughly classified, according to [12], into four categories:

- trusted certification,
- resource testing,
- costs and fees,
- trusted devices.

This is a classification for solutions to avoid sybils inside a network, but as already stated in Section 1, we hereby consider a different point of view, therefore sybils are present in our network scenario. Considering the specific threatened DHT operations, anti-sybils solutions may also follow an alternative classification, such as the following:

1. usage of diverse routing schemes (against routing attacks),
2. limitation on the number of peers that a physical node can admit to the network (against join attacks),
3. verification of some types of requirements, such as computational resources (against join and store attacks),
4. periodical refresh of the routing tables or of the IDs (against routing and storage/retrieval attacks).

The presence of a centralized Certification Authority (CA) is also claimed to be one of the best solutions, if not the only one [7]. The whole analysis of countermeasures as well as their advantages and drawbacks is investigated in Table 1. The study is done according to different levels: considering the specific threatened DHT operations (mainly $find\_node/find\_value$ and *join* procedures), then the malicious activities involved and then the general approach of the solution. Finally a brief description of the solution along with some advantages and drawbacks is given.

In the following a brief summary of the findings highlighted in Table 1 is provided.

As concerns the DHT $find\_node/find\_value$ procedure, it may result vulnerable to incorrect routing attacks or incorrect routing update attacks. The possible solutions [13] regard the usage of redundancy techniques, where one can find diverse routing strategies, not based upon the classic closeness routing, or techniques that exploit further constraints, such as the number of intersections of random routes, failure tests on the routing primitive, etc.

Regarding incorrect routing updates, instead, the possible solutions [14] are based upon a general resource testing scheme, upon redundancy or upon trusted certification. For the first countermeasure identity legitimacy controls can be enacted through messages sent in pre-assigned channels or through connectivity degree of a node. For the redundancy solutions more than one routing table can be used and finally for the trusted certification the possession of some cryptographic material. A specific malicious activity against the $find\_value$ DHT procedure is the storage and retrieval attack. Against this type of attack one possible solution can be the testing of resources, for example through storage or computation operations [7].

As regards the *join procedure*, it is usually attacked through the creation of false identities in the overlay network. The possible countermeasures are limitation-based, trust-certification-based, resource-testing-based and cost-and-fee-based. Concerning the first ones, examples can be found in limiting the join permission for active peers, for the second one in the presence of central authorities issuing identities, maybe with the help of certificates or through different channels. Regarding the second type of countermeasures also the periodical invalidation of IDs by a trusted entity or the registration of a new ID by other nodes may be considered. For the third type of countermeasures, resource testing, there can be mechanisms that provide an admission control, hierarchical or distributed, with puzzle challenges, or that bind the virtual ID to real world identities. Finally “costs and fees” countermeasures are based upon certificates that must be paid with real money.

After this analysis we propose a distributed solution scheme, in order to maintain the inherently non-centralized nature of the P2P networks, based upon trust and reputation according to some successful suggestions found in [22]. Refs. [23,24] are frequently called upon as the best schemes, while [25] limits the application of trust to P2P file sharing in unstructured networks. The solution in [26] operates in structured networks but does not have an implementation in the real world. In [8] the author presents a first approach of using a trust metric within the Kademlia routing process, whereas in [27] he extends his ideas also to the store/retrieval process, making, however, a distinction between trust for LOOKUPs and for GET and PUT operations. Another work very close to ours is [28] where, nevertheless, certificates and private and public keys are used. Our idea is simpler in the formulas computing the trust and does not need the usage of any thresholds or Certification Authorities. The ideas presented in [27], moreover, introduce, as stated also by the author, a lot of overhead and still lack of simulation results and analysis. Anyway, the first papers, at the best of our knowledge, trying to use trust and reputation for solving routing and storage attacks caused by sybils in Kademlia are given in [8,27], and for this reason we will compare our algorithm to the ones presented in these works. Our balanced trust-based scheme is adapted to the classic Kademlia algorithm (both for routing and for storage and retrieval) and, more important, it is a completely P2P solution, in order to maintain the inherently not centralized nature of Kademlia. As regards trust and reputation we took advantage of some successful suggestions found in [22,29–31]. In this paper we also compare our balanced trust-based algorithm with the one in [27] and show that it reaches better performances in terms of successful operations in a similar risk environment.

3. Consequences of a Sybil attack in Kademlia

In this section we deeply analyze the effects sybils cause in a Kademlia network. In the following we first briefly recall Kademlia classic routing and storing procedure [4], then, by means of our simulation results, we analyze how Kademlia performances degrade in a Sybil scenario, showing the effects of these malicious nodes on the standard Kademlia algorithm.

Table 1
Analysis of the main methods to defend networks against sybil behaviors.

DHT opers.	Mal. activity	Solut. type	Protection technique	Advantages	Drawbacks	Ref.
			Diversity routing: ID lookup trust profile (histogram representing nodes appearance frequency on the lookup path towards the ID)	Adversary nodes introducing a lot of sybils may acquire high values in the trust profile, so nodes behind them will not be used	Inefficient under normal circumstances and no progress toward target node	[13]
			Mixed routing: a balance between classic routing and diversity routing	Balance between standard and diversity approach	It does not perform better than the diversity strategy in case there are many more sybils than honest nodes	[13]
Find node/find value	Incorrect routing	Redundancy-based	Zig-zag routing: alternation of diversity and classic routing	Outperforms standard and mixed routing when the number of malicious nodes is large	Reduced lookup speed	[13]
			Secure routing primitive: failure test and redundant routing to ensure, with high probability, that at least one message arrives at each correct replica root for a key	Solves a constrained routing problem: even a few malicious nodes can reduce the probability of successful delivery	Several attacks have been proven to weaken the failure test	[14]
			Random routes with a predefined number of hops; identity verification based on two random routes; check of number of intersections	No centralization elements; trust and reputation policies are applicable due to the character of nodes relationships (social network)	Routing loops reduce the effective length of the routes and the probability of intersections; small number of attack edges assumed; honest users may be compromised; the number of hops as critical parameter	[15]
			Use of 2 routing tables: one exploiting network proximity, another exploiting a constraint on closeness	Constrained routing tables have an average fraction of only a few random entries pointing to attacker-controlled nodes	Routing overhead, an attacker can reduce the successful delivery probability not forwarding messages; no flexibility in neighbor selection	[14]
		Redundancy-based	Periodic reset of optimized routing tables to constraint tables; tables updates rate limitation; unpredictable ID assignment.	Adversaries must make more intensive their activity to maintain their foothold.	Short-term effect; only Bamboo tested; effective at low routing tables poisoning level; system adaptability and routing security trade-off ; randomness server availability.	[16]
Find node/find value	Incorrect routing update		Identity legitimacy by sending check messages at time intervals on channels pre-assigned to neighbor nodes.	No centralization elements involved.	A node cannot communicate simultaneously on more than one channel.	[17]
		Resource testing	Node degree bounded to a threshold: nodes anonymously audit the connectivity of each other through certificates binding ID and public key.	Flexibility in the selection of neighbors nodes.	Mainly addressed to eclipse attack; mechanisms for maintaining auditor anonymity required; localized eclipse attacks could work.	[9]
		Trusted certif.s/devices	Random assignment of key-sets to each node; sybils detection checking the number of coinciding keys in sets of different nodes.	Distributed control mechanisms.	Possibility to stole IDs (masquerading attacks).	[17]
Find value	Storage and retrieval attack.	Resource testing.	Resource control through simultaneous check of minimum resource level: communication, storage and computation operations.	No centralized elements.	It is not practically feasible in large-scale distributed systems.	[7]

(*continued on next page*)

Table 1 (*continued*)

DHT opers.	Mal. activity	Solut. type	Protection technique	Advantages	Drawbacks	Ref.
			Hierarchy of cooperative admission controls based on the bootstrap graph. Challenging new nodes by puzzles solving in order to have a global cryptographic proof. Upper bound on the number of IDs.	Distributed access control; more difficult targeted attacks; easy malicious node removal by pruning; easy attack localization.	Simple Diffie–Hellman used to share a secret in the hierarchy: possible man in the middle. Computationally expensive signature verification. Tree root as single point of failure.	[18]
Join	False ID creation	Resource test	Requiring the prospective (joining) node to solve crypto puzzles.	Distributed access control.	Costs trade-off in the solving of puzzles: not too expensive for good nodes but enough to prevent attacks.	[14]
			Binding node IDs to real-world identities.	Effective in “virtual private” overlays (enterprise networks).	CA is a single point of failure.	[14]
			Random computational puzzles, with incorporated distributed locally-generated challenges, required to participate in the network; challenge broadcasting from each peer toward others.	Locally challenges prevent puzzles to be reused by attackers over time; gives a node flexibility to choose its position.	Opposite effect when legitimate users are stuck with out-of-date hardware whereas attackers can rely on high performance computers.	[19]
		Constraint	Limited join permission by active P2P network nodes.	A limited possibility for introducing sybils into the network through the same bootstrap-node.	Introduction of additional rules into the classical DHT model.	[13]
			Release of new identities by CAs.	For the moment, it is the only way to tackle the Sybil Attack effectively.	Introduction of centralization elements into the system; probability of single point of failure.	[7]
Join	False ID creation	Trusted certification	ID obtaining through double-key encryption of newcomer’s IP address by a central agent returning ID via SMS.	Significant reduction in the number of sybils.	Single point of failure; different SIMs use possible; PKI elements introduction; IP addresses and phone numbers list; central agents to avoid multiple IDs issue to same peer, additional memory resources and costs.	[20]
			Registration through a centralized authority, list of trusted identities.	Secure bootstrap.	Single point of failure, introducing centralization elements, stolen Ids.	[17]
			Central trusted certification authority assigning and signing *nodeId* certificates that bind ID to a public key and to the IP address.	Randomly chosen *nodeIDs* and ID forging prevented; binding IDs and IP addresses makes it harder for an attacker to move ID across the nodes.	Single point of failure; if an IP address is changed, the corresponding *nodeId* and certificate become invalid.	[14]
		Central. trusted certif.	Periodical invalidation of node IDs by a trusted entity broadcasting a different initialization vector for the hash computations.	It is hard for an attacker to accumulate many node IDs over time and to reuse them.	Legitimate nodes must periodically spend additional time to maintain their membership.	[14]
Join	False ID creation	Distributed trusted certif.	Registration of the ID (IP address and port hash) of a new node by other peers according to IP address prefix; bound to the maximum number of nodes per participant.	External ID usage, no centralization elements; no need to obtain new separate ID; easily applicable to other DHTs besides Chord.	Distributed ID assignment, Sybil-proof only for a limited time; only one node behind a NAT; nodes can obtain a huge number of IPv6 addresses.	[21]
		Costs or fees	Certificates binding the ID to a public key should be paid.	The cost of an attack grows with the size of the network; fees are supposed to fund the operations of the CA.	Single point of failure.	[14]

3.1. Kademlia lookup procedure

In this work we consider both the iterative and the recursive [32] routing of Kademlia. More precisely we consider only iterative Kademlia when we analyze routing algorithms, whereas we examine both when we consider the storage and retrieval process as, in this case, it does not matter whether the Initiator of a lookup request has or not the control over the whole routing process. As a matter of fact, in case the focus is on the storage, the only important thing is to reach a final list of locations for the resource to be stored in or to be retrieved from, whereas in the case of routing it is important, in order to correctly apply our balanced trust-based algorithm, that the Initiator has the control on the local temporary research list at each step of the research. In the iterative routing of Kademlia, the answer of a resource query, launched by an Initiator, is a list of nodes to ask at the next step until the node responsible for the searched resource is found. This list is then sent back to the Initiator at each iterative step by the queried nodes. Therefore, the Initiator is responsible for all the queries in the lookup process and it has also the full control over the whole procedure. On the other hand, in recursive Kademlia, also called R-Kademlia, the routing process is in charge, step by step, of the previously queried node. Only the final list, containing the closest nodes to the target resource key, is sent back to the Initiator that does not have full control over the whole routing process. In the following we briefly recall both the iterative and recursive standard routing and the subsequent standard storage or retrieval process.

Kademlia nodes store contact information to route query messages, keeping lists of (IP address; UDP port; Node ID) triples for nodes at distance from themselves between $2i$ and $2i + 1$ and for i ranging from 0 to 160 excluded. These lists are known as $k - buckets$, where k is both the length of the lists and a system parameter chosen such that any given k nodes are very unlikely to fail within an hour of each other (usually $k = 20$). When trying to store or retrieve some resources, the requesting peer, the Initiator, runs α parallel asynchronous queries, called *lookups*, for a given resource key at the same time, where α is another system parameter, usually set to 3. The α peers are chosen by the originator peer (we call both the actual Initiator or the intermediate peer in charge of forwarding the lookup request recursively thus) according to a certain policy among those k peers closest to the targeted resource; these peers are stored in a temporary list to be updated by the originator at each step, in the case of iterative Kademlia, or they are simply received by the current in charge peer, in the case of recursive Kademlia [32]. Each of the α peers contacted responds to the querying peer with the k nodes closest to the target resource key according to their peer routing tables. The querying peer can have, as a response, at most $\alpha \times k$ nodes; among these it selects for the next step only the k closest to the searched resource. Both the iterative and the recursive lookup processes end up when at step n, among the at most $\alpha \times k$ responses, the querying peer is unable to find at least one peer closer to the resource than any other already present in the temporary list updated at step $n - 1$. At this point the Initiator perform a GET or PUT operation on the closest nodes retrieved. In the case of a PUT operation this also guarantees redundancy, and therefore robustness to the P2P network.

3.2. Simulation conditions

In this section we briefly summarize the main conditions of our simulations and the tool we deployed to study a Kademlia network under a Sybil attack. As a simulator, we employed DEUS [33], a discrete event simulator, where simulation time is sampled in virtual seconds (vs). DEUS is developed in Java language by the Distributed System Group of the Information Engineering Department of University of Parma. Among all its capabilities, we obviously focused on its embedded Kademlia implementation, using opportunely modified XML simulation scripts, in order to allow for trust and reputation management. For simplicity's sake, we worked in conditions of steady state network so that, despite nodes joining and leaving, a constant number of nodes, on average, is always present in the network; this is obtained starting from approximately 10,000 vs from the start of the simulation. Particularly the number of good nodes, that is, nodes that behave correctly according to the standard Kademlia algorithm, is always 100 on average, whereas the number of bogus nodes, the sybils, varies from simulation to simulation: 0, 2, 10, 50, 100, and 200 on average. 100 and 200 sybils, even if unrealistic, were considered to study the performances of the network and of our balanced trust algorithm in conditions of great stress and in environments with an overwhelming probability of attack. Peers join or leave the network till about 10,000 vs according to a periodic process of period 100 vs for good nodes and random period for malicious nodes respectively. Other parameters of the Kademlia network are α and k. k is always equal to 20 as suggested in [4], while α, as it influences only routing operations, was fixed to 1 in evaluating storage and retrieval attacks, whereas it is a variable parameter when evaluating routing attacks. The key space size was set to 2000. In DEUS simulation model of Kademlia another parameter is present, namely *discoveryMaxWait*, fixed to 1000 vs; this is needed to simulate nodes answering too slowly. In our considerations we will only focus on correctly ended value lookup procedures that start only when the network is steady and with no expiration of the *discoveryMaxWait* parameter; if *discoveryMaxWait* expires the requests are labeled as "unended". Finally, we considered average values over 200 seeds, for each simulation set of parameters, in order to achieve a 94% confidence interval on the results.

In this work we were interested into both routing and storage or retrieval attacks coming from the sybils, that is why we considered various bad behaviors in our simulations: both as a response to a *PUT*/*GET* operation and during the routing of lookup requests. As regards routing, two malevolent behaviors were considered: sybils asked for a next hop can answer back with an empty list or they can respond with a list made of randomly chosen peers. These are only some of the possible malicious behaviors and they were selected as representative of routing attacks and to make some general considerations on Kademlia performances and on the effectiveness of our proposed trust-based solution. Considering storage and retrieval, three malicious behaviors were mainly considered: one regarding a PUT operation and two concerning a GET operation. In the first case the sybils asked of storing a certain resource refuse to do so, in the second case, those sybils requested for retrieving a certain resource respond with a null value or a corrupted resource (fake resource). It should also be noted that these are only some of the possible malicious behaviors; they were selected as the most representative of a storage and retrieval attack (see Section 2). Furthermore, considering always PUT and GET procedures, we briefly investigated also gray-hole attacks to assess our trust-based scheme in presence of more sophisticated onrushes. As highlighted also in Section 5.2, we took the following misbehaviors into consideration: the acceptance of a PUT and the subsequent refusal to a GET, and a selective refusal of PUTs and GETs on the basis of the target resource.

3.3. Effects of sybils on standard Kademlia

In spite of various metrics, which can be used to evaluate a Sybil attack [34,35], we have concentrated mainly onto the analysis of the following features:

- as regards the *routing process*, the percentage of filling of the *k*-buckets averaged over all the nodes in the network and the average number of successful lookup procedures;
- as regards the *storage and retrieval*, the average number of successful PUT and GET procedures over all the PUT and GET operations carried out, respectively.

As successful lookups we considered researches arrived correctly at the nodes closest to the target resource in a $find\_node$ Remote Procedure Call (RPC), whereas for a correct PUT operation we took into account *store* RPCs where all final nodes accepted the storing request. As regards successful GET operations we considered $find\_value$ RPCs where the retrieved resource is genuine. As incorrect or malevolent behaviors we considered the following:

- for the *routing processes*: nodes responding with a null or a random list;
- for the *storage process*: nodes refusing to accept and store incoming resources for which they should be responsible in the overlay. In this case sybil nodes perform a sort of denial of service attack [36].
- as regards the *retrieval process*: nodes responding with null or not genuine resources.

For our first set of simulations, we focused on the routing process, so in Figs. 2 and 3 we analyzed *k*-bucket filling and the number of successful lookups. The results depicted in the aforementioned figures refer to sybil nodes responding with a null list and show the dramatic decrease of performances, in both analyzed metrics, as the number of sybils increases. In these graphs the α parameter is fixed to 1 so no parallelism is exploited.

In Fig. 2 the percentage of *k*-bucket filling is not particularly affected by a small number of sybils (namely 2), and this confirms [37] in saying that there is a minimum percentage of sybils in the routing tables to make a Sybil attack effective. However the filling of *k*-buckets drops for a number of sybils greater than 50 nodes, reaching a level close to zero when the number of fake IDs is twice the number of the true ones. In Fig. 3 the number of unsuccessful searches is somehow limited considering a number of sybils less than 10, while it significantly increases and overtakes successful searches from

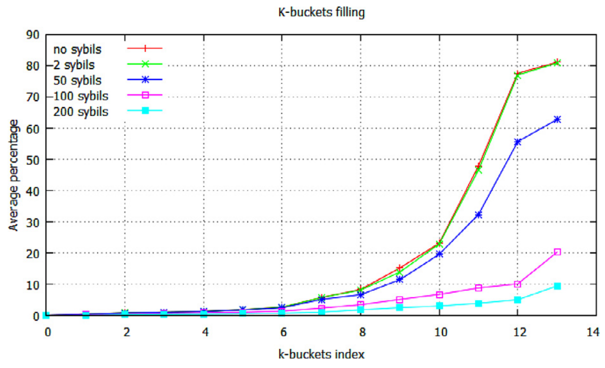


Fig. 2. *K*-buckets filling versus *k*-bucket index with variable number of sybils in the network.

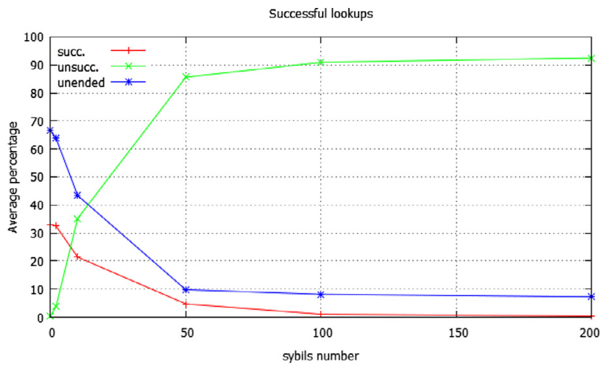


Fig. 3. Successful, unsuccessful and unended lookup procedures with sybils answering with a null list and $\alpha = 1$.

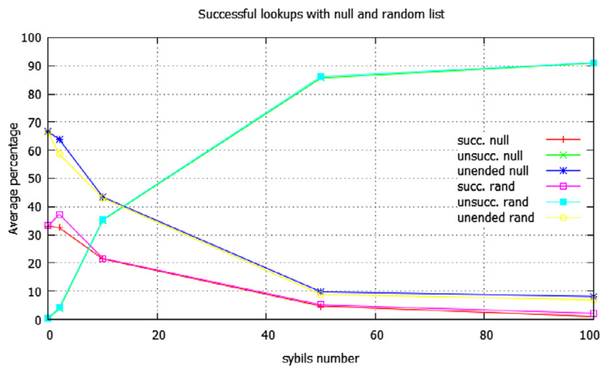


Fig. 4. Comparison between two malicious behaviors in routing process.

10 on. The decrease in not ended searches with an increasing number of sybils is also relevant: probably the more the sybils, the more their ability to divert correct ongoing searching paths to dead points.

In Fig. 4 we compared the performance degradation when the malicious behaviors take place in a look up procedure:

1. the case in which sybils answer with a null list of nodes or
2. with a random list.

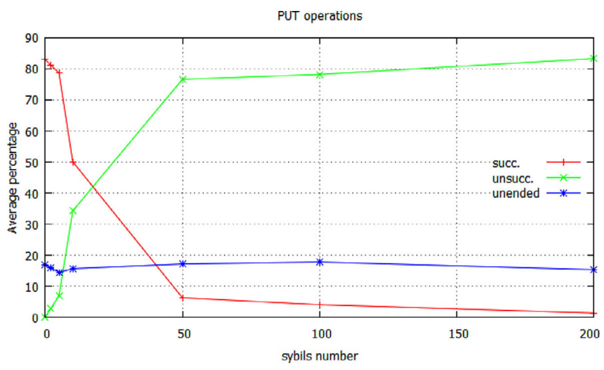


Fig. 5. Successful, unsuccessful and unended PUT procedures with sybils performing a DoS attack.

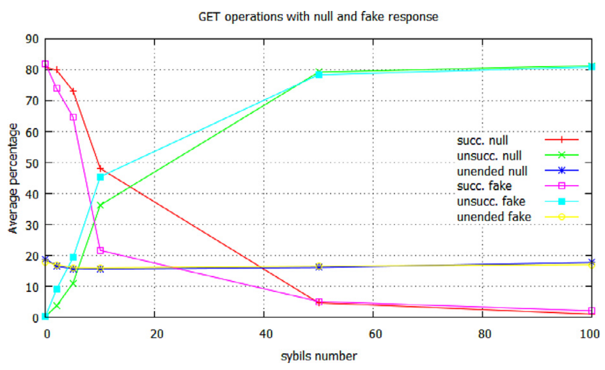


Fig. 6. Comparison between two malicious behaviors in response to a GET request.

From Fig. 4 we can notice that the difference is very slight. This could be explained considering the functioning of Kademlia standard algorithm as recalled in Section 3.1. In case a node answers with a *null* list the first node of the temporary list does not surely update, therefore this behavior forces to go into the searching *k* nodes phase. From this point the procedure, as described above, could stop or proceed with only one step more than if no malicious nodes where found on the path. The same happens if we consider a malicious node answering with a random list even if in theory there could be the chance for the local list to acquire a new first node anyway. This could explain why the graphs in Fig. 4 are very similar.

As a second set of simulations, we studied the effects of sybil nodes on the PUT and GET procedures. In this case the α parameter was fixed to 1 as it does not influence storage or retrieval but only routing parallelism. The results depicted in Fig. 5 refer to PUT operations, refusing to store the resource. Our results show the impressive decrease of successful storage operations as the number of sybils in the network increases. Anyway the number of unended procedures is almost constant, as we did not considered malicious activities of the sybils in the routing process in this case. Moreover, in Fig. 5, we verified that, also in this case, a small number of sybils (namely 2 or 5) cannot significantly affect the performances, and this confirms the findings in [37] yet again. Anyway the performances decrease, also in this case, for a number of sybils greater than 50 nodes, reaching a level close to zero when the number of fake IDs becomes preponderant.

Given these conditions, the number of unsuccessful PUT requests is somehow limited considering a number of sybils less or equal to 10, while it significantly increases and overtakes successful searches from a certain value, little greater than 10, onwards. The saturation behavior the curves experience when the number of sybils is greater than 50, besides the constancy in not ended operations with an increasing number of sybils, is also relevant. This effect, present also in Fig. 3 for the routing process, can lead to think there is a sort of threshold in the number of sybil nodes, after which the negative effects on the network are ensured. This threshold surely depends on the number of nodes in the network and the relative fractions of good and bad nodes.

In Fig. 6 we studied the difference in performance degradation between the main considered malicious behaviors in a GET procedure:

1. the case in which sybils answer with a null value or
2. with a fake value.

From Fig. 6 we can see that there is a very slight difference in the two cases for a large number or a very little number of sybils in the network. On the contrary, considering a number of sybils ranging from 2 to 10 the difference between the two curves enlarges. This can be explained with the following considerations. In case a node answers with a null value the originator node has the ability to perform a second request on the Kademlia network and therefore, in this second case, it may succeed in reaching a not null resource with a greater probability. In the other case, the originator cannot verify whether the retrieved resource is genuine or not and so it does not perform a further GET request on the network. This leads to a worse degradation of the overall performances, but this effect disappears when the number of sybils becomes extremely preponderant in the network, and the probability of obtaining a genuine resource performing a second GET request drops dramatically. These considerations could explain why the graphs in Fig. 6 are very similar when the number of sybils is very small or very large and why there is a significant difference when their number ranges from 2 to 10.

After these considerations we addressed towards the research of some solutions, or better, some ways to mitigate the effects of malicious nodes on both the LOOKUP and the PUT or GET procedures. With this aim in mind we focused on a solution concerning trust and reputation like in [31] or other solutions present in the literature; this seems a suitable technique in P2P networks, as, like in any human community, nodes interact with each other, create new contacts, and progressively gain their own experience and reputation about each other, but, at our knowledge, this was applied to Kademlia networks only in [8,27,38].

4. S-Kademlia: balanced trust-based DHT algorithm

We propose to improve Kademlia resilience to incorrect operations caused by sybils, introducing trust in the standard Kademlia algorithm. Our approach has been studied from several different viewpoints: we applied trust both to

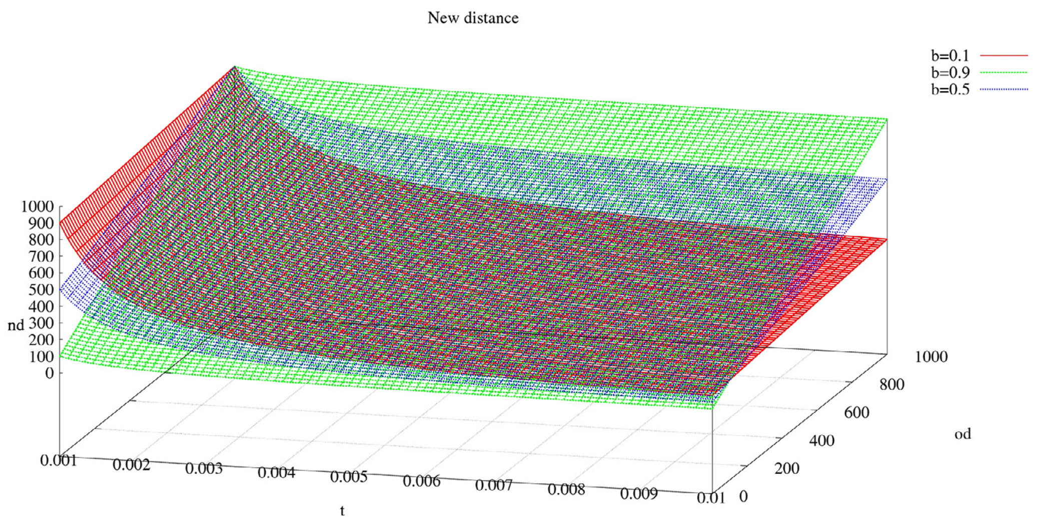


Fig. 7. New distance function.

the iterative routing process and to the storage and retrieval process, in this case independently whether the iterative or recursive scheme is employed. The main concept is taking a trust value into account when ordering the local temporary routing list or the final list of k peers where to store or retrieve the searched resource. Anyway, differently from [8], we thought that the use of a brute force ordering of such lists just on the basis of trust could not work or work very worse than standard Kademlia, maybe protecting from some consequences of a Sybil attack but resulting not convenient. So we tried to define a new metric, with which ordering the aforementioned lists, that takes advantage of some interesting results in load balancing [39,40]. We called it "new distance" (nd), and computed it according to Eq. (1):

$$nd = od \times b + (1 - b) \times \frac{1}{t} \qquad (1)$$

where nd is the new distance that taking trust and reputation into account as well, od the old distance computed according to the XOR operation as it is the case with standard Kademlia, b a balancing factor ranging from 0 to 1 and t the reliability factor ranging from a very small number near zero (say ϵ) and infinite. The nodes having the highest new position, or the lowest new distance, which amounts to the same, are chosen as the candidates for storing or retrieving the resource. The function representing the new distance is plotted in Fig. 7 with different values of b. The aforementioned figure shows the two different dependences: linear on the old distance and hyperbolic on trust. The balance factor b can be used to aptly tune the weight of the old distance or of trust.

The new balanced distance, as already stated, can be applied either to the routing or to the storage and retrieval process. In the following we describe the new S-Kademlia algorithm, from both the routing point of view and the storage viewpoint.

Speaking about routing we considered, as already stated, only iterative Kademlia. So at each iterative step the Initiator of a *lookup* chooses the nodes having the smallest new distance as the candidates to be queried for the next hop. The new balanced routing algorithm can be seen throughout the following steps:

- *step 1*: the Initiator of the search, looking for a determined resource, chooses α nodes to query among the k nearest (according to the *old distance*) he knows;
- for the *following steps*, each chosen node undertakes the same procedure. It chooses other α nodes for the next step and sends back to the Initiator their addresses to enable it to contact them. At each step of our balanced algorithm the Initiator orders the local temporary search list with the first position represented by the node with smallest new distance among the k possible ones obtained from one of the α nodes of the previous step. In case the nd values are all the same the choice could be done according to a different policy, e.g., randomly.

Considering storage and retrieval, we considered both iterative and recursive Kademlia. The storing or retrieval algorithm can be described as follows:

- *step 1*: the peer currently in charge of the search, looking for a determined resource to retrieve or to put, gets the final list of peers, closest to the correct position of the resource according to standard Kademlia algorithm. Nodes in this list are ordered according to a certain position according to the *old distance* metric;
- *step 2*: the in charge peer reorders such a list on the basis of its local trust assigned to each peer present in the final list. This reordering is originator-dependent as, in the case of iterative Kademlia, it is the actual Initiator of the *find_value* or *store* RPC to be in charge of reordering the list according to its local perceived trust into other peers, whereas, in the case of recursive Kademlia, it is the final peer in charge to reorder the list before performing the PUT or GET operation. For a PUT operation only peers at position greater than a certain threshold are chosen, in order to guarantee the intrinsic redundancy of Kademlia, whereas for a GET operation only peers at the very first position are elected.

4.1. Trust management

Trust information, in our model, is computed according to the definition found in [31] rather than the one in [27] or [8]. That is, we accounted for an environment risk factor in case no interactions with nodes were previously enacted, in order to avoid the grace phase of [8], and we gave much more importance to direct experience than to recommendations, which are not considered in our model. The trust factor, in its whole form, is an opportune combination of risk and reputation, performed with the contribution of numerical weights. Similarly to PET we considered the trust factor (t) defined as follows:

$$t = W_{R_e} \times R_e + W_{R_r} \times R_r \tag{2}$$

where R_e and R_r are respectively the reputation and the risk, whereas W_{R_e} and W_{R_r} are respectively the weight for the reputation and for the environment risk (values ranging from 0 to 1). We hereby simplified the PET model considering only one bad service level (L = low grade). Our computation of the risk R_r, e.g., the level of vulnerability of the network, is therefore reduced to the following formula:

$$R_r = \frac{N_L h_L}{h_L N_T} = \frac{N_L}{N_T} \tag{3}$$

N_L is always the number of low-grade interactions experienced till now in the network, but it encompasses different behaviors according to the three scenarios we considered:

1. applying trust *only to the routing process*: in this case N_L regards lookup timeout, resource not reached, etc.;
2. applying trust *only to the storage or retrieval process*: in this case N_L considers storage refusal, null retrieved resource, etc.;
3. applying trust *both to the routing and to the storage or retrieval processes*: in this case N_L is made by all the aforementioned malicious responses.

h_L is the PET map function for level L (but as seen in Eq. 3 it is inconsequential in our simplified model), whereas N_T is the total number of requests launched by the peers in the network (only *find_node* for the first case, only *find_value* and *store* for the second case and all possible RPCs for the third case).

The evaluating node stores in a local table different trust values for each node of the P2P network; in case no interactions with a certain node have been yet experienced the value of t is simply represented by the risk part $W_{R_r} \times R_r$; otherwise t also comprises the reputation part that is updated as follows:

1. in case only routing operations are considered: $+\frac{1}{N_p}$ to each node present in paths leading to the correct resource and $-\frac{2}{N_p}$ to the nodes of incorrect paths,.
2. in case only storage or retrieval operations are considered: $+\frac{1}{N_p}$ to each peer accepting a PUT or responding with a correct resource and $-\frac{2}{N_p}$ to the peers denying a PUT or responding to a GET request with fake or null resources.
3. should the trust be applied to all Kademlia RPCs, t update is given by all the aforementioned considerations.

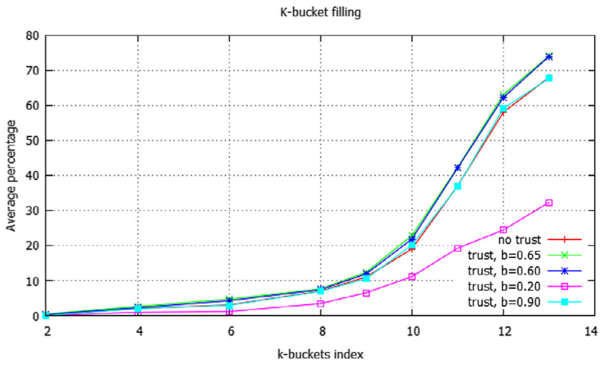


Fig. 8. *K*-buckets filling versus *k*-bucket index for trust and no trust algorithms for 50 sybils network and variable value of balancing factor.

N_p is the total number of requests (LOOKUPs in the first case, PUTs or GETs in the second case and all in the third case) received by the considered peer. We do not consider unchoking but, similarly to [8], trust values expire after 24 h in order to avoid that a malicious node may gain too much trust through collusion operations. As regards the routing part, the negative reputation update may concern only the last node, or a limited subset of nodes in the path anyway, in order to take into consideration the possibility that not all the nodes in a wrong path behaved maliciously.

5. Simulation results

In this section we present the results of our analysis of the performances of our new balanced trust-based approach over Kademlia with sybils. The majority of the parameters and of the simulation conditions are the same as the ones in Section 3.2. The value for the parameter b is critical and very important to opportunely tune the performances of our balanced trust-based algorithm. Its optimum value, obtained after a great simulation campaign and averaged over the malicious behaviors considered, is 0.65 as regards only routing operations, whereas it is 0.55 when we considered only storage and retrieval operations. In both situations W_{R_r} was set to 1 for a node joining for the first time and 0.5 for an already joined node, whereas W_{R_e} is 0 for new joining nodes and 0.5 for already joined ones. R_r is positive for new nodes that join the network. In these conditions we analyzed the routing and the storage and retrieval processes separately with our balanced trust-based algorithm.

5.1. Trust applied to routing processes

Considering routing operations, in Fig. 8 the k-bucket filling versus k-bucket index is analyzed considering 50 sybils in the network. Several values of the balancing parameter b were used and, as it can be inferred from the aforementioned figure, the optimum value is about 0.65, whereas a value of b closer to 1 leads the performances close to the ones of a network infected by sybils with no protection and a value of b close to 0 degrades the performances even worse than the presence of sybils. In the latter, case the results can be explained as the routing is done mainly according to the trust

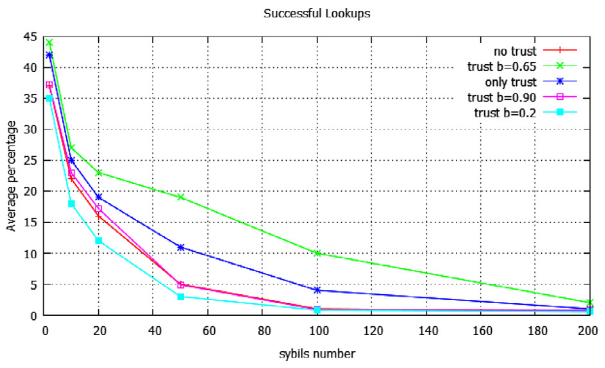


Fig. 9. Comparison of successful lookups for standard Kademlia, only trust method and our balanced routing method with variable balancing factor.

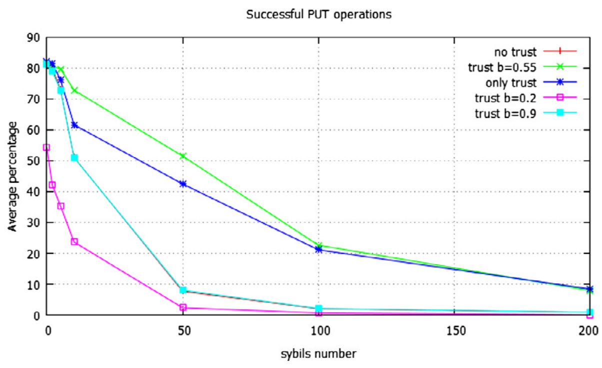


Fig. 10. Comparison of successful PUTs for standard Kademlia, only trust method and our balanced trust method.

metric rather than the standard Kademlia algorithm, which is the optimum.

On the other hand, in presence of a relevant number of sybils, our approach mitigates their negative effect even, albeit not reaching the performances of standard Kademlia algorithm with no sybils. This is evident also in Fig. 9 where we see how our balanced routing partly alleviates the negative effects of sybils on successful lookups, especially when the number of malicious nodes is less than 200. In the same figure we can see that a not optimized value of *b* may even help sybils in degrading the performances as it happened in the *k*-bucket filling. Moreover, always in Fig. 9, we compare our approach with the one described in [8] or [27] (labeled as "*only trust*") showing that a balanced approach is better than a pure trust scheme.

5.2. Trust applied only to storage and retrieval processes

As regards storage and retrieval operations we ran different sets of simulations for PUT and GET operations and for iterative and recursive Kademlia. The average optimum value for the *b* parameter is in this case slightly smaller than in the routing process, namely 0.55. This value was achieved making an average, over a great simulation campaign, of the main considered malicious behaviors: denying to store a resource and responding with a null or fake resource. Another important finding is that its value is independent from considering iterative or recursive Kademlia routing algorithm.

In Fig. 10 we see how our balanced storing algorithm is effective in the case of PUT operations as well, also in this case when the number of malicious nodes is far less than 200. Moreover, in the same figure, we compare our approach with the one described in [27] showing that a balanced strategy is better than a pure trust method in this case as well, especially when the number of sybils in the network is not preponderant. In this figure we considered successful a PUT request not denied but also correctly assigned to the node being theoretically right in charge. This is the reason why a preponderant value of trust leads to unwanted results and so to unsuccessful PUT operations.

Similar considerations can be drawn from Fig. 11, where we compared our approach and the only trust-based one in the case of GET operations with sybils responding with a fake resource, itself the worst scenario as explained in Section 3.3.

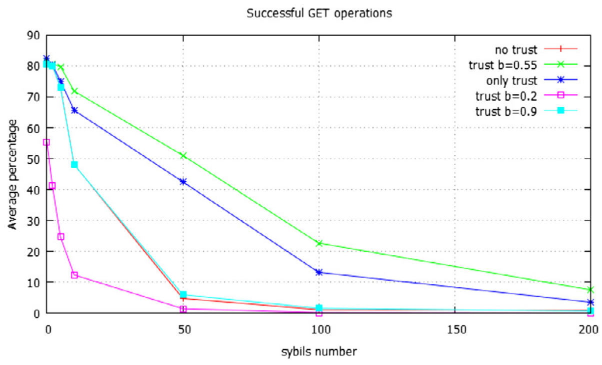


Fig. 11. Comparison of successful GETs for standard Kademlia, only trust method and our balanced trust method.

Furthermore, we noticed that, differently from the PUT operations, a better performance of our algorithm, compared to the only trust-based one, occurred also in presence of a great amount of sybils within the network.

Moreover, in this scenario, we tried to investigate some more sophisticated types of attack coming from the sybils further. In particular, the gray-hole behaviors considered are the following:

1. accepting a resource so that the PUT is successful and then refusing retrieval operations on that resource;
2. blocking the PUTs and GETs only as regards particular contents or resources.

The first behavior, intuitively, does not affect the success of PUT procedures so we focused on successful GET operations. In this case the performances of our trust-based approach may decrease as shown in Fig. 12, but it strongly depends on the number of retrievals compared with the number of PUTs, and on the number of sybils in the network. Considering *r* as the ratio of PUTs over GETs, in a fixed time interval (namely 10,000 *vs* in our simulations), we can notice from the analyses that its variation may lead to worse performances for our balanced algorithm. This occurs mainly when *r* is greater than 1, i.e., the number of PUTs overtakes the number of GETs, and with a number of sybils greater than 10. When the value of *r* is less than 1, the degradation in the performances of our algorithm starts later, when the number of sybils in the network gets greater than 50. Finally, we

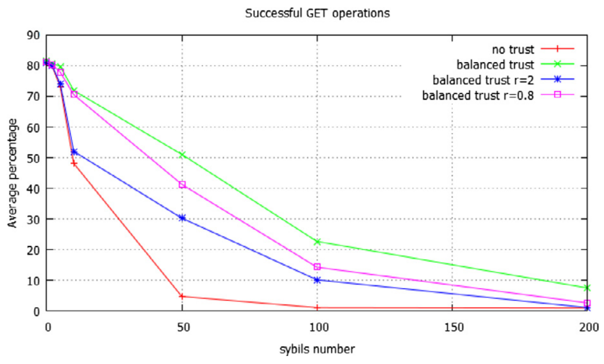


Fig. 12. Successful GETs according to standard algorithm and balanced trust algorithm and variable ratio r between PUTs and GETS, with b = 0.55 and gray-hole type 1 attacks.

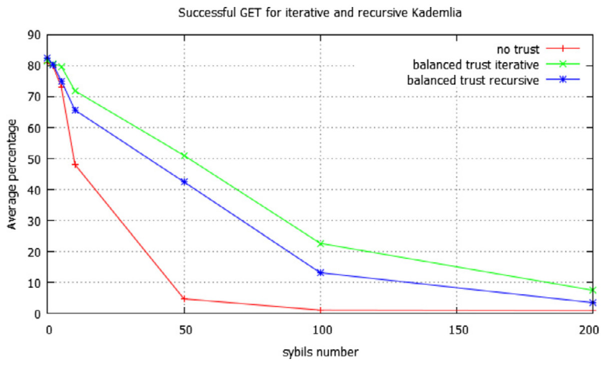


Fig. 14. Successful GET operations in Kademlia according to standard algorithm and balanced trust algorithm (both iterative and recursive) with b = 0.55.

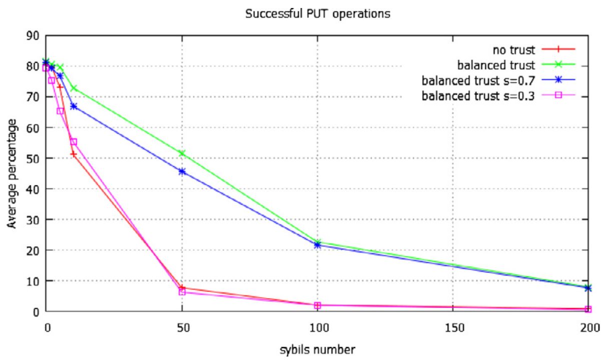


Fig. 13. Successful PUTs according to standard algorithm and balanced trust algorithm and variable popularity ratio s, with b = 0.55 and gray-hole type 2 attacks.

observe that when the number of sybils is not preponderant, the value of r is not so significant and the performances are similar to the case when no sophisticated attacks are enacted. In this situation, a possible way to soothe the attack may be to change a little the trust management: for example updating the trust value with $-\frac{3}{N_p}$ for nodes accepting a PUT and afterwards denying the possession of the target resource.

The second gray-hole behavior, always concerning storage and retrieval operations, may affect both PUTs and GETs. Anyway, the effectiveness of such an attack is to evaluate on the basis of the diffusion of the considered content. If the content is very popular, the attack may be thwarted through our proposed balanced trust-based algorithm with no changes, whereas in case the target resources are not so much widespread, the performances of our algorithm for the target resources decrease. This can be clearly seen in Fig. 13, where successful PUTs are depicted in the aforementioned cases. The spreading factor s is a measure of the popularity of the target resource, the only one considered for PUT procedures in those curves where s is specified in the key. s is defined as the ratio between the PUTs for the target resource and the overall storage operations. As we can see, when the value of s is greater than 0.5 (therefore indicating a very popular resource) the curve is very similar to our standard balanced trust-based algorithm, whereas when s is less than 0.5 the curve experiences an irregular behavior, being both below and above Kademlia standard algorithm, depending on the number of sybils in the network. A possible adaptation of the trust management could be useful under this attack, in order to correctly perform storage or retrieval for the target resource. This could affect both the risk evaluation, considering a further parameter, besides N_L, to take into account content specific bad behaviors, and the trust value update. Anyway an attacker should carefully consider whether to perform such a targeted attack or not. As a matter of fact, launching such a specific bad behavior may affect the effectiveness of the overall attack, as other PUT procedures are not compromised and performing a different attack for all possible contents may result too expensive in terms of computational resources.

Finally, always considering this scenario where trust is applied only to storage and retrieval operations, we were able to compare the iterative and recursive Kademlia versions as well. This is depicted in Fig. 14 where we analyzed our balanced trust approach in both versions of Kademlia, as regards the number of successful GET operations. Performances of standard Kademlia, both in the recursive and in the iterative case, are practically the same and so in the graph they are represented through a single line labeled ‘no trust’. Differently, the performances using our trust-based balanced algorithm for storing resources in recursive Kademlia seems slightly outperforming iterative Kademlia. This could be explained as, in our implementation of recursive Kademlia, the final in charge peer sends back, to all the contacted nodes, information regarding the success of the PUTs or the GETs, whereas in iterative Kademlia this information is available only to the actual Initiator of GET or PUT requests. This could lead to a better spreading of trust information among peers.

5.3. *Trust applied to all Kademlia RPCs*

Finally, we assessed the effectiveness of our balanced trust-based algorithm on the overall Kademlia algorithm. To reach this goal, we focused only on iterative Kademlia, in order to apply trust both to the routing process and to the subsequent storage or retrieval procedure. In these analysis conditions we focused on the percentage of overall successful RPCs, i.e., we took the number of successful lookup requests followed by a PUT or GET operation into account. A

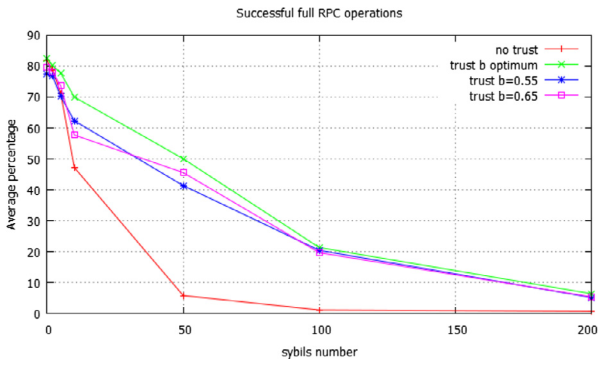


Fig. 15. Successful RPC operations in Kademlia according to standard algorithm and balanced trust algorithm with variable balancing factor.

successful request can be so deemed if it reaches the node responsible for a certain resource, according to standard Kademlia XOR distance, and it succeeds into storing in it the resource or to get from it a not fake resource. The main finding in this set of simulations is that an optimum value for the *b* parameter does not exist, but the maximum of the performances is achieved with a value, ranging from 0.55 to 0.65, which act, respectively, as lower and upper bound. The value of the balancing factor is not simple the mean value but it varies according to the total number of storage or routing operations launched in the network. The main results are depicted in Fig. 15. Obviously, the metrics analyzed in the previous subsections undergo worse performances, separately considered, however they perform better than in the case no trust is deployed.

6. Conclusion

In this paper we carried out a deep analysis of anti-sybil countermeasures present in the literature of the last years and we quantitatively analyzed the effects of a Sybil attack on some important features of a Kademlia network, namely routing and storage and retrieval operations. We presented a combined trust-based algorithm to make the *find_node*, *find_value* and *store* RPCs of a Kademlia network more resilient in presence of malicious nodes with many false identities, considering both some simple malicious behaviors and some more sophisticated attitudes. Our approach, based both on the standard Kademlia algorithm and on a trust-based algorithm, is not centralized and uses a new distance metric trying to vary the possible spectrum of nodes in order to mitigate the malicious behavior without the need for certificates, cryptography or Certification Authorities. The results of our simulations confirm that using a balanced trust algorithm may lead to better results when compared with standard Kademlia and that its performances are better even when compared with those methods using exclusively trust. Future works may consider how to effectively thwart the more sophisticated attacks highlighted in the paper, perhaps exploiting the suggested hints in the trust management, and finding an apt threshold after which the Sybil attack is ensured. Another future research topic could be analyzing the trust information spreading among peers in the recursive version of Kademlia or the variation of the optimum value for the balancing factor in different situations and levels of environmental risk in detail. Finally, finding a mathematical close form for the variation of the optimum value of the balancing factor according to the number of enacted routing or storage and retrieval operations could be very interesting.

Acknowledgments

The author would like to thank his colleague Luca Veltri, who provided insight and expertise that greatly assisted the research, and Antonio Enrico Buonocore for his precious support in proof reading the article.

References

[1] R. Pecori, L. Veltri, A key agreement protocol for P2P VoIP applications, Proceedings of the Seventeenth International Conference on Software, Telecommunications and Computer Networks, Hvar-Korcula-Split, Croatia, September 2009.
[2] R. Pecori, A PKI-free key agreement protocol for P2P VoIP applications, in: Proceedings of the IEEE International Conference on Communications (ICC), Ottawa, Canada, June 2012.
[3] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, in: Proceedings of the SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego, CA, USA, August 27th–31st, 2001.
[4] P. Maymounkov, D. Mazires, Kademlia: A Peer-to-peer Information System Based on the XOR Metric, in: Proceedings of the First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 2002.
[5] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer system, in: Proceedings of the Eighteenth IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany, November 2001.
[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: Proceedings of the SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego, California, USA, August 2001.
[7] J.R. Douceur, The Sybil Attack, in: Proceedings of the IPTPS First International Workshop, Cambridge, MA, USA, March 2002.
[8] M. Kohnen, Analysis and optimization of routing trust values in a Kademlia-based distributed hash table in a malicious environment, in: Proceedings of the Second Baltic Congress on Future Internet Communications (BCFIC), Vilnius, Lithuania, 25th–27th April, 2012.
[9] A. Singh, T.W. Ngan, P. Druschel, D.S. Wallach, Eclipse attack on overlay networks: Threats and defenses, in: Proceedings of the INFOCOM Twenty-fifth IEEE Conference on Computer Communications, Barcelona, Spain, April 2006.
[10] E. Sit, R. Morris, Security considerations for peer-to-peer distributed hash tables, Lecture Notes In Computer Science, vol. 2429.
[11] R. Pecori, L. Veltri, Trust-based routing for Kademlia in a sybil scenario, in: Proceedings of the Twenty-second International Conference on Software, Telecommunications and Computer Networks (SoftCOM), September 17th–19th 2014, Split, Croatia, 2014.
[12] B.N. Levine, C. Shields, N.B. Margolin, A Survey of Solutions to the Sybil Attack Technical report, University of Massachusetts, 2006.
[13] G. Danezis, C. Lesniewski-Laas, M.F. Kaashoek, R. Anderson, Sybil-resistant DHT routing, in: Proceedings of the Tenth European Symposium on Research in Computer Security (ESORICS), September 12th–14th, 2005, Milan, Italy, 2005, pp. 305–318.
[14] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D.S. Wallach, Secure routing for structured peer-to-peer overlay networks, in: Proceedings of Fifth Usenix Symposium on Operating Systems Design and Implementation, Boston, MA, USA, December 2002.
[15] H. Yu, M. Kaminsky, P.B. Gibbons, A. Flaxman, SybilGuard: Defending against Sybil attacks via social networks, in: Proceedings of the IEEE/ACM Transactions on Networking, Pisa, Italy, September 11th–15th, 2006.
[16] T. Condie, V. Kacholia, S. Sankararaman, J.M. Hellerstein, P. Maniatis, Induced Churn as Shelter from Routing-Table Poisoning, in: Proceedings of the Thirteen Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 2006.
[17] J. Newsome, E. Shi, D. Song, A. Perrig, The Sybil attack in sensor networks: Analysis and defenses, in: Proceedings of the Third Symposium on Information Processing in Sensor Networks (IPSN), Berkeley, CA, USA, April 26th–27th, 2004.

[18] H. Rowaihy, W. Enck, P. McDaniel, T.L. Porta, Limiting Sybil attacks in structured peer-to-peer networks, in: Proceedings of the Twenty-sixth IEEE International Conference on Computer Communications (INFOCOM), May 6th–12th, 2007, Anchorage, AK, USA, 2007.

[19] N. Borisov, Computational Puzzles as Sybil Defenses, in: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, September 6th–8th, 2006, Cambridge, UK, 2006.

[20] M. Steiner, T. En-Najjary, E.W. Biersack, Exploiting KAD: Possible uses and misuses, in: Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 37(5), October 2007.

[21] J. Dinger, H. Hartenstein, Defending the Sybil attack in P2P networks: Taxonomy, challenges, and a proposal for self-registration, in: Proceedings of the First International Conference on Availability, Reliability and Security (ARES), April 20th–22nd, 2006, Vienna, Austria, 2006.

[22] F.G. Marmol, G.M. Perez, State of the art in trust and reputation models in P2P networks, in: Xuemin Shen, et al. (Eds.), Handbook of Peer-to-Peer Networking, Springer, 2010, pp. 761–784.

[23] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in P2P networks, in: Proceedings of the Twelfth International Conference on World Wide Web, Budapest, Hungary, 2003, pp. 640–651.

[24] N. Griffiths, K.-M. Chao, M. Younas, Fuzzy trust for peer-to-peer systems, in: Proceedings of the Twenty-sixth IEEE International Conference on Distributed Computing Systems Workshops (ICDCS), Lisbon, Portugal, 2006.

[25] L. Mekouar, Reputation-based Trust Management in Peer-to-peer File Sharing Systems., University of Waterloo, Waterloo, Ontario, 2010.

[26] L. Xiong, L. Liu, PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities, in: Proceedings of the IEEE Transactions on Knowledge and Data Engineering, vol. 16, July 2004, pp. 843–857.

[27] M. Kohnen, Applying trust and reputation mechanism to a Kademlia-based distributed hash table, in: Proceedings of the ICC International Communication Conference, Ottawa, CA, June 2012.

[28] M. Kohnen, J. Gerbecks, E.P. Rathgeb, Applying certificate-based routing to a Kademlia-based distributed hash table, in: Proceedings of the Third International Conference on Advances in P2P Systems, Lisbon, Portugal, 2011.

[29] N. Fedotova, L. Veltri, Reputation management algorithms for DHT-based peer-to-peer environment, Comput. Commun. 32 (12) (2009) 1400–1409. (ISSN 0140–3664).

[30] N. Fedotova, et al., Reputation management techniques in DHT-based peer-to-peer networks, in: Proceedings of the International Conference on Internet and Web Applications and Services, Mauritius, May 13th–19th, 2007.

[31] Z. Liang, W. Shi, PET: A personalized trust model with reputation and risk evaluation for P2P resource sharing, in: Proceedings of the Thirty-eighth Annual Hawaii International Conference on System Sciences, January 3rd–6th, 2005, Island of Hawaii, HI, USA, 2005.

[32] B. Heep, R/Kademlia: Recursive and topology-aware overlay routing, in: Proceedings of 2010 Australasian Telecommunication Networks and Applications Conference, Auckland, NZ, 31st October–3rd November 2010.

[33] M. Agosti, M. Amoretti, Deus-project hosting on Google code [online], 2010. Available from: http://code.google.com/p/deus/.

[34] X. Liu, K. Cai, Y. Li, Measuring “Sybil attacks” in Kademlia-based networks, in: Proceedings of the Ninth IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), Sharm El-Sheikh, Egypt, 27th–30th December, 2011.

[35] M. Urueña, et al., A model to quantify the success of a Sybil attack targeting RELOAD/Chord resources, IEEE Commun. Lett. 17 (2) (2013) 428–431. ISSN 1089-7798.

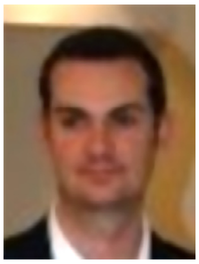
[36] A. Merlo, M. Migliardi, N. Gobbo, F. Palmieri, A. Castiglione, A denial of service attack to units networks using sim-less devices, IEEE Trans. Dependable Secur. Comput. 11 (3) (2014) 280–291.

[37] O. Jetter, J. Dinger, H. Hartenstein, Quantitative analysis of the Sybil attack and effective sybil resistance in peer-to-peer systems, in: Proceedings of the ICC International Conference on Communications, Cape Town, South Africa, 23rd–27th May, 2010.

[38] U.R. Gracia-Tinedo, et al., Sophia: Local trust for securing routing in DHTs, in: Proceedings of the Eleventh IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011), Newport Beach, CA, USA, 23rd 26th May, 2011.

[39] D. Macone, et al., A dynamic load balancing algorithm for quality of service and mobility management in next generation home networks, Telecommun. Syst. J. 53 (3) (2013) 265–283, doi:10.1007/s11235-013-9697-y.

[40] G. Oddi, et al., A decentralized load balancing algorithm for heterogeneous wireless access networks, in: Proceedings of the World Telecommunications Congress (WTC), 2014, pp. 1–6.



Riccardo Pecori received the 3-year program degree in Telecommunications Engineering on October 2004, “cum laude”, from University of Parma. On July 2007, he got the Master’s degree in Telecommunications Engineering “cum laude”, discussing a thesis entitled “Development of a Statistical Technique for Recognizing Applications over IP Networks”, awarded as a finalist in the Laurea degree awards competition from the Institute of Engineers of Perugia in December 2008. Since April 2008 to December 2008 he has been a recipient of a CNIT scholarship for the TERIT (TElecommunications Research in ITaly) project for the Multimedia Services and Networks theme. In 2010 he was also a teaching lecturer for the “Telecommunications Networks” e-learning course at University of Parma and commissioner for the “Telecommunications” and “Telecommunications Networks” at the Engineering School of eCAMPUS University. In 2011 he got his Ph.D. from the Information Engineering Department of the University of Parma and became adjunct professor of “Telecommunication Networks” at eCAMPUS University and since February 2014 also of “Telecommunications Didactics”. Since September 2014 he is adjunct professor also of “IT Security” and “Network Security” at eCAMPUS University and adjunct professor of “Informatics” and “Computer Science” at the Department of Veterinary Medicine of the University of Parma. Since May 2015 he is Assistant Professor of Information Technology at eCAMPUS University and Associate Researcher at the Information Engineering Department of the University of Parma.