Computer Networks 56 (2012) 2569–2589

Contents lists available at SciVerse ScienceDirect

Computer Networks

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

Identity management based on adaptive puzzles to protect P2P systems from Sybil attacks

Weverton Luis da Costa Cordeiro, Flávio Roberto Santos, Gustavo Huff Mauch, Marinho Pilla Barcelos, Luciano Paschoal Gaspary *

Institute of Informatics, Federal University of Rio Grande do Sul, Av. Bento Gonçalves, 9500, 91.501-970 Porto Alegre, RS, Brazil

ARTICLE INFO

Article history:
Received 10 August 2011
Received in revised form 24 March 2012
Accepted 31 March 2012
Available online 12 April 2012

Keywords:
Peer-to-Peer networks
Identity management
Computational puzzles
Counterfeit identities
Sybil attack

ABSTRACT

The Sybil attack consists on the indiscriminate creation of counterfeit identities, by a malicious user (attacker), in large-scale, dynamic distributed systems (for example, Peer-to-Peer). An effective approach to tackle this attack consists in establishing computational puzzles to be solved prior to granting new identities. Solutions based on this approach have the potential to slow down the assignment of identities to malicious users, but unfortunately may affect normal users as well. To address this problem, we propose the use of adaptive computational puzzles as an approach to limit the spread of Sybils. The key idea is to estimate a trust score of the source from which identity requests depart, calculated as a proportion of the number of identities already granted to (the) user(s) associated to that source, in regard to the average of identities granted to users associated to other sources. The higher the frequency (the) user(s) associated to a source obtain(s) identities, the lower the trust score of that source and, consequently, the higher the complexity of the puzzle to be solved. An in-depth analysis of both (i) the performance of our mechanism under various parameter and environment settings, and (ii) the results achieved with an experimental evaluation, considering real-life traces from a Peer-to-Peer file sharing community, has shown the effectiveness of the proposed mechanism in limiting the spread of Sybil identities. While comparatively more complex puzzles were assigned to potential attackers, legitimate users were minimally penalized with easier-to-solve puzzles.

1. Introduction

The indiscriminate creation of counterfeit identities (or Sybil attack [1]), although elementary, remains as one of the major threats to the security of distributed systems [2]. The main motivation for a malicious user (attacker) to launch such an attack is to control a large fraction of peer identities in the network. In doing so, there is a high probability that interactions between legitimate peers will be affected by a fraction of counterfeit identities, and manipulated for the benefit of the attacker [3]. In Peer-to-Peer (P2P) networks, the corruptive power of counterfeit identities (Sybil identities) is widely known, being the object of several studies in the literature [4,5]. The tampering of polling-based and/or reputation algorithms (to manipulate the reputation of peers or contents being shared), and even the launching of other attacks such as eclipse [6], free-riding [7], and pollution [8], are examples of malicious actions an attacker could accomplish if she manages to control several counterfeit identities.

A promising approach to protect from Sybil attacks consists in assigning computational puzzles to requesting users prior to granting them or renewing their identities [9]. This approach relies on the assumption that the amount of computational resources available to users (processing cycles, memory, etc.) is finite. Therefore, by

* Corresponding author.
E-mail addresses: weverton.cordeiro@inf.ufrgs.br (W.L. da Costa Cordeiro), frsantos@inf.ufrgs.br (F.R. Santos), ghmauch@inf.ufrgs.br (G.H. Mauch), marinho@inf.ufrgs.br (M.P. Barcelos), paschoal@inf.ufrgs.br (L.P. Gaspary).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.03.026

demanding users to prove that they have a certain fraction of these resources for each identity requested, one can limit the number of counterfeit identities an attacker can create.

Previous work has demonstrated the potential benefits of using computational puzzles for identity management in P2P networks [9,10]. However, existing proposals are limited as they do not distinguish between requests from legitimate users and those originated by attackers. Ideally, requests from malicious users would be assigned much higher costs than those originated by legitimate users. If so, for some arbitrary hardware capacity at the hands of an attacker, the number of identities she could obtain would be much smaller than in existing approaches. The challenge to this approach, however, would be to identify and separate malicious from legitimate requests. The key observation, which we explore in this paper, is that malicious users tend to recur much more frequently than normal ones.

In this paper we propose the use of adaptive puzzles as a mechanism to limit the spread of Sybils in P2P systems. In contrast to previous approaches, our mechanism estimates a trust score of the source from which identity requests originate, calculated as a proportion of the number of identities already granted to (the) user(s) associated to that source in regard to the average of identities granted to users associated to other sources. The higher the frequency (the) user(s) associated to a source obtain(s) identities, the lower its trust score and, consequently, the higher the complexity of the puzzle to be solved by (the) user(s) associated to that source (the concept of source is further detailed in Section 3.1).

To assess the effectiveness of the proposed mechanism in limiting the spread of Sybils, we (*i*) developed a prototypical implementation of an identity management system, (*ii*) evaluated its performance under various parameter and environment settings, and (*iii*) carried out a set of experiments using real traces of identity requests from a P2P file sharing community, considering scenarios with and without attack. An in-depth analysis of the results achieved evidenced that potential attackers have to face comparatively more complex puzzles, and thus the rate of identity assignment to malicious users is substantially reduced. For the sake of evaluation, we consider a P2P network using a weak identity scheme, in which a user obtains a new id upon joining the network (whose lifetime is the session of the user's P2P agent). Nonetheless, our mechanism can be straightforwardly adopted to address other identity schemes, for example one considering long-term identities. In this case, a continuous revalidation of these identities (through the resolution of new puzzles) during their lifecycle would have a similar effect as in the weak identity scheme addressed in our evaluation.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 presents the proposed mechanism for adaptive puzzles as a protection against Sybil attacks. Section 4 describes the evaluation carried out to analyze the sensitivity of the proposed mechanism to various parameter and environment settings, whereas Section 5 presents the results achieved with real life traces of identity requests. Section 6 discusses practical aspects related to the instantiation of our mechanism in P2P networks, and Section 7 closes the paper with concluding remarks.

2. Related work

Countermeasure proposals to tackle the Sybil attack generally fall in two categories, namely *weak* and *strong identity-based*, depending on the strategy employed to enforce peer authenticity [11]. The first category comprises mechanisms in which users have autonomy to create their own identities in the P2P system.

In this case, although it is not possible to avoid the existence of Sybil identities, it is possible to limit their amount to an "acceptable level". Such mechanisms may be useful, for instance, for applications that can tolerate a certain fraction of counterfeit identities. Examples of mechanisms in this category include those proposed by Yu et al. [12,13], Danezis and Mittal [11], and Tran et al. [14]. These mechanisms explore the concept of social networks to limit the spread of Sybil identities in a P2P system, also estimating an upper bound for the number of counterfeit identities that are "accepted". In addition to the violation of user anonymity, Yang et al. [2] has recently shown that Sybils do not aggregate themselves in a dense network of relationships, thus invalidating of the key assumptions upon which social network-based solutions relied on.

In the second category of proposals, peers may only obtain identities through certification authorities. Its main advantage is the difficulty imposed to peers that attempt to create and control several identities, or take control of someone else's identity. Nevertheless, such mechanisms may limit the scalability of the P2P network, force users to trust unknown certification authorities, and hamper the access of potential users (for example, when he/she needs to provide personal data or to pay fees to obtain one identity). The proposals that fit in this category [15–17] attempt to minimize some of these side-effects. For example, in [16] and [17], the authors have focused on the decentralization of the Public-Key Infrastructure (PKI). However, these solutions rely on the contribution of a certain fraction of peers to operate as expected.

In the paper that describes the Sybil attack [1], Douceur demonstrated that a robust and scalable solution for peer authentication in P2P networks requires a certain degree of centralization (for example, by employing certification authorities). Since then, and considering the severe constraints imposed by the use of certification authorities, an intermediate category of proposals that has attracted attention in recent years is to condition identity granting/renewal to the previous resolution of computational puzzles. Puzzles are typically assigned to users (and their solution, verified) by a *bootstrap service*, and aim at decreasing attackers' capabilities of creating counterfeit identities without compromising the intrinsic characteristics of P2P networks (such as scalability, decentralization, and peer autonomy).

Approaches based on computational puzzles have presented satisfactory results when using challenges created and/or verified in a distributed fashion. Borisov [9], for

example, has shown the technical feasibility of using puzzles generated periodically and distributedly, by proposing a mechanism in which peers that participate in the puzzle generation process are able to validate its solution. Rowaihy et al. [10], in turn, have proposed a mechanism based on multiple puzzle generation entities. It requires that, prior to obtaining identities, users contact one of these entities and solve a series of puzzles.

Despite the advances in computational puzzle-based identity management, existing mechanisms do not address the issue of adjusting puzzle complexity. More specifically, when assigning puzzles of equal computational complexity to any user, it becomes very hard to choose a complexity that effectively reduces the assignment of identities to malicious users, without severely compromising legitimate ones. On one extreme, puzzles having higher complexity penalize legitimate users (who often have less powerful hardware). On the other extreme, puzzles having lower complexity may favor potential attackers (which are generally equiped with high performance computing hardware). Therefore, the use of a uniform complexity for puzzles may benefit attackers at the expense of legitimate users.

In a previous work [18] we have proposed a mechanism in which the complexity of assigned puzzles is dynamically adapted, according to the relative recurrence of (the) user(s) behind each source in the network. Users associated to sources whose recurrence is lower or similar to the average recurrence of other sources are benefited with less complex puzzles. In contrast, users associated to sources whose recurrence is higher than the average of other sources are forced to cope with more complex puzzles to obtain identities. As a significant contribution to our previous work, in this paper we explore in more detail the potentialities of using adaptive puzzles to prevent the widespread dissemination of counterfeit identities in large-scale, dynamic distributed systems. Furthermore, we (*i*) present an enhanced version of the algorithm (and supporting mathematical formulation) used to calculate the trust score of sources of identity requests, (*ii*) introduce a formalism for the translation of trust scores into puzzles to be solved by users requesting identities, (*iii*) provide a detailed analysis to show the effectiveness of our mechanism considering various parameter and environment settings, also including a performance comparison with existing solutions, and (*iv*) show the effectiveness of using adaptive puzzles to prevent the creation of counterfeit identities considering real traces of identity requests, obtained from a BitTorrent community.

It is important to highlight that our mechanism (similarly to others based solely on computational puzzles [9,10]) falls in the category of *weak identity-based* mechanisms. This is because users are not required to provide "proof of identity" (e.g. personal data) and/or pay taxes when requesting identities (which are characteristics inherent of *strong identity-based* mechanisms). As a consequence, created identities do not fit for the purpose of strongly authenticating a given user of the P2P network (similarly to any other proposal that falls in the category of *weak identity-based* mechanisms).

As shown in the following sections, our mechanism not only assigns more complex puzzles to potential attackers (and benefits legitimate users with less complex ones), but also deals appropriately with scenarios in which legitimate users and attackers are associated to a same source.

3. Proposed mechanism

Given the problem of the Sybil attack in P2P networks, this paper proposes the establishment of adaptive computational puzzles for identity management. It involves solving five main sub-problems, namely: (*i*) identify the sources of identity requests; (*ii*) characterize the behavior of sources (or the behavior of users associated to them); (*iii*) calculate the trust score of a source based on the observed behaviors; (*iv*) deal with the dynamics of users' behavior in the calculation of the trust score; and (*v*) translate the trust scores into puzzles to be solved. Each of these sub-problems are addressed in the following subsections.

3.1. Using network-gathered information to identify sources of identity requests

The concept of *source of identity requests*, fundamental building block upon which the proposed mechanism is built, is defined as an aggregation of one or more users (either legitimate, malicious, or both), located in an specific portion of the network, from which identity requests originate. Therefore, the exact meaning of "a *source* requests and obtains identities" is "user(s), from a certain *source*, request(s) and obtain(s) identities". Fig. 1 (left) illustrates this concept, also highlighting the interactions between sources of identity requests and the bootstrap service (entity that assigns identities to users interested in joining a P2P network).

Following a traditional process of P2P identity creation/assignment, when a user becomes interested in joining a P2P network, it issues an *identity request* to the bootstrap service (flow 1 in Fig. 1). In the absence of a mechanism to control the assignment of peer identities, the bootstrap service replies to that request assigning an identity to the user (flow 2), so that he/she may use it to join a P2P network (flow 3). In our mechanism, identity requests are associated to a source according to user location. In Fig. 1, a source is represented as a "cloud", and the users associated to a given source are located within its respective cloud. Before granting an identity, the bootstrap service requires the user to solve a puzzle whose complexity depends on the behavior observed for that source (as discussed in the following subsections). To prevent tampering, the messages exchanged between the user and the bootstrap service, as well as granted identities, must be digitally signed by the bootstrap service (using its pair of public/private keys).

There are two general strategies that can be used (either combined or isolatedly) to delineate the portion of the network regarded as a single source. The first one is viewing an IP address, a sub-network, or a combination of non-contiguous sub-networks as a single source. The second main

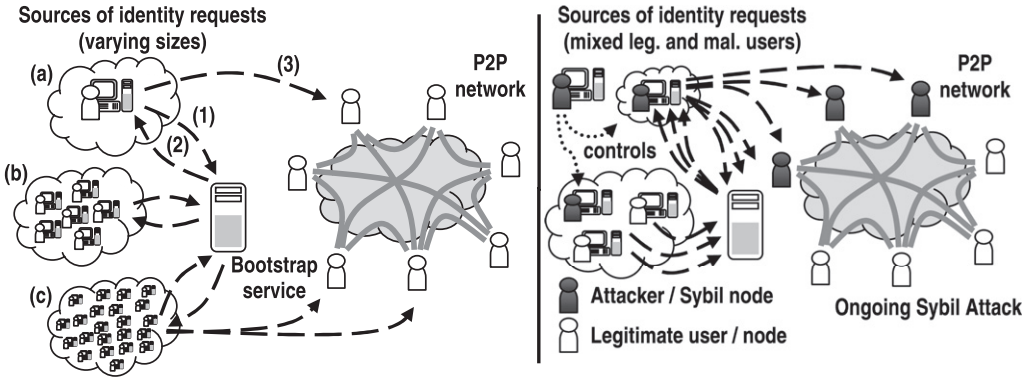


Fig. 1. Characterization of sources of identity requests (left), and launch of a Sybil attack onto the P2P network (right).

strategy considers the use of a network coordinate system (e.g. Vivaldi [19] and Veracity [20]) for distinguishing requests coming from certain regions, cities, states or even countries. One could also combine both strategies to distinguish a large number of users behind NAT (or behind a small number of IP addresses); conversely, it may be useful to use network coordinates to identify a single attacker using a large number of IP addresses. The combination of both strategies is out of scope of this paper, being envisaged as future research on the topic. However, we present in Section 4 an evaluation of the effectiveness of our mechanism against a coordinated attack originated from various sources, and also its impact to various users located in a same source.

The design decision of using IP addresses to materialize the concept of sources is a well established one, and was first used by Liang et al. [21]. In that work, sources are defined as an aggregation of various users, located in specific IP ranges, who upload contents (multimedia files, documents, software, etc.) to the P2P network. Similarly, our concept of sources of identity requests could be materialized as an aggregation of various users, behind specific IP ranges, who request identities to the bootstrap service.

Observe also that sources may vary in *granularity* and *composition*, regardless of the strategy used to delineate them. Fig. 1 (left) shows that the granularity (i.e. number of users associated to a source) may range from a single user, e.g. a user's workstation (entity *a*), to a group of several users, e.g. a local network (entity *b*) or an entire autonomous system (entity *c*). Fig. 1 (right) shows, in turn, that sources may be heterogeneous, i.e. may be shared between legitimate users and attackers. In fact, malicious requests may also originate from sources to which legitimate users are associated. Our mechanism performs satisfactorily regardless of the source granularity and composition, as shown in the analysis presented in Sections 4 and 5.

3.2. Employing recurrence rates to characterize behaviors

In order to characterize the behavior of sources of identity requests, it is important to keep track of the number of identities already granted to the users associated to a given source. In the context of this work, this number is defined as $\phi_i(t)$ for the ith source, at instant t (with $\phi_i(t) \in \mathbb{N}$). Based on this information, two metrics are defined: *source recurrence rate* ($\Delta\phi_i$) and *network recurrence rate* ($\Phi(t)$, with $\Phi(t) \in \mathbb{R}$ and $\Phi(t) \geqslant 1$). The former, defined as $\Delta\phi_i = \phi_i(t) - \phi_i(t - \Delta t)$, represents the number of identities the bootstrap service granted to users associated to some specific source i, within a given time interval Δt. The latter corresponds to the average number of identities that sources have obtained from the bootstrap service. Observe that the computation of these metrics is straightforward, as the bootstrap service has direct access to information regarding identity requests. Note that Δt is an input parameter, and therefore must be adjusted manually (we discuss in Section 4.1 a methodology for defining an appropriate setting for this parameter).

The network recurrence rate $\Phi(t)$ is computed using the simple mean of sources' recurrence rates, according to Eq. (1). In this equation, n is the number of currently active sources, i.e. those that have obtained at least one identity within the interval Δt. Note that when $\Delta\phi_k = 0$ for some source k, users associated to it have not obtained any identity (during Δt); such a source can be safely ignored.

$$\Phi(t) = \begin{cases} 1 & , \quad \text{if } n = 0 \\ \frac{1}{n} \times \sum_{i=1}^{n} \Delta\phi_i, & \text{if } n \geqslant 1 \end{cases} \tag{1}$$

With regard to using Δt as a bound for the portion of identity grants considered when computing the sources' (and the network) recurrence rates, it is important to observe that rates in which identities are granted to sources may vary across different times of the day, month or year. Thus, at certain times (e.g. on weekends) a larger number of users might be interested in joining the P2P network and, consequently, more identities might be granted. Without taking into account the timely patterns of users (and of the network), legitimate users may be regarded as suspicious if the source (s) to which these are associated obtain(s) new identities with higher frequency. Conversely, if all identity grants were considered (since the beginning), it would be easier for an attacker to launch Sybil attacks; this is because the number of identity grants would grow indefinitely, consequently obfuscating the higher recurrence rates of suspicious sources. Therefore, the time interval Δt functions as a "sliding window" that addresses the

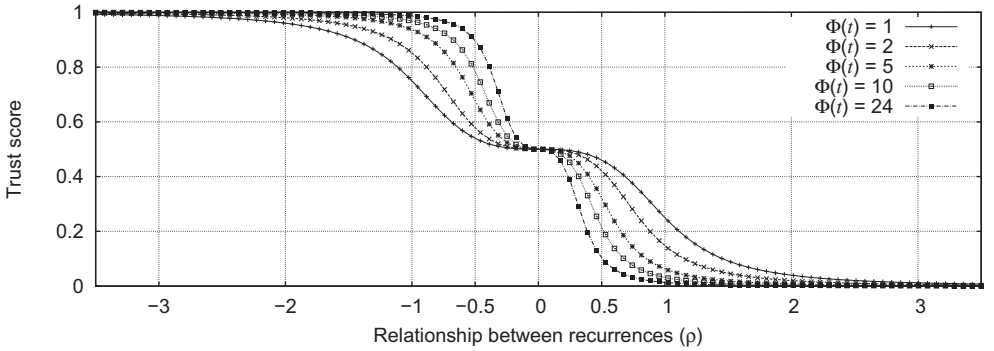


Fig. 2. Curves of Eq. (3) for the calculation of the source trust score, considering different values of network recurrence rates ($\Phi(t)$). Note that these values are arbitrarily chosen for the sake of illustration; their meaning depends on the interval chosen.

seasonality in the pattern of identity grants. As this window slides forward, older identity grants are gradually discarded, thus allowing room to newer ones which are more representative of the current state of the P2P network.

3.3. Calculating trust scores from observed behaviors

For a Sybil attack to succeed, the attacker must obtain a proportionally large number of identities from the bootstrap service. This behavior leads to the increase in the recurrence rate observed for the source associated to the attacker, as illustrated in Fig. 1 (right). Conversely, it is expected that the sources associated to legitimate users obtain fewer identities. Therefore, the underlying idea to control Sybil attacks is to assign more complex puzzles to users associated to sources whose recurrence rates are higher than the network recurrence rate.

By comparing the behavior of a given source i (inferred from $\Delta\phi_i$) and the network behavior (inferred from $\Phi(t)$), we calculate the *relationship between source and network recurrence rates* ($\rho_i(t)$, with $\rho_i(t) \in \mathbb{R}$). When negative, $\rho_i(t)$ indicates how many times the recurrence rate of the ith source is lower than the network recurrence rate. For example, $\rho(t) = -1$ means that, at instant t, $\Delta\phi_i$ is 50% lower than $\Phi(t)$ (i.e. $\Phi(t)$ is 100% higher than $\Delta\phi_i$). Likewise, a positive $\rho_i(t)$ expresses how higher is the recurrence rate of the ith source. For instance, if $\rho_i(t) = 2$, then $\Delta\phi_i$ is 200% higher than $\Phi(t)$. Eq. (2) provides the value of $\rho_i(t)$.

$$\rho_i(t) = \begin{cases} \frac{1}{\Phi(t)} - 1, & \text{if } \Delta\phi_i = 0 \\ 1 - \frac{\Phi(t)}{\Delta\phi_i}, & \text{if } 1 \leqslant \Delta\phi_i \leqslant \Phi(t) \\ \frac{\Delta\phi_i}{\Phi(t)} - 1, & \text{if } \Delta\phi_i > \Phi(t) \end{cases} \quad (2)$$

The relationship between the source and network recurrence rates ($\rho_i(t)$) is used to compute the *source trust score* ($\theta_i(t)$). This score, calculated at instant t according to Eq. (3), assumes values in the interval $(0, 1)$: on one extreme, values close to 1 denote a high trust on the legitimacy of (the) user(s) associated to the ith source; on the other, values close to 0 indicate high distrust, i.e. a high probability that there exists an attacker behind that source, who is launching a Sybil attack. The complexity of the computational puzzles applied to a user is then determined by the trust score of the source to which he/she is associated, in the moment a new identity is requested (this aspect will be further discussed in Section 3.5).

In Eq. (3), the constant *0.5* defines the mean trust score, which is incremented or decremented by 0.5 (therefore resulting in a trust score that ranges from 0 to 1) depending on $\rho_i(t)$. The function arctan (x) is used to map any possible value of $\rho_i(t)$, in the interval $(-\infty, +\infty)$, into a value in the interval $(-\frac{\pi}{2}, +\frac{\pi}{2})$. The term π in Eq. (3) normalizes the result of the arctan function, to the interval $(-0.5, 0.5)$ (see *Prop. 1* next). The metric $\rho_i(t)$ is raised to the power of 3 in order to create an interval around $\rho_i(t) = 0$ for which the trust score varies minimally (see *Prop. 2*). Finally, the term $\rho_i(t)^3$ is weighted by the current network recurrence rate ($\Phi(t)$) to regulate the conservativeness of the proposed mechanism in face of the current network behavior (see *Prop. 3*).

$$\theta_i(t) = 0.5 - \frac{\arctan\left(\Phi(t) \times \rho_i(t)^3\right)}{\pi} \quad (3)$$

Fig. 2 presents five different configurations that illustrate how the trust score obtained for the ith source varies as a function of $\rho_i(t)$. In each of these configurations, the current network recurrence rate $\Phi(t)$ controls how conservative the proposed mechanism should be regarding the current perception of a source recurrence rate. The plot reveals three important properties that Eq. (3) holds. These are enumerated below.

- **Property 1.** The function that maps values of $\rho_i(t)$ into trust scores is asymptotic in 0 and 1; therefore, for $\rho_i(t) \to -\infty$ or $\rho_i(t) \to +\infty$, there is always a corresponding value of trust score.
- **Property 2.** The trust score varies minimally for values of $\rho_i(t)$ close or equal to 0; this corresponds to a situation in which the ith source behaves similarly or equals to the network average $\Phi(t)$.
- **Property 3.** The proposed mechanism becomes more conservative as the current network average $\Phi(t)$ increases, and less conservative otherwise.

Observe that *Prop. 2* allows a certain degree of tolerance in the evaluation of source behavior. To illustrate, consider

the curve for $\Phi(t) = 1$ shown in Fig. 2; variations of $\rho_i(t)$ lying in the interval $[-0.5, 0.5]$ indicate a source behavior similar to the pattern observed in the network, and thus have minimal impact. Behaviors that deviate significantly from this interval, however, will be assigned lower (or higher) trust scores.

Now focusing on *Prop. 3*, it enables weighting appropriately the value of $\rho_i(t)$ taking into account the current network behavior. To illustrate, consider the case of $\rho_i(t) = 0.5$ in Fig. 2. Regardless of the current network recurrence rate, this value of $\rho_i(t)$ indicates that the ith source has obtained 50% more identities than the average of all sources. Considering as an example the case of $\Phi(t) = 2$, the currently active sources in the network have requested 2 identities in the interval Δt on average, whereas the ith source has obtained 50% more identities in the same interval, i.e. 3 identities; as a consequence, a trust score of approximately 0.4 is assigned to subsequent identity requests originating from this source. In the case of $\Phi(t) = 24$, however, the same value of $\rho_i(t) = 0.5$ means that the ith source obtained 36 identities in the interval Δt (50% more than the average of 24 identities requested by the currently active sources). Although the proportion of identities obtained (in the interval Δt) in regard to the average of the network remained the same (50%), the surplus of identities obtained in the latter case ($36 - 24 = 12$ identities) was much higher than in the former one ($3 - 2 = 1$ identity only); as a consequence, our mechanism becomes more conservative, and establishes a trust score of 0.1 to future identity requests originating from this source.

3.4. Dealing with the dynamics of users' behavior

Peer autonomy is an important characteristic of P2P networks and has several implications. Peers may arbitrarily join and leave the network at any moment, or become unavailable. One possible effect of such dynamics is the variation in the behavior of both individual sources and the network as a whole. Next, we discuss how the proposed mechanism deals with the dynamics of observed behaviors.

The trust score provided by Eq. (3) is instantaneous (at some time t). This variable is limited in the sense it does accommodate fluctuations in the recurrence rate of a source (which could originate identity requests in bursts). Thus, a source with historically lower recurrence rates should be entitled to demand higher rates for some time without being penalized. To accomplish this, a smoothing factor is used to properly represent the trust score of a given source in light of its past behavior.

The smoothed trust score, defined as $\theta'_i(t_k)$ for the ith source in a given instant $t_k = t$, is calculated as shown in Eq. (4). The smoothing factor β determines the weight of present behavior in the calculation of the smoothed trust score, assuming values in the interval $(0, 1]$. Thus, values of β close to 0 assign a high weight to the historical behavior of the source under consideration, and vice versa. In the special case in which $\beta = 1$, the current trust score (as calculated through Eq. (3)) is fully considered, and the historical behavior, totally ignored. In Eq. (4), t_{k-1} refers to the instant in which the smoothed trust score was computed for the last time.

$$\theta'_i(t_k) = \begin{cases} \theta_i(t), & \text{if } k = 0 \\ \beta \times \theta_i(t) + (1-\beta) \times \theta'_i(t_{k-1}), & \text{if } k \geqslant 1 \end{cases} \quad (4)$$

The smoothing factor β is important to deal adequately with changes in the behavior of sources of identity requests. In particular, abrupt and/or intentional changes in the source behavior, caused by (a) user(s) interested in obtaining benefits (such as "betrayers"), are captured in the trust score of the source associated to that user(s). A "betrayer" is an attacker that aims at obtaining good scores in reputation-based systems and, once obtained, uses them to harm other peers or obtain unfair advantages. A proper dimensioning of β, in this context, may prevent an attacker from manipulating the proposed mechanism in such a way that the source in which he/she is located obtains (or recovers) a high trust score more quickly. Since the historical behavior is considered while asserting the present behavior, only those sources whose users present good historical behavior may be considered trustworthy.

3.5. Translating trust scores into adaptive puzzles

The trust score calculated for a source of identity requests is used as input to determine the complexity of the puzzles to be solved by users associated to that source. The mapping between trust and puzzle complexity is given by an abstract function $\gamma : \Theta \rightarrow \mathbb{N}^*$. In this function, the value of the trust score $\theta'_i(t_k) \in \Theta$ is mapped to a computational puzzle having complexity equivalent to $\gamma_i(t_k)$.

To illustrate, consider the computational puzzle presented by Douceur in [1]: given a sufficiently high random number y, find two numbers x and z such that the contatenation $x|y|z$, after processed by a secure hash function, leads to a number whose γ least significant bits are 0. The time required to solve the proposed puzzle is proportional to $2^{\gamma-1}$, and the time to assert the validity of the solution is constant. Considering this puzzle, an example of mapping function for computing γ is the one illustrated in Eq. (5). In this function, the factor 18 defines the maximum complexity allowed for an assigned puzzle, whereas the constant 1 defines the minimum possible complexity.

$$\gamma_i(t_k) = \lfloor 18 \times (1 - \theta'_i(t_k)) + 1 \rfloor \quad (5)$$

Any feasible puzzle can be employed with our mechanism. There are examples in the related literature, such as [1,9,10], so there is no need to invent a new one.

4. Sensitivity analysis of the proposed mechanism

In order to evaluate the technical feasibility of using adaptive puzzles to tackle Sybil attacks in P2P networks, we implemented a bootstrap service for use in a simulation environment. In summary, the implemented service aggregates the functionalities of management of identity requests from users interested in joining the network, assignment of puzzles for each identity request, validation of solutions received for assigned puzzles, and granting (or

denial) of requests (according to the correctness of received solutions).

An instance of the bootstrap service has then been used to carry out several experiments, using a combination of both realistic traces of identity requests and synthetic ones. The experiments aimed at confirming that (*i*) puzzles proposed to legitimate users minimally penalize them; (*ii*) puzzles assigned to potential attackers have comparatively higher computational complexity; (*iii*) the proposed mechanism is robust and resilient even when there is a large fraction of attackers in the system; and (*iv*) the number of counterfeit identities effectively granted by our mechanism is kept to a minimum, without impacting significatively the requests originating from legitimate sources.

In this section we evaluate how well the proposed mechanism ameliorates the negative impact of a Sybil attack considering various distinct settings. The parameters involved in the evaluation, summarized in Table 1, are separated in two classes: *Environment-related parameters* characterize the P2P network under consideration; and *Input parameters*, those that may be adjusted to regulate the performance of the proposed mechanism in tackling Sybil attacks. In the following section, we present and discuss the results obtained with the proposed mechanism using real traces of identity requests of a closed BitTorrent community.

In our analysis, we evaluate the dynamics of identity granting considering different sizes for the sliding window (Section 4.1); various values for the smoothing factor (Section 4.2); sharing of legitimate sources with the attacker (Section 4.3); different sizes of sources (Section 4.4); increasing computing power of the attacker (Section 4.5); and varying proportions of sources in control of the attacker (Section 4.6).

The values for the other environment-related parameters are defined as follows. A number of $L_u = 160{,}000$ legitimate users perform $L_r = 320{,}000$ identity requests to the bootstrap service, during a period of 1 week, or 168 h ($T = 604{,}800$ s). This makes an average of two identities requested per user, during 1 week.

To evaluate our mechanism, we developed a discrete event simulator using Java. In our simulator, the arrival of an identity request and the resolution of a puzzle are regarded as events. The global clock is used mainly for event dispatch and to keep track of identity grants that should be gradually discarded by the mechanism (an effect of the sliding window). The simulator also keeps a queue of events, which is dynamically ordered considering the timestamp in which events will occur in the simulation. The global clock always evolves to the timestamp of the next event in the queue. The ending condition in our simulator is the instant when, in the absence of the mechanism, the last arrival of a legitimate identity request occurs.

In order to model the delay caused by solving computational puzzles, we considered the puzzle presented by Douceur in [1] and described in Section 3.5. Further, for the sake of simplicity, we considered that a puzzle having complexity $\gamma_i(t_k)$ takes $2^6 + 2^{\gamma_i(t_k)-1}$ seconds to be solved in a powerful but standard, off-the-shelf computing hardware (which has a normalized computing power of 1, for reference); a computer two times faster takes half of this time to solve the same puzzle.

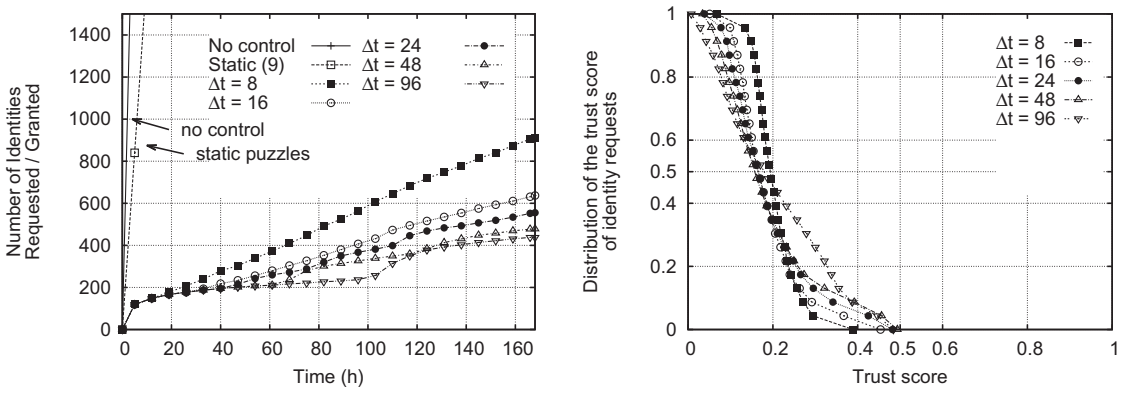
The *computing power* of users' workstations follows an exponential distribution ($\lambda_{cp} \approx 0.003$), ranging from 0.1 to 2.5 times the computing power of the standard, off-the-shelf hardware used as reference. This is an arbitrary modeling for the distribution of computing power of users' stations, for the sake of investigation only, since as far as we are aware of there is no such a model that is generally representative of P2P users. Our choice is also more conservative in comparison to evaluation scenarios considered in previous investigations [10], in which both legitimate users and the attacker are assumed to have a fixed and equal computing power. The amount of time required to solve a puzzle is then obtained by dividing the total number of complexity units (of the assigned puzzle) by the computing power of the user's station.

Based on this model, an identity request performed by a user associated to a source i having $\theta'_i(t_k) = 0.5$ would be assigned a puzzle whose goal is to obtain two numbers x and z so that the concatenation $x|y|z$, after processed by a secure hash function, leads to a number where the $\lfloor 18 \times (1 - 0.5) + 1 \rfloor = 10$ least significant bits are 0. This puzzle has complexity of $2^6 + 2^{10-1} = 576$ units. Considering the reference computing power, this puzzle takes 576 s to be solved; in the case of a hardware two times faster, the same puzzle takes 288 s.

As mentioned earlier in this paper, we consider a P2P network using a weak identity scheme, in which a user obtains a new identity upon joining the network. In this scenario, an attacker, provided with a limited amount of resources, launches a Sybil attack in an attempt to obtain (and thus control) a minimum of 1/3 of peer identities in

Table 1
Summary of parameters involved in the evaluation of the proposed mechanism.

Parameter/variable	Taxonomy	Description
S	Environment-related parameter	Number of (legitimate) sources in the system
L_r	Environment-related parameter	Number of identities requested by legitimate users
L_u	Environment-related parameter	Number of legitimate users
L_s	Environment-related parameter	Number of legitimate users per source
M_r	Environment-related parameter	Number of identities requested by the attacker
M_u	Environment-related parameter	Number of sources in hands of the attacker
M_c	Environment-related parameter	Number of high-performance workstations the attacker has available
T	Environment-related parameter	Simulation duration (in time units)
β	Input parameter	Smoothing factor for the source trust score
Δt	Input parameter	Size of the time interval, or "sliding window" (in time units)



(a) Counterfeit identities created (b) CCDF of the trust score of malicious sources

Fig. 3. Effectiveness of the proposed mechanism in controlling malicious identity requests, under different sizes for the sliding window Δt.

the network. This proportion was chosen since it exceedes the amount of counterfeit identities in the P2P network that most of the Sybil-tolerant solutions are able to cope with [12,13].

4.1. Time interval Δt

The idea of the analysis presented in this subsection (and also in the following one) is to gradually adjust the parameters of the mechanism. Once adjusted, these parameters will serve as basis for the in-depth analysis of our mechanism under various characteristics of the environment (e.g. increasing number of sources and computing power of the attacker), and considering the real traces of identity requests.

In this subsection, we evaluate how the time interval Δt influences the performance of the proposed mechanism in tackling Sybil attacks launched by an attacker using various distinct locations (sources). As the goal of the attacker is controlling 1/3 of the identities in the system, the total number of counterfeit identities to be requested is set to 82,425 ($M_r = 82{,}425$). We initially focus on the scenario where no puzzles are assigned prior to granting identities (no control). Then, we concentrate on the effectiveness of our mechanism in choking attacks (and on the undesired impact caused to legitimate requests). For the sake of comparison, we also show results obtained with static puzzles [9].

In this analysis (and in the following one), the $L_u = 160{,}000$ legitimate users are evenly distributed along 10,000 sources ($S = 10{,}000$); this makes a total 16 legitimate users per source ($L_s = 16$). The number of legitimate requests per source follows an exponential distribution, with $\lambda_r \approx 0.0634$ and bounded between 16 (making at least one request per legitimate user) and 128 (making eight requests per user). The *first arrival* of an identity request of each source is normally distributed during the simulation period, with $\mu_f = 302{,}400$ (half of the week) and $\sigma_f = 100{,}800$; the *time interval* between requests of each source ranges from 1 min to 2 h, and follows an exponential distribution (with $\lambda_i \approx 9.94 \times 10^{-4}$). It is important to highlight that the choice for these distributions (exponential for the number of requests per source, normal for the first arrival of a user's request, and exponential for the time between requests) was based on an analysis of real traces of identity requests from a BitTorrent community [22]. These traces also serve as basis for the experimental evaluation presented in Section 5.

For requesting the counterfeit identities, the attacker employs 10 high-performance computers ($M_c = 10$), each eqquiped with a hardware 2.5 times faster than the reference one (the most powerful hardware considered in our evaluation). The attacker controls 10 of the legitimate sources ($M_u = 10$), and uses them to originate her $M_r = 82{,}425$ identity requests; each of these sources originate around 8,242 malicious requests, one every 73.3 s approximately $\left(\frac{604{,}800}{82{,}425} \times 10\right)$. We consider such a request rate since it represents the best choice for the attacker – given the design of our mechanism, and the exponentially increasing complexity of puzzles. Otherwise, some malicious requests would receive even lower trust scores (and therefore, cope with extremely complex puzzles), should the attacker choose a different balance of requests per source per unit of time (rather than evenly dividing requests among sources, and throughout time). Recall also that an attacker is able to manipulate the network recurrence rate iff she manages to quickly solve the assigned puzzles (regardless of their complexity).

For this analysis we consider the following values of Δt: 8, 16, 24, 48, and 96 h. The smoothing factor is set to *0.125* ($\beta = 0.125$). The mechanism based on static puzzles [9] uses a puzzle with 2^9 complexity units, which takes between 3 and 85 min to solve depending on the computer capacity.

Fig. 3a shows that, without being required to solve computational puzzles (curve "No control"), an attacker is able to obtain all 82,425 identities in 168 h (the *y-axis* is limited in the figure for the sake of legibility). When using static puzzles, the number of counterfeit identities created within the same period is decreased to 28,390 (a reduction of 65.55%). The use of adaptive puzzles changes this scenario significantly: no more than 910 identities are granted to the attacker during the same period of 168 h, when a sliding window of 8 h is used (curve "$\Delta t = 8$"). This represents

a reduction of 98.89% when compared to the no control scenario, and an improvement of 96.79% over the use of static puzzles. The reason for such a poor attack performance when our mechanism is employed is that the average trust score of the attacker's identity requests is extremely low, thus resulting in the assignment of extremely complex puzzles. Note from Fig. 3b, for example, that only approximately 10.98% of the malicious requests received a trust score higher or equal than 0.3. This proportion falls to 2.07% if we consider only those requests for which a trust score of 0.5 was calculated. No malicious requests were assigned a score higher than 0.5. Such low trust scores may be explained by the recurrence rate of each malicious source, which is comparatively higher than the average recurrence rate of the network.

Observe from the plots in Fig. 3a that the higher the duration of the sliding window, the more restrictive the proposed mechanism becomes for the attacker. Increasing the sliding window from $\Delta t = 8$ to $\Delta t = 96$ results in a gain as high as 51.86% (i.e. from 910 counterfeit identities created over 168 h to only 438 identities). The reason is that a larger duration for the time interval Δt makes a higher fraction of the history of sources to be considered when calculating their recurrence rates $\Delta\phi$ (as discussed in Section 3.2). Such an increase in their recurrence rates has two effects. First, $\Phi(t)$ also increases, and the mechanism becomes more conservative when computing trust scores to those requests coming from sources having higher $\Delta\phi$ (see *Prop. 3* in Section 3.3). And second, the $\Delta\phi$ of sources in hands of the attacker, which is already high, becomes even larger; consequently, even lower trust scores are assigned to future requests originating from them.

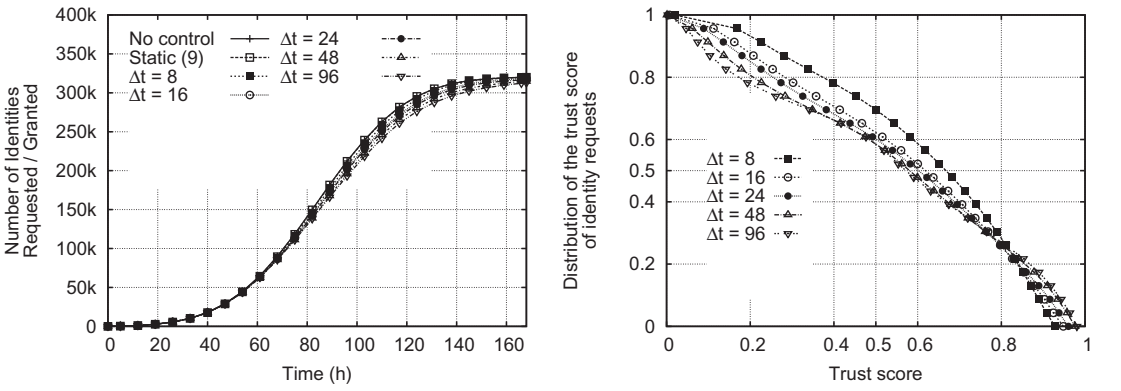
After evaluating the effectiveness of the proposed mechanism in limiting the spread of counterfeit identities, we now focus on the overhead caused to the identity requests originated from legitimate sources. From the curves shown in Fig. 4a, it is clear that the overhead to legitimate users was negligible in all scenarios, regardless of the interval Δt used; and more importantly, it was in general similar to the overhead caused to legitimate requestes when static puzzles are used. This is because the majority of identity requests from legitimate sources received a value of trust score higher or equal to 0.5 (above 60% in all cases, as shown in Fig. 4b). In addition, as legitimate sources requested only a few identities each, they were able to cope with (and solve more quickly) puzzles having higher complexity – in contrast to malicious sources, which had a large number of puzzles to solve using a limited number of computers.

One may note from Fig. 4b that the overall trust score of identity requests decreases (though marginally) as the duration of the sliding window Δt increases, thus resulting in a higher overhead to legitimate users. The reason for such a behavior follows the same logic as the one previously explained for the case of malicious requests. On the one extreme, lower values of Δt cause the proposed mechanism to consider a smaller fraction of the recent history of sources. Therefore, with fewer identity requests being considered, the network recurrence rates tend to become lower, and thus higher trust scores are assigned to future identity requests. On the other extreme, higher values of Δt make the proposed mechanism to consider a larger fraction of the recent history. Consequently, the network recurrence rate becomes higher, and the mechanism becomes more conservative when computing trust scores.

The appropriate setting of the sliding window strictly depends on the nature of the P2P application, and on what is considered a reasonable behavior for users of the application in question. For an application whose users typically obtain few identities a day (e.g. four identities – one per login session), the use of $\Delta t = 48$ (or $\Delta t = 24$) is appropriate, since it will refrain malicious users from attempting to obtain a significantly higher number of identities than normally expected. The lower an ordinary user is expected to obtain identities from the bootstrap service, the higher Δt should be, and vice versa. For the analysis presented in the following subsections, we chose to focus on a value of $\Delta t = 48$, as it is more representative of the file sharing application explored in this paper.

4.2. Smoothing factor (β)

In this subsection we focus on the effect that varying values for the smoothing factor β causes to the identity



(a) Legitimate identities granted (b) CCDF of the trust score of legitimate sources

Fig. 4. Impact of the proposed mechanism to legitimate identity requests, under different sizes for the sliding window Δt.

requests of these sources. We concentrate on the following values of β: 0.125, 0.25, 0.5, 0.75, and 1. For this set of experiments, Δt is defined as 48. The number of sources the attacker controls remains unchanged ($M_u = 10$), as well as the number of high-performance workstations used to solve the assigned puzzles ($M_c = 10$). The other characteristics of the attack, as well as other environment-related settings, are kept at their original values. Again, for the sake of comparison, we include the results achieved without control, and with the use of static puzzles.

Fig. 5a presents the dynamics of creation of counterfeit identities over the period of 168 h, in each of the evaluated scenarios. Observe that increased values of β improve the performance of our mechanism, and the total number of counterfeit identities created is decreased from 478 (with $\beta = 0.125$) to 315 (with $\beta = 1.0$). One may also observe that regardless of the value of β used, the performance of our mechanism in tackling the attack is several orders of magnitude higher than what is achieved using static puzzles. In regard to legitimate users, Fig. 5b evidences that the overhead measured with our mechanism is low (between 1.6% and 2.4% of legitimate requests are not granted an identity during 168 h, with β ranging from 0.125 to 1.0), and yet similar to the overhead caused when using static puzzles. As β increases, so increases the negative impact of the proposed mechanism to the trust score of legitimate requests. In spite of this, the proportion of requests assigned with a trust score higher or equal to 0.5 was above 45% in all scenarios, and above 62% in the particular case of $\beta = 0.125$.

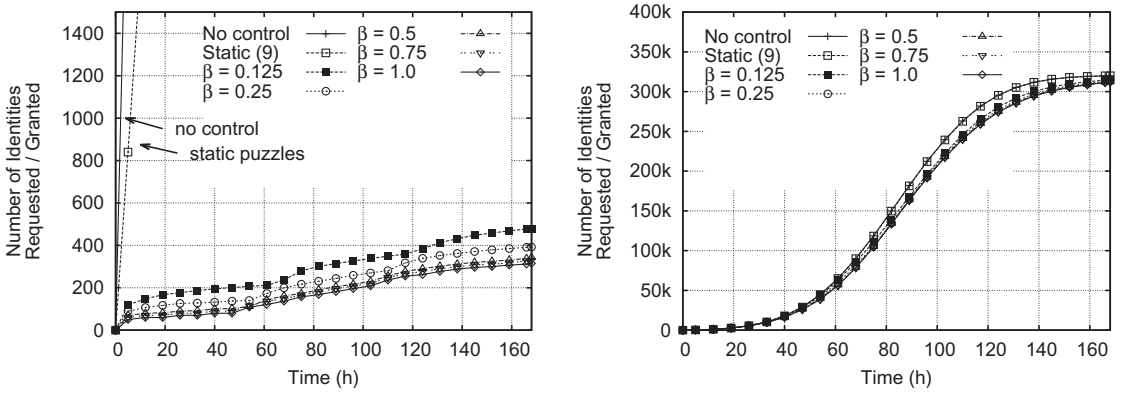
In principle, the higher the value of β, the more restrictive the proposed mechanism becomes for both legitimate and malicious requests. To understand this, first recall (from Section 3.4) that higher values of β assign higher weight to the instantaneous trust score $\theta_i(t)$ (upon the calculation of the smoothed trust score $\theta'_i(t_k)$); conversely, lower values of β assign higher weight to the historical behavior of $\theta_i(t)$. Recall also that both $\Delta\phi_i$ and $\Phi(t)$ may vary significantly through time, either because of newly granted identities or because of gradual disposals of older grants from the sliding window Δt, thus making the instantaneous trust score $\theta_i(t)$ to experience high fluctuations. Therefore, the higher values of β make the smoothed trust score to reflect even more the fluctuations seen in the instantaneous trust score, which ultimately results in lower trust scores (and thus puzzles of higher complexity) being eventually assigned to identity requests. As a consequence, the total number of identities created over 168 h is decreased. This observation holds mainly for the case of the attacker, who has a limited set of resources and requests a proportionally higher number of identities. In contrast, lower values for β reduce this fluctuation of $\theta'_i(t_k)$. Therefore, identity requests are assigned with lower trust scores (and thus puzzles of higher complexity) only if the source from which they originate maintains the discrepant behavior of requesting a higher number of identities (in regard to the network) for a longer period.

Given the discussion above, we chose a value of $\beta = 0.125$ for the analyses that follows, as it makes the proposed mechanism more robust to sources that continuously request more identities than the network average, whereas allows a legitimate user to request more identities during a trasient failure (e.g. unstable network connectivity) without being penalized for that.

4.3. Sharing legitimate sources with an attacker

In the previous subsections, we provided a detailed analysis on the means the proposed mechanism may be adjusted to undermine the attacker's ability of creating counterfeit identities. In summary, the results showed that the effectiveness of launching a Sybil attack is drastically reduced, even when compared to the results achieved when static puzzles [9] are used. Now we concentrate our evaluation on the impact that varying environment-related settings may have over the proposed mechanism.

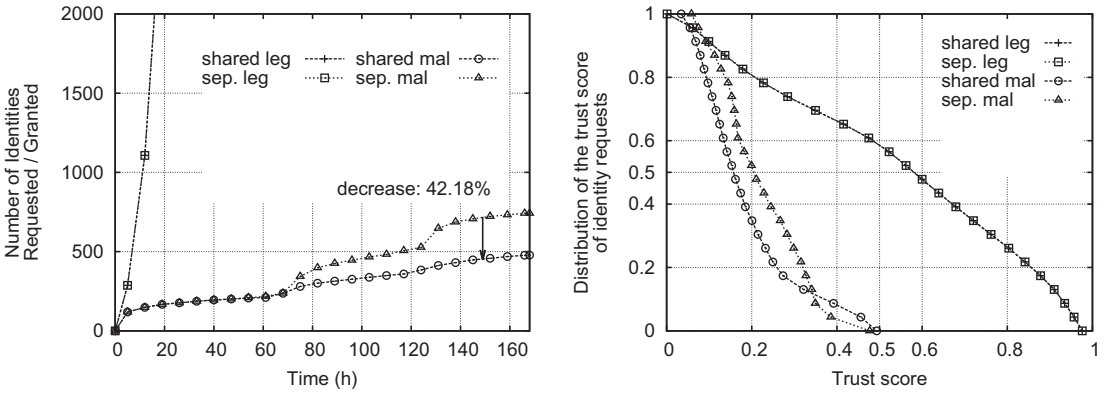
In this analysis, we compare the performance of our mechanism when malicious requests depart from sources shared with legitimate users, in contrast to a scenario in which the attacker uses her own sources to request identities. In practice, the first scenario could correspond to an attacker behind networks using NAT, in which other



(a) Counterfeit identities created

(b) Legitimate identities granted

Fig. 5. Effectiveness of the proposed mechanism under various values for smoothing factor β.



(a) Identities granted (b) Trust score of identity requests

Fig. 6. Effectiveness of the proposed mechanism considering sharing of sources.

legitimate users can also be found. Conversely, the second scenario corresponds to an attacker using its own set of valid IP addresses to request the counterfeit identities.

The number of sources the attacker controls and the computing power available to solve the assigned puzzles remains the same, i.e. $M_u = 10$ and $M_c = 10$. Note that in the first scenario (shared sources), there are $S = 10{,}000$ sources in the system (10 out of them are used for both legitimate and malicious requests), whereas in the latter (separated sources) there is a total of $S = 10{,}010$ sources. Finally, we adopt the following parameter setting for the proposed mechanism: $\Delta t = 48$, and $\beta = 0.125$.

Fig. 6 shows that sharing sources with legitimate users clearly degrades the effectiveness of the attack. In the scenario in which the attacker uses her own sources (curve "sep. mal"), 742 counterfeit identities are created. This number is reduced in 35.57% (to 478 identities) when the malicious requests depart from sources shared with legitimate users (curve "shared mal"). Although shared sources also impact the legitimate users associated to them, this impact is restricted, as legitimate users request very few identities each: only 0.001% of identity requests (4 identities) were not granted when sources are shared. From the experiment above, only 10 sources (out of 10,000) were shared. Given that there is an average of 16 legitimate users per source (as discussed in Section 4.1), only 160 users (out of 160,000) – or 320 identity requests (out of 320,000) – were directly affected. Focusing on the trust score of identity requests, observe from Fig. 6b that the score of legitimate requests remains virtually unchanged, whereas malicious requests are largely affected (even though a few requests obtain better values of trust score).

The reason for the poorer performance of the attack is the increase in the recurrence of sources from which malicious requests depart, in case the attacker shares sources with legitimate users. To understand the degradation of the attack performance, consider the scenario where legitimate users do not share sources with an attacker; in this scenario, the recurrence rate of a source that originates legitimate requests only is $\Delta\phi_l = x$, and the recurrence rate of a source originating malicious requests is $\Delta\phi_m = y$ (with $x, y \in \mathbb{N}$ and $x, y > 0$). In the scenario in which the attacker shares sources with legitimate users, the recurrence of a single source originating both legitimate and malicious requests is now $\Delta\phi_{l+m} = x + y$. This increased recurrence rate ultimately results in lower values of trust scores assigned to requests coming from this source – and consequently puzzles of higher complexity. While these lower values of trust scores affect both legitimate and malicious requests, it is worth noting that a legitimate user is minimally penalized, as it performs only a few identity requests. An attacker, in contrast, should be more penalized as her number of identity requests will be significantly larger (in compliance with the goal of launching a Sybil attack into the network). This indicates that sharing sources with legitimate users is not a good strategy for the attacker – on the contrary, she might even have the effectiveness of her attack severely affected.

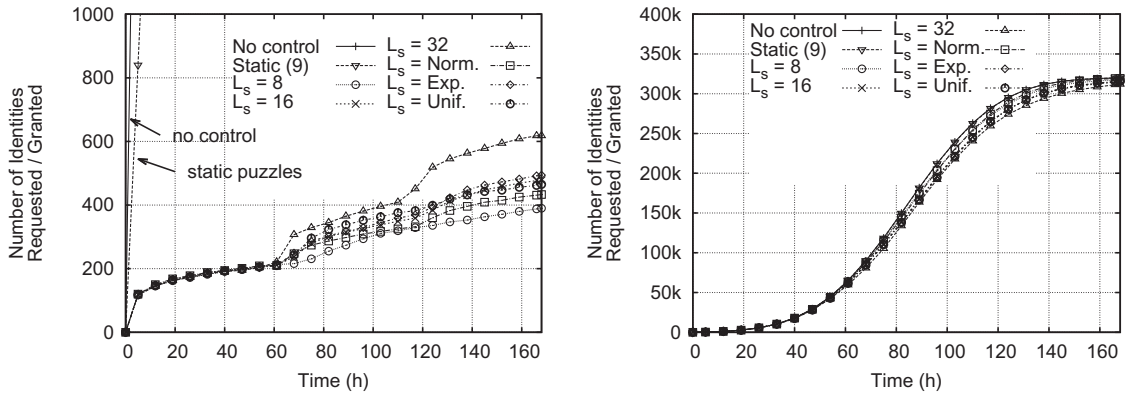
4.4. Size of sources of identity requests (L_s)

As discussed in Section 3.1, a varying number users may be associated to a single source of identity requests – therefore resulting in sources of varied sizes. This is the case of legitimate users behind networks using NAT, when the network addressing scheme is used to delineate sources; and also the case of users located in a same geographical location (e.g. same building), when network coordinate systems are used to distinguish the sources. In order to evaluate the impact of such a situation (various users behind a same source) on the performance of the proposed mechanism, we carried out several experiments considering different sizes of sources, and also different distributions for sources sizes. In this subsection we discuss the most prominent ones.

For this analysis, we considered six scenarios: three in which the size of sources is fixed, and other three in which it varies following a given distribution. Table 2 presents in more detail the characteristics of each scenario. For the sake of evaluation, in this evaluation (and also in the following ones) malicious requests depart from sources shared with legitimate users. Recall from Section 4.1 that

Table 2
Characteristics of the sources in each of the evaluated scenarios.

Scenario	Number of Sources	Distribution of sources size	Users per source	Distribution of sources requests	Requests per source
$L_s = 8$	$S = 20{,}000$	Fixed	8	Exponential ($\lambda \approx 0.03168$)	Between 32 and 256
$L_s = 16$	$S = 10{,}000$	Fixed	16	Exponential ($\lambda \approx 0.06337$)	Between 16 and 128
$L_s = 32$	$S = 5000$	Fixed	32	Exponential ($\lambda \approx 0.12674$)	Between 8 and 64
$L_s = Norm.$	$S = 10{,}000$	Normal ($\mu = 15$, $\sigma = 5$)	Between 1 and 31	Exponential ($\lambda \approx 0.14787$)	between 16 and 64
$L_s = Exp.$	$S = 10{,}000$	Exponential ($\lambda \approx 0.1126$)	Between 1 and 64	Exponential ($\lambda \approx 0.08872$)	Between 16 and 96
$L_s = Unif.$	$S = 10{,}000$	Uniform	Between 1 and 31	Exponential ($\lambda \approx 0.14787$)	Between 16 and 64



(a) Counterfeit identities created (b) Legitimate identities granted

Fig. 7. Effectiveness of the proposed mechanism under varying source sizes.

the choice for an exponential distribution for the number of requests per source (column "Distribution of sources requests" in Table 2) is based on an analysis of sources recurrences in real traces of identity requests from a BitTorrent community [22].

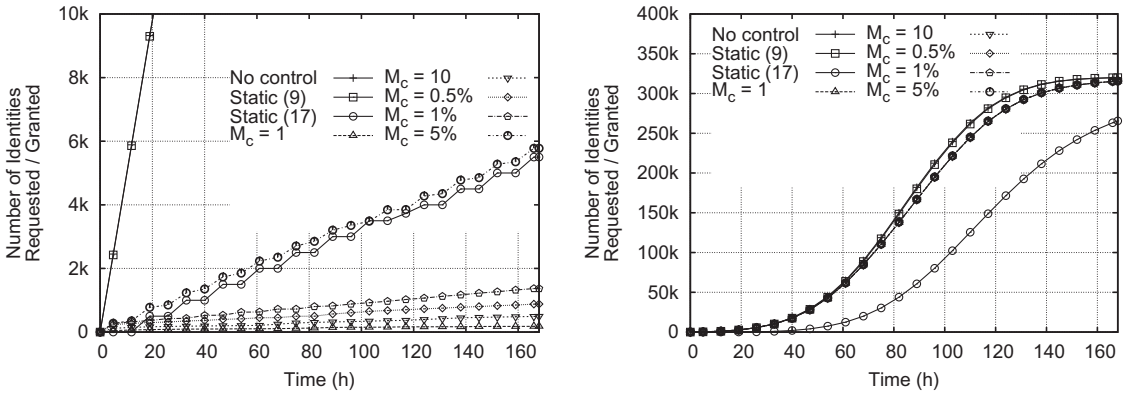
Fig. 7a shows that the higher the average size of sources is, the better for the attacker. In the scenario "$L_s = 8$" (8 legitimate users behind each source), the attacker was able to obtain 390 identities only. This number increased to 618 (or 58.46%) in the scenario "$L_s = 32$". This is because of the higher average of source recurrence rates ($\Delta\phi_i$, for the ith source), which in turn increase the network recurrence rate $\Phi(t)$; as a consequence, the sources in hands of the attacker become less suspicious (as the relationship between source and recurrence rate $\rho_i(t)$ decreases), and puzzles of lower complexity are assigned to requests coming from them. It is important to highlight that, despite such increase, the attacker remained far from achieving the desired number of identities ($M_r = 82{,}425$). With regard to the scenarios in which the size of sources followed a distribution, the results achieved were in general similar to the scenario "$L_s = 16$". As for the impact to the requests of legitimate users, again it remained marginal – as one can see from Fig. 7b. These results, although not exhaustive, suggest that the proposed mechanism should perform consistently regardless of the arrangement of users among sources in the system.

4.5. Increasing the computing power of the attacker (M_c)

In this analysis, we measure the effectiveness of our mechanism when the attacker increases the computing power available to solve the assigned puzzles. The number of sources the attacker uses to request identities remains the same ($M_u = 10$). An analysis considering an increasing number of sources in hands of the attacker is shown in the following subsection. For the sake of this evaluation, we consider the same number of sources, distribution of sizes of sources, and distribution of recurrence of sources considered in the scenario "$L_s = Exp.$" (from the previous subsection). The choice for this scenario is supported on a recent study by Maier et al. [23], which suggests that the number of unique end-hosts behind networks using NAT – in residential DSL lines of a major European ISP – follows an exponential distribution.

The total number of counterfeit identities to be requested is again 82,425 ($M_r = 82{,}425$). For this attack, we consider the availability to the attacker of the following number of high-performance computers (M_c): 1, 10, 0.5% (with regard to the number of sources S, i.e. 50 computers), 1% (100 computers), and 5% (500 computers). We also present results achieved with static puzzles, considering two scenarios: one with puzzles of 2^9 complexity units, and another with puzzles of 2^{17} complexity units (which take between 14 and 364 h to solve, depending on the computer capacity). In both cases, the computing power of the attacker is the highest considered ($M_c = 5\%$ high-performance computers). Finally, we adopt the following parameter setting for the proposed mechanism: $\Delta t = 48$, and $\beta = 0.125$.

Fig. 8a shows that the more computing power the attacker has, the higher number of identities it obtains from the system. However, even when 500 high-performance computers are used (curve "$M_c = 5\%$"), the attacker is not



(a) Counterfeit identities created (b) Legitimate identities granted

Fig. 8. Legitimate and malicious identity requests granted, considering a varying computing power of the attacker.

able to repeat the same performance as in the "No control" scenario, and only 5,598 counterfeit identities are created (a reduction of 93.2%). In the scenario in which the attacker is extremely limited in resources ($M_c = 1$), only 184 identities are created during the period of 168 h. The reason for such a poor attack performance is that the recurrence rate of the sources in hands of the attacker becomes extremely high. Recall that the attacker controls (in this experiment) only $M_u = 10$ sources; consequently, the average number of malicious requests per source is $\frac{82,425}{10} \approx 8242$. On one hand, with a very limited computing power available (e.g. $M_c = 1$ high-performance computer), the rate in which the attacker solves the assigned puzzles is slower; therefore, the trust score of subsequent malicious requests does not drop significantly. On the other hand, with a higher computing power available (e.g. $M_c = 5\%$, or 500 high-performance computers), puzzles are solved more quickly, in a first moment. The recurrence rate of the sources controlled by the attacker then becomes extremely higher (than the average network recurrence rate). As a consequence, even more complex puzzles will be assigned to future malicious requests – therefore undermining the attackers' hability to keep solving puzzles quickly.

There are two important observations regarding the results shown in Fig. 8. First, the gain obtained by the attacker – when our mechanism is used – increases almost linearly with the number of high-performance computers employed in the attack. As shown in Fig. 8a, with $M_c = 0.5\%$ the attacker is able to obtain 881 identities. This number increases to 1369 identities with $M_c = 1\%$, and to 5778 with $M_c = 5\%$. This behavior indicates that the attacker, in order to improve the performance of the attack, must dedicate a directly proportional amount of computational resources to solve the assigned puzzles. In other words, higher success rates comes at even higher (and increasing) costs for the attacker.

The second observation is that the mechanism based on static puzzles is completely taken over (i.e. becomes uneffective) when easier-to-solve puzzles are assigned to requests (curve "Static (9)" in Fig. 8a). Conversely, it only presents a similar performance to our mechanism for the worst-case scenario ($M_c = 5\%$), when an extremely difficult puzzle – with 2^{17} complexity units – is used (curve "Static (17)"). Observe from Fig. 8b, however, that such a puzzle imposes a very high cost for legitimate users as well – and 54,469 legitimate requests are not granted an identity within the period of 168 h. In the case of the proposed mechanism, the identity requests of legitimate users experience a marginal impact, despite the computing power the attacker has available. This observation highlights the major trade-off involved with the use of static puzzles to mitigate Sybils, between effectiveness (in preventing the indiscriminate creation of counterfeit identities) and flexibility (to legitimate users).

In summary, from the discussion presented in this subsection, two main conclusions may be drawn. First, simply increasing the computing power dedicated for the attack is not sufficient for circumventing the proposed mechanism, as the higher recurrence rate of malicious sources becomes a bottleneck for the success of the attack. And second, the negative impact that the proposed mechanism caused to the requests of legitimate users was negligible, even when the attacker had an extremely high computing power available. In the following subsection, we evaluate how increases in the number of malicious sources impacts both the number of counterfeit identities the attacker obtains, and the requests of legitimate users.

4.6. Varying proportions of malicious sources (M_u)

In this analysis, we evaluate the performance of the proposed mechanism in a different scenario: instead of solely increasing the computing power available (using a high-performance cluster), the attacker controls a botnet and uses its machines to request identities and process the associated puzzles. The total number of counterfeit identities to be requested is again 82,425 ($M_r = 82,425$). For this attack, we consider the availability to the attacker of a botnet of high-performance computers, each associated to its own source, having the following sizes (M_u): 1, 10, 0.5% (with regard to the number of sources S, i.e. 50 zombie machines), 1%, and 5%. It is important to keep in mind that

M_u = 5% is a very extreme case, in which still makes sense (for the attacker) launching a Sybil attack. Should the attacker have a higher number of workstations connected to the Internet, she would already have outnumbered legitimate users connected to the P2P network, and launching a Sybil attack would not be necessary.

Observe that the main difference of this analysis in contrast to the described in the previous subsection is that each computer in the botnet has its own source; in the previous analysis, the number of sources was limited to M_u = 10 regardless of the number of computers used. For the sake of comparison, we also include the results achieved with static puzzles (for the scenarios with puzzles of complexity 2^9 and 2^{17}).

In the case of M_u = 0.5% (a botnet of 50 zombie machines), around 1648 malicious requests originate from each machine/source, i.e. a request every 366 s, per source. Analogously, in the case of M_u = 1%, around 824 requests are performed from each machine/source in the botnet; a request every 733 s, per source. In the extreme case of M_u = 5%, 164 requests are performed per source, one every 3665 s (1 h). Recall that the lower the number of identities requested by a machine and the higher the interval between requests, the better the trust score of requests coming from that machine, and consequently the lower the complexity of the puzzle to be solved upon each request. All other properties of the analysis remain the same.

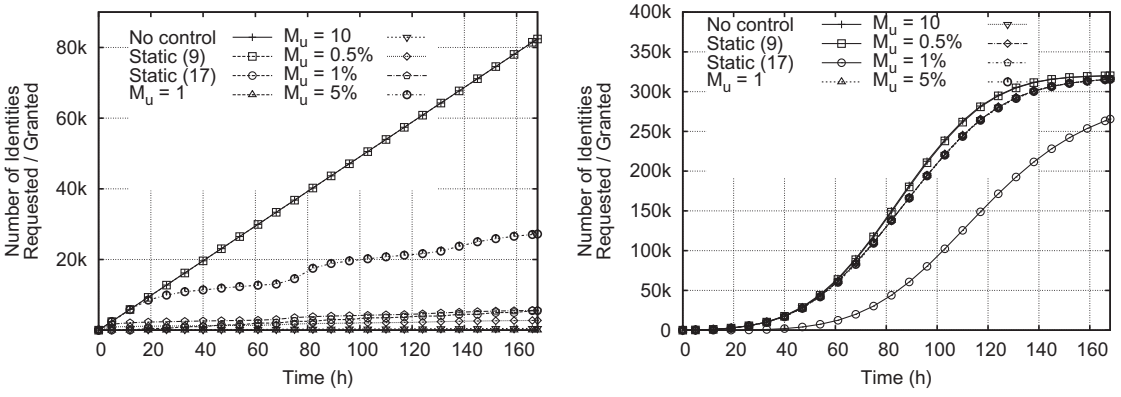
Fig. 9a shows that the attacker is able to create only 53 counterfeit identities when using only one machine (curve "M_u = 1"). As the number of zombie machines used in the attack increases, so increases the number of counterfeit identities created. With M_u = 10, a number of 494 identities are created. This number increases linearly with the amount of resources dedicated to the attack (as seen in the previous subsection): 2760 identities with 50 machines ("M_u = 0.5%"), 5624 identities with 100 machines ("M_u = 1%"), and 27,209 with 500 machines ("M_u = 5%"). In spite of the increased performance, however, the attacker does not achieve the same success as in the "No control" scenario. In the extreme case where 500 zombie machines are used (curve "M_u = 5%"), the attacker only achieves 33% of its goal during 168 h. Now focusing on the overhead caused to legitimate users, Fig. 9b shows that it remained stable and minimal. These results not only evidence that our mechanism is able to limit the creation of counterfeit identities, but also shows its better performance (specially under extreme conditions, for which launching Sybil attacks still makes sense for the attacker) when compared to the mechanism based on static puzzles.

5. Evaluation using real life traces of identity requests

Having evaluated our mechanism using a synthetically generated trace, and analyzed its behavior under the influence of various environment and parameter settings, we now focus on its performance in a real life scenario. For the sake of comparison, we also show results obtained with static puzzles [9] and when control is absent (no control scenario). We attempt to answer two fundamental questions with this analysis. First, how well does the proposed mechanism undermines the attacker's ability of getting hold of counterfeit identities across the P2P network? And second, how large is the overhead of our mechanism to legitimate users (specially in scenarios without attack)? In order to answer these questions, we have both evaluated and compared scenarios with and without attack, for each of the mechanisms analyzed. In the following subsections we cover the settings employed for the evaluation (Section 5.1) and the results achieved (Section 5.2).

5.1. Evaluation settings

The traces used in the evaluation were extracted from a log file from a BitTorrent community that register sessions of users' participating in torrent swarms. Each line in this log (whose excerpt is illustrated in Table 3) represents a session in the swarm, which is identified by the torrent ID (1st field of the log line, anonymized). The log also reports in each line, among other information, the user participating in the session (identified by an anonymized IP address and port, in the 2nd field), and the session duration (the difference between the end and begin timestamps, which are located in the 3rd and 4th fields, respectively).



(a) Counterfeit identities created

(b) Legitimate identities granted

Fig. 9. Effectiveness of the proposed mechanism with varying proportions of malicious sources.

Table 3
Excerpt of the BitTorrent log file employed in the evaluation.

1	2579139627397792460|3111354312011519843|2006-01-01 00:32:49|2006-01-02 18:18:49|0.5981862991410045|LEECHER|no|0.0
2	−1976250767827230296|-3161433101217960484|2006-01-01 00:33:01|2006-01-02 18:19:01|0.21703508372128355|LEECHER|no|0.0
3	−594805617465889591 3|2855973049255676852|2006-01-01 00:37:00|2006-01-02 18:18:00|0.1325415990077356|LEECHER|no|0.0
4	−527687489399158853 2|-2553693720321829928|2006-01-01 00:38:22|2006-01-02 18:19:22|0.11622309767807151|LEECHER|no|0.0
5	1985783098817489872|-2553693720321829928|2006-01-01 00:38:31|2006-01-02 18:19:31|0.11705653824459557|LEECHER|no|0.0
6	−594805617465889591 3|7056542205054267382|2006-01-01 00:44:00|2006-01-02 18:18:00|0.20494416938186288|LEECHER|no|0.0
7	−803560730911763113 9|3691911145490117 34|2006-01-01 00:44:12|2006-01-02 22:54:51|0.16915404929047936|LEECHER|no|32.27539
8	1703653772333122919|-1891502691723097807|2006-01-01 00:46:30|2006-01-02 18:19:30|2.9296278965678595|LEECHER|no|0.0

Table 4
Characteristics of the traces used in the evaluation.

	Trace 1	Trace 2	Trace 3
First identity request date/time	Sunday January 1 00:00:47 2006	Wednesday February 1 00:00:00 2006	Wednesday March 1 00:01:34 2006
Last identity request date/time	Saturday January 7 20:53:09 2006	Monday February 6 14:48:08 2006	Tuesday March 7 23:59:49 2006
Total number of identity requests	203,060	738,587	545,134
Total number of sources of identity requests	44,066	50,512	47,968
Trace duration (h)	164.87	134.80	167.97
Average number of identity requests per minute	20.52	91.31	54.09
Lowest and highest source recurrences	1 and 273	1 and 521	1 and 134
Average number of identity requests per source	4.60808	14.62201	11.36453
Standard deviation of recurrences	4.57688	35.44363	15.44093
Mode and median of source recurrences	1 and 3	1 and 5	1 and 6
Harmonic mean of recurrences	2.39177	3.15381	3.23839

The methodology employed to extract the trace of identity requests from the information contained in this log is described as follows. First, observe that there may exist one or more overlapping sessions for a given user, as he may participate in multiple swarms during the usage of his BitTorrent client. This is the case, for example, of lines 4 and 5 from the trace excerpt shown in Table 3. There is a high probability that these multiple, overlapping sessions (when associated to a same IP address) actually refer to a single BitTorrent client used to download (or upload) multiple files. This is so because BitTorrent clients generate locally an identity when they are started, and use this same id to join any swarm until the client is terminated. Based on this assumption, the resulting sequence of identity requests is obtained by merging the overlapping sessions associated to a same IP address. To illustrate, lines 4 and 5 in Table 3 refer to a single identity request event, performed on 2006-01-01 00:38:22. By employing this methodology, we processed and generated three distinct traces, whose relevant characteristics for the evaluation presented hereafter are shown in Table 4. Our choice for more than one trace reflects our goal of evaluating our mechanism under an average identity request behavior that varies not only during a day or a week, but also across months.

For each of the traces used, three attack scenarios were evaluated: without attack, with $M_u = 1\%$, and with $M_u = 10\%$ malicious sources. The second scenario ($M_u = 1\%$) corresponds to an attacker with limited computing power to solving puzzles and possessing various distinct locations from which identity requests depart. In the third scenario ($M_u = 10\%$), the attacker has a botnet at his service and uses it to launch the attack. Note here that we increased even further the amount of resources in hands of the attacker, in contrast to the scenario evaluated in Section 4.6. We assume the worst case scenario for the botnet, in which all *zombie* machines forming it are equipped with the most powerful hardware considered in our experiments (i.e. 2.5 times higher than of the off-the-shelf, reference hardware). Table 5 describes these scenarios in more detail. For the computing power of legitimate users, it follows an exponential distribution ($\lambda_{cp} \approx 0.003$), ranging from 0.1 to 2.5 times the computing power of the reference hardware.

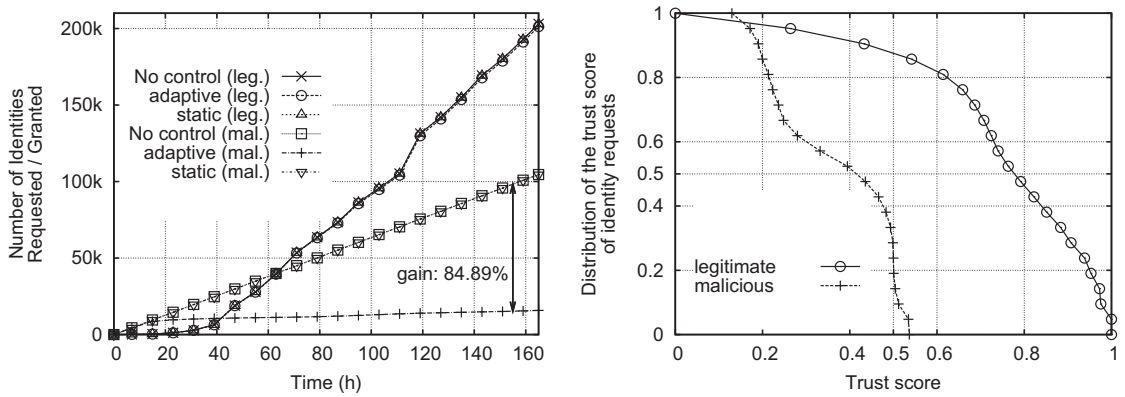
The specific parameters of each mechanism were defined as follows. The sliding window Δt is equal to 48 h, and the smoothing factor β is set to 0.125, values chosen according to the evaluation presented in the previous subsection. The mechanism based on static puzzles [9] uses a puzzle with 700 complexity units, which takes between 5 min and 2 h (approximately) to solve depending on the computer capacity.

5.2. Results achieved

Figs. 10–12 show the results obtained for each of the compared mechanisms, for Traces 1, 2, and 3, respectively. For the sake of clarity and space constraints, only the results obtained for the second attack scenario ($M_u = 1\%$) are shown in these figures.

Table 5
Characteristics of the scenarios evaluated for each trace.

	Trace 1			Trace 2			Trace 3		
Counterfeit identities requested	104,606			380,484			280,826		
Time between requests (sec)	5.67			1.27			2.15		
Attack scenarios evaluated	No attack	1%	10%	No attack	1%	10%	No attack	1%	10%
Amount of malicious sources	–	440	4,406	–	505	5,051	–	479	4,796
Requests per source (avg)	–	237.74	23.74	–	753.43	75.32	–	586.27	58.55
Time between requests per source	–	41.6 min	6.94 h	–	10.7 min	1.78 h	–	17.2 min	2.86 h
Number of attacker's computers	–	100	4,406	–	100	5,051	–	100	4,796
Comp. power per source (avg)	–	≈0.56	2.5	–	≈0.5	2.5	–	≈0.52	2.5



(a) Identities granted (b) Values of trust score of identity requests

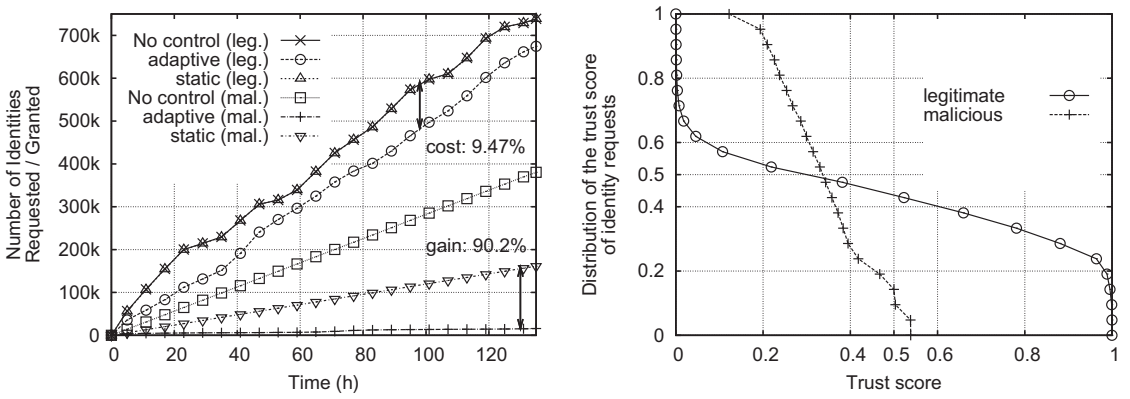
Fig. 10. Legitimate and malicious identities granted, and trust scores assigned (when the proposed mechanism is used), for each of the scenarios evaluated with Trace 1.

Observe from Fig. 10a that our mechanism outperforms both static puzzles [9] and the absence of control in limiting the creation of counterfeit identities. The proposed mechanism (curve "adaptive (mal.)" in Fig. 10a) reduced in 84.9% the number of counterfeit identities granted, in comparison to the "no control" scenario (curve "no control (mal.)"). This represented an effective gain of 84.89% over the scenario where static puzzles were used (curve "static (mal.)"); the use of such puzzles reduced marginally the number of granted counterfeit identities (0.05% only). Such a performance of the static puzzles is because the time required to solve them (5 min, in the case of the attacker) was overall smaller than the interval between identity requests. In regard to the overhead caused to legitimate users, observe from Fig. 10a that the curve "adaptive (leg.)" (and also "static (leg.)") overlaps with the curve "no control (leg.)", thus indicating that the imposed overhead was negligible.

We now focus on the trust scores assigned by the proposed mechanism. In Fig. 10b, the close the curves are from $x = 1$ and $y = 1$ axes, the higher the proportion of identity requests that received higher trust scores. Thus, it is clear that a higher proportion of legitimate identity requests received better trust scores, than malicious requests. Looking at these results in more detail, one may see that over 43% of legitimate requests were assigned a value of trust score higher or equal to 0.8. This proportion increases to 87% if we also consider those requests assigned with a trust score higher or equal to 0.5. This represents in fact that 87% of identity requests coming from legitimate users were not considered suspicious, and consequently received puzzles of lower complexity. Conversely, 78% of malicious requests were assigned with a value of trust score lower or equal to 0.5.

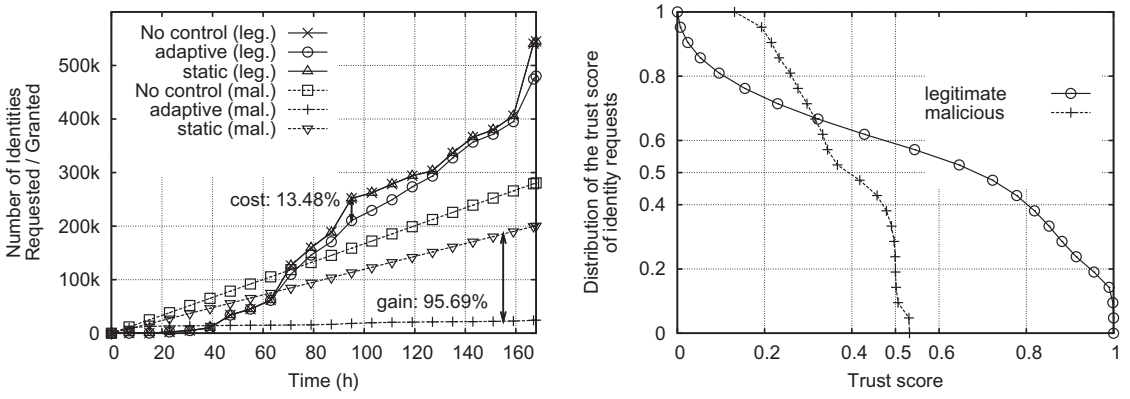
Table 6 presents a summary of the results achieved using Trace 1 as input, for each scenario evaluated. The table shows, per scenario evaluated, statistics for both *legitimate requests* and *malicious requests*, and *proportion* of counterfeit (malicious) identities over the total number of identities granted (recall that the goal of the attacker is to obtain 34% of identities in the network). The statistics (for both legitimate and malicious requests) include: (*i*) number of identities granted (field *granted*), (*ii*) proportion of identities granted in comparison to the "no control" scenario (field % "*no control*"), (*iii*) number of identity requests that were assigned a puzzle of lower complexity than the one given by the *Static Puzzles* mechanism [9] (<*static*), and (*iv*) proportion of requests that were assigned puzzles of lower complexity, in comparison to the total number of identity requests performed (% *requests*). For the sake of clarity, the results achieved with our mechanism are highlighted in gray.

Observe from Table 6 that our mechanism effectively prevented the creation of counterfeit identities across the



(a) Identities granted (b) Values of trust score of identity requests

Fig. 11. Legitimate and malicious identities granted, and trust scores assigned (when the proposed mechanism is used), for each of the scenarios evaluated with Trace 2.



(a) Identities granted (b) Values of trust score of identity requests

Fig. 12. Legitimate and malicious identities granted, and trust scores assigned (when the proposed mechanism is used), for each of the scenarios evaluated with Trace 3.

Table 6
Results obtained for each of the scenarios evaluated, for Trace 1.

Scenario	Legitimate requests				Malicious requests				Proportion
	Granted	% "No control"	<Static	% Requests	Granted	% "No control"	<Static	% Requests	
No attack, Static	202,889	99.92	–	–	–	–	–	–	–
No attack, Adaptive	200,869	98.92	161,064	79,32	–	–	–	–	–
M_u = 1%, Static	202,889	99.92	–	–	104,554	99.95	–	–	34.01
M_u = 1%, Adaptive	201,251	99.11	178,960	88,13	15,796	15.10	5,837	36.72	7.28
M_u = 10%, Static	202,889	99.92	–	–	104,554	99.95	–	–	34.01
M_u = 10%, Adaptive	201,545	99.25	186,593	91,89	95,678	91.47	22,694	22.69	32.19

network, in both scenarios with attack (M_u = 1% and M_u = 10%), whereas imposing a marginal overhead to legitimate requests. In contrast, the mechanism based on static puzzles did not cause any impact to malicious requests, and the attacker easily succeeded in obtaining 34% of identities in the network. Even in the worst case scenario (M_u = 10%), our mechanism prevented 8.53% of malicious requests from being granted a (counterfeit) identity. As previously discussed in Section 4.6, note that M_u = 10% is a very extreme case, in which still makes sense (for the attacker) launching a Sybil attack. Furthermore, the relative "success" comes at a very high cost to the attacker, who either has to control a large number of high-performance, *zombie* workstations in the Internet, or possess a large number of high-performance computers dedicated to solving puzzles.

Table 7
Results obtained for each of the scenarios evaluated, for Trace 2.

Scenario	Legitimate requests				Malicious requests				Proportion
	Granted	% "No control"	<Static	% Requests	Granted	% "No control"	<Static	% Requests	
No attack, Static	738,443	99.98	–	–	–	–	–	–	–
No attack, Adaptive	674,426	91.31	311,827	42.22	–	–	–	–	–
M_u = 1%, Static	738,443	99.98	–	–	161,190	42.36	–	–	17.92
M_u = 1%, Adaptive	674,544	91.33	313,263	42.41	15,801	4.15	2190	13.77	2.29
M_u = 10%, Static	738,443	99.98	–	–	380,252	99.94	–	–	33.99
M_u = 10%, Adaptive	676,244	91.56	320,520	43.40	195,336	51.34	38,398	19.17	22.41

Table 8
Results obtained for each of the scenarios evaluated, for Trace 3.

Scenario	Legitimate requests				Malicious requests				Proportion
	Granted	% "No control"	<Static	% Requests	Granted	% "No control"	<Static	% Requests	
No attack, Static	544,921	99.96	–	–	–	–	–	–	–
No attack, Adaptive	478,694	87.81	286,173	52.50	–	–	–	–	–
M_u = 1%, Static	544,921	99.96	–	–	200,804	71.50	–	–	26.93
M_u = 1%, Adaptive	480,149	88.08	315,409	57.86	8,636	3.08	5837	23.89	1.77
M_u = 10%, Static	544,921	99.96	–	–	280,689	99.95	–	–	34.00
M_u = 10%, Adaptive	484,519	88.88	336,373	61.70	42,228	15.04	22,694	12.09	8.02

Another important metric for measuring the efficiency of our mechanism, presented in Table 6, is the number of puzzles of lower complexity – than the complexity of those used for the "static puzzles" mechanism – assigned to identity requests. Ideally, this number should be as high as possible for legitimate requests, and as low as possible for malicious requests. Note that the proportion of legitimate requests that received a puzzle of lower complexity ranged from 79.32% (no attack scenario) to 91.89% (scenario M_u = 10%). Conversely, the proportion of malicious requests that received a puzzle of higher complexity ranged from 36.72% to 22.69%. These results evidence that our mechanism not only prevented the spread of counterfeit identities across the network, but also distinguished satisfactorily between identity requests originated from correct users and attackers, and dealt with them accordingly.

The results obtained for Trace 2 and Trace 3, depicted in Figs. 11 and 12a, confirm the positive results our mechanism achieved, in contrast to existing solutions. In the case of Trace 2, while the attacker was able to obtain 161,190 counterfeit identities (over 135 h) when static puzzles were used, the use of our mechanism reduced this number to 15,801 identities; this represents a gain of approximately 90.2% over the results achieved with static puzzles, and a reduction of 95.84% in the number of counterfeit identities that would be granted (380,483) if no control was imposed. For Trace 3, the effective gain our mechanism provided was of 95.69%.

In regard to the overhead caused to the requests of legitimate users (cost), although higher than observed in the case of Trace 1, both were comparatively low. For Trace 2, only 9.47% of the identity requests coming from legitimate users were not granted, during the period of 135 h; for Trace 3, the measured overhead was 13.48%.

Now concentrating on the trust scores assigned to legitimate and malicious requests, Fig. 11b shows that approximately 40% of legitimate requests were assigned a value of trust score higher or equal to 0.6. No malicious request was assigned such a value of trust score, and only 10% of malicious requests were assigned a value of trust score higher or equal to 0.5. Fig. 12b, in turn, shows that approximately 60% of legitimate requests were assigned with a value of trust score higher or equal to 0.5, whereas approximately 86.19% of malicious requests were assigned a value of trust score lower or equal to 0.5.

Tables 7 and 8 clearly evidence that our mechanism outperforms the static-puzzle-based mechanism even in the worst case scenario, for both Trace 2 and Trace 3, respectively. While static puzzles were completely uneffective, our mechanism allowed the attacker to control at most 22.41% of identities in the system, in the case of Trace 2, and only 8.02% for Trace 3. Further, our mechanism assigned puzzles of lower complexity to at least 40% of legitimate requests (over than 50%, in the case of Trace 3); less than 25% of malicious requests (up to 20% in the worst case scenarios) received such a puzzle.

In summary, the results shown above not only evidence that adaptive puzzles are a promising solution for tackling the spread of Sybils, without causing severe restrictions to legitimate requests. More importantly, our mechanism showed to be effective in providing puzzles of higher complexity to malicious requests, and lower complexity to legitimate ones. The potentialities of our mechanism contrasts to what may be achieved assigning puzzles of same complexity to every request, regardless of their origin. Instead, static puzzles only bring the difficulty of finding an appropriate puzzle complexity that tackles the ongoing attack without severely penalizing legitimate users.

6. Considerations on the proposed mechanism

As shown in the previous sections, our mechanism provides substantial improvement over existing mechanisms based on static computational puzzles: while being able to limit even further the widespread dissemination of counterfeit identities, it causes a significantly lower overhead to legitimate users. In this section, we present some considerations regarding the deployment and operation of our mechanism in real P2P systems.

It is important to highlight that the deployment of our mechanism in real P2P systems is expected to require minimal changes to entities that compose the network. Considering the instantiation of the proposed mechanism in a P2P system such as BitTorrent [24], we envisage the need to slightly modify user agents and trackers so as to include the following interactions: (*i*) agent requests an identity to the tracker; (*ii*) tracker demands from the requesting agent the resolution of a puzzle; (*iii*) agent replies to the tracker with the solution to the proposed puzzle; and (*iv*) tracker checks the validity of the solution and, if correct, issues an identity to the agent. From this moment on, user agent and tracker interact using the in-place BitTorrent protocol. In this scenario, the most significant modifications would be restricted to the tracker – in such a way that it keeps information about sources' recurrence rates, compute their respective trust scores, and define the complexity of the computational puzzle to be solved as a function of the current trust score.

Regarding its scalability, observe that multiple bootstrap services could be deployed to eliminate the existence of a central point of failure. To this end, we envisage the same strategies proposed by Rowaihy et al. [10] to instantiate multiple certification entities in their mechanism based on static puzzles. For example, the bootstrap service could be replicated (holding the same public/private key pair) across multiple hosts. In this scheme, DNS redirection could be used to balance the load of identity requests among these hosts. Alternatively, a *master certification authority* could be designed to certificate multiple bootstrap services. In this scenario, a technique based on two random choices [25] could be used to balance the load of identity requests among them.

With respect to the parameters β and Δt of the proposed mechanism, updates of their values might be required to reflect long-term changes in the network behavior (for example, substantial increase in the number of users or changes in the community profile). In this case, periodic updates on the parameter values (in the order of months, for example) might be performed by the community administrator, taking into account his/her own experience, or with support of heuristics and/or automated mechanisms. It is important to mention, however, that adjustments of the parameters to adapt the proposed solution to shorter-term changes are not necessary; this is because of metrics *source recurrence rate* and *network recurrence rate*, which can capture variations in the behavior of the network that may occur during this period of time.

Another important discussion is related to the validity of a puzzle. An attacker may explore a period of time in which the source she is associated to achieves a high trust score, and request a potentially high number (e.g. millions) of identities. Because the trust score varies only with the number of identities *granted* (rather than the number of identities *requested*), the attacker would obtain millions of easy-to-solve puzzles in this scenario. She would then solve the received puzzles and obtain millions of identities at a significantly lower cost (in contrast to the case in which a new identity request comes only after the puzzle of the previous request is solved). To prevent this attack, puzzles may carry a validity timestamp, defined based on the time it is expected to be solved using a standard, off-the-shelf hardware. Once this timestamp is expired, the puzzle is no longer valid, and its solution may be refused by the bootstrap service. To remediate the case in which low capable machines do not timely solve the puzzles, another puzzle may be assigned, having a validity timestamp that is double the time it is expected to be solved using a standard hardware (and so on).

7. Final considerations

The use of computational puzzles is a promising alternative in tackling the occurrence of Sybil attacks in P2P networks. In spite of this, the lack of mechanisms that take advantage of the discrepant behavior observed between legitimate users and attackers, allied to a proper treatment of situations in which there is a gap of computing power between legitimate users and attackers, has hampered a more widespread and disseminated use of computational puzzles in the P2P realm. To address this limitation, in this paper we proposed the use of adaptive puzzles as a controlling factor to Sybil attacks.

The experiments carried out showed the efficacy of the proposed mechanism in decreasing the attackers' capabilities of creating an indiscriminate number of counterfeit identities, whereas not compromising legitimate users, which were, in general, minimally penalized. When computing lower trust scores to sources having higher recurrence rates, (malicious) users associated to these sources had to cope with more complex computational puzzles. Conversely, users associated to presumably legitimate sources (and that made fewer use of the bootstrap service to obtain new identities) received less complex computational puzzles (given the higher trust scores determined for the great majority of these sources in the P2P network).

As prospective directions for future research, we intend to (*i*) investigate a mechanism to support the proper assignment of values to the parameters of our mechanism, taking into account communities having distinct characteristics; and (*ii*) instantiate and evaluate the proposed mechanism as an extension of an existing P2P file sharing system, being BitTorrent a strong candidate.

Acknowledgements

The authors thank Prof. Nazareno Andrade (Federal University of Campina Grande, Brazil), for providing the historical traces of identity requests from Bitsoup.org used

in the experimental evaluation presented in this paper. The authors also thank Prof. Francisco Brasileiro (Federal University of Campina Grande, Brazil), John Douceur and Jacob Lorch (Microsoft Research Redmond, USA), and the various anonymous reviewers, for the valuable comments and suggestions that helped improving this work.

The research work presented in this paper was supported in part by funding from Project 560226/2010-1, Granted by CNPq (Edital MCT/CNPq No. 09/2010 PDI – Pequeno Porte).

References

[1] J.R. Douceur, The sybil attack, in: 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002), 2002, pp. 251–260.
[2] Z. Yang, C. Wilson, X. Wang, T. Gao, B.Y. Zhao, Y. Dai, Uncovering social network sybils in the wild, 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC'11), ACM, New York, NY, USA, 2011, pp. 259–268.
[3] M.P. Barcellos, D. Bauermann, H. Sant'anna, M. Lehmann, R. Mansilha, Protecting BitTorrent: design and evaluation of effective countermeasures against DoS attacks, in: Proceedings of the 2008 Symposium on Reliable Distributed Systems (SRDS '08), IEEE Computer Society, Washington, DC, USA, 2008, pp. 73–82.
[4] M.A. Konrath, M.P. Barcellos, R.B. Mansilha, Attacking a swarm with a band of liars: evaluating the impact of attacks on BitTorrent, 7th IEEE International Conference on Peer-to-Peer Computing (P2P '07), IEEE Computer Society, Washington, DC, USA, 2007, pp. 37–44.
[5] O. Jetter, J. Dinger, H. Hartenstein, Quantitative analysis of the sybil attack and effective sybil resistance in peer-to-peer systems, in: 2010 International Communications Conference (ICC 2010), Cape Town, South Africa, 2010, pp. 1–6.
[6] A. Singh, T.-W. Ngan, P. Druschel, D.S. Wallach, Eclipse attacks on overlay networks: threats and defenses, in: 25th Conference on Computer Communications (INFOCOM 2006), Barcelona, Catalunya, Spain, 2006, pp. 1–12.
[7] M. Feldman, C. Papadimitriou, J. Chuang, I. Stoica, Free-riding and whitewashing in peer-to-peer systems, IEEE Journal on Selected Areas in Communications 24 (5) (2006) 1010–1019.
[8] F.R. Santos, W.L. da Costa Cordeiro, L.P. Gaspary, M.P. Barcellos, Choking polluters in BitTorrent file sharing communities, in: 12th IFIP/IEEE Network Operations and Management Symposium (NOMS 2010), 2010, pp. 559–566.
[9] N. Borisov, Computational puzzles as sybil defenses, in: 6th IEEE International Conference on Peer-to-Peer Computing (P2P 2006), 2006, pp. 171–176.
[10] H. Rowaihy, W. Enck, P. McDaniel, T. La Porta, Limiting sybil attacks in structured P2P networks, in: 26th IEEE International Conference on Computer Communications (INFOCOM 2007), Anchorage, Alaska, USA, 2007, pp. 2596–2600.
[11] G. Danezis, P. Mittal, SybilInfer: Detecting sybil nodes using social networks, in: 2009 Network and Distributed System Security Symposium (NDSS 2009), The Internet Society, San Diego, California, USA, 2009.
[12] H. Yu, M. Kaminsky, P.B. Gibbons, A. Flaxman, SybilGuard: defending against sybil attacks via social networks, 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06), ACM Press, New York, NY, USA, 2006, pp. 267–278.
[13] H. Yu, P.B. Gibbons, M. Kaminsky, F. Xiao, SybilLimit: a near-optimal social network defense against sybil attacks, IEEE Symposium on Security and Privacy, IEEE Computer Society, 2008, pp. 3–17.
[14] N. Tran, J. Li, L. Subramanian, S. Chow, Optimal sybil-resilient node admission control, in: 30th IEEE International Conference on Computer Communications (INFOCOM 2011), Shangai, China, 2011, pp. 3218–3226.
[15] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, D.S. Wallach, Secure routing for structured peer-to-peer overlay networks, in: 5th Usenix Symposium on Operating Systems Design and Implementation (OSDI 2002), 2002, pp. 299–314.
[16] K. Aberer, A. Datta, M. Hauswirth, A decentralized public key infrastructure for customer-to customer e-commerce, International Journal of Business Process Integration and Management (2005) 26–33.
[17] R. Morselli, B. Bhattacharjee, J. Katz, M.A. Marsh, KeyChains: A Decentralized Public-Key Infrastructure, <http://hdl.handle.net/1903/3332>.
[18] W. Cordeiro, F.R. Santos, G.H. Mauch, L.P. Gaspary, M.P. Barcellos, Securing P2P systems from sybil attacks through adaptive identity management, in: Mini-conference Proceedings of the 7th International Conference on Network and Service Management (CNSM 2011), 2011, pp. 1–6.
[19] F. Dabek, R. Cox, F. Kaashoek, R. Morris, Vivaldi: a decentralized network coordinate system, SIGCOMM Computer Communications Review 34 (4) (2004) 15–26.
[20] M. Sherr, M. Blaze, B.T. Loo, Veracity: practical secure network coordinates via vote-based agreements, in: USENIX Annual Conference (USENIX '09), 2009.
[21] J. Liang, N. Naoumov, K.W. Ross, Efficient blacklisting and pollution-level estimation in p2p file-sharing systems, 1st Asian Internet Engineering conference on Technologies for Advanced Heterogeneous Networks (AINTEC'05), Springer-Verlag, Berlin, Heidelberg,, 2005, pp. 1–21.
[22] Bitsoup.org, Bitsoup.org – The Number One Site for Your Torrent Appetite, 2010, <http://bitsoup.org/>.
[23] Gregor Maier, Fabian Schneider, Anja Feldmann, NAT usage in residential broadband networks, in: Neil Spring, George Riley (Eds.), Passive and Active Measurement, Lecture Notes in Computer Science, vol. 6579, Springer, Berlin/Heidelberg, 2011, pp. 32–41.
[24] B. Cohen, Incentives Build Robustness in BitTorrent, in: 1st Workshop on Economics of Peer-to-Peer Systems, 2003, pp. 1–5.
[25] M. Mitzenmacher, A. Richa, R. Sitaraman, The power of two random choices: a survey of techniques and results, in: S. Rajasekaran et al. (Eds.), Handbook of Randomized Computing, vol. 1, Kluwer Academic Publishers., 2001, pp. 255–312.

Weverton Luis da Costa Cordeiro is a Ph.D. student at the Institute of Informatics of the Federal University of Rio Grande do Sul, Brazil. He holds a B.Sc. degree in Computer Science from the Federal University of Pará (2007), and a M.Sc. degree in Computer Science from the Federal University of Rio Grande do Sul (2009). His research interests include peer-to-peer systems, network management and security, Information Technology Service Management, and quality of service in wireless ad hoc/mesh networks.

Flávio Roberto Santos is a Ph.D. student at the Institute of Informatics of the Federal University of Rio Grande do Sul, Brazil. He holds a B.Sc. degree in Computer Science from the Federal University of Campina Grande (2007). His research interests include algorithms, peer-to-peer and live streaming systems, reputation mechanisms, and grid computing.

Gustavo Huff Mauch is a system analyst at Teracom Telemática Ltd. (DATACOM). He holds a M.Sc. degree in Computer Science from the Federal University of Rio Grande do Sul (2010), and a B.Sc. degree in Computer Science from the Federal University of Rio Grande do Sul (2003). His research interests include computer communications, network security, and peer-to-peer systems.

Marinho Pilla Barcellos received B.Sc. and M.Sc. degrees in Computer Science from Federal University of Rio Grande do Sul (1989 and 1993, respectively) and PhD degree in Computer Science from University of Newcastle Upon Tyne (1998). In 2003–2004, he worked in a joint project between University of Manchester and British Telecomm research labs on high-performance multicast transport. Since 2008 Prof. Barcellos has been with the Federal University of Rio Grande do Sul (UFRGS), where he is an Associate Professor. He has authored many papers in leading journals and conferences related to computer networks, network and service management, distributed systems, and computer security, also serving as TPC member and chair. He is the appointed chair of the Special Interest Group on Computer Security of the Brazilian Computer Society (CESeg/SBC) 2011–2012. He is a member of SBC, IEEE and ACM. His interests are security of peer-to-peer, virtualized and cloud-oriented networks. See http://www.inf.ufrgs.br/~marinho for further details.

Luciano Paschoal Gaspary received the Ph.D. degree in Computer Science from the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2002. In 2006, he joined the same institute, where he is now an associate professor. In addition to his appointment at UFRGS, Prof. Gaspary is currently director of the National Laboratory on Computer Networks (LARC) and administrative director of the Brazilian Computer Society (SBC). He is also a member of the IEEE and the ACM. Prof. Gaspary is author of more than 100 full papers published in leading journals and conferences related to computer networks, network and service management, and computer security. Furthermore, he has been directly involved in the organization and served as technical program committee member of several IEEE, IFIP and ACM conferences including IM, NOMS, DSOM, IPOM, GLOBECOM and ICC. See http://www.inf.ufrgs.br/~paschoal for further details.