Computer Networks 95 (2016) 97–114

Contents lists available at ScienceDirect

Computer Networks

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

Making puzzles green and useful for adaptive identity management in large-scale distributed systems

Weverton Luis da Costa Cordeiro[a,*], Flávio Roberto Santos[b], Marinho Pilla Barcellos[a], Luciano Paschoal Gaspary[a], Hanna Kavalionak[d], Alessio Guerrieri[c], Alberto Montresor[c]

[a] Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
[b] Chaordic Systems, Florianópolis, Brazil
[c] Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy
[d] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy

ARTICLE INFO

Article history:
Received 4 May 2015
Revised 23 October 2015
Accepted 18 December 2015
Available online 23 December 2015

Keywords:
Peer-to-peer networks
Identity management
Proof of work
Computational puzzles
Fake accounts
Sybil attack

ABSTRACT

Various online systems offer a lightweight process for creating accounts (e.g., confirming an e-mail address), so that users can easily join them. With minimum effort, however, an attacker can subvert this process, obtain a multitude of fake accounts, and use them for malicious purposes. Puzzle-based solutions have been proposed to limit the spread of fake accounts, by establishing a price (in terms of computing resources) per identity requested. Although effective, they do not distinguish between requests coming from presumably legitimate users and potential attackers, and also lead to a significant waste of energy and computing power. In this paper, we build on adaptive puzzles and complement them with waiting time to introduce a *green* design for lightweight, long-term identity management; it balances the complexity of assigned puzzles based on the reputation of the origin (source) of identity requests, and reduces energy consumption caused by puzzle-solving. We also take advantage of lessons learned from massive distributed computing to come up with a design that makes puzzle-processing *useful*. Based on a set of experiments, we show that our solution provides significant energy savings and makes puzzle-solving a useful task, while not compromising effectiveness in limiting the spread of fake accounts.

1. Introduction

Online systems such as Facebook, Twitter, Digg, and BitTorrent communities (among various others) offer a lightweight process for creating identities[1] (e.g., confirming a valid e-mail address; the actual requirements may vary depending on the system), so that users can easily join them. Such convenience comes with a price, however: with minimum effort, an attacker can obtain a multitude of *fake accounts*[2] (Sybil attack [1]), and use them to either perform malicious activities (that might harm legitimate

* Corresponding author. Tel.: +55 51 3308 9450.
E-mail addresses: weverton.cordeiro@inf.ufrgs.br, weverton.cordeiro@gmail.com, wlccordeiro@inf.ufrgs.br (W.L.d.C. Cordeiro), barata@chaordicsystems.com (F.R. Santos), marinho@inf.ufrgs.br (M.P. Barcellos), paschoal@inf.ufrgs.br (L.P. Gaspary), hanna.kavalionak@isti.cnr.it (H. Kavalionak), guerrieri@science.unitn.it (A. Guerrieri), alberto.montresor@unitn.it (A. Montresor).

http://dx.doi.org/10.1016/j.comnet.2015.12.005
1389-1286/

[1] In this paper, the terms "account" and "identity" are used interchangeably to refer to an informational abstraction capable of distinguishing users in a given system.

[2] We use the terms "fake account", "counterfeit identity", and "sybil" interchangeably to refer to those identities created and controlled by an attacker, and which are used with the purpose of harming the system and/or its users.

users) or obtain unfair benefits. The corruptive power of counterfeit identities is widely known, being the object of several studies in the literature [2–4].

It is extremely challenging (if not impossible) to devise a *one-size-fits-all* solution for identity management. As a consequence, the research community has focused on the design of system-specific solutions, in scenarios having a well-defined set of purposes, requirements, and constraints. In this paper, we approach the issue of fake accounts in large-scale, distributed systems. More specifically, we target those based on the peer-to-peer paradigm and that can accommodate lightweight, long-term identity management schemes [5] (e.g., file sharing and live streaming networks, collaborative intrusion detection systems); *lightweight* because users should obtain identities without being required to provide "proof of identity" (e.g., personal documents); and *long-term* because users should be able to maintain their identities (e.g., through renewal) indefinitely.

In the scope of these systems, strategies such as social networks [2,4,6,7] and proof of work (e.g., computational puzzles) [8–11] have been suggested as promising directions to tackle fake accounts. In spite of the potentialities, important questions remain. A number of investigations [12,13] has shown that some of the key assumptions on which social network-based schemes rely (e.g., and sybils form tight-knit communities) are invalid. More importantly, the use of social networks for identity verification might violate user's privacy. This is an extremely sensitive issue, especially because of the growing concern and discussion about privacy issues in social networks [14–16].

Puzzle-based schemes inherently preserve users' privacy (since no personal information is required to obtain identities) and therefore represent an interesting approach to stop sybils. Existing schemes focus on the users' computing power, and use cryptographic puzzles of fixed complexity to hinder attackers [8,9]. However, puzzle-solving incurs in considerable energy consumption, which increases proportionally to the system popularity and the interest of attackers in controlling counterfeit identities. Furthermore, users waste computing resources when solving puzzles. These aspects lead to the following research questions: 1) Is it possible to force potential attackers to pay proportionally higher costs than legitimate users for each identity they request? 2) Can one reduce resource consumption required for puzzle-solving, without compromising its effectiveness? 3) How could one take advantage of puzzle-solving to perform useful processing?

To tackle this issue, we build on adaptive puzzles [17] – a mechanism that defines puzzle complexity based on the frequency in which users (sources of requests) obtain new identities – and complement them with waiting time to introduce a *green* design for lightweight, long-term identity management. Our design balances the complexity of assigned puzzles based on the measured reputation of the source of identity requests, and also reduces energy consumption incurred from puzzle-solving (hence *green*). We also take advantage of lessons learned from massive distributed computing to come up with a design that makes puzzle-processing *useful* – it uses real-life data processing jobs in replacement to cryptographic puzzles.

To answer the research questions posed earlier, we evaluated our solution by means of simulation, analytically, and using the PlanetLab environment. The results obtained show that it provides significant energy savings and makes puzzle-solving a useful task, while being effective in limiting the spread of fake accounts.

The remainder of this paper is organized as follows. We briefly review related work in Section 2, and revisit the concept of trust score of identity requests in Section 3. Then, in Section 4, we introduce our design for lightweight, long-term identity management based on *green* and *useful* puzzles. The protocol for identity lifecycle management is described in Section 5. In Sections 6 and 7 we present the set of experiments carried out to evaluate our solution, along with major findings. We then close the paper in Section 8.

2. Related work

The name "Sybil attack" was coined by Douceur, [1] to designate the creation of fake accounts in online systems. In the paper that describes the attack, Douceur proved that in the absence of a logically centralized entity, an unknown entity can always present himself to other entities in the system using more than one identity, unless under conditions and assumptions that are unfeasible for large-scale distributed systems. Since then, investigations have focused on limiting the spread of fake accounts and also on mitigating their potential harm. There are various solutions that bound the number of valid fake accounts in a given system. Danezis et al. [5] was the first to categorize these solutions, classifying them as either *strong* or *weak identity-based* management schemes.

The first category comprises solutions in which users may only obtain identities certified by trusted third-parties. Its main advantage is the difficulty imposed to users that attempt to create and control several identities, or take control of someone else's identity. The solutions that fall in this category can be further classified as certification authorities [18–20] and trusted computing [21]. In spite of their potentialities, both require a wide availability and acceptance of the trusted entities that will vouch for the validity of identities (among other drawbacks).

The second category comprises mechanisms in which identities are not created with strong authentication guarantees (i.e., they do not serve for strongly authenticating the user/device using it). In this case, although it is not possible to avoid the existence of sybils, 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. The solutions in this category may be further classified according to the strategy employed to enforce authenticity. Next we enumerate and discuss some of them.

1. *Blacklisting*. The solutions that fit in this category [22,23] rely on a list of addresses or subnets that presented "bad behavior" in the past (e.g., sending spam).

The maintenance of these blacklists must be supervised by a human operator (to prevent legitimate subnets from being regarded as suspicious), and must be updated often. They also rely on algorithms that are prone to misidentification of corrupt subnets.

2. *Social networks*. The use of social networks to detect and/or limit the occurrence of fake accounts has gained significant attention, with several prominent solutions proposed [4,6,7,24]. In addition to detecting/limiting fake accounts, they also estimate an upper bound for the number of fake accounts that are "accepted". Those solutions have some important drawbacks, however. For example, they might violate user anonymity. Also, some investigations have shown that key assumptions upon which those solutions rely on are invalid [12,13].
3. *Trust and reputation.* These systems also have been used to detect fake accounts [25,26]. The actual goal is not limiting the dissemination of fake accounts, but detecting them once they behave suspiciously. However, such solutions have been vulnerable to white-washing attacks, except when strategies to make identity assignment a non-trivial task (e.g., moderation by a human operator) are adopted.
4. *Proof of work*. Solutions based on this approach (e.g., computational puzzles) [8–11] have been effective in limiting the spread of fake accounts or stopping malicious behavior in general (e.g., spamming). Despite the advances, existing mechanisms do not adjust puzzle complexity. More specifically, when assigning puzzles of equal computational complexity to everyone, it becomes hard to choose one that effectively hinders attackers, without severely compromising legitimate users.

In our research, we took advantage of proof of work and complemented it with the concept of trust score for coming up with a design for lightweight, long-term identity management that (i) preserves users' privacy, (ii) adaptively balances the price (in terms of computing resources) to be paid per identity requested (thus making potential attackers to be severely penalized, without significantly harming presumably legitimate users), and (iii)bounds the number of fake accounts an attacker can control.

3. Revisiting the concept of trust score of identity requests

The research reported in this paper is built on the notion that users have to dedicate a fraction of resources to obtain identities, as a strategy to prevent the dissemination of fake accounts. As an important step towards dealing appropriately with presumably legitimate identity requests and those potentially malicious, we introduced in a previous work the concept of trust score [17]. It is a reputation index that establishes likeliness that some identity request, originated from a certain *source of identity requests*, is presumably legitimate or part of an ongoing attack. With such an index, proof of work based approaches can balance the price (in terms of resources) per identity requested, by using the values of trust score as a parameterization factor.

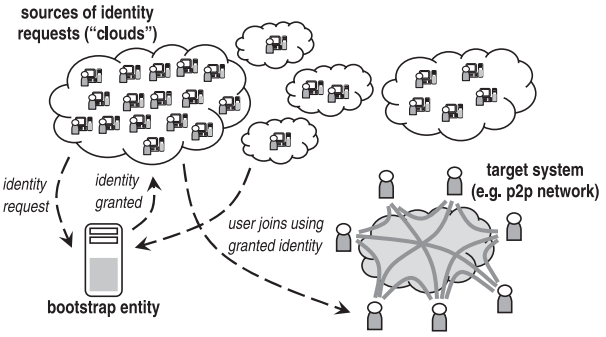


Fig. 1. Sources of identity request.

The concept of source of identity requests, illustrated in Fig. 1, is defined as an aggregation of one or more users (legitimate and/or malicious), that share locational characteristics (e.g. are behind a same IP address or sub-network, or are within a same region in the map), 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" [17].

3.1. Characterizing users' recurrence patterns

Our research for a trust score function was driven by the idea that a good candidate should consider the dynamics of users' recurrence patterns in large scale distributed systems, and ensure that they be assigned good values of trust score. To this end, it is important to characterize users' recurrence in these systems, and assess a baseline regarded as "regular behavior". The literature is rich in investigations analyzing the profile of traffic exchange (amount of data transferred, download and upload speeds, etc.), swarm sizes, etc. in traces publicly available [27]. However, the pattern of users' participation in swarms has been largely neglected. We focus our analysis on this aspect.

Our characterization basically consisted in evaluating aspects such as users' time and frequency of arrivals. From the files of users' participation in torrent swarms analyzed, we extracted traces that rebuild users' identity request events.[3] In summary, we obtained five distinct traces, whose relevant characteristics for our evaluation are shown in Table 1.

In our analysis of users' arrival distribution, we observed a consistent behavior over the week, apart from a few access peaks, characterized by an increase in the number of joins during daytime (UTC), and decrease overnight. As for users' inter-arrival, we observed that some users left and re-joined the system within relatively short time

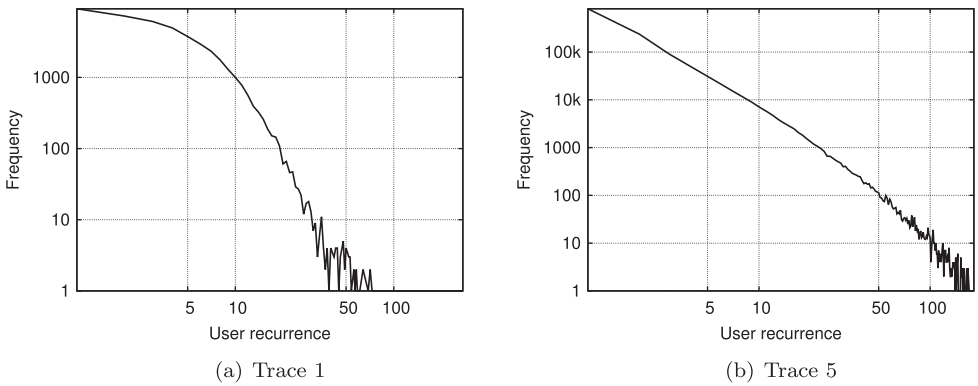
[3] For ethical reasons, the traces collected have been fully anonymized a priori, in order to make it impossible any sort of user identification/tracking. We also assume that the traces contain legitimate identity request activity only, since no evaluation of peer identity vs. address was carried out.

Table 1
Characteristics of part of the traces used in our analysis.

	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5
First identity request date/time (MM-DD-YY HH:MM)	01-01-06 00:00	02-01-06 00:00	03-01-06 00:01	06-15-13 15:00	10-09-13 21:00
Last identity request date/time (MM-DD-YY HH:MM)	01-07-06 20:53	02-06-06 14:48	03-07-06 23:59	06-22-13 15:00	10-16-13 21:00
Total number of identity requests	203,060	738,587	545,134	3,315,363	7,426,316
Total number of sources of identity requests	44,066	50,512	47,968	1,320,074	2,194,519
Trace duration (h)	164.87	134.80	167.97	168	168
Average number of identity requests per minute	20.52	91.31	54.09	328.905	736.737

Table 2
Statistical summary of the frequency of recurrences, for each of the traces considered.

Traces	Minimum	1st decile	Mean	Standard deviation	Mode	Median	Harmonic mean	9th decile	Maximum
Trace 1	1	1	4.608	4.576889	1	3	2.39177	9	273
Trace 2	1	1	14.62	35.44363	1	5	3.15381	28	521
Trace 3	1	1	11.36	15.44093	1	6	3.23839	29	134
Trace 4	1	1	2.511	4.766563	1	1	1.342308	5	180
Trace 5	1	1	3.384	5.422933	1	2	1.538539	7	82



(a) Trace 1 (b) Trace 5

Fig. 2. Distribution of users' recurrences, shown in log–log plots.

intervals. A number of hypotheses can explain this behavior, ranging from transient network failure (which causes users to disconnect often), to monitoring crawlers (which connect just to obtain a list of online users, and then disconnect). In summary, these two aspects suggest that a candidate design for a trust score function should accommodate seasonal changes in users' behavior (within a window of hours, days, or even weeks), and enable users to rejoin the system with a certain frequency without being much penalized.

The analysis of users' recurrence in the system provided the most interesting insights for the design of our trust function. Fig. 2 shows the frequency of recurrence for Traces 1 and 5. Observe from Fig. 2(a) that the vast majority of users joined the system very few times in one week. Observe also that the recurrences followed exponential/power-law distributions; this observation was confirmed for several other traces studied.

Table 2 presents a statistical summary of the frequency of recurrences. Note that the amplitude of recurrences is large in each of the traces. For example, amplitude observed in Trace 1 was 272 (1 and 273). However, at least 90% of users joined the system no more than 9 times. For Trace 2, at least 90% of users joined the system no more than 28 times during one week; this is similar for Trace 3 (29 times).

As for the 10% of users that joined the system more frequently, some interesting patterns were observed. For example, some users joined/left the system every 5 min, on average. This behavior is consistent with robots that periodically collect statistical information from the network, or that attempt to identify users disseminating copyrighted content [28]. Apart from these outliers, one important conclusion from our analysis, which we explored in our design for a trust score function, is that the majority of users tend to join the swarm in a relatively low frequency during a given period.

In summary, the trace analysis has shown that a candidate trust score function should accommodate seasonal changes in users' behavior, and enable them to rejoin with a certain frequency without being much penalized. More importantly, it has shown that a vast majority of users tend to access online systems with a relatively low frequency, during a given period. Attackers on the other hand shall request a higher number of identities from a limited number of sources. Therefore, keeping track of sources' recurrence becomes a promising approach for the design of our trust score function.

3.2. Computing values of trust score for sources of identity requests

In the context of this research, trust score (θ) is a reputation index that establishes the likeliness that some identity request, originated from a certain source, is presumably legitimate or potentially part of an ongoing attack. Here we provide a brief overview of our trust score function and related equations (please refer to our previous paper [17] for an in-depth discussion about trust scores and supporting mathematical formulation).

The main input for computing θ is the number of identities already granted to the users associated to a given source. This number is defined as $\phi_i(t)$ for the *i-th* source, at instant t (with $\phi_i(t) \in \mathbb{N}$). Based on this information, we formally define the *source recurrence* ($\Delta\phi_i(t)$) and *network recurrence* ($\Phi(t)$, with $\Phi(t) \in \mathbb{R}$ and $\Phi(t) \geq 1$) metrics. The former, given by $\Delta\phi_i(t) = \phi_i(t) - \phi_i(t - \Delta t)$, represents the number of identities granted to users associated to some source i, in the last Δt units of time. The latter corresponds to the average number of identities granted in the same period.

The network recurrence metric $\Phi(t)$ is computed using the simple mean of the values of sources' recurrence, 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 $\Delta\phi_k(t) = 0$ for some source k, users associated to that source have not obtained any identity (during Δt); such a source can be safely ignored.

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

Observe that Δt serves as a bound for the portion of identity grants considered when computing the sources' (and the network) recurrence metrics, thus functioning as a "sliding window" that addresses the seasonality of users' access patterns. As the 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 system.

Recall from the trace analysis that a large fraction of users presented a similar, consistent behavior in terms of users' recurrence. For this reason, we use the network average as baseline for "normal behavior". In this context, by comparing the behavior of a given source i (inferred from $\Delta\phi_i(t)$) and the network behavior (inferred from $\Phi(t)$), we calculate the *relationship between source and network recurrences* ($\rho_i(t)$, with $\rho_i(t) \in \mathbb{R}$). When negative, $\rho_i(t)$ indicates how many times the recurrence of the *i-th* source is lower than the recurrence of the network. Eq. (2) provides the value of $\rho_i(t)$.

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

The value of $\rho_i(t)$ serves then as input for computing the source trust score ($\theta_i(t)$). It is calculated at instant t according to Eq. (3), and assumes values in the interval (0, 1): on one extreme, 1 denotes a high trust on the legitimacy of (the) user(s) associated to that source; on the other, 0 indicates high distrust, i.e., a high probability that there exists an attacker "behind" that source. We refer to our previous paper [17] for a detailed, in-depth discussion on the rationale for this equation and trust properties it holds.

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

Abrupt but momentarily changes in their behavior should also be taken into account by our trust score function. For this reason, we compute the *smoothed trust score*. Defined as $\theta'_i(t)$ for the i-th source at instant t, it is calculated as shown in Eq. (4). The smoothing factor β determines the weight of present behavior in the calculation of the trust score, assuming values in the interval (0, 1); values of β close to 0 assign a high weight to historical behavior, and vice-versa. In Eq. (4), $\theta'_i(t')$ refers to the last computed value of smoothed trust score.

$$\theta'_i(t) = \begin{cases} \theta_i(t), & \text{if } \theta'_i(t) \text{ was never computed} \\ \beta \times \theta_i(t) + (1 - \beta) \times \theta'_i(t'), & \text{otherwise} \end{cases} \quad (4)$$

4. Green and useful puzzles for identity management

In this section we briefly discuss the concept of adaptive puzzles (Section 4.1). Then, we introduce our proposal for making puzzles green and useful (Section 4.2).

4.1. Adaptive puzzles

Although being effective in protecting large-scale distributed systems from Sybil attacks, traditional puzzle-based defense schemes do not distinguish between identity requests from (presumably) legitimate users and attackers, and thus require both to afford the same cost per identity requested. To address this problem, we use trust scores to parameterize the complexity of the puzzles to be solved by users prior to obtaining identities. The mapping between trust and puzzle complexity is given by an abstract function $\gamma : \Theta \rightarrow \mathbb{N}^*$ (where Θ is a set of all possible values of trust), which depends essentially on the nature of the adopted puzzle; for being effective, the puzzle must belong to the complexity class NP-complete. In this function, the trust score $\theta_i(t) \in \Theta$ (of the i-th source) is mapped to a puzzle having exponential complexity, equivalent to $O(2^{\gamma_i(t)})$.

An example of mapping function is given in Eq. (5); note that the puzzle complexity is defined based on a maximum possible complexity Γ. In this equation, the constant 1 defines the minimum possible puzzle complexity.

$$\gamma_i(t) = \lfloor \Gamma \cdot (1 - \theta_i(t)) \rfloor + 1 \quad (5)$$

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 concatenation $x|y|z$, after processed by a secure hash function, leads to a number whose γ least significant bits are 0. This

problem can be solved through a brute-force search algorithm, whose complexity is proportional to $O(2^{\gamma})$, whereas the time to assert the validity of the solution is constant. Any puzzle having similar characteristics can be employed with our solution. There are examples in the related literature, such as [1,8,9], so there is no need to invent a new one.

4.2. Making puzzles green and useful

In the following sections we describe our proposal for making traditional puzzles green and useful, without degrading their effectiveness against fake accounts.

4.2.1. Towards green puzzles

Cryptographic puzzles are effective in stopping abusive behavior (e.g. dissemination of fake accounts) because of their time (and resource) consuming nature. This type of puzzle however poses a trade-off between complexity and effectiveness. Puzzles that take one second to be solved (less complex) will barely stop attackers. To keep attackers away, it is important to assign more puzzles (which take longer to be solved).

A side effect of making puzzles more complex is an increase in energy consumption (to power the processor that will solve them). This problem is negligible considering a handful of puzzles. However, it becomes considerable as more and more people have to solve them (e.g. before obtaining an account). Given the growing concern about rational usage of natural resources, "green puzzles" (which demand less resources, but remain effective in limiting fake accounts) becomes paramount.

In our research, we propose reducing the average complexity of assigned puzzles by complementing them with "wait time". Take as an example the one illustrated in Fig. 3: suppose that keeping attackers away requires assigning puzzles that take five time units to solve. In our approach, instead of assigning such (traditional) puzzle, we assign a (green) puzzle that takes two time units; as a complement, we "ask" him to wait three more time units (without processing any puzzle) before obtaining an identity.

The value of trust score is used for estimating both the puzzle complexity and the wait period. The strategy we envisage for estimating them depends on the process currently taking place, which can be either an *identity request* or *renewal*.

(1) Identity request process. It involves the assignment of a puzzle to be solved and a wait time to be obeyed by the user. The puzzle complexity $\gamma_i(t)$ is estimated as a function of the trust score $\theta_i(t)$, and considers a higher value of maximum complexity, $\Gamma = \Gamma_{req}$. The value of $\theta_i(t)$ used to estimate the puzzle complexity is saved in the identity (this aspect is discussed in more detail in Section 5). Eq. (5) (shown in the previous section) represents a design we consider for assessing the puzzle complexity.

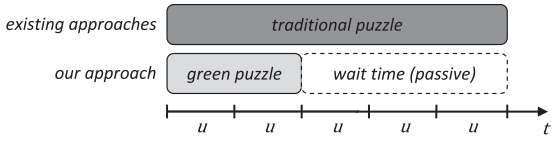


Fig. 3. Illustration of the concept of green puzzles and passive wait time.

For the waiting time, we calculate it considering a similar approach as in the case of puzzle complexity (recall that the defined wait time must complement the complexity of the assigned puzzle). In other words, it increases exponentially, proportionally to 2^{ω}, where ω is a wait factor also defined as a function of $\theta_i(t)$. Therefore, small decreases in the value of trust score will translate into higher increases in the wait time to be obeyed by the user. The design we consider for computing ω is given in Eq. (6). In this function, Ω represents the maximum factor for the waiting time.

$$\omega_i(t) = \Omega \cdot (1 - \theta_i(t)) \tag{6}$$

To prevent that an attacker request a large amount of puzzles and "parallelize" her wait for all of them, the current value of the source trust score must be compared to the one used to estimate the wait period; if their difference exceeds a treshold $\Delta\theta$ (i.e., $\theta_i(t) - \theta_i(t') \geq \Delta\theta$), the identity request process should be interrupted (similarly to what happens if the user does not obey the wait time). Recall that the trust score of a source changes overtime, especially after new identities are granted to it.

(2) Identity renewal process. It involves solely the assignment of a puzzle to be solved; no wait time is assigned. The puzzle complexity is defined as a function of the value of trust score stored in the identity (represented by $I\langle\theta\rangle$), saved during the identity request process. This is an incentive so that users renew their identities, and thus take advantage of an increasingly better reputation. Eq. (7) represents a design we consider for assessing the puzzle complexity in this process. Once completed, $\theta_i(t)$ must be saved in the identity ($I\langle\theta\rangle \leftarrow \theta_i(t)$), for use during future renewal processes.

$$\theta_i(t) = \beta \cdot 1 + (1 - \beta) \cdot I\langle\theta\rangle \tag{7}$$

The puzzle complexity is defined considering a lower complexity factor, $\Gamma = \Gamma_{renew}$. If the identity has already expired, another value of maximum puzzle complexity, $\Gamma = \Gamma_{reval}$, must be used. As a general recommendation to encourage users to maintain their identities and renew them before expiration, $\Gamma_{renew} < \Gamma_{reval} < \Gamma_{req}$.

4.2.2. Towards useful puzzles

There are several proposals of cryptographic puzzles that can be used with our design to establish a cost for the identity renewal process [1,8,9]. An important characteristic of such puzzles is that their processing does not result in actual useful information. For example, consider again the one proposed by Douceur [1]. The solutions for that type of puzzle hardly can be used (if not) as useful input to some other process. We thus propose a different class of puzzles, which takes advantage of users' processing cycles to compute actually useful information.

To assign a puzzle to be solved, every identity request or renew messages must be replied with (i) an URL that contains a piece of software that implements the puzzle (which can be a *useful* puzzle or a cryptographic one) and

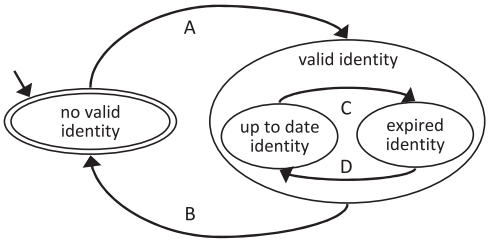


Fig. 4. Possible states in an identity lifecycle.

(*ii*) a set $\mathcal{J}$ of jobs (where each job is comprised of a number of input arguments to the downloaded piece of software). The puzzle complexity is given by $|\mathcal{J}|$.

An example of puzzle is a software that runs a simulation and generates the results using plain text. In this context, $\mathcal{J}$ contains a number of seeds that must be used as input to the simulation. Supposing that $\gamma_i(t_k) = 4$ (as computed from Eq. (5)), then $|\mathcal{J}| = 2^4 = 16$. To prevent that an attacker provide fake solutions, any protection mechanism adopted in massive distributed computing or crowdsourcing could be used. For example, $\mathcal{J}$ could contain some "test jobs" (for which the result is already known); this approach follows the same strategy used in schemes such as ReCAPTCHA [29], which aims at keeping robots away from websites and helps digitizing books. In this case, the attacker would not be able to distinguish which are test jobs and real ones.

5. Conceptual design for identity management

Here we focus on the conceptual design that forms the basis of our solution. In the description that follows, *bootstrap* is the entity responsible for granting/renewing identities to users; a *source* is the location, identified by the bootstrap, from which a given user requests an identity.

5.1. Identity schema and lifecycle

In our design, we assume that identity I contains the following information: $I = \langle i, t, \theta \rangle$. In this tuple, $I\langle i \rangle$ is a unique, universal identifier, and $I\langle t \rangle$ is the last time it was processed by the bootstrap (e.g., during a renewal). Finally, $I\langle \theta \rangle$ represents the trust score associated to the identity (it will be discussed in the following section). To prevent tampering, identities should be digitally signed by the bootstrap upon processing.

We also envisage two system-wide parameters for supporting the identity lifecycle management: V and E (with $V \geq E$). V indicates the amount of time provided for the validity of an identity, whereas E indicates the time for identity expiration. Being T the current timestamp, an identity is valid for contacting the bootstrap only if $I\langle t \rangle + V \geq T$; otherwise, the user must obtain a new identity (which will incur in a comparatively higher cost, as discussed earlier). Likewise, an identity is valid for identifying a user in the system only if $I\langle t \rangle + E \geq T$. Observe that I may have expired, but the identity can still be valid for renewal; this is true as long as $I\langle t \rangle + V \geq T$ does not hold yet. Fig. 4 depicts the set of possible states of an identity, and possible transitions between them. Next we describe the conditions that trigger each transition.

5.2. Protocol for identity lifecycle management

Fig. 5 (a) illustrates a simplified view of the protocol for managing the lifecycle of identities, and the entities involved. The messages exchanged with this protocol should be digitally signed, to prevent tampering. Fig. 5(b), in turn, focuses on the state machine for this protocol. The gray regions in these figures highlight the extension we propose here considering any traditional, general identity management scheme.

In an initial state, the user has no valid identity (either because he has never joined the system, or his identity is no longer valid). Transition *A* (arrow A in Fig. 4) after the user obtains an identity from the bootstrap, by means of an identity request process. This process takes place when he issues a BeginHandshake message (first arrow in Fig. 5(a)) to the bootstrap. In response, the bootstrap issues a task to be performed, using message CompleteTask (second arrow). In our protocol, *task* can be of any type that is commonly understood between the entities involved. In the context of this research, we envisage *task* as a puzzle (or wait time), with complexity (or duration) determined according to the methodology described in the previous sections.

After completing the task (e.g., solving the assigned puzzle, or obeying the determined wait time), the user

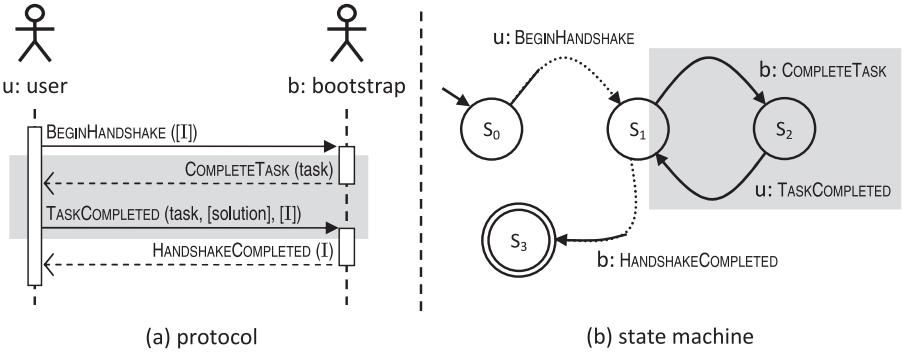


Fig. 5. Proposed extension for an identity management protocol (left), and state machine (right).

contacts the bootstrap and issues the TaskCompleted message (third arrow). This message has an optional parameter, *solution*, which essentially depends on the task nature. Once verified that the task was correctly completed (e.g., by assessing the validity of *solution*), the bootstrap issues a new identity to the user, by sending message HandshakeCompleted (fourth arrow). At this stage, the user evolves to the "up to date identity" state, which means he has a valid, unexpired identity. It is important to observe from Fig. 5(b) that the protocol enables as many iterations CompleteTask, TaskCompleted as the bootstrap sees fit. For example, in the case a solution based on green and useful puzzles is used for identity management, a first iteration of CompleteTask, TaskCompleted may serve for puzzle assignment, and a second iteration may serve for establishing a wait time.

Depending on the system nature, users might be required to renew their identities periodically. To this end, we envisage a renewal process following the same protocol depicted in Fig. 5(a). The difference is that the messages exchanged now contain the identity to be renewed (parameter *I*). Observe that the bootstrap entity is the sole responsible for task assignment and validation of returned solutions. This responsibility could be decentralized to some degree. It is important however that the entities in charge be trusted (to prevent tampering).

In case the user does not renew its identity and $I\langle t\rangle + E \geq T$ holds, transition *C* (Fig. 4) takes place; the identity then becomes no longer valid for joining the system (state "expired identity"), but it can still be renewed. If the user renews it before $I\langle t\rangle + V \geq T$ holds (transition *D*), it becomes valid again (state "up to date identity"). Otherwise, transition *B* takes place and the identity becomes useless; the user must then go through the process of obtaining a new identity as if he had never had one.

6. Analytical evaluation

We begin our evaluation by establishing an analytical model for assessing the effectiveness and robustness of green and useful puzzles for identity management. Based on the research questions posed in the introduction, here (and in the section that follows) we focus on the evaluation aspects enumerated next: (i) what is the impact of our scheme to legitimate users and attackers? (ii) what are the potential energy savings? (iii) is it technically feasible to reuse processing cycles to compute useful information? (iv) what is the monetary cost of launching an attack against our scheme (in terms of acquiring the resources necessary for deploying it)? To help answering these questions, we define the following evaluation metrics:

- *Reduction of fake accounts* (*R*). This metric indicates the proportion in which the amount of valid fake accounts was reduced. Formally, we have

$$R = 1 - \frac{\int_{t_i}^{t_f} f_s(t)dt}{\int_{t_i}^{t_f} f_b(t)dt} \tag{8}$$

In the equation above, $f_s(t)$ and $f_b(t)$ are functions that describe the number of valid fake accounts in the system at instant *t*, in the scenario where our solution and a baseline mechanism are used, respectively. t_f and t_i represent the period of evaluation (in our experiments, t_i equals to 0 and t_f equals to the duration of the experiment). Possible values for this metric range in the interval $(-\infty..1]$. A value of $R = 1$ is a global optimum; it means our solution did not allow any creation of fake accounts. Due to the nature of the data collected from the experiments, we use Riemman Summation to compute both integrals.

- *Proportion of energy savings* (*D*). It indicates the ratio of energy savings provided by our solution, when compared to a baseline scenario. Formally, we have

$$D = 1 - \frac{\min(\text{consumption }(\textit{baseline}), \text{consumption }(\textit{solution}))}{\text{consumption }(\textit{baseline})} \tag{9}$$

In the equation above, consumption(*solution*) is the total energy consumption caused by our solution, and consumption(*baseline*) is the energy consumption observed for the mechanism regarded as baseline. Possible values for this metric range in the interval [0..1). A value of $D \approx 1$ is a global optimum; it means our solution provided the highest savings possible, when compared to the baseline scenario.

6.1. Model

Legitimate and malicious requests arrive following some distribution $f_k(t;\ldots)$; index k can be either *leg* (for legitimate requests) or *mal* (for malicious ones). Note that either distribution can be Gaussian, exponential, uniform, fixed, or any other. In this case, we can derive as $\lambda_k(t) = U_k^{-1} \cdot f_k(t;\ldots)$ the arrival rate for each legitimate ($k = leg$) and malicious ($k = mal$) source. In these equations, U_k stands for the number sources.

The analysis in our model evolves in rounds, with duration Δt. In this case, the number of identity requests per source within a given round i ($\forall i > 0$) is given by Eq. (10). In this equation, k can be either *leg* or *mal*, and R_k is the total number of requests of type k made during the evaluation.

$$\Lambda_k(i) = R_k \int_{(i-1)\cdot\Delta t}^{i\cdot\Delta t} \lambda_k(t)\,dt \tag{10}$$

To accommodate identity renewal in our model, we define a function $\Lambda'_k(i)$ (computed following Eq. (11)), which is the sum of the number of requests per source that arrived for the first time in the network (defined by $\Lambda_k(i)$), and the number of identities per source that expired in the previous round and thus must be renewed (see Eq. (18)), $\Psi_k(i-1)$. U_k is the number of sources of type k in the system. For simplicity, we assume here that both legitimate users and attackers renew their identities once expired.

$$\Lambda'_k(i) = \frac{\Psi_k(i-1)}{U_k} + \Lambda_k(i) \tag{11}$$

The rate in which identities are granted per second (service rate), defined for a given round i, is given by

Eq. (12) (for $k \in \{leg, mal\}$). Observe that it captures the penalties imposed by both cryptographic puzzles and wait time. The former is captured through the exponential complexity $O(2^{\gamma(t)})$ of the brute force search algorithm used to solve the puzzles. The latter, following our design for green puzzles, is a passive wait that grows exponentially with $2^{\omega(t)}$. In this equation, C_k is the total number of computing devices of type k, and P_k is the average computing power of each device in possession of a given user (either a legitimate user or an attacker). A value of $P_k = 1$ represents a standard, off-the-shelf hardware (for reference); a value of $P_k = 2$ denotes a hardware twice as fast. The values of $\gamma_k(i)$ (puzzle complexity factor) and $\omega_k(i)$ (waiting time factor) are computed according to Eqs. (5) and (7), respectively (as discussed in Section 3).

$$\mu_k(i) = \frac{1}{\frac{2^{\gamma}+2^{\gamma_k(i)}}{P_k \cdot C_k \cdot U_k^{-1}} + 2^{\omega} + 2^{\omega_k(i)}} \tag{12}$$

Based on the service rate defined above, the maximum number of identities that one can obtain in each round can be computed using Eq. (13).

$$M_k(i) = \Delta t \cdot \mu_k(i) \tag{13}$$

For defining the service rate (as shown in Eq. (12)), we need the average trust score of the sources of identity requests (either legitimate or malicious). To estimate it, we must first estimate the average recurrence of sources, and the network recurrence rate. The former is given by Eq. (14), whereas the latter is given in Eq. (16).

$$\widehat{\Delta\phi_k}(i) = \min(\Lambda'_k(i) + \psi_k(i-1)\,,\, M_k(i-1)) \tag{14}$$

The rationale for computing $\widehat{\Delta\phi_k}(i)$ is the following. In each round i, $\Lambda_k(i)$ requests arrive in the system. However, the number of identities granted in previous rounds can be smaller ($\Lambda_k(i) > M_k(i)$). In practice, it means that some user may be either solving puzzles or respecting the waiting period. As a consequence, there will be a number of requests that will not be granted in that round, and thus will retry. Therefore, for computing $\widehat{\Delta\phi_k}(i)$ in the i-th round, we take the minimum between the number of requests (i) from round $\Lambda'_k(i)$ plus those still in process from previous rounds $\psi_k(i-1)$, and (ii) those already granted an identity in the previous round $M_k(i-1)$. Note that the service rate in the next round depends on the arrival rate in the previous one. Similarly, the number of requests that have not been granted an identity and therefore will retry in the next round ($\psi_k(i)$, given in Eq. (15)) is defined as the difference between the number of identities that arrived and the estimated value of $\Delta\phi_k(i)$.

$$\psi_k(i) = \Lambda'_k(i) + \psi_k(i-1) - \widehat{\Delta\phi_k}(i) \tag{15}$$

The network recurrence rate (Eq. (16)) is given by the minimum between 1 (in case no request was granted yet) and the average of already granted requests. In this equation, $\mathcal{K}$ is a set that indicates what types of users have requested at least one identity during that round ($\mathcal{K} \subseteq \{leg, mal\}$). In case $\mathcal{K} = \varnothing$ (i.e., neither legitimate users nor attackers have obtained identities during that round), it returns 1.

$$\widehat{\Phi}(i) = \max\left(1\,,\, \frac{1}{\max(1, |\mathcal{K}|)} \cdot \sum_{k \in \mathcal{K}} \widehat{\Delta\phi_k}(i-1)\right) \tag{16}$$

Finally, the number of identities of type k that are valid after i rounds (where i can be given by the total period of analysis T divided by Δt, i.e., $\lfloor T/\Delta t \rfloor$) is given by $G_k(i)$, as shown in Eq. (17). In this equation, ν is the number of rounds in which an identity is valid, and is given by the maximum validity $\nu = V/\Delta t$.

$$G_k(i) = \left\lfloor U_k \cdot \sum_{j=i-\nu}^{i} \widehat{\Delta\phi_k}(j) \right\rfloor \tag{17}$$

It is important to keep track of the number of identities that expire, so that they re-enter the system as new requests. This number, denoted as $\Psi_k(i)$ for the i-th round, and computed according to Eq. (18), is defined as the difference between the total number of requests that have arrived since the first round, and the number of identities currently valid plus the number of requests that were not yet granted.

$$\Psi_k(i) = U_k \cdot \sum_{j=0}^{i} \Lambda_k(j) - (G_k(i) + \psi_k(i) \cdot U_k) \tag{18}$$

6.2. Complexity analysis of the attack

The number of fake accounts an attacker may obtain basically depends on the amount of resources provided to the attack, and conditions in which these resources are used. The algorithm for implementing an attack against our solution is straightforward. Given as input the number of fake identities that must be created, number of malicious sources available, and total period for the attack, the attacker must determine, for each malicious source, (*a*) how many requests will originate from it, and (*b*) when each request will be made. Algorithmically, the complexity of determining an allocation of number of identity requests per sources, and instant in which each request will be performed, is equivalent to $O(r)$, where r stands for the number of malicious requests.

While the complexity of the algorithm above is relatively small, implementing a successful attack depends on the strategy used to materialize steps (*a*) and (*b*). We argue that the best strategy an attacker can use to maximize her profit, given a finite amount of resources (computing power and sources), is: (*i*) making an equal allocation of malicious requests among those sources in her hands, (*ii*) uniformly spread those malicious requests overtime, and (*iii*) prevent sharing malicious sources with legitimate users.

To illustrate (*i*), consider $\gamma = \omega = 0$, and $P_{mal} = 1$. In order to obtain puzzles of lower complexity, the trust score $\theta(t)$ of the malicious sources must be as high as possible, or at least around 0.5 ($\theta(t) \gtrapprox 0.5$). To this end, the recurrence of a source must be as low as possible, or at most around the same value of the measured network recurrence rate ($\Delta\phi(t) \lessapprox \Phi(t)$). To achieve this, and at the same time perform r malicious requests, the attacker must divide them as uniformly as possible among the u sources

available (i.e., $\Delta\phi(t) = \frac{r}{u}$), so as to keep the average recurrence per source low and thus appear less suspicious. Suppose that, by doing this, the recurrence of malicious sources becomes $\Phi(t)$ (i.e., $\Delta\phi_i(t) = \Phi(t)$, where i is the index of a malicious source). We thus have $\theta_i(t) = 0.5$. From Eq. (12), the attacker's capability to solve puzzles is then $\mu = 1/(2^{0.5\cdot\Gamma} + 2^{0.5\cdot\Omega} + 2)$.

The logic behind statement (*ii*) is analogous. First, consider $\gamma = \omega = 0$, and $P_{mal} = 1$. Let Δt be the window considered for computing $\Delta\phi(t)$ and $\Phi(t)$. Consider also that the time period begins in T_0 and ends in T_f. For simplicity, we consider that the attacker controls a single source. The results presented here, however, can be generalized for multiple sources. In order to obtain puzzles of lower complexity, the value of $\theta(t)$ of the malicious source must be as high as possible, or at least around 0.5 ($\theta(t) \gtrapprox 0.5$). To this end, the recurrence of that source, $\Delta\phi(t)$, must be as low as possible, or at most around $\Phi(t)$. To achieve this, and at the same time perform r malicious requests in T units of time, the attacker must divide them as uniformly as possible along T (i.e., $\Delta\phi(t) = r/T \cdot \Delta t$), so as to keep the average recurrence of her source low within the sliding window Δt. Suppose that, by doing this, the recurrence of the malicious source becomes equal to $\Phi(t)$. In this case, we achieve $\theta_i(t) = 0.5$. From Eq. (12), the attacker's capability to solve puzzles is then $\mu = 1/(2^{0.5\cdot\Gamma} + 2^{0.5\cdot\Omega} + 2)$.

Finally, the logic behind statement (*iii*) is similar to that of statements (*i*) and (*ii*): sources will be assigned higher values of trust score should they be associated with fewer identity requests. Therefore, sharing sources of identity requests ultimately means increasing their recurrence, which of course degrades the measured trust score.

6.3. Attacker's strategy

Here we provide proof of the validity of the attack strategy discussed above, showing that it is in fact the best the attacker could use.

Theorem 1. *Let $S_{mal} = u$ be a set of sources in hands of the attacker, and $R_{mal} = r$ the number of identities to be obtained. The strategy that maximizes the profit of the attacker (i.e., that minimizes the overall complexity of the puzzles to be solved) is to make an equal allocation of the S_{mal} requests among the R_{mal} sources.*

Proof. Suppose that some different, generic attack strategy in which malicious requests are *not* evenly divided is the best one an attacker could use. Therefore, for some source(s), its recurrence will be larger than $\Phi(t) = \frac{r}{u}$. Conversely, the recurrence of other malicious sources will be proportionally lower (and smaller than $\Phi(t)$ as well). Assume that k sources (with $k \in \mathbb{N}^*$ and $k < u$) request less x identities each (with $x \in \mathbb{N}^*$). $\Delta\phi(t)$ will decrease to $\frac{r}{u} - x$. Consequently, the trust score of these k sources will increase to $\theta'_1 = 0.5 + \vartheta_1$ (with $\vartheta_1 \in (0, 0.5]$); the attacker's capability to solve puzzles becomes $\mu'_1 = 1/(2^{\theta'_1\cdot\Gamma} + 2^{\theta'_1\cdot\Omega} + 2)$.

The number of identities which remain to be requested (from the attacker's goal of r identities) is $k \cdot x$. These requests must be originated from the remaining $u - k$ sources. If the remaining requests are evenly distributed among sources, their recurrence rate will increase to $\Delta_i\phi(t) = \frac{r}{u} + k \cdot \frac{x}{u-k}$. Such a recurrence will be higher than $\Phi(t) = \frac{r}{u}$; the trust score of these k sources will decrease to $\theta'_2 = 0.5 - \vartheta_2$ (with $\vartheta_2 \in (0, 0.5]$); the attacker's capability to solve puzzles becomes $\mu'_2 = 1/(2^{\theta'_2\cdot\Gamma} + 2^{\theta'_2\cdot\Omega} + 2)$. Comparing this situation with the previous one (in which requests are evenly divided among sources), we have $\mu'_2 \ll \mu \ll \mu'_1$. Note that, while the attacker has a gain with the sources that request fewer identities, the overhead with the sources that request more identities increases significantly, given the exponential growth rate of puzzles. This proof evidences that any attack strategy that does not evenly divide requests among sources will result in even more complex puzzles being assigned. □

Theorem 2. *Let r be the number of identities an attacker must obtain, and T the time period during which these identities should be requested. The strategy that maximizes the profit of the attacker is to spread the r requests throughout period T.*

Proof. Suppose that some different, generic attack strategy in which malicious requests are *not* uniformly requested along T is the best one an attacker could use. Therefore, for some period $\Delta t \in T$, its recurrence will be larger than $\Delta\phi(t) = r/T \cdot \Delta t$ (and thus larger than $\Phi(t)$). Conversely, for the rest it will be proportionally lower (and smaller than $\Phi(t)$ as well). Assume that from T_{i-1} to T_i (with $0 < i < f$) that source request less x identities (with $x \in \mathbb{N}^*$). $\Delta\phi(t)$ will decrease to $\frac{r-x}{T} \cdot (T_i - T_{i-1})$ and, therefore, it will be lower than $\Phi(t)$. Consequently, the trust score of these k sources will increase to $\theta'_1 = 0.5 + \vartheta_1$ (with $\vartheta_1 \in (0, 0.5]$); the attacker's capability to solve puzzles becomes $\mu'_1 = 1/(2^{\theta'_1\cdot\Gamma} + 2^{\theta'_1\cdot\Omega} + 2)$.

The number of identities which remain to be requested is x, so that all r requests are performed. They must be performed in the remaining $T_f - T_i$ period of time. If the remaining requests are evenly distributed among sources, their recurrence will increase to $\Delta\phi(t) = \frac{r+x}{T} \cdot (T_f - T_i)$. Such a recurrence will be higher than $\Phi(t)$; the trust score of these k sources will decrease to $\theta'_2 = 0.5 - \vartheta_2$ (with $\vartheta_2 \in (0, 0.5]$); the attacker's capability to solve puzzles becomes $\mu'_2 = 1/(2^{\theta'_2\cdot\Gamma} + 2^{\theta'_2\cdot\Omega} + 2)$. Comparing this situation with the previous one (in which requests are evenly divided throughout time), we have $\mu'_2 \ll \mu \ll \mu'_1$. Note that, while the attacker has a gain with the sources that request fewer identities, the overhead with the sources that request more identities increases significantly. This is because the complexity of the puzzle increases exponentially. This proof evidences that any attack strategy that does not evenly divide requests throughout the time will result in even more complex puzzles being assigned. □

Theorem 3. *Let $\Delta\phi_l(t)$ be the recurrence of sources shared by legitimate users, and let $\Delta\phi_m(t)$ be the recurrence of a source originating malicious requests. In order to achieve as high values of trust score as possible for the malicious requests, the attacker must originate those requests from sources not being shared with legitimate users.*

Proof. Consider the scenario where legitimate users do not share sources with an attacker; in this scenario, the recurrence of a source that originates legitimate requests only is $\Delta\phi_l(t) = x$, and the recurrence of a source originating malicious requests is $\Delta\phi_m(t) = y$ (with $x, y \in \mathbb{N}^*$). In the scenario the attacker shares sources with legitimate users, the recurrence of a single source originating both legitimate and malicious requests is now $\Delta\phi_{l,m}(t) = x + y$. This increased recurrence 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. □

6.4. Dynamics of identity assignment

Fig. 6 presents the results of an evaluation to understand how our solution would behave in a large scale setting. In this analysis, we considered 3.6 million legitimate users that attempt to obtain one identity each, during one year (366 days). An attacker, controlling a botnet of 10,000 machines, attempts to control 1.8 million identities in the same period (one third of legitimate identities). We adopted the following distributions for the arrival of legitimate users: fixed rate, exponential, and Gaussian. The attacker evenly divides her malicious requests over time, and among the sources in her control. The other parameter settings for the analysis are: $P_{leg} = P_{mal} = 1$; $\gamma = 6$ (which means puzzles will have a fixed complexity which takes 64 s to solve on an standard hardware); $\Gamma = 13$ (meaning that most complex puzzles will take at most 2.275 h to solve); $\omega = 0$; and $\Omega = 17$ (meaning a wait time assigned between 0 and 36 h, depending on the source trust score). The identities assigned are valid for a period of four days, or twice the duration of the sliding window.

With regard to the setting for the sliding window Δt and smoothing factor β, we refer to our previous work [17], for a sensitivity analysis and methodology for determining a proper setting those parameters, considering the identity manegement scheme based on adaptive puzzles. In summary, the sensitivity analysis shown in our previous work suggested that the values of $\Delta t = 48$ h and $\beta = 0.125$ are promising to control the dissemination of fake accounts without hampering legitimate users. Of course, a proper setting these parameters requires considering the particularities of users' behavior (e.g. expected frequency of one's identity requests) and conservativeness with regard to changes in users' behavior.

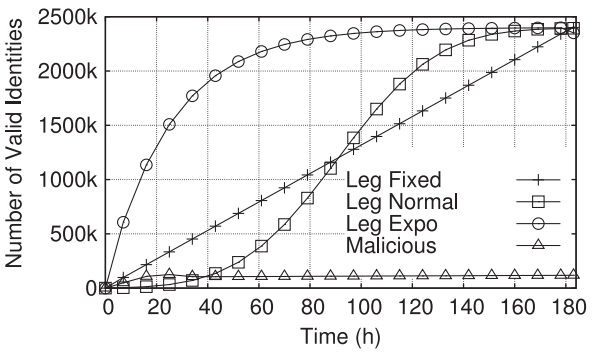


Fig. 6. Number of valid identities for legitimate and malicious users.

As one can see, legitimate users were minimally penalized, and managed to control 70% of the identities ($\approx$ 2.5 millions) by the end of the analysis, in all of the considered scenarios. In contrast, the attacker was able to achieve only 6% of her goal (of controlling 1.8 million identities). Considering only the valid identities in the system, the attacker was able to maintain 4.8% of the identities only. These results, in addition to confirming the results achieved with simulation and experimentation, provide strong evidence that our solution not only is able to control the dissemination of fake accounts, but remains effective in environments comprised of millions of users.

7. Simulation and experimentation with PlanetLab

Next we present the results achieved with simulation and experimentation using the PlanetLab environment. Here we take advantage of the analytical model, in order to cross-validate the outcome of both the simulation and experimental evaluation.

7.1. Simulation

We developed a simulation environment[4] that mimics the dynamics of puzzle assignment and resolution for our evaluation. In summary, it aggregates the functionalities of managing users' identity requests, puzzle assignment and validation, and granting (or denial) of requests (according to the correctness of received puzzle solutions). Although based on simulation, we made sure to use in our evaluation real life traces of identity requests. It is also important to emphasize that our simulation environment is system-agnostic, thus being suitable for evaluation of other (peer-to-peer) systems that can accommodate a puzzle-based identity management scheme.

We have evaluated scenarios with and without attack, considering the following solutions: without control, based on static puzzles (as proposed by Rowaihy et al. [9]), and our solution. To evaluate them through simulation, the following aspects had to be observed: (*i*) behavior of legitimate users, (*ii*) their computing power, (*iii*) puzzle complexity, and (*iv*) attacker's goal. Each of these aspects is discussed next.

7.1.1. Parameter setting of the simulation environment

In a previous work [17], we carried out an extensive sensitivity analysis to understand the influence of various parameter settings on the effectiveness of adaptive puzzles; here we take advantage of its major findings to define parameter setting. It is important to emphasize that

[4] The simulator and trace used in our evaluation are available for download at http://www.inf.ufrgs.br/~wlccordeiro/adaptive_puzzles/.

adopted settings characterize scenarios favorable to the attacker.

The simulation has duration of 168 h. In this period, 160,000 users (from 10,000 sources) arrive 320,000 times in the system. The number of users per source is exponentially distributed (between 1 and 32). This choice is supported on a study [30] that suggests that the number of unique end-hosts behind NAT networks – in residential DSL lines of a major European ISP – follows an exponential distribution. The arrival per source is exponentially distributed (bounded between 1 and 64).

The first arrival of each user is normally distributed throughout the simulation; inter-arrivals follow an exponential distribution, bounded between 1 min and 48 h; each user's recurrence is uniformly distributed, between 1 and 4; finally, the computing power of legitimate users is normalized and exponentially distributed, bounded between 0.1 and 2.5 times the capacity of a standard, off-the-shelf hardware used as reference.

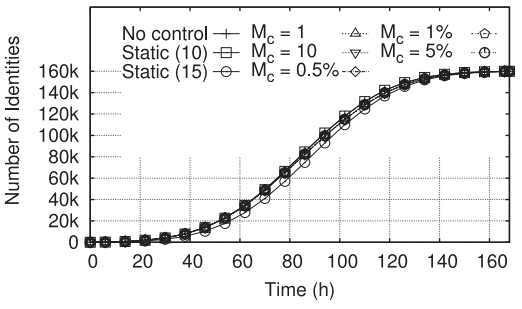
To model the delay incurred from puzzle-solving, we consider that a puzzle of complexity $\gamma_i(t_k)$ takes $2^6 + 2^{\gamma_i(t_k)-1}$ s to be solved using the standard, off-the-shelf hardware; a computer twice as fast takes half of that time. As for the waiting time, we consider that a factor of $\omega_i(t_k)$ results in $2^{\omega_i(t_k)}$ s of wait.

We consider two situations for the attacker: one in which she is able to increase her computing power (using a cluster of high-performance computers), and another in which she increases both the computing power and the number of sources from which her requests depart (e.g., using a botnet of high-performance computers).

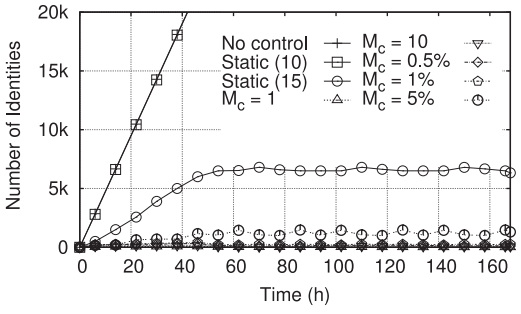
The other parameters in our evaluation are defined as follows. For adaptive green and useful puzzles, $\Delta t = 48$ h and $\beta = 0.125$. We use $\Gamma_{req} = 15$ (as maximum possible complexity when the user does not have a valid identity), $\Gamma_{reval} = 14$ (when the user has a valid but expired identity), and $\Gamma_{renew} = 13$ (otherwise). The waiting factor is $\Omega = 17$. Given the short scale of our simulation (one week), we consider that identities expire $E = 24$ h after created/renewed, and become invalid after $V = 48$ h. For static puzzles, we consider a scenario with complexity $\Gamma = 10$ (which take around 17 min to solve, depending on the hardware) and $\Gamma = 15$ (which take around 9 h to solve). We also compare the performance of our solution with the traditional adaptive puzzles [17]; the adopted parameter setting is $\Gamma = 17$, $\Delta t = 48$ h, and $\beta = 0.125$.

7.1.2. Effectiveness in mitigating fake accounts

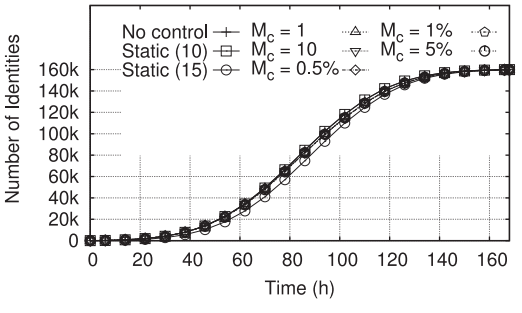
Fig. 7 shows the results achieved when the attacker increases her computing power to solve puzzles more quickly. We consider scenarios in which the attacker has $M_u = 10$ sources (Fig. 7(a) and (b)), and $M_u = 500$ sources (Fig. 7(c) and (d)) in her control. The number of high-performance computers available for the attack is defined proportionally to the number of legitimate sources: $M_c =$ 1 computer; $M_c = 10$; $M_c = 0.5\%$ (50 computers); $M_c = 1\%$ (100 computers); and $M_c = 5\%$ (500 computers). In the attack scenarios where static puzzles are used, the number of high-performance computers available for the attack is $M_c = 5\%$. For the sake of legibility, we omit curves for adaptive puzzles (see Table 4 for a summary). It is important to mention that, for legitimate users, the total number



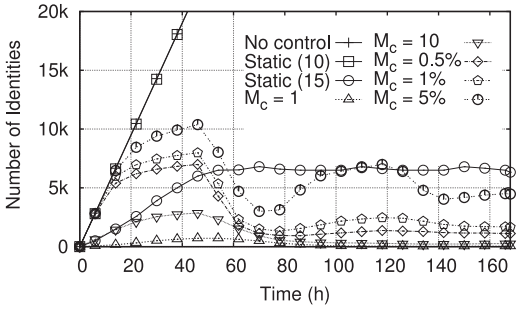
(a) Number of legitimate identities created (accumulated view), in the $M_u = 10$ attack scenario



(b) Instantaneous number of valid fake accounts in the system, in the $M_u = 10$ attack scenario



(c) Number of legitimate identities created (accumulated view), in the $M_u = 5\%$ attack scenario



(d) Instantaneous number of valid fake accounts in the system, in the $M_u = 5\%$ attack scenario

Fig. 7. Number of accounts (legitimate and fake), in the scenario where the attacker increases her computing power to solve puzzles.

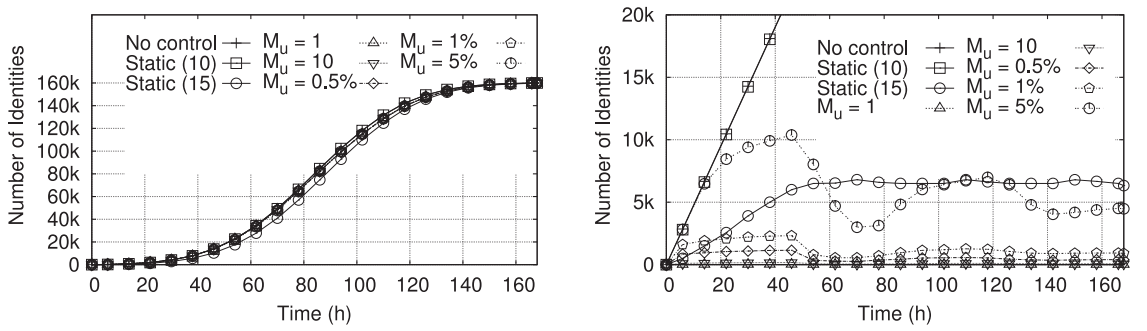


Fig. 8. Number of legitimate and fake accounts in the *botnet* scenario.

of identities created is depicted in the plots; as for the attacker, the plots depict the number of valid identities she controls in a given instant. This choice is because legitimate users request only one (or very few) identities during the simulation period, for use in a period shorter than its validity. As for the attacker, his goal is both creating and maintaining valid fake accounts.

One can see in Fig. 7(b) that the use of green and useful puzzles (also referred to as "our solution" in the remainder of this section) clearly limits the number of fake accounts the attacker can control, in contrast to the scenario where static puzzles [9] are used (curves "Static (10)" and "Static (15)"). This observation holds even for the worst case scenario to our solution, i.e., when the attacker has a cluster of $M_c = 5\%$ high-performance computers: our solution reduced in 81% ($R = 0.81$) the number of fake accounts, comparing to the scenario "Static (15)" ($\int_0^{168} f(t)dt = 936,087$ for static puzzles, and $\int_0^{168} f(t)dt = 169,619$ for our solution). As for the overhead imposed to legitimate users, Fig. 7(a) shows that it was negligible.

Fig. 7(d) evidences that increasing the computing power available is the only way the attacker can circumvent our solution; even so, she is not able to control more identities than would happen in the case of static puzzles with complexity $\Gamma = 15$. Fig. 7(c) shows that legitimate users remain unaffected.

From the results described above, two major conclusions can be drawn. First, our solution makes it more expensive for an attacker to control fake accounts in the system, and she cannot repeat the same performance as seen in traditional approaches. Second, static puzzles impose an important trade-off between effectiveness (in mitigating fake accounts) and overhead (to legitimate users); with $\Gamma = 10$, static puzzles were totally ineffective; with $\Gamma = 15$, there was a considerable overhead imposed to legitimate users, in exchange for some improvement in mitigating fake accounts.

The attack does not improve much when the attacker uses a botnet to solve puzzles. Fig. 8 shows that she only achieves a relative success in the extreme scenario where the botnet represents 5% of the total of sources. However, such success is not sustainable, because of the poor reputation of malicious sources; by the end of the evaluation, she has only a few identities more than she would have if static puzzles were used. Again, the overhead to legitimate users was minimal.

It is important to emphasize here that our solution does not necessarily distinguishes users between either legitimate or malicious. Instead, we compute the likeliness that an identity request, coming from a certain source, is presumably legitimate or part of an ongoing attack. We have chosen such design because our solution must deal with a potentially large number of users (in the order of millions), having a diverse set of identity request patterns, in scenarios more suitable to weak-based identity schemes. Therefore, we have focused on a solution in which legitimate users be minimally penalized, while imposing a significant penalty to a large fraction of malicious requests. To this end, our trust score metric (used to establish the "reputation" of some request) varies between 0 and 1, being 0 total distrust and 1 complete trust on that request.

Having said that, we analyzed for the botnet scenario the perceived reputation of legitimate and malicious identity requests. In summary, the results achieved were promising. At least 61% of legitimate requests were assigned a good reputation (i.e. 0.5 or higher), thus being regarded as presumably legitimate and minimally penalized. Another 34% of legitimate requests were assigned a value of trust score that, although suggesting a worse reputation, resulted into puzzles whose complexity were by far smaller than those assigned to malicious requests. Conversely, less than 5% of malicious requests (considering a handful requests made) were assigned good values of trust score. These results evidence that our solution is able to effectively identify legitimate requests as being so, with a low rate of false positives.

In Table 3 we compare the results achieved with the analytical model to those obtained through simulation. Observe that the model enables us to assess identity assignment trends for a given set of parameters, also establishing an upper bound for the number of malicious identities created/valid. It is important to emphasize that the difference observed when comparing the number of valid malicious identities occurs due to the simplified nature of our analytical model. For example, the model uses a same value of trust score for all identity requests within a given round (which has the duration of an entire window Δt). The simulation, in contrast, updates the value of trust score after every identity is assigned (similarly to what would take place in the wild).

Table 4 presents a statistical summary of puzzle resolution times, in the botnet scenario. The results show that in

Table 3
A comparison of the number of identities created (fake and legitimate) considering the results obtained through simulation and through the analytical model.

Scenarios	1		10		0.5%		1%		5%	
	Model	Sim.	Model	Sim.	Model	Sim.	Model	Sim.	Model	Sim.
Legitimate, $M_u = 10$	159,636	159,918	159,636	159,917	159,636	159,917	159,636	159,917	159,636	159,921
Malicious, $M_u = 10$	463	10	1039	72	1168	115	1187	273	1202	1303
Legitimate, $M_u = 5\%$	159,636	159,918	159,636	159,917	159,636	159,918	159,636	159,918	159,636	159,924
Malicious, $M_u = 5\%$	1336	40	6767	256	24,073	1115	35,453	1661	45,714	4476
Legitimate, *botnet*	159,636	159,918	159,636	159,917	159,636	159,917	159,636	159,916	159,636	159,925
Malicious, *botnet*	103	5	1039	72	5203	400	10,431	908	45,714	4476

Table 4
Puzzle resolution times (s) in the botnet scenario.

Scenario		Mean	Std. dev.	Median	9th decile
Adaptive puzzles	legitimate users, "$M_u = 10$"	2409	6641	82	7623
	attacker, "$M_u = 10$"	12,330	9758	13,140	26,242
	legitimate users, "$M_u = 5\%$"	2960	7880	63	11,178
	attacker, "$M_u = 5\%$"	10,050	9426	6582	26,242
Green and useful puzzles	legitimate users, "$M_u = 10$"	697	1766	60	2093
	attacker, "$M_u = 10$"	1716	1647	1666	3305
	legitimate users, "$M_u = 5\%$"	754	1850	55	2592
	attacker, "$M_u = 5\%$"	1338	1550	847	3305
Static puzzles, $\Gamma = 10$	legitimate users, "$M_u = 5\%$"	517	228	471	629
	attacker, "$M_u = 5\%$"	412	0	412	412
Static puzzles, $\Gamma = 15$	legitimate users, "$M_u = 5\%$"	16,420	6546	14,990	20,025
	attacker, "$M_u = 5\%$"	13,110	0	13,110	13,110

the case of green and useful puzzles, legitimate users are assigned easier-to-solve ones (each puzzle took 697 s on average to be solved, in the scenario "$M_u = 10$"). In contrast, the attacker took 146% more time on average to solve puzzles (1716 s in the same scenario). More importantly, 90% of users took at most 2093 s to solve puzzles; for the attacker, it took 3305 s.

For static puzzles, legitimate users and attacker took on average almost the same time to solve them (which was extremely high for "$\Gamma = 15$"); the difference observed is because the computing power of legitimate users is not uniform, and different from the attacker (which is fixed). In summary, these results evidence that green and useful puzzles represent an interesting approach for tackling fake accounts, as they (i) impose a significant overhead per identity obtained by the attacker, (ii)minimizes the burden caused to legitimate users, and (iii) decreases the overall processing effort required for puzzle solving, without losing effectiveness in stopping attackers.

7.1.3. Energy efficiency

We calculated an estimate for energy consumption based on the resolution of puzzles written in *python*, on an Intel Core i3-350M notebook, with 3MB of cache memory, 2.26 GHz CPU clock, and Windows 7. We used JouleMeter[5] for measurement, and considered only the processor energy consumption. After 150 runs, we observed an average consumption of 1.215 J. It is important to emphasize that, although we do not consider the various existing hardware and processor types, this estimate remains as an important indicator – neglected in previous investigations – for the average energy consumption expected for a puzzle-based solution.

The total energy consumption (summing up legitimate users and the attacker) caused by green puzzles in the $M_u = 5\%$ botnet scenario (330 MJ) is significantly lower compared to the measured for adaptive puzzles (1490 MJ); the savings ratio is $D = 0.71$. More importantly, it is only 4.9% of the consumption estimated for static puzzles (6742 MJ: 319,433 puzzles of complexity 15 assigned to legitimate users, and 23,176 to the attacker). In this comparison, the savings ratio is $D = 0.95$, i.e., a difference of 6412 MJ (1.78 MWh); this is equivalent to 69% of the annual energy consumption *per capita* in Brazil [31]. These results not only emphasize the need for green puzzles, but also highlight the potentialities of using waiting time to materialize them.

An important observation regarding energy consumption on the bootstrap side is that the puzzle verification load remains similar when comparing adaptive puzzles with green and useful puzzles, and also with static puzzles. This is because the cost of verifying the solution of a puzzle is constant and deemed negligible, and does not change regardless of its complexity. As these figures are similar, we do not take into account bootstrap-side energy consumption for puzzle verification on our evaluation.

7.1.4. Monetary analysis of the attack

Here we present a budgetary analysis of the attack, considering each identity the attacker could obtain during the 168 h simulation period. For estimating prices, we

[5] JouleMeter page: http://research.microsoft.com/en-us/projects/⟨?PMU?⟩joulemeter/.

Table 5
Budgetary analysis of an attack launched against our solution.

Scenario	Cluster	$M_u = 10$			$M_u = 5\%$		
		Identities (% of attackers' goal)	Total	US$/id	Identities (% of attackers' goal)	Total	US$/id
$M_c = 1$	1	10 (less than 0.1%)	$14	1.40	40 (less than 0.1%)	$259	6.47
$0.8M_c = 10$	10	72 (less than 0.1%)	$102	1.41	256 (0.3%)	$347	1.35
$M_c = 0.5\%$	50	115 (0.1%)	$492	4.27	1115 (1.3%)	$737	0.66
$0.8M_c = 1\%$	100	273 (0.3%)	$979	3.58	1661 (2%)	$1224	0.73
$M_c = 5\%$	500	1303 (1.6%)	$4877	3.74	4476 (5.5%)	$5122	1.14

considered a 2-vCPU Amazon EC2 Compute Optimized solution (for providing computing power), which can be hired for as low as US$0.116/h each, US$0.058/h per vCPU (an equivalent Microsoft Azure D-series virtual machine solution costs US$0.094/h) [32,33]. For spoofing sources of identity requests, we considered a botnet consulting service offered in the black market for US$500 per 1000 premium, non-blacklisted, zombie computers ([34,35]. Table 5 presents an overview of the results achieved for our solution. The values in field "cluster" are displayed in computing units hired. Field "identities" displays the number of valid fake accounts by the end of the simulation (and percentage of the attackers' goal fulfilled); field "total" contains the budget of the attack in each scenario considered (in US dollars); and field "US$/id" contains the cost per fake account. In the scenarios where static puzzles were used, the number of cluster instances hired was 500. In total, the budget for the attack was US$ 4,872 (for a 168 h computing power rental period). In the case of $\Gamma = 10$, each of the 79,945 identities created (99.9% of attackers' goal) had a cost of $\approx$ US$ 0.06; in the case of $\Gamma = 15$, it rose to $\approx$ US$ 0.77 for each of the 6327 identities created (7.9% of attackers' goal).

There is no precise estimate on the profit one can make with fake accounts. According to one study by Barracuda Labs, cited by The New York Times [36], the average price for 1000 twitter followers ranges around US$ 18 in the black market; prices can be as low as US$ 11 depending on the reseller [37,38]. There are also dealers offering a bulk of 100 Gmail accounts for US$ 10 [39]. In both cases, the minimum expected value one can obtain with fake accounts (US$ 0.018 per fake follower, or US$ 0.10 per fake mail account) is surpassed by the cost of obtaining a single fake account that our solution imposes; in the scenario the attacker maximizes the number of identities obtained, each had a cost of US$ 1.14. Observe also that the cost per identity with our solution is much higher compared to static puzzles.

7.2. PlanetLab

The primary goal of this evaluation – carried out using BitTornado – is to assess the technical feasibility of our solution. We also compare it with existing approaches. It is important to emphasize that the parameter setting adopted in this evaluation attempted to replicate, in a smaller scale, the scenarios considered in our simulations. In summary, we considered an environment having 240 legitimate sources and 20 malicious ones. The legitimate users request 2400 identities during one hour. The first request of each user is uniformly distributed during this period; their recurrence follows an exponential distribution, varying from 1 to 15 min. The interval between arrivals is also exponentially distributed, between 1 and 10 min. The attacker requests 1200 identities (1/3 of the requests of legitimate users), making an average of 60 identities per malicious source; their recurrence follows a fixed rate of one request per minute. Our evaluation (including the behavior of legitimate users and the attacker, and duration of the evaluation) was defined observing the technical constraints imposed by the PlanetLab environment (e.g., limited computing power, unstable nodes, and network connectivity); due to these constraints, the identity renewal aspect of our solution could not be evaluated. As for the proportion of 1/3 of identities, it was chosen since it exceeds the fraction of fake accounts that sybil-resilient solutions tolerate [40,41].

To make puzzles useful in our design, we used a software that emulates a small simulation experiment; it receives a list of random number generator seeds, and generates a single text file containing the results (for all seeds informed). The puzzle complexity is determined by the number of seeds informed, which in turn is proportional to $2^{\gamma_i(t)-1}$. For static puzzles, we considered the one proposed by Douceur [1] (discussed in Section 4.1).

The other parameters were defined as follows. For our solution, $\Delta t = 48$ h, $\beta = 0.125$, $\Gamma_{pl,req} = 22$ (which is equivalent to $\Gamma = 4$ used in the simulation model), $\Gamma_{pl,reval} = 21$ ($\Gamma_{reval} = 3$), $\Gamma_{pl,renew} = 20$ ($\Gamma_{renew} = 2$), and $\Omega = 10$. For the mechanism based on static puzzles, we considered three scenarios: $\gamma_{pl,1} = 16$ ($\gamma = 1$), $\gamma_{pl,2} = 20$ ($\gamma = 2$), and $\gamma_{pl,3} = 24$ ($\gamma = 6$). The adjust in the puzzle complexity, comparing the simulation model with the evaluation presented next, was necessary for adapting the puzzle-based mechanisms to the computing power constraints in the PlanetLab environment.

Fig. 9 shows that the dynamic of identity assignments to legitimate users with the proposed solution (curve "Our solution") is similar to the no control scenario ("No control"). In contrast, it evidences the overhead/ineffectiveness of using static puzzles for identity management. Focusing on the attacker, our solution reduced significantly the number of fake accounts she created (compared to the scenario without control).

The energy consumption estimates obtained also indicate the efficacy of our solution. While static puzzles with $\gamma_{pl,1} = 16$, $\gamma_{pl,2} = 20$, and $\gamma_{pl,3} = 24$ consumed 58.70 KJ, 533.85 KJ, and 803.92 KJ (respectively), our solution

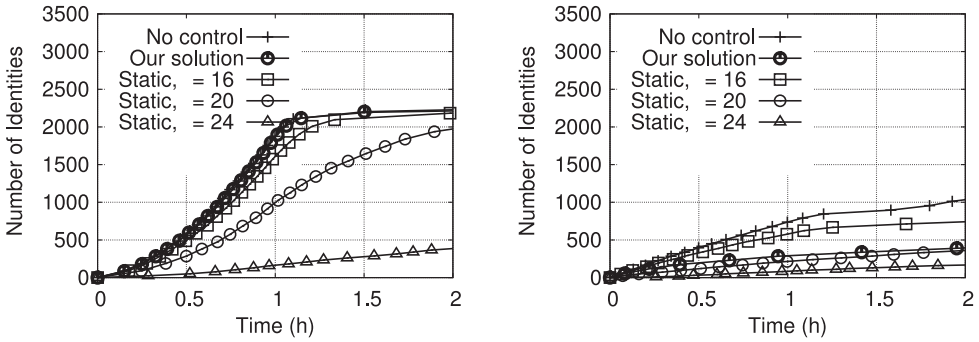


Fig. 9. Results achieved with the PlanetLab environment.

led to a consumption of 13.39 KJ only. It represents 22.81% ($D_{\gamma_{pl,1}} = 0.7718$), 2.41% ($D_{\gamma_{pl,2}} = 0.9749$), and 1.66% ($D_{\gamma_{pl,3}} = 0.9833$) of the estimated consumption with static puzzles.

Our experiments with the PlanetLab environment also served to prove the effectiveness of using data processing jobs in replacement of cryptographic puzzles, as a strategy to make puzzles useful. In summary, the experiments carried out in the PlanetLab environment not only confirmed the results achieved through simulation, but also evidenced the technical feasibility of using adaptive puzzles, waiting time, and massive distributed computing for *green* and *useful* identity management.

8. Final considerations

The use of puzzles to limit the spread of fake accounts has been hampered, among other reasons, due to the lack of mechanisms that deal properly with situations in which there is a gap of computing power between legitimate users and attackers. Existing solutions do not take advantage of discrepant behaviors observed between legitimate users and attackers as an approach to weight the complexity of assigned puzzles. To bridge this gap, we proposed a lightweight scheme for long-term identity management, based on adaptive puzzles, waiting time, and massive distributed computing.

The experiments carried out have shown the effectiveness of our solution in decreasing the attackers' ability of creating an indiscriminate number of counterfeit identities, without compromising legitimate users, using puzzles that consume resources in an efficient and useful manner. In summary, our solution has shown to be possible to force potential attackers to pay substantially higher costs for each identity; conversely, legitimate users received more easier-to-solve puzzles than the attacker, and took 43% less time on average to solve them. The use of waiting time, technique traditionally used in websites to limit the access to services, led to significant energy savings (at least 94% when compared to static puzzles [9]). More importantly, we observed an improvement of 81% in the mitigation of fake accounts when compared to the state-of-the-art mechanisms; this provides evidence to our claim that a puzzle-based identity management scheme can be modified so as to reduce its resource consumption, and without compromising its effectiveness. Finally, the use of massive distributed computing has shown to be technically feasible (considering several experiments carried out in environments such as PlanetLab) for providing utility for the processing cycles dedicated to solve puzzles.

The experiments carried out evidenced two major issues associated to puzzles. First, it was confirmed the unfeasibility of using static puzzles, given the difficulty in establishing a complexity that is effective against attackers and less harmful to legitimate users. Second, cryptographic puzzles have not shown to be reliable in assuring that an attacker will be penalized as expected. For example, the resolution time of a puzzle having a given complexity, in a certain multi-core hardware, varied from a few seconds to hundreds of minutes. The use of lessons learned from massive distributed computing, and the replacement of cryptographic puzzles with real data processing jobs (in our evaluation, simulation jobs), has shown to be a promising direction to deal with this issue.

In spite of the progresses reported, many opportunities for research remain. The most prominent one is the instantiation of our solution in the context of online social networks. Our idea is to introduce an admission process in which newly created accounts are regarded as "verified" only after a number of users already in the system (determined as a function of a trust score value) have vouched for them.

Acknowledgments

This research work was supported in part by funding from Project 462091/2014-7, granted by CNPq (MCTI/CNPQ/Universal 14/2014 - Faixa A).

References

[1] J.R. Douceur, The sybil attack, in: Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002), 2002, pp. 251–260.
[2] O. Jetter, J. Dinger, H. Hartenstein, Quantitative analysis of the sybil attack and effective sybil resistance in peer-to-peer systems, in: Proceedings of International Communications Conference (ICC 2010), Cape Town, South Africa, 2010, pp. 1–6.
[3] H. Yu, Sybil defenses via social networks: a tutorial and survey, SIGACT News 42 (3) (2011) 80–101, doi:10.1145/2034575.2034593.

[4] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, A. Panconesi, Sok: the evolution of sybil defense via social networks, in: Proceedings of IEEE Symposium on Security and Privacy (SP), 2013, pp. 382–396.
[5] G. Danezis, P. Mittal, Sybilinfer: detecting sybil nodes using social networks, in: Proceedings of Network and Distributed System Security Symposium (NDSS 2009), The Internet Society, San Diego, California, USA, 2009.
[6] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, in: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI 2012), USENIX Association, Berkeley, CA, USA, 2012, p. 15.
[7] P. Liu, X. Wang, X. Che, Z. Chen, Y. Gu, Defense against sybil attacks in directed social networks, in: Proceedings of the 19th International Conference on Digital Signal Processing (DSP), 2014, pp. 239–243.
[8] N. Borisov, Computational puzzles as sybil defenses, in: Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P 2006), 2006, pp. 171–176.
[9] H. Rowaihy, W. Enck, P. McDaniel, T. La Porta, Limiting sybil attacks in structured p2p networks, in: Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007), Anchorage, Alaska, USA, 2007, pp. 2596–2600.
[10] B. Groza, B. Warinschi, Cryptographic puzzles and dos resilience, revisited, Des. Codes Cryptogr. 73 (1) (2014) 177–207, doi:10.1007/s10623-013-9816-5.
[11] T.L.Q. Bui, Using Spammers' Computing Resources for Volunteer Computing, Portland State University, 2014 (Master's thesis). http://archives.pdx.edu/ds/psu/11031
[12] A. Mohaisen, A. Yun, Y. Kim, Measuring the mixing time of social graphs, in: Proceedings of the 10th annual Conference on Internet Measurement, ACM, New York, NY, USA, 2010, pp. 383–389.
[13] Z. Yang, C. Wilson, X. Wang, T. Gao, B.Y. Zhao, Y. Dai, Uncovering social network sybils in the wild, in: Proceedings of ACM SIGCOMM Conference on Internet Measurement Conference (IMC'11), ACM, New York, NY, USA, 2011, pp. 259–268.
[14] J. Angwin, J. Singer-Vine, Selling you on facebook, Wall Street J. (2012). http://online.wsj.com/article/SB10001424052702303302504577327744009046230.html
[15] B. Fung, Whisper: the 'anonymous' messaging app that reportedly tracks your location and shares data with the pentagon, 2014, [Online]. Available: URL: http://www.washingtonpost.com/blogs/the-switch/wp/2014/10/16/whisper-the-anonymous-messaging-app-that-reportedly-tracks-your-location-and-shares-data-with-the-pentagon/.
[16] S. Jayson, Social media research raises privacy and ethics issues, USA Today (2014). http://www.usatoday.com/story/news/nation/2014/03/08/data-online-behavior-research/5781447/
[17] W. Cordeiro, F.R. Santos, G.H. Mauch, M.P. Barcelos, L.P. Gaspary, Identity management based on adaptive puzzles to protect p2p systems from sybil attacks, Comput. Netw. 56 (11) (2012) 2569–2589.
[18] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, D.S. Wallach, Secure routing for structured peer-to-peer overlay networks, in: Proceedings of the 5th Usenix Symposium on Operating Systems Design and Implementation (OSDI 2002), 2002, pp. 299–314.
[19] K. Aberer, A. Datta, M. Hauswirth, A decentralized public key infrastructure for customer-to customer e-commerce, Int. J. Bus. Process Integr. Manag. 1 (1) (2005) 26–33.
[20] R. Morselli, B. Bhattacharjee, J. Katz, M.A. Marsh, Keychains: A Decentralized Public-key Infrastructure, 2006 URL: http://hdl.handle.net/1903/3332.
[21] D. Levin, J.R. Douceur, J.R. Lorch, T. Moscibroda, Trinc: small trusted hardware for large distributed systems, in: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2009.
[22] J. Liang, N. Naoumov, K.W. Ross, The index poisoning attack in p2p file-sharing systems, in: Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006), Barcelona, Catalunya, Spain, 2006, pp. 1–12.
[23] SpamHaus, The SpamHaus Project, 2013. URL: http://www.spamhaus.org/
[24] N. Tran, J. Li, L. Subramanian, S. Chow, Optimal sybil-resilient node admission control, in: Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM 2011),Shangai, China, 2011, pp. 3218–3226.
[25] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decis. Support Syst. 43 (2) (2007) 618–644.Emerging Issues in Collaborative Commerce
[26] A. Jøsang, Robustness of trust and reputation systems: Does it matter? in: Proceedings of 6th IFIP WG 11.11 International Conference on Trust Management VI (IFIPTM 2012), in: IFIP Advances in Information and Communication Technology, vol. 374, Springer, 2012, pp. 253–262.
[27] B. Zhang, A. Iosup, J. Pouwelse, D. Epema, The Peer-to-Peer Trace Archive, 2012. URL: http://p2pta.ewi.tudelft.nl/pmwiki/?n=Main.Home
[28] R.B. Mansilha, L.R. Bays, M.B. Lehmann, A. Mezzomo, G. Facchini, L.P. Gaspary, M.P. Barcellos, Observing the bittorrent universe through telescopes, in: Proceedings of the 2011 IFIP/IEEE International Symposium on Integrated Network Management, IEEE Computer Society, Washington, DC, USA, 2011.
[29] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, M. Blum, Recaptcha: human-based character recognition via web security measures, Science 321 (5895) (2008) 1465–1468.
[30] G. Maier, F. Schneider, A. Feldmann, Nat usage in residential broadband networks, in: N. Spring, G. Riley (Eds.), Passive and Active Measurement, Lecture Notes in Computer Science, vol. 6579, Springer, Berlin/Heidelberg, 2011, pp. 32–41.
[31] EPE, Anuário Estatístico de Energia Elétrica–population, Consumption and Per Capita Consumption, 2015 URL: http://www.epe.gov.br/AnuarioEstatisticodeEnergiaEletrica/Forms/Anurio.aspx.
[32] Microsoft Azure, Virtual Machines Pricing–Linux, 2015. URL: http://azure.microsoft.com/en-us/pricing/details/virtual-machines/#Linux
[33] AMAZON.COM, Amazon ec2 Pricing, 2015. URL: http://aws.amazon.com/ec2/pricing/.
[34] J. Leyden, Need an Army of Killer Zombies? Yours for Just $25 per 1,000 pcs, 2013. URL: http://aws.amazon.com/ec2/pricing/
[35] B. Prince, Botnet Business Continues to Thrive: Fortinet, 2013. URL: http://www.eweek.com/security/botnet-business-continues-to-thrive-fortinet/
[36] NEW YORK TIMES, Fake Twitter Followers Become Multimillion-dollar Business, 2013. URL: http://bits.blogs.nytimes.com/2013/04/05/fake-twitter-followers-becomes-multimillion-dollar-business/
[37] SOCIAL TIMES, Market for Fake Twitter Accounts is Booming, 2013. URL: http://socialtimes.com/market-for-fake-twitter-accounts-is-booming_b131197.
[38] eWEEK, Market for Fake Social Network Accounts Still Booming, 2013. URL: http://buygmailaccounts.org/.
[39] Buy Gmail Accounts, 2013. URL: http://buygmailaccounts.org/.
[40] H. Yu, M. Kaminsky, P.B. Gibbons, A. Flaxman, Sybilguard: defending against sybil attacks via social networks, in: Proceedings of 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06), ACM Press, New York, NY, USA, 2006, pp. 267–278.
[41] H. Yu, P.B. Gibbons, M. Kaminsky, F. Xiao, Sybillimit: a near-optimal social network defense against sybil attacks, in: Proceedings of IEEE Symposium on Security and Privacy, IEEE Computer Society, 2008.

Weverton Luis da Costa Cordeiro is a Postdoctoral Fellow at the Institute of Informatics of the Federal University of Rio Grande do Sul, Brazil. He holds a Ph.D. degree in Computer Science from the Federal University of Rio Grande do Sul (2014). His research interests include large scale distributed systems, information technology service management, software defined networks, network security and monitoring, future internet, and mobile ad hoc networks.

Flávio Roberto Santos holds a Ph.D. degree in Computer Networks from the Federal University of Rio Grande do Sul, Brazil. He completed his bachelors in Computer Science at the Federal University of Campina Grande. His research interests include algorithms, peer-to-peer and live streaming systems, reputation mechanisms, and grid computing.

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 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. He is member of the SBC, IEEE, and 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, GLOBECOM and ICC. See http://www.inf.ufrgs.br/~paschoal for further details.

Hanna Kavalionak received the M.Sc. degree in physics, from Belarusian State University in 2008, and the Ph.D. degree in Information and Communication Technologies from the University of Trento, Italy, in 2013. Since 2014, she has been a Postdoctoral Researcher at Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche. Her research focuses on distributed systems, autonomous resource management in P2P and hybrid P2P-Cloud systems, gossip-based algorithms, and data stream management. She authored several papers in international conferences.

Alessio Guerrieri is a Ph.D. student in Computer Science at the University of Trento, Italy. He obtained Dual Master Degree in Computer Science at University of Trento and at Georgia Institute of Technology. His research interests interests lie in decentralized and distributed algorithms in general.

Alberto Montresor is Associate Professor at the University of Trento since 2005. Before that, he has been on the faculty of the University of Bologna (2002–2005). He has authored more than 70 papers on large- scale distributed systems, cloud computing and P2P networks. His main goal is to develop distributed protocols and systems that "survive": to large scale, to dynamism, to failures, to adversarial environments. He is Steering Committee Chair of the IEEE Conference on P2P Computing, Associate Editor of Springer Computing and has served as General Chair and Program Chair in several conferences (DOA, DAIS, SASO, P2P).