Make it Green and Useful: Reshaping Puzzles for Identity Management in Large-scale Distributed Systems

Weverton Luis da Costa Cordeiro, Flávio Roberto Santos,
Marinho Pilla Barcellos, Luciano Paschoal Gaspary

Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
{wlccordeiro, frsantos, marinho, paschoal}@inf.ufrgs.br

***Abstract*—A vast number of large-scale distributed systems offer a lightweight process for creating new accounts, so that users can easily join them. Although convenient, such lightweight process fosters the spread of fake accounts (Sybil attack). Existing identity management schemes lack mechanisms to make identity creation easier for honest users and at the same time increasingly harder for an attacker. In this paper, we focus on identity lifecycle management as an (alternative) approach in order to augment the cost of possessing several identities, and thus reduce the volume of counterfeit ones. We build on adaptive puzzles and combine them with waiting time to introduce a *green* design for lightweight, long-term identity management; it minimally penalizes honest users (by assigning easier-to-solve puzzles to them), and reduces the energy consumption caused by puzzle-solving (by adopting passive wait to reduce their average complexity). We also take advantage of lessons learned from massive distributed computing to come up with a design that makes puzzle-processing *useful*. We evaluate our proposal via simulation and experimentation using PlanetLab. In summary, we show that an attacker must dedicate a large amount of resources to control a given fraction of identities. We also provide evidence that the overhead imposed to honest users is kept to a minimum.**

I. Introduction

Online systems such as Facebook, Twitter, Digg, Skype, and BitTorrent communities (among various others) offer a lightweight process for obtaining identities (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 horde of fake acounts (Sybil attack [1]), and use them to either perform malicious activities (that might harm honest users) or obtain unfair benefits. The corruptive power of counterfeit identities (or sybils) is widely known, being the object of several studies in the literature [2].

There exists no silver bullet or "one size fits all" solution to this problem. The major challenge faced by the operations & management community is then to devise identity management schemes that support a multitude of users, using heterogeneous devices, in environments having a diverse set of purposes, requirements, and constraints (e.g., large-scale distributed systems, Internet-of-Things [3], and Future Internet [4]). More importantly, such schemes should be resilient enough so as to make it easier for honest users to obtain their identities and, at the same time, increasingly harder for an attacker.

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 accomodate lightweight, long-term identity management schemes [5] (e.g. file sharing and live streaming networks, collaborative intrusion detection systems, among others); *lightweight* because users should obtain identities without being required to provide "proof of identity" (e.g., personal documents) and/or pay taxes; and *long-term* because users should be able to maintain their identities (e.g., through renewal) for an indefinite period.

In the scope of these systems, strategies such as social networks [2], [6] and proof of work (e.g., computational puzzles) [7], [8] have been suggested as promising directions to limit the spread of fake accounts. In spite of the potentialities, important questions remain. Recent research [9], [10] has shown that some of the key assumptions on which social network-based schemes rely (e.g., social graphs are fast mixing, and sybils form tight-knit communities) are invalid. More importantly, the use of social networks for identity verification might violate the user's privacy. This is an extremely sensitive issue, specially in a moment when there is a growing concern and discussion about privacy issues in social networks [11].

Puzzle-based schemes inherently preserve the 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 limit the spread of sybils [7], [8]. However, puzzle-solving incur 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 honest users for each identity?
2) Can one reduce resource consumption required for puzzle-solving, without compromising its effectiveness?
3) Can one compute useful information with the processing cycles allocated for puzzle-solving?

To tackle this issue, we build on adaptive puzzles [12] – a mechanism that defines puzzle complexity based on the frequency in which users (sources of requests) obtain new identities – and combine them with waiting time to introduce a *green* design for lightweight, long-term identity management. Our design minimally penalizes presumably honest users with easier-to-solve puzzles, and also reduces energy consumption incurred from puzzle-solving. 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 data processing jobs in replacement to cryptographic puzzles. This is similar to the phylosophy of the ReCAPTCHA project, which aims at keeping robots away from websites and helps

978-3-901882-50-0

digitizing books [13]. In summary, we make the following contributions to the state of the art:

- An identity management scheme for large-scale distributed systems, in which honest users are less penalized for obtaining/controlling identities than attackers;
- The concept of "wait time" as a strategy to decrease the average puzzle complexity, and consequently the energy consumption incurred from puzzle-solving;
- The reshaping of puzzles based on lessons learned from massive distributed computing, to provide utility to the processing cycles dedicated to solving puzzles.

To assess the effectiveness of our design, we evaluated it by means of simulation and experiments using PlanetLab. The results obtained show that an attacker must dedicate a large amount of resources to control $1/3$ of the identities in the system (proportion considered since it exceedes the amount of fake accounts in large-scale distributed systems that most of the sybil-tolerant solutions are able to cope with [14], [15]). The results also show that presumably honest users are minimally affected (being assigned easier-to-solve puzzles), and that the overall energy consumption is comparatively lower than in existing puzzle-based identity management schemes.

The remainder of this paper is organized as follows. In Section II we introduce our design for identity management, whereas in Section III we describe our proposal for making puzzles green and useful. In Section IV we present and analyze the results obtained. Finally, we briefly review related work in Section V, and close the paper in Section VI with concluding remarks and perspectives for future research.

II. Conceptual Design for Identity Management in Large-scale Distributed Systems

Our design for identity management is built upon the notion that a user has to dedicate a fraction of resources to obtain and renew identities in large-scale distributed systems. The primary goal of our design is to minimize the number of conterfeit identities that an attacker can obtain and control. The secondary goal is to reduce as much as possible the resource consumption incurred from puzzle-solving, and make their processing something useful. Our design makes the following considerations about the target system:

- The amount of resources available to users (processing cycles, memory, time, etc.), although unknown, is finite. Therefore, it is possible to bound the number of identities a single entity can control by establishing a "price" for the possession of each identity;
- To control at least $1/3$ of the identities in a system, an attacker must launch several requests to the system identity management entity. By tracking these requests back to their sources of origin, it is possible to assign higher costs to those coming from sources that have performed a larger number of requests;
- It is often difficult (and in certain circumstances, impossible) to reliably track sources of requests. With some effort, an attacker may obfuscate the source of her requests so that they appear to originate from various, distinct ones. For this reason, our design requires that identities be periodically renewed (as in [8]), in order to make it even more prohibitive for an attacker to accumulate them.

Our design is suitable for systems with various degrees of centralization (purely centralized, with a number of distributed servers, and structured networks). It is not suitable, however, for purely decentralized systems, where there is no central service providing any sort of coordination to participating users. In the remainder of this paper we discuss how our proposed scheme exploits the characteristics enumerated above to limit the number of fake accounts an attacker can control.

A. Identity Management Overview

To aid the presentation of our design, we use the following terminology. We call an entity interested in obtaining an identity a *user*. The *bootstrap service* is the entity responsible for granting/renewing identities to users. From the moment a user has a working identity, we call it a *peer*. A *source* is the location (identified by the bootstrap service) from which a given user requests her identity. The definition of a source accomodates the situation in which both several users and/or several identity requests are associated to a single location. An *attacker* is a person interested in controlling a large fraction of fake accounts in the system.

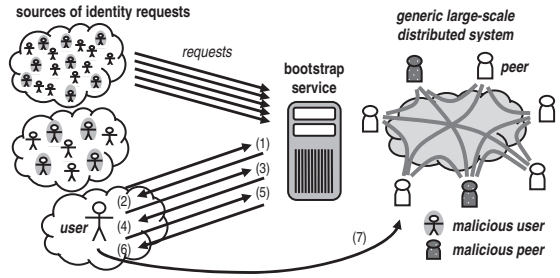


Fig. 1. Overview of the proposed identity management scheme

Fig. 1 gives an overview of the identity lifecycle management process (further discussed in Subsec. II-B). Before joining the system, a user must contact the bootstrap service in order to obtain an identity (arrow 1 in Fig. 1). In response, it estimates the complexity of the puzzle the user has to solve (based on the trust score of the source the user is associated to) in order to proceed with the request. Once the bootstrap is done, it replies the user with the puzzle to be solved and its parameters (arrow 2). The user then solves the assigned puzzle and returns to the bootstrap both the puzzle and its solution (3). The bootstrap validates the authenticity of the puzzle, the correctness of the solution, and then replies the user with a "waiting time" (4). This waiting time, also computed based on the source trust score, must be respected by the user before she contacts the bootstrap again to finally obtain a working identity (5). The bootstrap then delivers the requested identity (6), which can be used to join the system (7). This procedure is similar for renewing an existing identity, as a registered user is expected to provide fresher evidence about her identity.

B. Proposed Identity Lifecycle and Supporting Protocol

In our scheme, we envisage an identity as a tuple $I = \langle i, t, v, e, \theta, s \rangle$. 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). Element $I\langle v \rangle$ represents the validity timestamp. Being T the current time, an identity is valid for

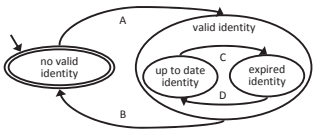


Fig. 2. Possible states in an identity lifecycle

contacting the bootstrap only if $I\langle v\rangle \leq T$ (i.e., $I\langle v\rangle$ has not yet passed); otherwise, the user must obtain a new identity (which will incur in a comparatively higher cost than in the case of renewal, as it will be discussed in the following subsections).

Element $I\langle e\rangle$ denotes the expiration timestamp (with $I\langle e\rangle \leq I\langle v\rangle$). An identity is valid for identifying a user in the system only if $I\langle e\rangle \leq T$. Observe that $I\langle e\rangle$ may have expired, but the identity can still be valid for renewal; this is true as long as $I\langle v\rangle$ is not yet passed. Element $I\langle\theta\rangle$ represents the trust score associated to the identity (it will be discussed in the following subsection). Finally, $I\langle s\rangle$ is a digital signature of the identity (computed by the bootstrap service upon its creation or renewal), and is used to assert its authenticity. The elements $I\langle v\rangle$ and $I\langle e\rangle$ play an important role in defining the current state of an identity. The set of possible states, along with the transitions between them, is depicted in Fig. 2. The conditions that trigger each of the transitions depicted in this figure are described in the following paragraphs.

We envisage two parameters to be used by the bootstrap service to support the identity lifecycle management: V and E (with $V \geq E$). These parameters are used to update $I\langle v\rangle$ and $I\langle e\rangle$ upon creation and renewal of identities, using the current time T as base. Whenever an identity I is processed (created or renewed), the bootstrap must make $I\langle t\rangle \leftarrow T$, $I\langle v\rangle \leftarrow T+V$, and $I\langle e\rangle \leftarrow T+E$.

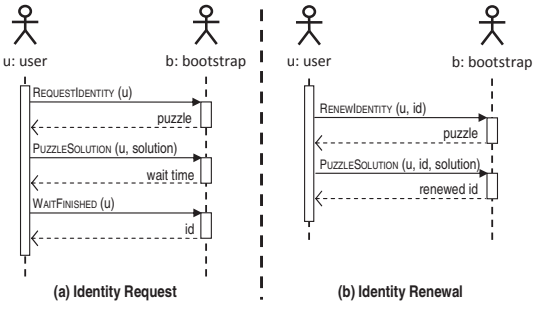


Fig. 3. Protocol for obtaining (a) and renewing (b) identities

Fig. 3 presents a general 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. The initial state of a user is "no valid identity" (either because she had never joined the system, or her identity is not valid anymore). Transition A (arrow 1 in Fig. 2) takes place when she contacts the bootstrap service to request a new identity.

The process to obtain a new identity (Fig. 3(a)) initiates when the user issues a REQUESTIDENTITY message to the bootstrap service (first arrow in the sequence diagram of Fig. 3(a)). The bootstrap replies with a puzzle to be solved (second arrow), whose complexity is defined according to the frequency of requests of the source from which the request departed. After solving the puzzle, the user issues the PUZZLESOLUTION message (third arrow). Once verified the correctness of the solution, the bootstrap increments the number of identities obtained by the source from which the request departed; after that, it calculates (and informs) a waiting period to be obeyed (fourth arrow). After the wait time is expired, the user issues the WAITFINISHED message (fifth arrow), and in response obtains the requested identity (sixth arrow). Note that the user can send this message only after finished the waiting period. Otherwise, the bootstrap can terminate the request process (as a penalty), and the user will have to restart it from scratch. To ensure that users will in fact obey the waiting time, the WAITFINISHED message should include the original message (along with its signature, for verification) in which the bootstrap informed the waiting time. Once the request identity process is completed successfully, the user evolves to the "up to date identity" state, which means she has a valid, unexpired identity.

As previously mentioned, the user must renew her identity periodically to keep contacting other peers. To this end, she performs the renewal process (Fig. 3(b)). This process is initiated when she issues a RENEWIDENTITY message (first arrow in the sequence diagram of Fig. 3(b)), informing the identity to be renewed. In response, the bootstrap assigns a puzzle to be processed (second arrow). After solving it, the user issues a PUZZLESOLUTION message informing the puzzle solution; if valid, the bootstrap renews her identity. The waiting time is not used in this process.

In case the user does not renew its identity and $I\langle e\rangle$ is passed, transition C (Fig. 2) takes place; the identity then becomes no longer valid for joining the system (state "expired identity"), but it still can be renewed by the bootstrap. If the user renews it before $I\langle v\rangle$ is passed (transition D), it becomes valid again for joining the system (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 she had never been in the system.

III. MAKING PUZZLES GREEN AND USEFUL

In this section we discuss the model for measuring the reputation of identity requests and renewals (Subsec. III-A), the concept of adaptive puzzles (Subsec. III-B) and waiting time (Subsec. III-C), and how real data processing jobs are used to materialize puzzles in our design (Subsec. III-D).

A. Model for Pricing Identity Requests/Renewals

In our identity management scheme, the complexity of puzzles is defined according to the measured trust score of the sources from which the identity requests departed. In this subsection we discuss this aspect in detail.

Identifying the Sources of Identity Requests

A source of identity requests is defined as an aggregation of one or more users, sharing locational similarities, from where identity requests depart. In Fig. 1, a source is represented as a "cloud", with one or more users (either honest or malicious) in it. It is important to emphasize that the bootstrap service has no way to distinguish users of a single source.

Two strategies have been traditionally used to identify the source from where service requests depart (in the context of this work, identity requests/renewals): IP addresses and

network coordinate systems [16]. Although effective, these strategies have important limitations. The distinction of sources considering IP addresses may penalize users located in networks using NAT. Network coordinate systems, in turn, may not precisely distinguish users located in a same region (e.g., city, neighborhood, or even a same building). In addition, both strategies can be tampered by a user that aims at forging her real location and/or faking various distinct ones.

In order to minimize the negative impact of the shortcoming discussed above, we propose to provide incentives to those users who already possess and maintain (i.e., constantly renew) their identities. The rationale is that the users who request new identities are not yet known, and therefore no strong assumption can be made on whether they are honest users or a potential attacker. The incentive we explore in this paper is to consistently decrease the complexity of puzzles these users must solve, and release them from obeying a wait time, prior to renewing their identities. In contrast, users requesting new identities are required to solve puzzles with a differentiated, higher complexity, and obey a waiting time.

Measuring the Source and Network Recurrence Rates

The identification of the source from which a request departed and the characterization of the behavior of sources is the first step towards granting a new identity to a user. To this end, it is important to maintain information of the identities effectively granted to users associated to a given source. This information, defined as $\phi_i(t)$ for the i-th source at instant t (with $\phi_i(t) \in \mathbb{N}$), serves as basis for the computation of the *source recurrence rate* and *network recurrence rate* ($\Phi(t)$). The former, defined by $\Delta\phi_i(t) = \phi_i(t) - \phi_i(t - \Delta t)$ in function of a sliding window Δt, represents the number of identities granted to a source i in the past Δt units of time. The latter is given by the total sum of identities granted, averaged by the number sources currently online (those which have obtained at least one identity in the past Δt units of time, i.e. $\Delta\phi_i(t) > 0$).

Computing the Trust Score of Identity Requests

The trust score is used for estimating both the puzzle complexity and the wait period. The procedure for computing this value depends on the process currently taking place.

1) Identity Request Process: The second stage in this process initiates with the computation of an index that reflects the relationship between the source ($\Delta\phi_i(t)$) and network recurrence rate ($\Phi(t)$). This index, defined as $\rho_i(t)$ for a given source i, is given by Eq. 1. It assumes negative values to indicate how many times the recurrence of the i-th source is smaller than of the network, and positive to indicate otherwise.

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

The $\rho_i(t)$ index serves as input for calculating a partial value of the source trust score, $\theta_i(t)$. Defined according to Eq. 2, it assumes values in the interval $[0, 1]$, with the extreme 0 meaning total distrust and 1, total trust on the legitimacy of the users associated to the i-th source.

$$\theta_i(t) = 0.5 - \frac{\arctan\left(\Phi(k) \cdot \rho_i(k)^3\right)}{\pi} \quad (2)$$

From $\theta_i(t)$ we finally obtain a smoothed trust score, which takes into account the historic behavior of the user(s) associated to the i-th source. Denoted as $\theta'_i(t_k)$ for instant t_k, it assigns a weight β (with $\beta \in (0, 1]$) to the partial value of the source trust score $\theta_i(t)$, and weight $1-\beta$ to the last known value of smoothed trust score, computed at instant t_{k-1}.

$$\theta'_i(t_k) = \begin{cases} \theta_i(t_k) & \text{, if } k = 0 \\ \beta \cdot \theta_i(t_k) + (1 - \beta) \cdot \theta_i(t_{k-1}) & \text{, if } k > 0 \end{cases} \quad (3)$$

The bootstrap must save the value of $\theta'_i(t_k)$ in the identity, i.e. $I\langle\theta\rangle \leftarrow \theta'_i(t_k)$, as it will be used during the identity renewal.

2) Identity Renewal Process: In this process, the value of $\theta'_i(t_k)$ used to estimate the puzzle complexity is computed based on the trust score saved in the identity, $I\langle\theta\rangle$, according to Eq. 4. Once the renewal process is complete, the bootstrap must save $\theta'_i(t_k)$ in the identity ($I\langle\theta\rangle \leftarrow \theta'_i(t_k)$), for use during future renewal processes.

$$\theta'_i(t_k) = \beta \cdot 1 + (1 - \beta) \cdot I\langle\theta\rangle \quad (4)$$

It is important to emphasize that the equation above represents an approach to encourage users to renew their identities. This incentive comes in the form of increasing, after each identity renewal, the value of trust score, until it reaches the extreme 1 (situation in which puzzles having the lowest complexity possible are assigned as a requirement for renewing identities). Other equations can be used, given that they provide incentives for renewing identities.

B. Defining the Puzzle Complexity

In this stage, we use as input the value of smoothed trust score $\theta'_i(t_k)$ to estimate the complexity of the puzzle to be solved. The mapping function, defined abstractly as $\gamma : \Theta \rightarrow \mathbb{N}^*$, depends essentially on the nature of the adopted puzzle. In this function, the value of the trust score $\theta'_i(t_k) \in \Theta$ is mapped to a computational puzzle having exponential complexity, equivalent to 2^γ.

An example of mapping function for computing γ is given in Eq. 5; note that the puzzle complexity is defined based on a maximum possible complexity Γ. The resulting value, $\gamma_i(t_k)$, can then be used to assess the difficulty of the puzzle.

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

As an example of puzzle, consider the one proposed by Douceur [1]: given a high random number y, find two numbers x and z such that the contatenation $x|y|z$, after processed by a secure hash function, leads to a number whose $\gamma_i(t_k)$ least significant bits are 0. The time required to solve the proposed puzzle is proportional to $2^{\gamma_i(t_k)-1}$, and the time to assert the validity of the solution is constant.

When a user requests a new identity, she must solve a puzzle defined over a differentiated, higher value of maximum

complexity, $\Gamma = \Gamma_{req}$. In order to renew her identity, she must solve a puzzle considering a lower value of maximum complexity, $\Gamma = \Gamma_{renew}$. If the identity has expired by the time the user renews it, the bootstrap service must use another value of maximum puzzle complexity, $\Gamma = \Gamma_{reval}$. As a general recommendation to encourage users to maintain their identities and renew them before expiration, $\Gamma_{renew} < \Gamma_{reval} < \Gamma_{req}$.

C. Estimating the Wait Time

Similarly to the puzzle complexity, the waiting time should increase exponentially (e.g., proportionally to 2^{ω}, where ω is a wait factor), and be defined as a function of $\theta'_i(t_k)$. The design we consider for computing ω is given in Eq. 6. In this function, Ω represents the maximum factor for the waiting time.

$$\omega_i(t_k) = \Omega \cdot (1 - \theta'_i(t_k)) \tag{6}$$

A possible attack to this protocol is the "parallelization" of the wait period: an attacker could request n identities in a same instant, solve the puzzles and use a same time interval to obey simultaneously the n assigned wait periods. To mitigate this attack, the bootstrap service must compare the current value of the source trust score with the one used to estimate the wait period (which is also included in the WAITFINISHED message); if the difference between these values exceeds a treshold $\Delta\theta$ (i.e., $\theta'_i(t_{k-1}) - \theta'_i(t_k) \geq \Delta\theta$), the bootstrap can interrupt the identity request process (similarly to what happens if the user does not obey the wait time). The rationale is that the user has not fully paid the price for obtaining an identity, as $\theta'_i(t_k)$ (the trust score of her request) decreased more than $\Delta\theta$ (maximum tolerable) since the request initiated.

D. Using Massive Distributed Computing

There are several proposals of cryptographic puzzles in the literature that can be used with our design to establish a cost for the identity renewal process [1], [7], [8]. In this paper, we propose a different type of puzzle, which takes advantage of the users' processing cycles to compute useful information.

To assign a puzzle to be solved, the bootstrap service replies to REQUESTIDENTITY and RENEWIDENTITY messages (*i*) an URL that contains a piece of software that implements the puzzle (which can be a *useful* puzzle or a cryptographic one) and (*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, chosen by the bootstrap, that must be used as input to the simulation. Supposing that $\gamma_i(t_k) = 4$ (as computed from Eq. 5), then $|\mathcal{J}| = 2^4 = 16$.

It is important to observe that one cannot fully trust on the results received, as an attacker may fake job results. For this reason, the bootstrap must inform in $\mathcal{J}$ some "test jobs" (for which the result is already known); this approach follows the same strategy used in other solutions based on massive distributed computing (e.g., ReCAPTCHA [13]). In this case, the attacker will not be able to distinguish which are "test jobs" and "real jobs"; she will have to solve all of them correctly to avoid the risk of faking the result for a test job, and having her request process terminated by the bootstrap.

IV. EVALUATION

In this section we present and discuss the results achieved by means of simulation (Subsec. IV-A) and experimentation in the PlanetLab environment (Subsec. IV-B).

A. Simulation

In this evaluation, we attempt to answer two main questions: what is the impact of our scheme to honest users and attackers? And what are the potential energy savings that our scheme can provide? To answer them, we have evaluated scenarios with and without attack, considering the following identity management schemes: without control, based on static puzzles (as proposed by Rowaihy et al. [8]), and our scheme.

Characteristics of the Simulation Environment

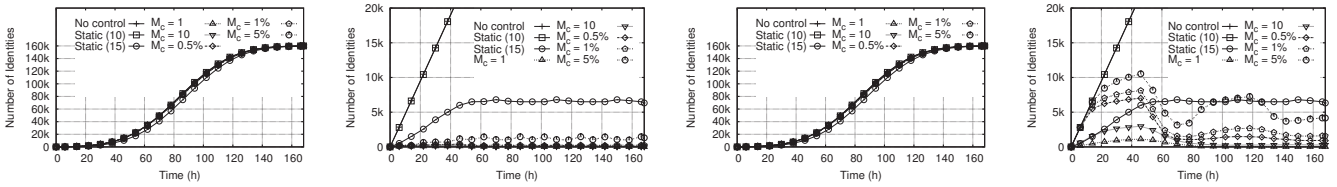
The simulation has a duration of 168 hours. In this period, 160,000 users (from 10,000 distinct sources) arrive 320,000 times in the system. The number of users behind a given source follows an exponential distribution, and varies between 1 and 16. The number of arrivals per source is exponentially distributed, bounded between 16 and 64.

The first arrival of each user is normally distributed throughout the simulation; the time between arrivals follows an exponential distribution, bounded between one minute and two hours; the user recurrence is uniformly distributed, between 1 and 2 recurrences at most per user; finally, the computing power of honest users is normalized and exponentially distributed, bounded between 0.1 ($1/10$ of the capacity of a standard, off-the-shelf hardware used as reference) and 2.5. It is important to emphasize that the above choices are arbitrary, for the sake of investigation only; as far as we are aware of, there is no such model that is generally representative of the behavior of users in large-scale, distributed systems.

To model the delay incurred from puzzle-solving, we consider that a puzzle of complexity $\gamma_i(t_k)$ takes $2^6 + 2^{\gamma_i(t_k)-1}$ seconds 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 $2^{\omega_i(t_k)}$ seconds of wait period the user must obey.

The goal of the attacker is controlling $1/3$ of the identities in the system, i.e. $80,000$. To achieve this goal, we consider two scenarios for the attacker: one in which she increases her computing power (using a cluster of high-performance computers, i.e. 2.5 times faster than the reference hardware), or increase both her computing power and the number of distinct sources from which her requests depart (using a botnet of high-performance computers).

The other parameters in our evaluation are defined as follows. For our identity management scheme, $\Delta t = 48$ hours 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 time factor is $\Omega = 17$. Given the short scale of our simulation (one week), we consider that identities expire $E = 24$ hours after created/renewed, and become invalid after $V = 48$ hours. For the puzzle-based scheme, we consider a scenario with puzzles of complexity $\Gamma = 10$ (which take around 17 minutes to be solved, depending on the hardware capacity) and $\Gamma = 15$ (which take around 9 hours to be solved).



(a) Legitimate, $M_u = 10$ (b) Fake, $M_u = 10$ (c) Legitimate, $M_u = 5\%$ (d) Fake, $M_u = 5\%$

Fig. 4. Number of legitimate and fake accounts, in the scenario where the attacker increases her computing power to solve the puzzles.

Effectiveness of our Scheme in Mitigating Fake Accounts

Fig. 4 shows the results achieved when the attacker increases her computing power as an strategy to control more counterfeit identities. The scenarios considered are one in which the attacker has $M_u = 10$ sources in her control (Figures 4(a) and 4(b)), and another in which she controls $M_u = 500$ sources (Figures 4(c) and 4(d)). The number of high-performance computers available for the attack is defined proportionally to the number of legitimate sources considered: $M_c = 1$ computer; $M_c = 10$; $M_c = 0.5\%$ (50 computers); $M_c = 1\%$ (100 computers); and $M_c = 5\%$ (500 computers).

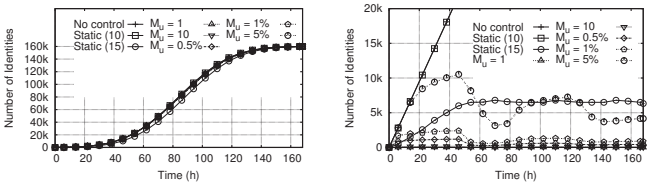
One can see in Fig. 4(b) that our scheme clearly limits the number of fake accounts the attacker can control, in contrast to the scenario where static puzzles [8] are used (curves "Static (10)" and "Static (15)"). This observarion holds even for the worst case scenario to our scheme, i.e. when the attacker has a cluster of $M_c = 5\%$ high-performance computers: our scheme reduced in 79% the number of fake accounts she can control, comparing to the scenario "Static (15)" (from 6,237 valid identities to 1,309). As for the overhead imposed to honest users, Fig. 4(a) shows that it was negligible. Observe, however, that the use of static puzzles with complexity $\Gamma = 15$ imposed a non-negligible overhead to honest users.

Fig. 4(d) evidences that increasing the computing power available is the only way the attacker can circumvent our scheme; even so, she is not able to control more identities than would happen in the case of static puzzles with complexity $\Gamma = 15$; the gain with our scheme was of 34.2% when compared to static puzzles (from 6,237 identities to 4,161). Fig. 4(c) shows that honest users remain unaffected.

From the results described above, two major conclusions can be drawn. First, our scheme 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 honest users); with $\Gamma = 10$, static puzzles were totally ineffective; with $\Gamma = 15$, there was a considerable overhead (in terms of resources to be allocated to solve puzzles) imposed to honest 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. 5 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; 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 honest users was minimum.

Table I presents the average time that honest users and



(a) Legitimate (b) Fake

Fig. 5. Number of legitimate and fake accounts in the *botnet* scenario

TABLE I. PUZZLE RESOLUTION TIMES (SECONDS)

Scenario		Mean	Std. Dev.	Median	9th decile
Proposed scheme	honest users, "$M_c = 10$"	534.5	1,317.5	55	1,751
	attacker, "$M_c = 10$"	6,224.1	1,388.5	6,582	6,582
	honest users, "$M_c = 5\%$"	627.9	1,555.1	54	1,886
	attacker, "$M_c = 5\%$"	1,322.7	1,541.8	847	3,305
Static puzzles	honest users, "$\Gamma = 10$"	497.6	198.5	455	596
	attacker, "$\Gamma = 10$"	412	0	412	412
	honest users, "$\Gamma = 15$"	15,825.4	6,032.2	14,466	19,003
	attacker, "$\Gamma = 15$"	13,110	0	13,110	13,110

the attacker take to solve puzzles, in each of the scenarios evaluated. The table also shows the standard deviation of the resolution times, median, and the 9th decile. Results show that in our scheme honest users are assigned easier-to-solve puzzles (which took 534 seconds on average to be solved, in the scenario "$M_c = 10$"). In contrast, the attacker took eleven times more on average to solve the puzzles assigned to her (6,224 seconds in the same scenario). More importantly, 90% of honest users took at most 1,751 seconds to solve the assigned puzzles; for the attacker, this time was 6,582 seconds.

In the case of static puzzles, the honest users and the attacker took on average almost the same time to solve the puzzles (and extremely high, in the scenario "$\Gamma = 15$"); the difference observed is because the computing power of honest users is not uniform, and different from the attacker (which in turn is fixed). These results evidence that long-term identity management based on adaptive puzzles and waiting time is a promising direction to tackle fake accounts.

Energy Efficiency of our Identity Management Scheme

The estimate for the energy consumption is based on the resolution of puzzles written in *python*, ran on an Intel Core i3-350M notebook, with 3MB of cache memory, 2.26 GHz CPU clock, and Windows 7. We used JouleMeter[1] to collect the measurements, and considered only the processor energy consumption with its usage around 100%. In an average of 10 runs of the puzzle for each value of complexity considered, we observed that the consumption is constant and equals to 1.215

[1]JouleMeter page: http://research.microsoft.com/en-us/projects/joulemeter/

TABLE II. PUZZLE COMPLEXITY AND ENERGY CONSUMPTION (ESTIMATES)

Puzzle complexity	Resolution (seconds)	Consumption (*joules*)	Proposed scheme			
			# leg.	KJ leg.	# mal.	KJ mal.
0	65	78.97	75,179	5,937.26	0	0
1	66	80.19	20,229	1,622.16	0	0
2	68	82.62	18,026	1,489.31	0	0
3	72	87.48	16,489	1,442.46	0	0
4	80	97.2	16,900	1,642.68	0	0
5	96	116.64	18,321	2,136.96	0	0
6	128	155.52	20,838	3,240.73	261	40.59
7	192	233.28	21,781	5,081.07	4,477	1,044.39
8	320	388.8	17,807	6,923.36	1,253	487.17
9	576	699.84	16,658	11,657.93	2,081	1,456.37
10	1.088	1,321.92	16,644	22,002.04	2,871	3,795.23
11	2.112	2,566.08	17,953	46,068.83	3,599	9,235.32
12	4.160	5,054.4	19,127	96,675.51	4,464	22,562.84
13	8.256	10,031.04	17,722	177,770.09	3,704	37,154.97
14	16.448	19,984.32	5,926	118,427.08	959	19,164.96
15	32.832	39,890.88	0	0	0	0
Total			319,600	502,117.48	23,669	94,941.85

joules. 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 – which was neglected in previous investigations – for the average energy consumption expected for a puzzle-based mechanism.

Table II gives an overview of the energy consumption caused by our scheme for the scenario $M_u = 5\%$ depicted in Fig. 5. In this table we present the values of puzzle complexity considered in the evaluation, estimates of the time required to solve such puzzle (column *resolution*), and the estimated energy consumption measured in *joules*.

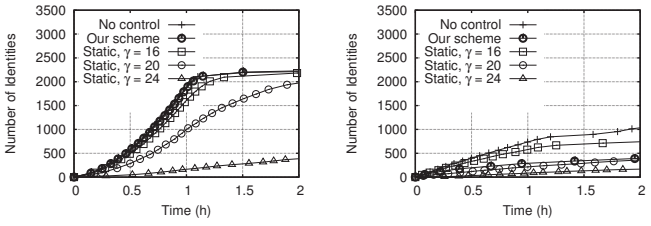
The total energy consumption (summing up honest users and the attacker) caused by our scheme (597 MJ) is only 3,84% of the consumption estimated for the mechanism based on static puzzles (15,530 MJ; consequence of 319,722 puzzles of complexity 15 assigned to honest users, and 23,174 assigned to the attacker). This represents a difference of 14,945 MJ (4.15 MWh), or 28.25% of the annual energy consumption *per capita* in Brazil [17]. These results not only emphasize the need for "green" puzzles, but also highlight the potentialities of using waiting time to materialize them.

B. Experimental Evaluation using PlanetLab

The primary goal of this evaluation – carried out using the BitTornado framework – is to assert the technical feasibility of our scheme in hampering the spread of fake accounts. We also compare our scheme with existing approaches.

In this evaluation we consider 240 honest sources and 20 malicious ones. The honest users request 2,400 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. The interval between arrivals is also exponentially distributed, between 1 and 10 minutes. The attacker requests 1,200 identities ($1/3$ of the requests of honest 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 honest users and the attacker) was defined observing the technical constraints imposed by the PlanetLab environment (e.g., limited computing power, and unstable nodes and network connectivity); due to these constraints, the identity renewal aspect of our scheme could not be evaluated.

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 $2^{\gamma_i(t_k)-1}$. For the mechanism based on static puzzles, we considered the one proposed by Douceur [1] (discussed in Subsec. III-B).



(a) Legitimate (b) Malicious

Fig. 6. Results achieved with the PlanetLab environment

The other parameters were defined as follows. For our scheme, $\Delta t = 48$ hours, $\beta = 0.125$, $\Gamma_{pl,req} = 22$ (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: $\gamma_{pl} = 16$ ($\gamma = 1$), $\gamma_{pl} = 20$ ($\gamma = 2$), and $\gamma_{pl} = 24$ ($\gamma = 6$). It is important to mention that the difference in the puzzle complexity, comparing the simulation model with the evaluation presented next, was necessary to adapt the puzzle-based mechanisms to the computing power constraints present in the PlanetLab environment.

Fig. 6 shows that the dynamic of identity assignments to honest users with the proposed scheme (curve "Our scheme") is similar to what is observed in the scenario without control ("No control"). In contrast, it evidences the overhead/ineffectiveness of using static puzzles for identity management. Focusing on the attacker, our scheme reduced significantly the number of fake accounts she created (compared to the scenario without control).

The estimates of energy consumption obtained also indicate the efficacy of our scheme. While static puzzles with $\gamma_{pl} = 16$, $\gamma_{pl} = 20$, and $\gamma_{pl} = 24$ caused an estimated consumption of 58.70 KJ, 533.85 KJ, and 803.92 KJ (respectively), our scheme led to a consumption of only 13.39 KJ. It represents 22.81%, 2.41%, and 1.66% of the estimated consumption with static puzzles. 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 compuring for *green* and *useful* identity management.

V. RELATED WORK

In 2002, Douceur [1] coined the term "Sybil attack" to designate the creation of fake accounts in online systems. The author proved that, in the absence of a central certification authority, an unknown entity can always present himself to other entitites in the system using more than one identity, unless under conditions and assumptions that are unfeasible for large-scale distributed systems. Since then, investigations on this subject have focused on limiting the dissemination of fake accounts and also on mitigating their potential harm.

According to the mechanism employed to authenticate users, these investigations can be classified into *strong* and

weak-based identity schemes [5]. The first category comprise solutions in which users obtain identities through certification authorities [18]. Although these solutions virtually prevent sybils, such mechanisms either force users to trust unknown authorities, or require them to pay fees and/or provide personal identification data to obtain identities. In the second category, users create their own identities or obtain them from bootstrap services. The major goal of the solutions that fit in this category is to bound the number of sybils to an "acceptable limit". This approach may be useful, for example, to applications that can tolerate a certain fraction of sybils. The solutions in this category may be further classified according to the strategy employed to enforce authenticity: social networks [2], [6], trust and reputation [19], IP blacklisting [20], and computational puzzles [7], [8], to cite the most prominent ones. However, these solutions are limited as they either rely on invalid assumptions and expose the user privacy, are subject to spoofing, free-riding and white-washing attacks [21], or cause the waste of a significant amount of resources.

The operations & management community has also dedicated considerable efforts on research in the topic of identity management, and many prominent schemes have been proposed in various contexts [3], [4]. These efforts, however, have mainly focused on strongly authenticating users (through certification authorities) and ensuring that services are accessed only when the user possess the required privileges; they are not suitable for those distributed systems in which weak identity-based schemes are more appropriate.

VI. Final Considerations

The use of computational puzzles has been long considered a potential strategy to mitigate the dissemination of fake accounts in large-scale, distributed systems. However, the lack of mechanisms that be fair with honest users and severe with attackers, and at the same time consume resources in an efficient and useful manner, has hampered the adoption of this strategy. To bridge this gap, we proposed a novel, lightweight scheme for long-term identity management, based on adaptive puzzles, waiting time, and massive distributed computing to limit the spread of fake accounts.

With regard to the research questions posed in the introduction, we make the following considerations. First, the results achieved have shown that it is possible to force potential attackers to pay substantially higher costs for each identity; honest users received more easier-to-solve puzzles than the attacker, and took 52.9% less time on average to solve them. Second, the use of waiting time, technique traditionally used in websites to limit the access to services, led to significant energy savings (at least 77.1% when compared to static puzzles [8]). More importantly, we observed an improvement of 34% 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. As for the third question, 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.

In spite of the progress reported in the paper, much research opportunities remain. Examples include (*i*) a refinement of the mechanism to track sources of identity requests (to make it increasingly difficult for an attacker to impersonate multiple sources), and (*ii*) the extension of the proposed scheme to make it suitable for those systems having no central service providing coordination to participating users (e.g., P2P networks such as Gnutella).

Acknowledgements

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

References

[1] J. Douceur, "The sybil attack," in *1st International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, 2002, pp. 251–260.

[2] O. Jetter, J. Dinger, and H. Hartenstein, "Quantitative Analysis of the Sybil Attack and Effective Sybil Resistance in Peer-to-Peer Systems," in *ICC 2010*, Cape Town, South Africa, may 2010, pp. 1–6.

[3] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, 2010.

[4] K. Lampropoulos and S. Denazis, "Identity management directions in future internet," *Communications Magazine, IEEE*, vol. 49, no. 12, pp. 74 –83, december 2011.

[5] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil Nodes using Social Networks," in *NDSS 2009*. San Diego, California, USA: The Internet Society, 2009.

[6] Q. Cao *et al.*, "Aiding the detection of fake accounts in large scale social online services," in *NSDI 2012*. Berkeley, CA, USA: USENIX Association, 2012.

[7] N. Borisov, "Computational Puzzles as Sybil Defenses," in *P2P 2006*, September 2006, pp. 171–176.

[8] H. Rowaihy *et al.*, "Limiting Sybil Attacks in Structured P2P Networks," in *INFOCOM 2007*, 2007, pp. 2596–2600.

[9] A. Mohaisen, A. Yun, and Y. Kim, "Measuring the mixing time of social graphs," in *IMC '10*. New York, NY, USA: ACM, 2010, pp. 383–389.

[10] Z. Yang *et al.*, "Uncovering Social Network Sybils in the Wild," in *IMC'11*. New York, NY, USA: ACM, 2011, pp. 259–268.

[11] The Wall Street Journal, "Selling You on Facebook," 2012. [Online].

[12] W. Cordeiro *et al.*, "Identity management based on adaptive puzzles to protect p2p systems from sybil attacks," *Computer Networks*, vol. 56, no. 11, pp. 2569 – 2589, 2012.

[13] L. von Ahn *et al.*, "recaptcha: Human-based character recognition via web security measures," *Science*, vol. 321, no. 5895, 2008.

[14] H. Yu *et al.*, "SybilGuard: Defending against Sybil Attacks via Social Networks," in *SIGCOMM '06*. New York, NY, USA: ACM Press, 2006, pp. 267–278.

[15] H. Yu *et al.*, "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2008.

[16] M. Sherr, M. Blaze, and B. T. Loo, "Veracity: Practical Secure Network Coordinates via Vote-based Agreements," in *USENIX '09*, June 2009.

[17] IBGE, "Sustainable Development Indexes 2012 (in Portuguese)," 2012. [Online]. Available: http://www.ibge.gov.br/home/presidencia/noticias/noticia_visualiza.php?id_noticia=2161

[18] K. Aberer, A. Datta, and M. Hauswirth, "A decentralized public key infrastructure for customer-to customer e-commerce," in *Intl. Journal of Business Process Integration and Management*, 2005, pp. 26–33.

[19] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618 – 644, 2007.

[20] J. Liang, N. Naoumov, and K. W. Ross, "Efficient blacklisting and pollution-level estimation in p2p file-sharing systems," in *AINTEC'05*. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 1–21.

[21] M. Feldman *et al.*, "Free-riding and whitewashing in peer-to-peer systems," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 5, pp. 1010–1019, 2006.