UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

WEVERTON LUIS DA COSTA CORDEIRO

Limiting Fake Accounts in Large-Scale Distributed Systems through Adaptive Identity Management

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Ph.D. Luciano Paschoal Gaspary
Advisor

Porto Alegre, January 2014

CIP – CATALOGING-IN-PUBLICATION

Cordeiro, Weverton Luis da Costa

Limiting Fake Accounts in Large-Scale Distributed Systems through Adaptive Identity Management / Weverton Luis da Costa Cordeiro. – Porto Alegre: PPGC da UFRGS, 2014.

132 f.: il.

– Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2014. Advisor: Luciano Paschoal Gaspary.

1. Identity management. 2. Peer-to-peer systems. 3. Sybil attack. 4. Fake accounts. 5. Collusion attacks. 6. Proof of work. I. Gaspary, Luciano Paschoal. II. Título.

“You’ll never get to Heaven if you’re scared of gettin’ high.”
— KYLIE MINOGUE (RED BLOODED WOMAN)

AGRADECIMENTOS / *ACKNOWLEDGMENTS*

Parece que depois de todo esse tempo de doutorado eu fiquei ruim com agradecimentos. Mas enfim, vou tentar escrever algo legal aqui; as pessoas que contribuiram para a conclusão de mais essa etapa na minha vida merecem esse reconhecimento!

There are a lot of people to thank here, from Brazil and abroad. Therefore, I will write interchangeably here in Portuguese and English in order to reach both the local and the foreign audience. I will try to write something really nice here; you guys who have helped me during my journey really deserve some recognition!

Vou começar pela epígrafe deste documento, pois ela tem uma história curiosa! Por volta de 2006 eu escutei pela primeira vez a música Red Blooded Woman (Kylie Minogue) e, no pouco conhecimento de expressões idiomáticas inglesas que tinha na época, achei a frase "You'll never get to heaven if you're scared of gettin' high" legal, meio que uma "frase de inspiração" (quanta inocência!). De tão legal que eu achei a frase, resolvi colocá-la no meu trabalho de conclusão de curso da graduação. Lógico, ela também veio junto para a minha dissertação de mestrado. E foi somente na defesa do mestrado que alguém, Prof. Marinho, durante a arguição da banca, no seu último comentário, explicou o verdadeiro sentido da frase. Óbvio que a platéia caiu na risada! Na época, fui desafiado a manter a frase na tese de doutorado. Então, cá está ela (risos)!

Let me begin by discussing the epigraph of this document, it's a curious story! By 2006 I heard Red Blooded Woman (Kylie Minogue) for the first time and, given the little knowledge of idiomatic expressions I had by then, I found the excerpt "You'll never get to heaven if you're scared of gettin' high" pretty cool, sort of a upliftment quote (how innocent!). I found it so cool that I decided to have it on my undergraduate thesis. Of course, I also wrote it on my master's thesis. It was only by my master's defense that someone, Prof. Marinho, during the committee briefing, on his very last comment, explained its actual meaning. Obviously, everyone in the audience laughed out loud! Back then, someone dared me to keep that quote in my Ph.D. thesis. So here it is (lol)!

Explicações feitas, vou começar os agradecimentos de verdade registrando o apoio incondicional dos meus pais, Weliton de Lima Cordeiro, e Ana Margarete da Costa Cordeiro. Sem dúvida todos os ensinamentos deles desde a minha infância até hoje foram essenciais para o cumprimento de mais essa etapa. Eu definitivamente não teria conseguido chegar tão longe não fosse pelo apoio dos meus pais. Meus irmãos Werley e Liz Marina também foram muito importantes na minha caminhada. Eu dedico mais essa conquista a todos vocês!

Eu não posso esquecer da minha estadia em Belém, durante a graduação. Minha tia Marinês Vieira da Costa foi essencial no período de transição, quando me mudei de Itaituba (PA) para Belém. Meu agradecimento mais do que especial ao Ladislau Cavalheiro da Costa (*in memorian*) e Aldete das Graças Serra da Costa, um casal que me adotou

como um filho, e que portanto se tornou uma segunda família para mim! Meus agradecimentos aos meus grandes irmãos Anderson Jorge, Ana Carolina, e Maria Auxiliadora da Silva Tavares. Meus reconhecimentos também aos professores da UFPA Rodrigo Reis, Carla Lima Reis, e Antônio Abelém, pelas orientações e ensinamentos.

Em terras Porto-alegrenses, não posso deixar de começar os agradecimentos pelo meu orientador e amigo, Luciano Gaspary. Sou muito grato a ele por ter me aceito como seu aluno, no mestrado e no doutorado. Aliás, não deve ter sido fácil me orientar (sou muito cabeça-dura); na verdade, vou ficar admirado se ele continuar no ramo da orientação de alunos depois dessa (risos). Durante esse período, ele me deu bastante liberdade para definir os rumos da pesquisa, e sempre me apoiou de alguma forma nas minhas decisões. Ele me ensinou quais problemas valem a pena ser abordados, e quais devem ser ignorados. Sobretudo, ele me ensinou a ser pragmático e a raciocinar sobre os problemas por uma perspectiva analítica. Sou muito grato pelos seus ensinamentos e pela sua paciência! A propósito, aprendi a atuar no segmento de transporte internacional de cargas por conta dele, também (risos). O Prof. Marinho Barcellos foi outra pessoa que também contribuiu bastante para a minha formação como pesquisador. A propósito, eu agradeço a ele por ter me explicado o verdadeiro sentido de *gettin' high* (risos).

I am also very grateful for the support and advisory I received from Jay Lorch, my former supervisor at Microsoft Research Redmond. He was always readily available to discuss my research and provide me with important guidance to solve the issues I had. Other people from Microsoft Research to whom I am very thankful are Brian Zill, Jitu Padhye, John Douceur, and Jon Howell. I am also thankful to Prof. Alberto Montresor (Univ. Trento, Italy), Olivier Festor (INRIA Nancy, France), and Burkhard Stiller (Univ. Zurich, Switzerland) for hosting me as a visiting researcher in their respective groups.

Os amigos também foram igualmente essenciais para vencer mais essa etapa. Em especial, eu agradeço ao meu grande irmão e parceiro Flávio Roberto Santos (Tchê Barata), por todas as discussões técnicas bem como pelas viagens e momentos de descontração. Foi graças a ele que conheci as belezas e peculiaridades do sertão nordestino, bem como os *canyons* do São Francisco; valeu mesmo, *mizera*! Outro agradecimento muito especial ao grande parceiro Rodrigo Mansilha. Ele desempenhou um papel crucial na reta final do meu doutorado, além de ter sido grande companhia em diversos momentos de descontração. Da mesma forma, não posso esquecer do Gustavo Huff Mauch, aluno de mestrado que começou o caminho dessa pesquisa. Também foi muito legal poder contar com o apoio e a companhia dos amigos do IFPA (Itaituba), da UFPA (Belém), da UFRGS (em especial aos parceiros do Grupo de Redes), e da antiga Pensão do Eraldo e da Housing 164 (Porto Alegre); teve muita festa legal nesse período! Meu reconhecimento também à Sarita Loureiro, por todo o aprendizado e pela amizade que ficarão para a vida inteira.

I also made some great friends and had a lot of fun abroad. I thank my friends from Hewlett Packard, Microsoft, and Microsoft Research for all the caipirinhas and amazing parties we had together. I also had a very pleasant stay in Trento (Italy), Zurich (Switzerland), and Nancy (France) with all the great friends I made in those cities. Finally, I had some crazy and amazing moments and parties in Amsterdan (The Netherlands) with some great friends I made there. Hey guys, you made this journey much more pleasant and fun!

Quero dedicar também essa conquista a uma grande amiga e parceira para todas as horas, Kênia Lílian Figueredo Ribeiro. Ela me acompanhou durante todo o mestrado e doutorado, e sem dúvida desempenhou um papel essencial para que eu concluísse essa etapa. Essa conquista também é dela! Afinal, só ela mesmo pra ter me aguentado durante tanto tempo (mas eu juro que não sou tão chato assim). Eu aprendi muito com ela, e as

lembranças e os ensinamentos desse período ficarão marcados pra sempre!

O mestrado/doutorado na UFRGS foi um período que eu conheci o mundo. Foram tantas coisas incríveis que eu vi, desde as auroras boreais no Pólo Norte até as águas do Mar Egeu na paradisíaca Ilha de Samos (Grécia). Portanto, não posso deixar de agradecer ao pessoal do PPGC/UFRGS por toda a ajuda em viabilizar essas viagens (Jorge Cunha, Maria Cristina e Cláudia Quadros), via auxílios do Instituto de Informática. Em especial à Cláudia Quadros, a quem eu tanto testei a paciência nesse período; Cláudia, a boa notícia é que não tenho mais auxílios pra receber (risos).

Por fim, meus reconhecimentos à CAPES e CNPq, pelos projetos financiados e pelas bolsas de doutorado que recebi durante o período. Meu reconhecimento em especial também à Microsoft Research pelo *fellowship* que me foi agraciado, o qual foi essencial para financiar uma boa parte da minha pesquisa.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

AS	Autonomous System
CA	Certification Authority
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CCDF	Complementary Cumulative Distribution Function
CPU	Central Processing Unit
DSL	Digital Subscriber Line
HTTP	HyperText Transport Protocol
ID	Identity
IP	Internet Protocol
ISP	Internet Service Provider
KJ	Kilojoules
MJ	Megajoules
NTP	Network Time Protocol
KWh	Kilowatts-hour
NAT	Network Address Translation
P2P	Peer-to-Peer
PKI	Public Key Infrastructure
TPM	Trusted Platform Module
UTC	Coordinated Universal Time

LIST OF SYMBOLS

α	
β	Smoothing factor
B	Set of bootstrap entities (trackers)
D	Proportion of energy savings
$\Delta\phi_i(t)$	Recurrence of the i-th source
Δt	(Chapter 3) Time interval between captured snapshots
Δt	(Chapter 5) Duration of the "sliding window"
E	Time to expire (updated by the bootstrap upon renewal of an identity)
E_l	Proportion of easier to solve puzzles assigned to an attacker
E_m	Proportion of easier to solve puzzles assigned to legitimate users
E_t	Set of edges linking nodes belonging to V_t in system G_t
F	Fairness in the assignment of puzzles
$\gamma_i(t)$	Complexity of the puzzle to be solved by user associated to source i (units)
Γ	Maximum possible puzzle complexity (units)
G_t	(Chapter 3) Finite system, comprised of a set of nodes and their relationships, at instant t
$G_k(i)$	(Chapter 6) Number of identities of type k that are valid after i rounds
h_N	N-th harmonic number
I	User identity in the system
$I\langle i\rangle$	Unique, universal identifier of the user's identity
$I\langle t\rangle$	Timestamp of the user's identity last update
$I\langle v\rangle$	Validity timestamp of the user's identity
$I\langle e\rangle$	Expiration timestamp of the user's identity
$I\langle\theta\rangle$	Trust score of the user's identity
$I\langle s\rangle$	Digital signature of the user's identity
$\mathcal{J}$	Set of jobs used as a puzzle

$\mathcal{K}$	Set indicating what types of users have requested at least one identity at a given round ($\mathcal{K} \subseteq \{leg, mal\}$)
λ_{cp}	Rate of the exponential distribution for users' computing power.
λ_i	Rate of the exponential distribution for users' inter-arrival.
λ_r	Rate of the exponential distribution for users' requests per source.
$\Lambda_k(i)$	Number of identity requests of type k (which can be legitimate or malicious) at round i
L	Peer list (BitTorrent)
L_r	Number of identities requested by legitimate users
L_u	Number of legitimate users
L_s	Number of legitimate users per source
μ_f	Mean of the normal distribution for users' first arrival
$M_k(i)$	Number of identities granted at round i
M_c	Number of high-performance workstations the attacker has available
M_r	Number of identities requested by the attacker
M_u	Number of sources in hands of the attacker
ϕ	(Chapter 3) Number of modified snapshots
$\phi_i(t)$	(Chapter 5) Number of identities obtained by the i-th source, at instant t
$\Phi(t)$	(Chapter 5) Network recurrence rate
ψ	(Chapter 3) Number of false positive snapshot corrections
$\psi_k(i)$	(Chapter 6) Number of identities that must be requested at round i
P_k	Computing power of users of type k (which can be legitimate or malicious)
P_l	Proportion of legitimate accounts created
P_m	Proportion of fake accounts created
$\Psi_k(i)$	Number of identity renewals of type k (which can be legitimate or malicious) at round i
$\rho_i(t)$	Relationship between source recurrence and network recurrence rate
R_k	Total number of identities of type k (which can be legitimate or malicious) to be requested
σ_f	Variance of the normal distribution for users' first arrival .
S	Number of sources in the system
$\mathcal{S}$	Snapshot set
τ	Number of accurately corrected snapshots
$\theta_i(t)$	Source trust score
$\theta'_i(t)$	Smoothed source trust score

Θ	Trust score domain
T	(Chapter 4) Current timestamp in the system
T	(Chapters 4 and 6) Duration of the experimental evaluation
U	Proportion of useful puzzles
V	Validity period (updated by the bootstrap upon renewal of an identity)
V_t	Set of nodes in the system G_t at instant t
ω	(Chapter 3) Number of missing positive snapshots
$\omega_i(t)$	(Chapter 6) Wait time to be obeyed by user associated to source i (units)
Ω	(Chapter 6) Maximum possible wait time (units)

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Online systems such as Facebook, Twitter, Digg, 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 accounts (Sybil attack), and use them to either perform malicious activities (that might harm legitimate users) or obtain unfair benefits.

It is extremely challenging (if not impossible) to devise a single identity management solution at the same time able to support a variety of end-users using heterogeneous devices, and suitable for a multitude of environments (e.g., large-scale distributed systems, Internet-of-Things, and Future Internet). As a consequence, the research community has focused on the design of system-specific identity management solutions, in scenarios having a well-defined set of purposes, requirements, and constraints.

In this thesis, we approach the issue of fake accounts in large-scale, distributed systems. More specifically, we target systems based on the peer-to-peer paradigm and that can accommodate lightweight, long-term identity management schemes (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., passport) and/or pay taxes; and *long-term* because users should be able to maintain their identities (e.g., through renewal) for an indefinite period.

Our main objective is to propose a framework for adaptively pricing identity requests as an approach to limit Sybil attacks. The key idea is to estimate a trust score for identity requests, calculated as a as function of the number of identities already granted in a given period, and considering their source of origin. Our approach relies on proof of work, and uses cryptographic puzzles as a resource to restrain attackers. In this thesis, we also concentrate on reshaping traditional puzzles, in order to make them "green" and "useful". The results obtained through simulation and experimentation have shown the feasibility of using green and useful puzzles for identity management. More importantly, they have shown that profiling identity requests based on their source of origin constitutes a promising approach to tackle the dissemination of fake accounts.

Keywords: Identity management, peer-to-peer systems, Sybil attack, fake accounts, collusion attacks, proof of work.

Gerenciamento Adaptativo de Identidades em Sistemas Distribuídos de Larga Escala

RESUMO

Sistemas *online* como Facebook, Twitter, Digg, e comunidades BitTorrent (entre vários outros) oferecem um processo leve para a obtenção de identidades (por exemplo, confirmar um endereço de e-mail válido; os requisitos podem variar dependendo do sistema), de modo que os usuários possam cadastrar-se facilmente nos mesmos. Tal conveniência vem com um preço, no entanto: com um pequeno esforço, um atacante pode obter uma grande quantidade de contas falsas (ataque Sybil), e utilizá-las para executar atividades maliciosas (que possam prejudicar os usuários legítimos) ou obter vantagens indevidas.

É extremamente desafiador (senão impossível) desenvolver uma única solução de gerenciamento de identidades que seja ao mesmo tempo capaz de oferecer suporte a uma variedade de usuários usando dispositivos heterogêneos e adequada para uma diversidade de ambientes (por exemplo, sistemas distribuídos de larga escala, Internet das Coisas, e Internet do Futuro). Como consequência, a comunidade de pesquisa tem focado no projeto de soluções de gerenciamento de identidades customizadas, em cenários com um conjunto bem definido de propósitos, requisitos e limitações.

Nesta tese, abordamos o problema de contas falsas em sistemas distribuídos de larga escala. Mais especificamente, nos concentramos em sistemas baseados no paradigma para-par e que podem acomodar esquemas de gerenciamento de identidades leves e de longo prazo (ex., sistemas de compartilhamento de arquivos e de *live streaming*, sistemas de detecção de intrusão colaborativos, entre outros); *leves* porque os usuários devem obter identidades sem precisar fornecer "provas de identidade" (ex., passaporte) e/ou pagar taxas; e *longo prazo* porque os usuários devem ser capazes de manter suas identidades (ex., através de renovação) por um período indefinido.

Nosso principal objetivo é propor um arcabouço para precificar adaptativamente as solicitações de identidades como uma abordagem para conter ataques Sybil. A idéia chave é estimar um grau de confiança para as solicitações de identidades, calculada como função do número de identidades já concedidas em um dado período, considerando a origem dessas solicitações. Nossa abordagem baseia-se em prova de trabalho e usa desafios criptográficos como um recurso para conter atacantes. Nesta tese, nós também concentramos esforços na reformulação dos desafios tradicionais, de modo a torná-los "verdes" e "úteis". Os resultados obtidos via simulação e experimentação mostraram a viabilidade técnica de usar desafios verdes e úteis para o gerenciamento de identidades. Mais importante, eles mostraram que caracterizar as solicitações de identidades com base na origem das mesmas constitui uma abordagem promissora para lidar com a redução substancial da disseminação de contas falsas.

Palavras-chave: gerenciamento de identidades, sistemas par-a-par, ataque Sybil, contas falsas, ataques em conluio, prova de trabalho.

1 INTRODUCTION

Identity and access management infrastructures play an important role in the digital era, enabling networked systems to determine who has access to them, what rights and permissions one has to managed resources, when and how these resources can be accessed, among others (TRACY, 2008). These infrastructures support a variety of functions, such as identity lifecycle management (e.g., creation, update, and revocation of identities), authentication, and access control, among others (HANSEN; SCHWARTZ; COOPER, 2008). In this thesis, we concentrate on identity lifecycle management.

In an ideal scenario, each person should possess one identity in a given system. Depending on the system nature (e.g., peer-to-peer networks, *ad-hoc* wireless mesh networks, and collaborative intrusion detection systems), this relationship could be interpreted as "one device, one identity". The reality is far from such an ideal scenario, however. 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: with minimum effort, an attacker can obtain a horde of *fake accounts*[2] (Sybil attack (DOUCEUR, 2002)), and use them to either perform malicious activities (that might harm legitimate users) or obtain unfair benefits. The corruptive power of counterfeit identities is widely known, being the object of several studies in the literature (JETTER; DINGER; HARTENSTEIN, 2010). With regard to media coverage, Facebook Inc. recently announced that the proportion of duplicate accounts had risen to 5% in the last quarter of 2012, and that the proportion of fake accounts in the system was 7.6% (EDWARDS, 2013). Google Inc. did not provide concrete numbers on YouTube fake accounts; however, the company stripped around two billion fake views from videos published by Universal Music and Sony using their YouTube accounts (GAYLE, 2012). It is worth mentioning that YouTube implements a reward policy, in which users whose videos are frequently viewed receive a fraction of the revenue with advertisement.

It is extremely challenging (if not impossible) to devise a single identity management solution at the same time able to support a variety of end-users using heterogeneous devices, and suitable for a multitude of environments (e.g., large-scale distributed systems, Internet-of-Things (ATZORI; IERA; MORABITO, 2010), and Future Internet (LAMPROPOULOS; DENAZIS, 2011)). As a consequence, the research community has fo-

[1] In this thesis, 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", "sybil identity", "sybil", and "counterfeit identity" 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 the users in it.

cused on the design of system-specific identity management solutions in scenarios having a well-defined set of purposes, requirements, and constraints.

1.1 Problem definition and scope

In this thesis, 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 leverage lightweight, long-term identity management schemes (DANEZIS; MITTAL, 2009) (e.g., file sharing and live streaming networks, collaborative intrusion detection systems, online social networks, among others); *lightweight* because users should obtain identities without being required to provide "proof of identity" (e.g., passport) 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 the systems mentioned earlier, strategies such as social networks (JETTER; DINGER; HARTENSTEIN, 2010; CAO et al., 2012) and *proof of work* (e.g., computational puzzles) (BORISOV, 2006; ROWAIHY et al., 2007) have been suggested as promising directions to limit the spread of fake accounts. In spite of the potentialities, important questions remain. A number of investigations (MOHAISEN; YUN; KIM, 2010; YANG et al., 2011) have 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 users' privacy. This is a very sensitive issue, specially in a moment when there is a growing concern about online privacy (ANGWIN; SINGER-VINE, 2012).

Puzzle-based schemes, in turn, inherently preserve the users' privacy (since no personal information is required to obtain identities), and therefore represent an interesting approach to limit Sybils. Existing schemes focus on the users' computing power, and use cryptographic puzzles of fixed complexity to limit the spread of Sybils (BORISOV, 2006; ROWAIHY et al., 2007). However, existing proposals are limited as they do not distinguish between requests from legitimate users and those originated by attackers. Ideally, requests from malicious users would be assigned much higher costs than those originated by legitimate users. If so, for some arbitrary hardware capacity at the hands of an attacker, the number of identities he could obtain would be much smaller than in existing approaches. The challenge to this approach, however, is to identify and separate malicious from legitimate requests. The key observation, which we explore in this thesis, is that malicious users tend to recur much more frequently than normal ones.

1.2 Objectives

Our main objective is to propose a framework for adaptively pricing identity requests as an approach to limit Sybil attacks, in the context of large scale distributed systems. We base our framework on the hypothesis that "**one can separate presumably legitimate identity requests from those potentially malicious by observing their source of origin and users' identity request patterns**[3]".

[3]In the context of this thesis, "a *source* requests identities" means in fact "user(s), from a certain *source*, request(s) identities". *Source* may refer to a user's workstation, a local network, an Autonomous System (AS), etc. (identified by an IP address or prefix). In substitution or as a complement, *source* may be associated with a network coordinate provided by a system such as Vivaldi (DABEK et al., 2004) and

Based on the hypothesis above, the key idea is therefore to estimate a trust score of the source from which identity requests depart, calculated as a proportion of the number of identities already granted to (the) user(s) associated to that source, in regard to the average of identities granted to users associated to other sources. The higher the frequency (the) user(s) associated to a source obtain(s) identities, the lower the trust score of that source and, consequently, the higher the price that must be paid per identity requested. Note that exploring this hypothesis to design such a framework basically requires us answering the following research questions:

- How effective would a mechanism based on the hypothesis that *one can filter potentially malicious requests* be in detecting them?
- What is the overhead that such a mechanism would cause to legitimate users?
- Would an attacker, being aware of the inner-working of the mechanism, be able to subvert it? If so, under which conditions?

Building on top of the concept of trust score, in this thesis we take advantage of proof of work strategies to propose the notion of adaptive puzzles. The goal is to adaptively adjust the complexity of puzzles assigned to those users requesting identities, based on the measured value of trust score. Therefore, those requests likely to be involved in a Sybil attack will be assigned puzzles having higher complexity, whereas presumably legitimates will be assigned less complex ones. In this thesis, we focus on the users' computing power and use cryptographic puzzles, as seen in previous approaches (BORISOV, 2006; ROWAIHY et al., 2007). In order to decrease energy-consumption caused by puzzle-solving, and make a useful usage of the processing cycles dedicated for that task, in this thesis we discuss a promising direction for reshaping puzzles, taking advantage of waiting time and lessons learned from massive distributed computing (AHN et al., 2008). In the end, we come up with a design for lightweight, long-term identity management that makes puzzles *green* and *useful*.

In order to answer the research questions posed earlier and assess the effectiveness of our design, we evaluated our framework by means of simulation and also through experiments using PlanetLab. The results obtained show that potential attackers must dedicate a larger amount of resources to control a certain fraction of the identities in the system. The results also show that presumably legitimate users are minimally affected (being assigned easier-to-solve puzzles). In the various stages of this research, our framework was evaluated considering a fixed set of real traces of identity requests, and also synthetically-generated ones, always focusing on reproducing real environments.

1.3 Contributions

As mentioned earlier, we propose a framework for lightweight, long-term identity management in large-scale distributed systems. Our framework, illustrated in Figure 1.1, is composed of a number of building blocks, described next.

The notion of *sources of identity requests*, *frequency of identity requests*, and *network recurrence rate* are important pieces for building the concept of *trust score* – the most fundamental contribution of this thesis (MAUCH et al., 2010; CORDEIRO et al., 2011).

Veracity (SHERR; BLAZE; LOO, 2009). The concept of source is further detailed in Chapter 4.

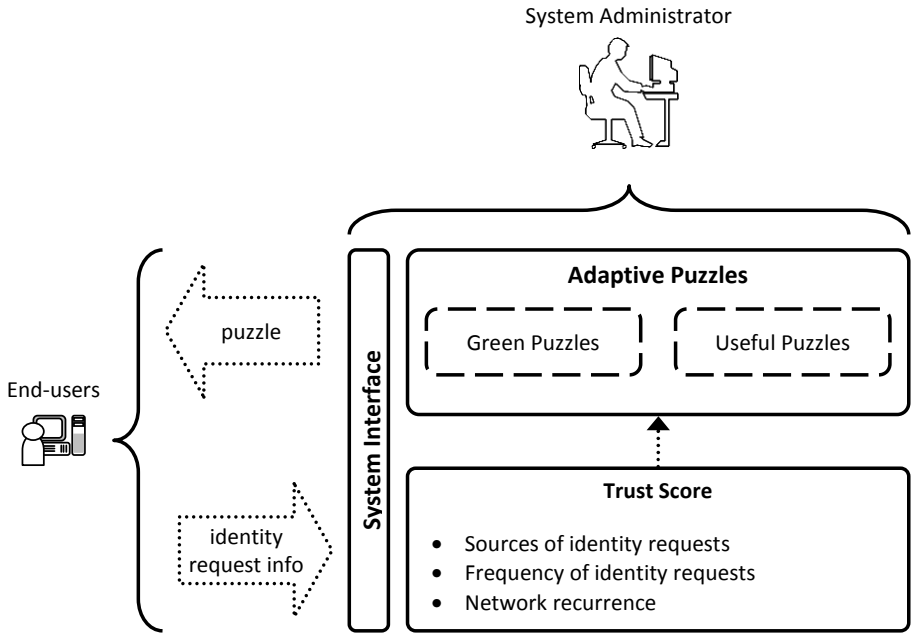


Figure 1.1: Illustration of the proposed framework.

It is important to mention these pieces are abstract notions; i.e., the concept of trust score is agnostic on how these notions are instantiated in a real deployment.

Taking advantage of the concept of trust scores, we make a second contribution in this thesis: the use of *adaptive puzzles* – a solution to make it expensive for potential attackers the process of obtaining a single identity, without penalizing presumably legitimate users (CORDEIRO et al., 2012).

A third contribution of this thesis is the reshaping of traditional puzzles to make them *green* (in terms of energy required to solve the puzzles) and *useful* (by taking advantage of lessons learned from massive distributed computing) (CORDEIRO et al., 2012, 2013). In the end, the framework becomes an enabler of the lightweight scheme for long-term identity management, in which legitimate users are less penalized for obtaining/controlling identities than attackers (CORDEIRO et al., 2013).

It is important to mention that the use of the "green" and "useful" modules are not mandatory in the framework, which could be instantiated solely with traditional, adaptive puzzles. This provides flexibility in scenarios where high efficiency in limiting the dissemination of counterfeit identities is more important than instantiating a green and useful solution for identity management.

In order to assess the validity of our hypothesis, we carried out an extensive analysis considering various traces of identity requests. Some of these traces had inaccuracies, mainly because of the methodology considered to capture them. A fourth contribution of this thesis is therefore a methodology to improve the quality of these traces (CORDEIRO et al., 2014). Our methodology can be used for making these traces suitable not only for the analysis considered in this thesis, but also for any evaluation (e.g., simulation) that takes as input traces of real-world user workloads.

1.4 Organization

The remainder of the thesis is organized as follows. In Chapter 2 we review the literature on mechanisms for identity management in large-scale distributed systems, and we present a taxonomy for a comprehensive understanding of their underlying concepts and assumptions. In Chapter 3 we present a generic methodology for improving the accuracy of traces of users' usage sessions in large-scale distributed systems. In this chapter we also describe the set of experiments carried out to assess the validity of our hypothesis, and our major findings. The analysis described in this chapter is based both on traces processed with our methodology, and also ones available from public repositories. In Chapter 4 we provide an overview of the conceptual framework for identity management that forms the basis of our proposal, and we discuss the possible strategies an attacker can use to subvert our solution.

In Chapter 5 we introduce the concept of trust score, detailing how it is derived from the sources' frequency of identity requests. Its feasibility and robustness is evaluated by means of simulation, considering real traces of identity requests collected from a BitTorrent file sharing community. In Chapter 6 we describe how the values of trust score computed for a source are translated into puzzles of adaptive complexity and assigned to users requesting identities. Finally, in Chapter 7 we summarize the contributions and our key findings, and we present prospective directions for future research.

2 RELATED WORK

The Sybil attack (DOUCEUR, 2002) has been the subject of various research investigations reported in the literature. Several authors have (*i*) studied and quantified the benefits an attacker can obtain by launching it, (*ii*) investigated other attacks that can be deployed using fake accounts, and (*iii*) proposed solutions to prevent and/or detect the existence of fake accounts in various large-scale distributed systems. In this chapter we review some of the most prominent investigations.

Organization. The remainder of this chapter is organized as follows. We first describe, in Section 2.1, the Sybil attack and related ones. Then, in Section 2.2, we introduce a taxonomy for categorizing the proposals that form the state-of-the-art in the field. The taxonomy is followed, in Sections 2.3 and 2.4, by a brief survey of the literature for the most prominent solutions to tackle this attack. Finally, in Section 2.5, we close the chapter with a summary.

2.1 Identity management: vulnerabilities and attacks

The concern on mitigating the existence of fake accounts in online systems is not new (FIEGE; FIAT; SHAMIR, 1987; ELLISON, 1996). In spite of the research efforts carried out to this end, there is no proposal in the literature that enables a local entity to verify, in a fully-distributed system and without direct and/or physical knowledge of the remote entities it communicates to, if different identities in fact belong to distinct entities.

In 2002, Douceur (DOUCEUR, 2002) coined the term "Sybil attack"[1] to designate the creation of fake accounts in online systems[2]. In order to enable an attacker to create various identities and thus launch such attack, it is necessary that the identity management mechanism in place allows the creation of identities without asserting their authenticity (e.g., through verification of personal documents). The corruptive power of counterfeit identities is widely known, being the object of several studies in the literature. The interested reader may refer to the work of Jetter et al. (JETTER; DINGER; HARTENSTEIN, 2010) for a detailed, quantitative analysis of the Sybil attack.

The idea behind launching a Sybil attack is that an adversary can control the majority (or a large fraction) of the identities in the system. The benefit of possessing several coun-

[1]The term Sybil was suggested by Brian Zill from Microsoft Research, after the 1973 book "Sybil" (SCHREIBER, 1974). The book discusses the treatment of Sybil Dorsett (a pseudonym for Shirley Mason) for dissociative identity disorder, then referred to as multiple personality disorder. It was eventually discovered that much of the story portrayed in the book was fabricated, and even the actual diagnosis of Shirley Mason is subject of dispute (CBC BOOKS, 2011).

[2]Prior to that publication, "pseudo spoofing" was the most common term used in the literature to refer to the act of creating counterfeit identities in a distributed system (ORAM, 2001).

terfeit identities varies according to the nature of the system. In Facebook and YouTube, for example, fake accounts can be used to spoof friendships (therefore making someone look “popular”), likes (and thus promote people, fan-pages, videos, or advertisements), page hits, or video views. In peer-to-peer systems, possessing fake accounts increases the chance that an attacker intermediate the communication among any peers. An attacker possessing several identities may also subvert polling algorithms, thus manipulating the reputation of peers and contents shared in the network, among others. In addition, a Sybil attack may serve as basis for launching other attacks such as Eclipse, White-washing, Free-riding, Collusion, and Pollution. Next we will discuss each of them in more detail.

2.1.1 Eclipse

The Eclipse attack (SINGH et al., 2006) consists in dividing a peer-to-peer-based distributed system in an arbitrary number of segments. The goal is to enable the attacker to intermediate the communication between peers from different segments. Therefore, the attacker will be able to omit messages sent from/to peers in eclipsed segments. He will be also able to provide, intentionally, faulty answers to requests in both application and overlay levels. In the former case, by returning false information as a response to searches, in an attempt to censor the access to certain resources. In the latter, by providing bogus routing information, in order to partition the network. In addition, attackers can launch an Eclipse attack in other to perform other malicious activities such as traffic analysis, in those systems that aim to provide anonymity (GASPARY; BARCELLOS, 2006).

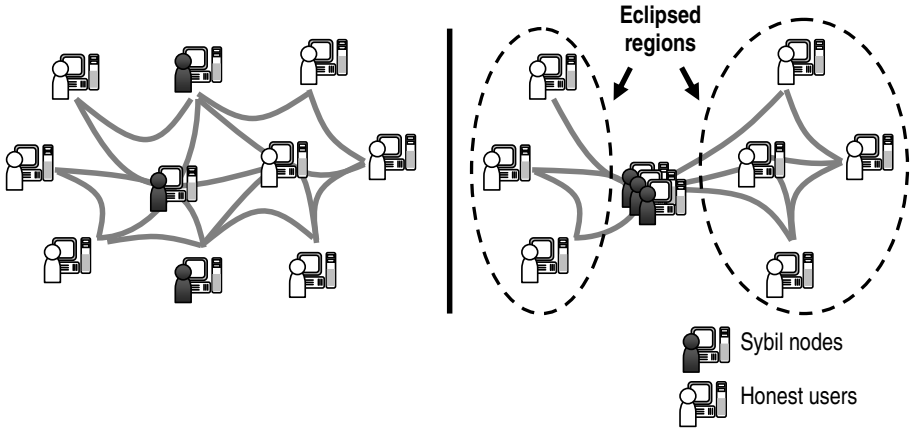


Figure 2.1: Illustration of a peer-to-peer network under an Eclipse attack.

Figure 2.1 illustrates a peer-to-peer network under Eclipse attack. In the left part of the figure the network is shown as it is perceived by the legitimate users, which are not aware that several of the identities they communicate with are controlled by a single entity. In the right part it is illustrated the actual view of the network, partitioned into two segments. In this setting, every communication between peers from each of the partitions will be intermediated (and possibly modified) by the entity controlling these fake accounts. The natural path for an attacker to obtain identities in order to create these fake peers is launching a Sybil attack beforehand.

Eclipse attacks are particularly harmful in structured peer-to-peer networks such as Pastry (DRUSCHEL; ROWSTRON, 2001), CAN (RATNASAMY et al., 2001), and Chord (STOICA et al., 2001), since that in this type of networks the identifier of each peer is assigned according to their position in the overlay. These aspects have been the subject of

investigation (TIMPANARO et al., 2011), in which the robustness of BitTorrent Mainline DHT against Eclipse attacks has been evaluated. From a set of experiments the authors have found that a relatively small number of counterfeit identities (20) were sufficient to launch a successful Eclipse attack in the network.

2.1.2 Free-riding

Free-riding is a violation to the principle of collaboration among the participants of peer-to-peer networks, especially to file sharing ones. It consists in one taking advantage of the network resources without providing one's own resources in exchange. The users that adopt such selfish behavior are called free-riders. In contrast to Eclipse, whose motivation is often clear, there are several reasons why a user would free-ride the system (ENGLE; KHAN, 2006):

- the intent of decreasing upload bandwidth consumption; this is the case when this resource is limited, or when charges per amount of data transmitted/received apply;
- the concern of sharing illegal contents (in the case of file sharing networks); a number of countries (e.g., United States and Switzerland) have specific legislation on this matter. In these countries, sharing copyrighted contents that have restrictions on copying and distribution for example constitutes violation to copyright laws;
- some people tend to abuse of certain resources if no form of payment is required for their usage; this behavior, whose explanation is related to "the tragedy of the commons" (HARDIN, 1968), is extremely harmful to those peer-to-peer networks which rely on collaboration to function as expected.

In those large-scale distributed systems that offer incentives for peers to contribute with their own resources, the creation of counterfeit identities can play an important role so that attackers can obtain more benefits without contributing to the network. For example, Pontes et al. (PONTES; BRASILEIRO; ANDRADE, 2007) described that an attacker, possessing a small number of identities (100 identities in a system having over than 50,000 peers, according to an experiment), can obtain higher download rates in BitTorrent file sharing communities, thus harming other users. In a more recent publication, Karakaya et al. (KARAKAYA; KORPEOGLU; ULUSOY, 2009) performed an extensive description of the potential of counterfeit identities to augment the harmful power of free-riders in a variety of large-scale distributed systems.

2.1.3 White-washing

The white-washing attack, first described in a survey about trust and reputation systems (JØSANG; ISMAIL; BOYD, 2007), consists in taking advantage of the neutral reputation that newcomers have to launch other attacks in the network, such as Free-riding.

In order to understand the motivation for this attack, it is important to have in mind the basic operation of peer-to-peer reputation systems (such as EigenTrust (KAMVAR; SCHLOSSER; GARCIA-MOLINA, 2003) and DHTrust (XUE et al., 2012)). According to Jøsang et al. (JØSANG; ISMAIL; BOYD, 2007), the basic idea of reputation systems is to enable peers to evaluate each other, for example, after completing a transaction. The aggregation of these evaluations is then used to estimate a trust or reputation score of each participant. One may then consider each others' trust score in his/her decision of

taking part in future transactions with them. Peers having lower reputation scores can be penalized by not having their transactions completed in the system.

In those systems that adopt reputation mechanisms, a user that free-rides potentially gets lower reputation. In a given moment, his/her reputation will not allow him to continue completing transaction with other peers. This situation then motivates him to leave the system, discard his/her own identity, and create a new one – which will have a higher reputation compared to the previous one. Since the reputation is associated to the identity the user possesses, a free-rider that repeatedly joins the system with a new identity will get rid of his/her lower reputation (FELDMAN et al., 2006), and thus continue being able to establish transactions with other peers. Note that white-washing brings advantages for an attacker only if creating identities is simple and cheap, and the effort required to regenerate his/her reputation in the system is relatively small.

2.1.4 Collusion

A collusion attack (MARTI; GARCIA-MOLINA, 2006) consists in multiple entities acting together to achieve specific goal, often malicious. For example, they may provide similar feedback about a given resource or entity in order to subvert reputation algorithms. According to Marti and Garcia-Molina (MARTI; GARCIA-MOLINA, 2006), investigations that target this issue consider that a group of colluding entities acts as a single entity, with each entity aware of each others' behavior and intentions.

A single attacker can take advantage of counterfeit identities to launch a collusion attack against the system. In this case, the attacker defines the role of each fake entity considering his/her major goal. In addition to promoting people and/or advertisements, colluding attacks may serve as basis for disseminating polluted content to users in a file-sharing network. The pollution attack and its relationship with collusion attacks are described in the following section.

2.1.5 Pollution

In the content pollution attack, a malicious user publishes a large number of *decoys* (same or similar meta-data), so that queries of a given content return predominantly fake/corrupted copies (e.g., a blank media file or executable infected with virus) (LIANG et al., 2005). Another attack is *meta-data pollution*, which consists of publishing a file with misleading meta-data, inducing users to download files that do not correspond to the desired content. A third kind of attack, known as *index poisoning* (LIANG; NAOUMOV; ROSS, 2006), consists in creating many bogus records that associate titles with identifiers of inexistent copies and/or false IP addresses/port numbers. Various studies have looked at the extension of these attacks in Gnutella, KaZaa, BitTorrent, and in live streaming systems (KUMAR et al., 2006; LEE et al., 2006; SANTOS et al., 2011; LIN et al., 2010).

In those systems that aim to tackle the dissemination of polluted content using reputation schemes (WALSH; SIRER, 2006; COSTA; ALMEIDA, 2007; VIEIRA et al., 2013; SANTOS et al., 2013), counterfeit identities have been widely used to collude by increasing the reputation of polluted contents in detriment of non-polluted versions. This is done, for example, by providing false testimonies about the legitimacy of a given version. This can also be done in combination with other attacks, such as Eclipse.

This attack (and also the previously discussed ones) relies on poor identity management in order to take place. In fact, Wallach (WALLACH, 2003) argued that a secure design for peer to peer networks requires the solution of three problems: securely assigning identities to users, securely maintaining the routing tables, and securely forwarding

messages; without an adequate solution for the first problem, the later two cannot be properly approached. In the following section we discuss some of the most prominent solutions proposed to date to deal with this problem.

2.2 Taxonomy of solutions

In the paper that describes the Sybil attack (DOUCEUR, 2002) 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 on this subject have focused on limiting the dissemination of fake accounts and also on mitigating their potential harm. We propose a taxonomy to organize these proposals, which is based on the broader categorization introduced by Danezis et al. (DANEZIS; MITTAL, 2009), namely *weak* and *strong identity-based* management schemes.

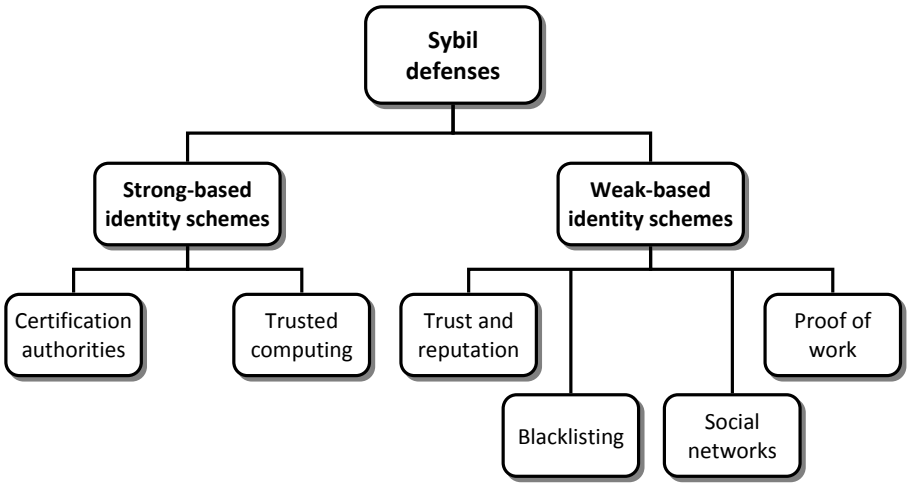


Figure 2.2: Taxonomy of Sybil-defense solutions.

In our taxonomy, depicted in Figure 2.2, existing solutions are grouped according to the mechanism employed for identity management or user authentication. Each of the groups are described in detail in the following sections.

2.3 Strong-based identity schemes

In this category, 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. Next we describe the most prominent sub-categories in this class: certification authorities and trusted computing.

2.3.1 Certification authorities

The certificate-based authentication schemes use external entities to provide identification services, typically Certification Authorities (CA) such as VeriSign^{TM}, Thawte^{TM}, and Comodo CA^{TM}. In this scheme, each user must obtain an identity from a given trusted entity prior to joining the system; that entity will be held responsible for ensuring the authenticity of the users' identities in the system. Therefore, it becomes implicit that all users should agree with that trusted entity. The requirements for creating new identities

may be diverse, ranging from the provisioning of personal information such as passport number, to the payment of taxes.

This scheme brings several interesting advantages to large-scale distributed systems. The most important one is that users cannot spoof and/or steal identities (assuming that the certification authority is not compromised). Without being able to create new identities, Sybil and white-washing attacks are virtually eliminated, given the difficulties imposed by the identity creation/certification process. Another advantage of this scheme is that it ensures users are in fact communicating with those they actually intend to.

This scheme might also present some drawbacks, depending on the system nature. First, the certification authority must be accepted for all users in the system. This aspect becomes more relevant for those certification authorities deployed for serving some specific system, or when users are unwilling to trust their personal information or pay taxes to obtain identities from unknown or obscure certification authorities. Well-known, online payment companies (e.g., PayPal) could provide authentication services, thereby solving this problem. In this case, an identity could be obtained by proving the possession of an account in such companies, which in turn authenticate users by validating their credit cards. Apart from the discussion on whether this is acceptable, we argue here that users should have options for authenticating himself/herself that do not require or rely on their personal, sensitive information.

Second, in those systems in which users are free to choose their certification authorities, a communication problem may arise when two users that wish to complete a transaction do not trust on each other's chosen certification authorities. Third, it might be unfeasible to deploy a dedicated certification authority for serving a specific system.

The proposals that fit in this category (CASTRO et al., 2002; ABERER; DATTA; HAUSWIRTH, 2005; MORSELLI et al., 2006) attempt to minimize some of the side-effects of using certification authorities. For example, Aberer et al. (ABERER; DATTA; HAUSWIRTH, 2005) and Morselli et al. (MORSELLI et al., 2006) have focused on decentralized public-key infrastructures (PKI) and web of trust to provide an authentication service to users. However, these solutions are vulnerable to selfish behavior, i.e., they rely on the contribution of a certain fraction of peers to operate as expected.

2.3.2 Trusted computing

Trusted computing is a relatively new platform, in which commodity hardware (e.g., computers) comes with Trusted Platform Module (TPM) – a secure crypto-processor that can store cryptographic keys that protect information (TRUSTED COMPUTING GROUP, 2013), and that is certified by the hardware vendor. This platform serves for attesting that the overall hardware and software in a given device have certain properties. It can be used, for example, to attest that users are running a certain software version, or for verification of remote computation in a grid computing scenario (to prevent that an attacker forges computation results).

The use of trusted computing could theoretically enable the assertion "one device – one identity" (discussed in the introduction), thus solving many security and privacy issues related to identity management in large-scale distributed systems. TrInc (LEVIN et al., 2009) is a first step towards this end – a small, trusted component designed to combat equivocation in large-scale distributed systems, by providing means for commodity hardware to provide unique, once-in-a-lifetime, verifiable attestations (for example, about the user's identity). The solutions based on trusted computing, however, might not be widely available, since participating devices must be equipped with a TPM.

2.4 Weak-based identity schemes

This category comprises mechanisms in which identities are not created with strong authentication guarantees (i.e., the identities do not serve for the purpose of authenticating the user/device using it). In this case, although it is not possible to avoid the existence of Sybil identities, it is possible to limit their amount to an "acceptable level". Such mechanisms may be useful, for instance, for applications that can tolerate a certain fraction of counterfeit identities.

The solutions in this category may be further classified according to the strategy employed to enforce authenticity: social networks (JETTER; DINGER; HARTENSTEIN, 2010; CAO et al., 2012), trust and reputation (JØSANG; ISMAIL; BOYD, 2007), blacklisting (LIANG; NAOUMOV; ROSS, 2005), and proof of work (BORISOV, 2006; ROWAIHY et al., 2007). Next we discuss each of them.

2.4.1 Blacklisting

There are a number of investigations that have proposed IP blacklisting as an strategy to prevent pollution and fake accounts in large-scale, distributed systems (LIANG; NAOUMOV; ROSS, 2005). The idea is simple: once a region within the Internet is found to be associated with nasty activities, these regions (typically a subnet address/submask) are recorded in a list and prevented from gaining access to the system. This is similar to the strategy adopted by organizations such as The SpamHaus Project (SpamHaus, 2013) to reduce the amount of spam messages sent across the Internet.

In spite of the potentialities of using blacklisting as a mean to limit Sybil attacks, there are a number of technical limitations. The most important is that the maintenance of these blacklists must be supervised by a human operator (in order to prevent legitimate subnets from being regarded as suspicious). They also rely on algorithms that are prone to misidentification of corrupt subnets. More importantly, spoofing IP addresses is a well known technique, which severely hampers the effectiveness of this approach.

In the realm of pollution in large-scale, distributed systems, Liang et al. (LIANG; NAOUMOV; ROSS, 2005) proposed the creation of a blacklist based on the evaluation of title meta-data collected in a P2P network. The list is comprised of IP ranges labeled as *polluters*, and built according to the density of corrupted files made available. The success of the approach depends on the frequent download of meta-data, potentially leading to substantial overhead. In a continuation of the previous investigation (LIANG; NAOUMOV; ROSS, 2006), a blacklist is created by a global system to keep track of sub-network reputation. In both investigations, there are limitations in characterizing a sub-network as polluter based on the high density of polluted versions. Furthermore, many participants of P2P networks are connected to the Internet via Network Address Translation in Internet Service Providers.

2.4.2 Social networks

The use of social networks as a resource to detect and/or limit the occurrence of counterfeit identities has gained significant attention from the research community, with several prominent solutions reported in the literature (YU et al., 2006, 2008; DANEZIS; MITTAL, 2009; TRAN et al., 2011; CAO et al., 2012). These mechanisms explore the concept of social networks to limit the spread of counterfeit identities in the system, also estimating an upper bound for the number of counterfeit identities that are "accepted".

Yu et al. (YU et al., 2006) proposed a mechanism to limit the negative influence of

counterfeit identities by taking advantage of the real world relationships between users. Based on the assumption that social networks are "fast-mixing" (i.e., that users form a dense network of relationships), clusters of counterfeit identities can be detected and eliminated through a mechanism of random walks in the social graph. In a subsequent research, Yu et al. (YU et al., 2008) improved the performance of the proposed mechanism, by decreasing the number of counterfeit identities that are accepted as being legitimate.

One of the most recent research initiatives in the field is SybilRank (CAO et al., 2012). Also relying on the assumption that social networks are fast mixing, Cao et al. proposed an approach to rank nodes in a social graph. The ranking is done according to the degree-normalized probability of a short random walk of starting and landing in a non-Sybil node. The lower-ranking nodes in this scheme are regarded as potentially fake with high probability, and can thus be banned from the network for example.

In spite of the potentialities, social networks have some important drawbacks. The use of social networks as a mean to tackle Sybils might violate user anonymity; as previously mentioned in this thesis, this is an extremely sensitive issue, specially in a moment when there is a growing concern and discussion about privacy issues in social networks (ANGWIN; SINGER-VINE, 2012). More importantly, Mohaisen et al. (MOHAISEN; YUN; KIM, 2010) and Yang et al. (YANG et al., 2011) have recently shown that most social networks are not fast-mixing and that Sybils do not aggregate themselves in a dense network of relationships (this is also an important finding from Cao et al. (CAO et al., 2012)). These findings invalidate one of the key assumptions upon which social network-based solutions relied on.

2.4.3 Trust and reputation

Trust and reputation systems also have been studied and used to detect suspicious identities as being counterfeit (JØSANG; ISMAIL; BOYD, 2007; JØSANG, 2012). The actual goal is not limiting the dissemination of counterfeit identities in the network, but detecting them once they behave suspiciously. According to Jøsang (JØSANG, 2012), there are two major challenges in this field: (*i*) design systems that are able to derive accurate values of trust score to the participating entities, and (*ii*) ensure that the computed values of trust score will reflect future behavior of these entities.

E-commerce companies such as eBay and Amazon.com also implement reputation indexes based on users' transaction history. In the distributed systems realm, proposed solutions have been vulnerable to white-washing attacks, except when strategies to make identity assignment a non-trivial task are adopted (e.g., closed BitTorrent communities such as BitSoup (BITSOUP.ORG, 2010)).

2.4.4 Proof of work

In the paper that describes the Sybil attack (DOUCEUR, 2002), Douceur demonstrated that a robust and scalable solution for peer authentication in P2P networks requires a certain degree of centralization (for example, by employing certification authorities). Since then, and considering the severe constraints imposed by the use of certification authorities, an intermediate category of proposals that has attracted attention in recent years is to condition identity granting/renewal to the previous resolution of computational puzzles. Puzzles are typically assigned to users (and their solution, verified) by a *bootstrap service*, and aim to decrease attackers' capabilities of creating counterfeit identities.

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

2006), for example, has shown the technical feasibility of using puzzles generated periodically and distributedly, by proposing a mechanism in which peers that participate in the puzzle generation process are able to validate its solution. Rowaihy et al. (ROWAIHY et al., 2007), in turn, have proposed a mechanism based on multiple puzzle generation entities. It requires that, prior to obtaining identities, users contact one of these entities and solve a series of puzzles.

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

2.5 Summary

Online systems have been a frequent target of attacks that aim to create a large number of counterfeit identities. These identities can be used for a variety of purposes, such as promoting or raising the reputation of advertisements and contents, obtaining unfair advantages in the system, and/or causing nasty activities that might harm other users. The lack of a proper identity management system may ultimately cause the failure of the system and compel the users to abandon it.

As mentioned earlier in this chapter, there is no solution that can virtually eliminate counterfeit identities in the absence of a central certification authority (DOUCEUR, 2002). The solutions proposed to limit Sybils in online systems have thus focused on the design of mechanisms that can bring the proportion of fake accounts in the system to an acceptable level, each having their own set of requirements and constraints. For example, the mechanisms based on online social networks require that the users in the system personally know each other. The proposed solutions also vary in complexity, degree of authentication, overhead (to legitimate users), robustness, and security (to identity owners).

Table 2.1: Summary of proposals and their attributes.

Solution	Type of identity scheme	Users' collaboration	Enforcement of legitimacy	Knowledge of other users
Certification authorities	Strong-based	Solution-specific	Before identity request	Solution-specific
Trusted computing	Strong-based	Not required	Before identity request	Not required
Blacklisting	Weak-based	Solution-specific	Before identity request	Not required
Social networks	Weak-based	Not required	After identity request	Required
Trust and reputation	Weak-based	Required	After identity request	Not required
Proof of work	Weak-based	Solution-specific	Before identity request	Not required

Table 2.1 provides a brief summary of the proposed solutions analyzed in this section. The solutions are classified in this table considering (i) type of identity scheme

(DANEZIS; MITTAL, 2009), (ii) need for users' collaboration to be effective, (iii) moment in which identity authenticity is enforced (before or after the identity is created and assigned to the user), and (iv) requirement for users know each other. With regards to users' collaboration, for example, it forms the basis of some solutions (e.g., trust and reputation), whereas in other cases it may depend on the design of the solution. For proof of work, some implementations use collaborative, distributed creation and verification of puzzle solution, whereas others use central creation and verification of puzzle solution.

One of the concerns that has driven the research reported in this thesis is that users should not need to reveal any information to the system in order to obtain identities, and that their privacy should be respected to its full extent (for example, no behavioral or access pattern analysis should be done to identity fake accounts – as it occurs in reputation systems). Therefore, we focus on the class of proof of work strategies as a prospective solution to limit the dissemination of Sybils. It is important to highlight that our framework (similarly to others based solely on computational puzzles (BORISOV, 2006; ROWAIHY et al., 2007)) falls in the category of *weak identity-based* mechanisms. As a consequence, created identities do not fit for the purpose of strongly authenticating users. In the following chapters, we describe our approach to tackle the issue of fake accounts.

3 PROFILE OF IDENTITY REQUEST PATTERNS

In the previous chapter we presented a survey of the most prominent solutions to address the issue of fake accounts in large-scale distributed systems. In this thesis, we approach this issue considering a perspective that users' privacy should not be violated; to this end, we explored an aspect largely neglected in previous investigations: the patterns of users' identity requests. In this chapter we discuss the results of an analysis of traces of identity requests, and present insights that can help us in the design of a solution that can separate presumably legitimate requests from those potentially malicious. It is important to mention that we used BitTorrent traces from a public repository maintained by the Technische Universiteit Delf P2P Trace Archive (ZHANG et al., 2012), and also traces we collected from some BitTorrent file sharing communities. In this chapter we also discuss the methodology we used to capture and process the traces we collected[1].

Organization. The remainder of this chapter is organized as follows. In Section 3.1 we present the approach we used for collecting traces. In Section 3.2 we discuss the methodology for processing the collected traces and making them suitable for our evaluation. In Section 3.3 we discuss our insights with the analysis of the studied traces. Finally, in Section 3.4, we close the chapter with a summary.

3.1 Collecting traces

As previously mentioned, we used BitTorrent traces from a public repository – maintained by the Technische Universiteit Delf P2P Trace Archive (ZHANG et al., 2012) – in our evaluation. In spite of the variety of traces available, most of them were collected in 2009 or earlier. For this reason, we dedicated some effort for obtaining more recent ones. For the sake of consistency, we focused on a BitTorrent file-sharing community.

There has been substantial research on approaches for obtaining traces based on the capture of users' usage information (e.g. arrival and departure, and duration of online sessions) (POUWELSE et al., 2005; YOSHIDA; NAKAO, 2011a; HOÃŸFELD et al., 2011). In summary, these approaches can be mapped into three basic strategies: (i) obtaining users' behavior information directly from the system database (GUO et al., 2007), (ii) running add-ons on each user (Alexa Internet, Inc., 2012), and (iii) taking periodic "snapshots" of users in the system, and building a trace from these snapshots (HOÃŸFELD et al., 2011). The third strategy has attracted significant attention from the community, mainly because of its simplicity, popularity (YOSHIDA; NAKAO, 2011b; CUEVAS et al.,

[1]Part of the concent of this chapter is based on the following submission: Cordeiro, W., Mansilha, R., Santos, F., Gaspary, L., Barcellos, M., Montresor, A.: *Were You There? Bridging the Gap to Unveil Users' Online Sessions in Networked, Distributed Systems*. In: 33rd Conference on Computer Communications (INFOCOM 2014), 2014, Toronto, Canada (submitted), 2014.

2010; BLOND et al., 2010; ZHANG et al., 2010) and, more importantly, the autonomy it provides for one to collect data (i.e. without requiring privileged access to any entities in the system). For these reasons, in this thesis we also took advantage of this strategy to perform our trace collection.

3.1.1 Overview

Here we provide a brief overview on the *modus operandis* of the strategy we use to collect traces; to this end, we use Figure 3.1 as basis. The target of our trace collection is some generic distributed system, for example a BitTorrent file sharing community. In such a system, a number of users (e.g. peers interested in sharing contents) join and leave at their will. In this context, labels $v_{1..3}$ represent the users that were seen at least once in the system; labels $s_{1..14}$ represent the snapshots taken, at a regular time interval Δt.

In Figure 3.1, 1 indicates that a given user v_i was seen in the system when snapshot s_t was taken (i.e., a "positive" snapshot), whereas 0 ("null") represents the opposite. Building the trace of users' online sessions is then a trivial task; in the example from Figure 3.1, the gray regions form an online session.

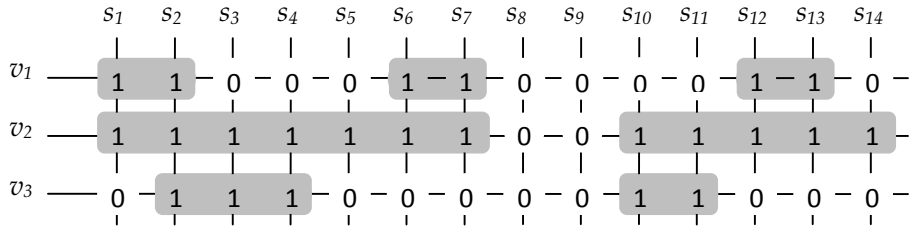


Figure 3.1: Users seen on each snapshot taken from the monitored system.

Zhang et al. (ZHANG et al., 2011) and Mansilha et al. (MANSILHA et al., 2011) use a set of instrumented clients (crawlers) to take snapshots from BitTorrent swarms. In our research, we used a partial instance of the TorrentU framework, which was proposed by Mansilha et al. (MANSILHA et al., 2011), to take snapshots. Next we describe the methodology that TorrentU uses to collect traces.

3.1.2 Using TorrentU to collect traces

We used an instance of the TorrentU framework tailored for the purpose of capturing users' arrival and departure in large-scale distributed systems. This instance uses a minimal implementation of a BitTorrent client software to collect traces, which is installed in a set of distributed nodes (crawlers). To materialize our instance, we used a set of PlanetLab nodes as crawlers. In this section we first describe, in a higher level of abstraction, the snapshot captuing process using TorrentU. Then, we dive into the details of the trace capturing process, using a high level algorithm as basis.

Trace collection deployment overview

Figure 3.2 provides an overview of the collection process. The first step in the setup of a new monitoring process is the upload of the monitoring scripts and software, from the master server to the crawlers used in the monitoring process (arrow 1 in Figure 3.2). The master server also sends to each of the nodes a set of parameters related to the monitoring, which include: (i) the monitoring start time, (ii) a list of torrents to be monitored, (iii) the address of the master server, and (iv) the periodicity of snapshots.

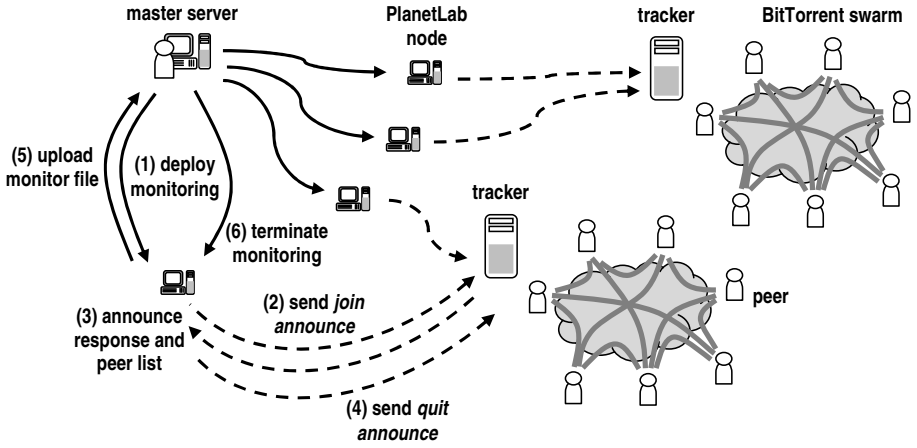


Figure 3.2: Simple schema of the TorrentU deployment.

Once the monitoring starts, each node begin collecting, periodically, a set of online users from the swarms indicated in the list of torrents given as parameter. To carry out this process, the node basically acts as a BitTorrent client. The methodology used by the modified client to obtain peer lists is sending two *announce* messages to the tracker. The first message, *join announce*, is used to join the swarm (arrow 2); in response, the tracker sends a list of peers already in the swarm (arrow 3). The second message, *quit announce*, is sent shortly after to leave the swarm (arrow 4). The reason for sending a *quit announce* message is twofold. First, it minimizes any interference to the swarm. By doing so, for example, the crawler's IP address will not be sent to other users, since the crawler itself is not going to download or upload content. Second, it will prevent blacklisting, since the crawler will not be regarded as subscribed to various trackers. Note that this process (illustrated through arrows 2 - 4) is repreated periodically, during the period of the snapshot capture process.

In our TorrentU instance, the node periodically uploads the set of peer lists obtained to the master server (arrow 5). This is done to protect from transient or permanent PlanetLab node failures, which may difficult or even make it impossible to retrieved the peer lists collected so far. Once the monitoring is done, the master server simply sends a signal to each crawler for gracefully terminating the snapshot collection process (arrow 6).

High level algorithm of the monitoring process

In each crawler, the BitTorrent client is invoked from within a monitoring script. The monitoring script, in turn, is remotely invoked from the master server, which coordinates the snapshot capturing process. A partial view of the monitoring script algorithm is presented in Algorithm 1.

The monitoring starts once the start time expires in each crawler (lines 1-4 in Algorithm 1). To ensure accuracy of collected data, crawlers and master server must be synchronized so that clock discrepancy among them is at most a couple of seconds; this setting can be easily achieved using NTP. As previously mentioned, the snapshot collection process is carried out periodically (i.e., in rounds). Therefore, in this algorithm an alarm is set (line 3), which will trigger every Δt units of time to create a thread for executing the RUNMONITORING procedure (line 8). For distinguishing each partial snapshot that will be created by some crawler having identifier Global.*crawlerId*, we maintain a variable called *snapshotId*. Once the monitoring starts, this variable is set to zero (line 2).

Algorithm 1 Script for taking periodic snapshots

```
1:  on SystemClock.currentTime == InputParam.startTime do
2:    snapshotId = 0                          ▷ counter for distinguishing partial snapshots
3:    alarmId ← setAlarm (InputParam.Δt, RunMonitoring)
4:  end on

5:  on event == SystemEvent.Interrupt do
6:    deleteAlarm (alarmId)
7:  end on

8:  procedure RUNMONITORING( )
9:    snapshotId ++                           ▷ identifies each snapshot file per round
10:   logFile ← createFile ('snapshotFile_' + Global.crawlerId + '_' + snapshotId)
11:   append (logFile, SystemClock.CurrentTime)
12:   for all torrent ∈ InputParam.torrentList do
13:     for all tracker ∈ torrent.trackerList do    ▷ on first run, tracker.minInterval is 0
14:       tracker.minInterval ← tracker.minInterval −InputParam.Δt
15:       if tracker.minInterval ≤ 0 then
16:         peerList, nextRequest ← runBitTorrentClientMod (torrent, tracker)
17:         tracker.minInterval ← nextRequest          ▷ update tracker.minInterval
18:         append (torrent.infoHash + ' ' + tracker.address + ' ' + peerList, logFile)
19:       end if
20:     end for
21:   end for
22:   close (logFile)
23:   uploadFile (InputParam.masterServerAddr, logFile)
24: end procedure
```

The RUNMONITORING procedure implements most of the logic of the monitoring process. The first step is creating a file to record the snapshot (line 10). The name of the file will be formed by the concatenation of the crawler identifier and the current snapshot index *snapshotId*. To distinguish the round to which each snapshot file refers to, *snapshotId* is incremented on each round (line 9).

The current timestamp (timestamp of the snapshot) is recorded in this file (line 11). Then, for each torrent in the torrent list given as input for the monitoring process (line 12), the following steps are taken. First, the torrent is open, to retrieve its tracker list. For each tracker in that list (line 13), the algorithm checks if that tracker should be queried (lines 14-19). If so, it launches the modified BitTorrent client to fetch a peer list from it (lines 16). The algorithm updates tracker.*minInterval* with the minimum time expected for a future request *nextRequest* informed by the tracker (line 17). Finally, the received peer list (plus the torrent *infohash* and tracker address) are appended to the file (line 18).

The decision on whether a given tracker should be queried on some round is necessary because trackers often impose a minimum interval between announces (*nextRequest*). If some node makes requests in an interval shorter than the one informed, it may be blacklisted, and thus prevented from obtaining peer lists from that tracker. To illustrate, suppose that a node sent an announce message to some tracker, and in reply it obtained a peer list and a value of 28 minutes. That value means that the node must wait at least

28 minutes before contacting the tracker again to send another announce message. Now suppose that $\Delta t = 5$ minutes. Given that tracker.*minInterval* has a value of 0 on the first run, the tracker is queried in the first monitoring round because tracker.*minInterval* $-\Delta t \therefore 0 - 5 = -5 \leq 0$ (lines 14-15). After the query is completed, tracker.*minInterval* is now equal to 28 (line 17). The tracker will be contacted again only in the sixth round since the last query, when tracker.*minInterval* will be equal to -2. Note in this example that more time is elapsed (30 minutes) than the tracker actually indicated (28 minutes). However, it is more important to ensure that snapshots are in fact taken in a regular time basis; in this case, speed is sacrificed for the sake of accuracy.

After the peer lists from trackers of all torrents are obtained, the next step in the algorithm is to close the file (line 22) and upload it to the master server (line 23, arrow 4 in Figure 3.2)). For the sake of optimization, this file could be uploaded after a certain number of rounds have been passed, for example.

The monitoring is globally terminated when the master server sends a signal to each of the PlanetLab crawlers (arrow 5). The arrival of that signal basically triggers the deletion of the alarm set up for periodically running the RUNMONITORING procedure (lines 5 - 7).

Note that the master server will receive partial snapshots from all crawlers involved in the snapshot capturing process. Building a complete snapshot from the system at instant t is then a trivial task. It is done by simply taking those partial snapshots collected from the various trackers of a given swarm in a same round and adding them together.

3.1.3 Considerations on our deployment for trace collection

There were five major issues that had to be addressed in order to deploy the monitoring architecture described in Figure 3.2. These issues are discussed next.

High values for nextRequest *received from trackers*

The period between snapshots Δt plays an important role in the accuracy of the obtained traces. On one hand, longer periods between snapshots increase the chances that a join/leave event of a user is missed. On the other hand, shorter periods increase the overhead caused to the monitored system.

The strategy that trackers use to define a value for *nextRequest* significantly affects the regularity required for capturing snapshots. In a set of monitoring experiments we carried out, for example, *nextRequest* was often set to 30 minutes. Using $\Delta t = 30$ might result in a poor trace, where a large fraction of join/leave events are missed. To address this problem, we divided the set of PlanetLab nodes into N groups, so that $\Delta t = {}^{E[\text{nextRequest}]}/_{N}$, where $E[\text{nextRequest}]$ is the expected value for *nextRequest*. In this case, group N_0 queries the trackers on instant $t_0 = 0$, group N_i on instant $t_i = i \cdot \Delta t$ and so on. When it comes the turn of group N_0, the expected amount of time $E[\text{nextRequest}]$ will have been already elapsed.

Number of trackers queried per node

Building a snapshot of online users takes some non-negligible amount of time. Among the aspects that may influence that amount of time are: (i) round-trip time, (ii) available bandwidth (to transmit peer lists), (iii) CPU power (to build peer lists and handle requests/connections), (iv) number of trackers to be queried, among others. Therefore, it is important to keep the number of trackers that each node queries down to a minimum, so that building a single snapshot takes as little time as possible. Otherwise, snapshots might

mix together users that were not necessarily online in a same instant.

To illustrate, suppose that querying each tracker takes 5 seconds on average (including the time required to initiate and close a connection following the BitTorrent protocol). Ideally, each node would query a single tracker. However, considering that each node will query a number of 6 trackers (from a number of torrents), the total time to build a snapshot will be 30 seconds.

Synchronization of the system clock of crawlers

The snapshot building process has a high degree of distribution, and partial snapshots must be taken approximately at the same instant in order to compose a full, accurate snapshot of the system. Therefore, it is important that each monitoring round starts approximately at the same time. However, because of process scheduling and other external factors, the difference of the start times of a given round across crawlers may increase significantly overtime. In the scope of our research, we used NTP to make sure crawlers are roughly synchronized. We also used real-time alarms to prevent any round start delays due to process scheduling overhead.

There are scenarios in which one cannot adjust the system clock and/or take advantage of real-time alarms. For monitoring practitioners dealing with them, we propose a design based on a global synchronization server to provide coordination to crawlers. In this design, as soon as a crawler is ready to take snapshots, it registers itself to the central server. Upon the start of a new round, that server contacts each registered crawler by sending a ROUNDSTART message. Although this process is subject to server-crawler communication latency, it would ensure that the round start offset across crawlers would not grow overtime.

Transient and permanent crawler failures

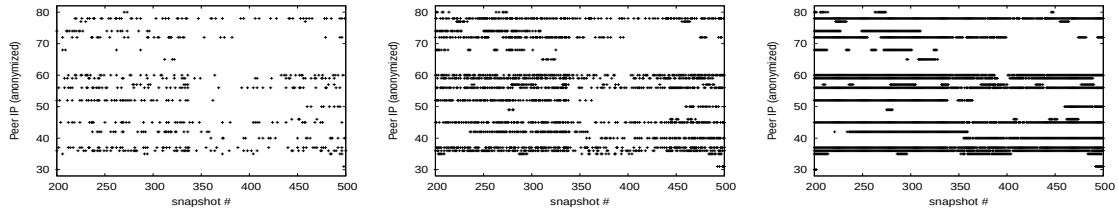
TorrentU uses a set of PlanetLab nodes as crawlers for taking snapshots. PlanetLab nodes have a convenient characteristic of geographic distribution. However, it comes at the cost of high node instability. Therefore, some nodes may transiently fail during the monitoring process, whereas others may fail permanently.

There are three candidate strategies to deal with this issue. The first is increasing crawler redundancy. However, this strategy might not be always suitable in those scenarios in which resource is limited. Given that most of nodes suffer from transient (rather from permanent) failures, one possible direction is to temporarily replace crawlers.

A second strategy (which we considered in our research) is to use keep-alive messages to test node availability. The central server would be responsible for periodically sending those messages and checking whether that node is still available. If some node fails to reply for a number of consecutive rounds, it is replaced by another backup node.

The strategy discussed above replaces crawlers permanently, and requires a certain number of rounds in order to ensure that the node actually went permanently offline, i.e. the fact that the node did not reply to successive keep-alive messages was not due to a transient communication failure. Most of PlanetLab node failures are transient, and often these nodes come back online quickly. However, during these transient failures, nodes are unable to collect peer lists.

In order to deal with such transient failures, we propose as a third strategy the use of special crawlers, denoted as *fire-crawlers*. To understand this strategy, consider the design for ensuring global synchronization of crawlers discussed earlier. Taking advantage of that design, if some crawler x fails to register as ready for the upcoming round, the server



(a) 1 crawler (b) 5 crawlers (c) 50 crawlers

Figure 3.3: Set of snapshots from a BitTorrent file sharing community, anonymized, shown in three resolutions.

would schedule a fire-crawler to take that snapshot instead. In case x suffered from a transient failure, it would later report to the server that it is ready to capture snapshot. In this case, the server would then de-allocate the fire-crawler, and send a *go* message to x in the following round.

Accuracy of snapshots

Technical limitations prevent that peer lists be able to contain every peer the tracker is aware of. In other words, only a fraction of peers some tracker is aware of will figure in each peer list it provides. A naive solution to overcome this problem is to request a certain number peer lists so that the probability that everyone figures at least once in any of those lists is maximized.

Note however that there is a minimal interval between announces imposed by trackers. Therefore, to increase snapshot accuracy, we must increase the number of crawlers dedicated to collect them. Figure 3.3 illustrates a trace taken using a varying number of crawlers: 1, 5, and 50 crawlers. In this figure, the x axis shows the snapshot number, whereas the y axis depitcs the peer identifier (anonymized). A dot in this plot means that peer having id y was online when snapshot x was taken. Note in this figure that increasing the number of crawlers improves trace resolution.

To estimate the number of crawlers required to take more accurate snapshots from a swarm population of certain size, we take advantage of the methodology proposed by Hoãÿfeld et al. (HOÃŸFELD et al., 2011). This methodology was formulated based on a variation of the coupon's collector problem with samples of size $k > 1$ (KOBZA; JACOBSON; VAUGHAN, 2007). In this methodology, an approximation for the number of required tracker's responses to obtain a full snapshot of the users' population in a given swarm ($E[X]$) can be obtained from Equation 3.1.

$$E[X] \approx \frac{E[N] \cdot h_N}{L_{max} \cdot |B|} \tag{3.1}$$

In the equation above, N is the expected number of users, and h_N the N-th harmonic number. The maximum size of a peer list is denoted by L_{max}. Considering for example an estimate of the swarm population around 4,000 users, $L_{max} = 200$ users (a typical setting found for most BitTorrent file sharing communities) and $|B| = 4$ trackers (i.e. 800 users obtained in each request), we obtain $E[X] \approx 44$ crawlers; for the sake of fault tolerance, one may further increase the number of crawlers.

This solution might not be applicable when crawlers are a scarce resource. As a consequence, some peers may fail to appear in snapshots. This failure has serious consequences to the quality of traces. In the next session we discuss this aspect in more detail,

and present a methodology for correcting traces collected using snapshot-based strategies.

3.2 Improving the accuracy of collected traces

The fact that some users may fail to appear in snapshots may introduce a harmful noise in the collected traces. To illustrate, consider Figure 3.4; the circles indicate a given user actually *was* in the system. Because snapshots s_8 and s_9 missed user v_2, the resulting trace will report two online sessions for that user, when the user actually joined and left the system once.

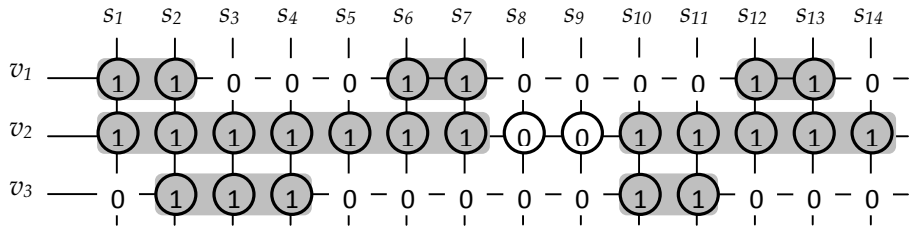


Figure 3.4: Users seen on each snapshot taken from the monitored system.

As previously discussed, this issue could be addressed by increasing the number of crawlers used for taking snapshots. However, there are scenarios in which the number of crawlers available may be insufficient. For example, according to the methodology proposed by Zhang et al. (ZHANG et al., 2011), monitoring a BitTorrent ecosystem with an average of 10,000 users requires around 489 crawlers; for 100,000 users, around 6,045 ones are required. As the target system grows in popularity, it becomes challenging to ensure the proper operation of these crawlers during the monitoring, not to mention that over-provisioning cannot ensure that truly accurate snapshots are in fact taken.

To deal with this issue, in the scope of this thesis we came up with a solution for improving the accuracy of captured traces. For the sake of generality, we focused on the design of a solution agnostic of system-specific properties or user-dependent, seasonal behaviors. As a consequence, our solution is not only suitable for applicability to snapshot sets collected from BitTorrent file sharing communities, but also to traces from distributed systems in general. In the following sections we describe our solution in detail.

3.2.1 System model and notation

In this section we introduce the system model and snapshot definition we adopt in the design of our solution for improving the accuracy of traces. We also discuss a set of assumptions and limitations related to our methodology.

System components and operation

Let $G = \langle V_t, E_t, B \rangle$ be a finite system comprised of nodes $V_t = \{v_1, \ldots v_m\}$, edges $E_t \subseteq V_t \times V_t$, and a set of bootstrap entities $B = \{b_1, \ldots b_n\}$. Each node corresponds to a user online in the system at instant t. Edges in this system indicate awareness of online presence: an edge from a user v_i to v_j means v_i is aware that v_j is online at instant t. By aware we mean that v_i had some recent communication with v_j, or that v_i learned this information from some other entity (bootstrap or other user). Each bootstrap entity is responsible for user admission. We assume that b_k is aware of v_i in the system at instant t if and only if b_k handled the admission of v_i, or v_i contacted b_k for some other purpose.

Let $e(v_i) \subseteq V_t$ be the set of online users that v_i is aware of. Similarly, let $e(b_k) \subseteq V_t$ be the set of online users that b_k is aware of.

Fetch Acquaintance List. This operation, launched by v_i to b_k (or v_j), consists in obtaining a (sub)list L of online users that b_k (or v_j) is aware of; $L(v_i, x)$ denotes the (sub)list obtained by v_i from x (x can be either a bootstrap entity $b_k \in B$ or a user $v_j \in V_t$). The maximum size of such a list is denoted by L_{max}. Observe that v_i may only query some user v_j if v_i is aware of v_j. By definition, $L(v_i, b_k) \subseteq e(b_k)$; similarly, $L(v_i, v_j) \subseteq e(v_j)$.

Building snapshots

Here we define formally the trace collection process based on snapshot capturing. Let $\mathcal{S}$ be a set of snapshots taken from users in G within some period. A snapshot $s_j \in \mathcal{S}$ is a list of users seen online in instant t_j. Snapshots are taken every Δt time units, at times $t_0, t_1, \ldots$ so that $t_j = t_0 + j \cdot \Delta t$. Let V be set of nodes seen at least once in the system; by definition, $V_t \subseteq V$. From $\mathcal{S}$ we can build a binary matrix $M_{|V|,|\mathcal{S}|}$, where $m_{i,j}$ represents the state of user v_i in snapshot s_j: 1 if $v_i \in s_j$ ("positive snapshot"), or 0 otherwise ("null snapshot").

The strategy for capturing users' usage information we approach here uses a set $V' \subseteq V$ of special users – *instrumented clients* (or *crawlers*) – to take snapshots. An instrumented client $v_i \in V'$ can contact the bootstrap entities or the online users (or both) to build a (partial) snapshot. A snapshot s is then formed as a union of the lists obtained by crawlers: $s = \bigcup L(v_i, x)$. Note that obtaining $L(v_i, x)$ has a non-negligible cost for v_i (e.g. due to bandwidth requirements, network latency, number of fetches allowed per period, among others).

Model assumptions

- **Users Arrival and Departure.** We make no assumptions regarding users' arrival and departure. It broadens the scope of our methodology to a variety of systems and scenarios (for example, flash-crowds). However, we assume that at least one bootstrap entity in the system becomes aware when a user joins the system. This assumption is actually a basic requirement for many systems to function; for example, a user must know the address of a bootstrap server to join the system.

- **Online Session Duration.** We consider that the longer a given users' online session already is [in the past], the longer it shall last [in the future] (YAO et al., 2009; STEINER; EN-NAJJARY; BIERSACK, 2009). This is the case of those users watching a video streaming, or downloading some large content. We take advantage of this observation to detect the likeliness that a given online user failed to appear in some snapshot. Apart from it, we do not assume any distribution for the duration of online sessions, neither any correlation between them.

- **Composition of Acquaintance List.** We assume that acquaintance lists are built randomly, following a uniform distribution. This assumption is related to a variant of the coupon's collector problem with random sample sizes (KOBZA; JACOBSON; VAUGHAN, 2007), and has been considered in a variety of scenarios (HOÃŸFELD et al., 2011; ZHANG et al., 2011). Addressing those cases in which acquaintance lists ared biased (or consider a distribution other than uniform) is envisaged as a direction for future research.

System-specific limitations

- **Awareness of Online Users.** Ideally, a snapshot taken at time t should be equal to $s_t \subseteq V_t$. However, there may have cases in which snapshots falsely report a user as being online. The reasons may vary; for example, v_i may not disconnect gracefully (e.g. crash), or b_k may fail to receive v_i's "disconnect" message. These cases are rather uncommon, often occurring due to some system-specific design. Therefore, we leave this aspect out of scope of this investigation, and envisage it as direction for future research.

- **Amplitude of Acquaintance Lists.** Ideally, $L = V_t$ (or $s_t = V_t$) at any point in time t. However, there are a number of system-specific limitations that prevent this from taking place. For example, constrained devices (e.g. embedded hardware or sensors) may not have sufficient memory for maintaining a large list of acquaintances. Second, some systems limit the acquaintance list size L_{max} for the sake of optimization or network-wide constraints (MANSILHA et al., 2011). It may even be unfeasible for an entity to build and transmit a full list of acquaintances as $|V_t|$ reaches the order of thousands or millions.

3.2.2 Formal definition of snapshot failures

In the scope of this work, we use the term *failure probability* (denoted herein as p) to refer to the chance that a given user $v_i \in V_t$ is missing in a snapshot s_t taken at instant t. Before introducing the methodology to estimate the failure probability for a given set of snapshots, we first provide in this section a set of definitions that emphasize the system properties and constraints involved in the problem.

Definition 1. *Let $e(b_k)$ be the list of users currently online that bootstrap entity b_k is aware of. We then have $\bigcup_{k=1}^{|B|} e(b_k) = V_t$ and $\bigcap_{k=1}^{|B|} e(b_k) \subseteq V_t$.*

Definition 1 evidences two aspects of our system model. First, as mentioned in the previous section, bootstrap entities handle user admission, and provide acquaintance lists upon request. In addition, a bootstrap entity b_k becomes automatically aware that v_i is online once v_i contacts b_k for admission. Therefore, any user in the system is recognized by at least one bootstrap entity as being online. The second one is similarly trivial. In our system model, one user may register with multiple bootstrap entities. Therefore, a user in the system may be recognized by more than one bootstrap entity as being online.

Definition 2. *Let $e(v_i)$ be the list of users currently online that v_i is aware of. We then have $\bigcup_{i=1}^{|V_t|} e(v_i) \subseteq V_t$ and $\bigcap_{i=1}^{|V_t|} e(v_i) \subseteq V_t$.*

Definition 2 emphasizes one subtle difference between users' and bootstrap acquaintance lists. While bootstrap entities are aware of any user currently online, users may not be fully aware of each other. For example, one recently admitted user will not be known by already online ones; this information will be gradually learned by others as they request novel acquaintance lists from the bootstrap entities.

One important conclusion can be drawn from the definitions above. A better accuracy in the snapshot building process can only be achieved if the bootstrap entities are used as central source of information for users currently online.

Definition 3. *Let $L(v_i, b_k)$ be the acquaintance list returned by b_k upon request from v_i. We then have $L(v_i, b_k) \subseteq e(b_k)$.*

Definition 4. *Let $L(v_i, v_j)$ be the acquaintance list returned by v_j upon request from v_i. We then have $L(v_i, v_j) \subseteq e(v_j)$.*

Definitions 3 and 4 evidence a system limitation. Assuming L_{max} bounded to an arbitrary size (for the sake of optimization or network-wide constraints), it turns out that the acquaintance list may not have room for every user b_k (or v_j) is aware of. The implication of this limitation is described next.

First, recall that gathering users' usage information has a cost (due to required bandwidth, network latency, etc.). Acquaintance lists must be obtained timely, in order to have a *true* instantaneous view of the system. Therefore, for building a single snapshot, each instrumented client send as few list requests as possible. This requirement makes it imperative taking into account the budget allocated for the snapshot capturing process (HOÃŸFELD et al., 2011). In this work, we consider such a budget in terms of number of instrumented clients used.

In this context, the conditions for obtaining a complete snapshot of users' population in the system are given by Theorems 1 and 2.

Theorem 1. *Let c be the budget required for capturing snapshots from G, and let $L(v_i, b_k)$ be the acquaintance list returned by $b_k \in B$ upon request from $v_i \in V'$. Considering that crawler v_i makes only one acquaintance list request per snapshot, we then have $s_t = \bigcup_{i=1}^{c} L(v_i, b_i) = V_t \iff L_{max} \to \infty$, $c \geq |B|$, and each bootstrap entity is queried at least once.*

Proof. $\boldsymbol{A \Rightarrow B}$.
Suppose that each user $v_i \in V$ is regarded as currently online by a single bootstrap entity $b_k \in B$ only. We thus have from Definition 1 that $\bigcup_{k=1}^{|B|} e(b_k) = V_t$ and $\bigcap_{k=1}^{|B|} e(b_k) = \emptyset$. From Definition 3, we have $L(v_i, b_k) \subseteq e(b_k)$. Now consider the restrictions that a crawler $v_i \in V'$ makes only one acquaintance list request per snapshot. In this case, obtaining a full snapshot of the system ($s_t = V_t$) requires contacting each of the bootstrap entities in the system, and that $L(v_i, b_k) = e(b_k)$. The condition for $L(v_i, b_k) = e(b_k)$ being true regardless of the size of a bootstrap acquaintance list is that $L_{max} \to \infty$, whereas the condition for $\bigcup_{i=1}^{c} L(v_i, b_i) = V_t$ is that there is at least one crawler for each bootstrap entity (i.e., $c \geq |B|$), and that each bootstrap entity be queried by a crawler at least once.

$\boldsymbol{B \Rightarrow A}$.
Suppose that $L_{max} \to \infty$. In this case, $L(v_i, b_k) = e(b_k)$. Suppose also that $c = |B|$ and that each bootstrap entity is queried at least once when building a snapshot set. In this case, we have $\bigcup_{i=1}^{c} L(v_i, b_i) = V_t$, which also respects the restriction that each crawler makes only one acquaintance list request per snapshot. □

Theorem 2. *Let c be the budget required for capturing snapshots from G, and let $L(v'_i, v_j)$ be the acquaintance list returned by $v_j \in V_t$ upon request from a instrumented client $v'_i \in V'$. Considering that client v'_i makes one acquaintance list request per snapshot, we then have $s_t = \bigcup_{i=1}^{c} L(v'_i, v_i) = V_t \iff \bigcup_{j=1}^{|V_t|} e(v_j) = V_t$, $L_{max} \to \infty$, $c \geq |V_t \setminus V'|$, and each user v_i is queried at least once.*

Proof. $\boldsymbol{A \Rightarrow B}$.
Suppose that each ordinary user $v_i \in V$ is regarded as online by at least one user $v_j \in V$ (i.e., $E_t = V_t \times V_t$). This situation may lead to several possible configurations in the acquaintance graph; it ranges from one in which the graph forms a single cycle, to another one in which just a single user v_i is aware of all other users v_k in the system, and

some other user v_j is aware of v_i as being online. We thus have from Definition 2 that $\bigcup_{j=1}^{|V_t|} e(v_j) = V_t$ and $\bigcap_{j=1}^{|V_t|} e(v_j) = \emptyset$. From Definition 4, we have $L(v_i, v_j) \subseteq e(v_j)$. Now consider the restrictions that a crawler $v_i \in V'$ makes only one acquaintance list request per snapshot. In this case, obtaining a full snapshot of the system ($s_t = V_t$) requires contacting each of the users in the system, and that $L(v'_i, v_j) = e(v_j)$. The condition for $L(v'_i, v_j) = e(v_j)$ being true regardless of the size of a user's acquaintance list is that $L_{max} \to \infty$. The condition for $\bigcup_{i=1}^{c} L(v'_i, v_i) = V_t$ is that (i) there is at least one crawler for each ordinary user ($c \geq |V_t \setminus V'|$), (ii) that each ordinary user be queried by a crawler at least once, and (iii) that every user be regarded as currently online by at least another user in the system (which results in $\bigcup_{j=1}^{|V_t|} e(v_j) = V_t$).

***B* ⇒ *A*.**
Suppose that $L_{max} \to \infty$. In this case, $L(v'_i, v_i) = e(v_i)$. Suppose also that $c \geq |V_t \setminus V'|$, that each ordinary user is queried at least once when building a snapshot set, and that every user be regarded as currently online by at least another user in the system (i.e., $\bigcup_{j=1}^{|V_t|} e(v_j) = V_t$). In this case, querying all ordinary users using the budget we have and making the union of the peer lists obtained, we obtain a full snapshot of the system ($\bigcup_{i=1}^{c} L(v'_i, v_i) = V_t$). Observe that since we have a crawler for each ordinary user, the restriction that each crawler makes only one acquaintance list request per snapshot can thus be respected. □

Observe from Theorems 1 and 2 that achieving $s_t = V_t$ (a complete snapshot) by querying bootstrap entities only is far more feasible (and likely) than achieving it by querying ordinary users only. While Theorem 1 states that we just need unbounded L_{max}, Theorem 2 makes it clear that achieving $s_t = V_t$ requires that everyone be aware of each other in the system (i.e. $E_t = V_t \times V_t$), in addition to unbounded L_{max}. It is important to emphasize that Theorems 1 and 2 consider unbounded L_{max}. With L_{max} bounded to some size, we have $s_t = \bigcup_{i=1}^{|V'|} \bigcup_{k=1}^{|B|} L(v_i, b_k) \subseteq V_t$ (if querying bootstrap entities) and $s_t = \bigcup_{i=1}^{|V'|} \bigcup_{j=1}^{|V_t|} L(v_i, v_j) \subseteq V_t$ (if querying ordinary users).

From the definitions and theorems presented earlier, one simple, straightforward strategy to estimate the probability that a snapshot missed a user would be given by Equation 3.2.

$$p = 1 - \frac{\min(L_{max} \cdot |V'|, |V_t|)}{|V_t|} \tag{3.2}$$

Note however that Equation 3.2 relies on the assumption that it is possible to obtain (an estimate) for $|V_t|$. In those more realistic scenarios in which this information cannot be obtained, the best one can do is checking if $|V'| = |B|$ and $\forall v_i \in V', |L(v_i, b_k)| < L_{max}$ (or $|L(v_i, v_j)| < L_{max}$) is *true*; in this particular case, and according to Theorem 1, the snapshot did not miss any online user. In the general case, the average size of lists obtained to build $|s_t|$ provide far from sufficient information to estimate p.

3.2.3 Unveiling more accurate users' sessions from ill-captured snapshots

In this section we describe our solution for correcting ill-captured snapshots and building traces of users' online sessions from them. We first provide a discussion on measuring failure probability for some system, and then elaborate on our methodology to estimate it from a given set of snapshots.

Assuming that (i) we have no knowledge of the system structure (e.g. users' and bootstrap awareness relationships), and (ii) no significant information can be obtained from the snapshot capturing process, in this investigation we rely solely on the information contained in the snapshot set to estimate its average *failure probability* p. We envisage the use of system-specific sources of information for estimating p as a prospective direction for research in the area.

Let X_i be the event that a given user v_i is online and present in snapshot s_j ($m_{i,j} = 1$), Y_i be the event v_i is not in s_j ($m_{i,j} = 0$) though online, and Z_i be the event v_i is offline ($m_{i,j} = 0$). The probability of a failed snapshot for v_i is then given by $P(Y_i) = p$, whereas the probability of a correct snapshot is $P(X_i) + P(Z_i) = 1 - p$. The challenge we address here is to estimate $P(Y_i)$, given that events Y_i and Z_i are indistinguishable in the snapshot.

Given that a user may fail to appear in a snapshot with probability p, our goal is to obtain an estimate for p ($\hat{p}$) and use it to correct the traces. To this end we explore two key observations, described in the propositions shown next. First, the longer a user has been in a system, the longer that user is expected to remain there (YAO et al., 2009; STEINER; EN-NAJJARY; BIERSACK, 2009); consequently, there is a high probability that a user's absence in the snapshot following a long session is an event of type Y (rather than Z):

Proposition 1. *Let k be a sequence of "positive" snapshots ($k \in \mathbb{N}^*$). The probability of a false negative null $k + 1$ one is expected to increase geometrically with k (Bernoulli process).*

Proof. Consider each snapshot a Bernoulli trial. Let *failure* be a trial in which a given user *is* online and in the snapshot (positive sample), and *success* be a trial in which that same user is online, but *is not* in the snapshot (false negative sample). In this case, the probability of a false negative sample after k consecutive, positive ones follows a Bernoulli process with a geometric distribution, and it is given by $1 - (1 - p)^{k+1}$; by definition, $1 - (1 - p)^{k+1}$ increases geometrically with k. □

Second, the longer is a sequence of null samples, it become more likely the user actually went offline during that period:

Proposition 2. *Let k be a sequence of null snapshots ($k \in \mathbb{N}^*$). The probability that all of them are failed ones decreases exponentially with k.*

Proof. Let $Y_{i,j}$ be the event that a given user i is online, but not present in the j-th snapshot, and $P(Y_{i,j}) = p$ be the probability of such event. Now assume that users have an uniformly distributed chance of being selected to compose an acquaintance list $L(v_i, x)$. It then turns out that failure events are statistically independent. Therefore, using the conditional probability and chain rule, we obtain

$$P\left(\bigcap_{j=1}^{k} Y_{i,j}\right) = \prod_{j=1}^{k} P\left(Y_{i,j} \,\middle|\, \bigcap_{l=1}^{j-1} Y_{i,l}\right) \therefore p^n$$

, which by definition decreases exponentially with n. □

We build on these propositions to obtain $\hat{p}$ with an error $1 - \alpha$, with $\alpha \in (0, 1)$. In this context, values of α closer to 0 represent a more conservative approach towards finding failed snapshots, in which false positives become less likely. Conversely, values closer to 1 represent a more relaxed approach, leading to a higher number of snapshot corrections.

Focusing on Proposition 1, what values of p allow snapshot $k+1$ to be a failed one? To find them, we make $1-(1-p)^{k+1} \geq 1-(1-\alpha)$. The probability p (for k consecutive snapshots) that satisfies this inequality is given by Equation 3.3.

$$p \geq 1-(1-\alpha)^{\frac{1}{k+1}} \tag{3.3}$$

The implication of Proposition 2 is illustrated in Figure 3.5. The probability that a given null snapshot be a false negative one, for a given user v, decreases with the number of preceding null snapshots. Note that this probability can be measured both onwards (curve 1) and backwards (curve 2). Therefore, checking if v actually went offline restricts to measuring the probability that snapshot s_i (median) is a false negative one for her. This is because s_i has the lowest failure probability considering both onward and backward directions simultaneously. Admitting an error $1-\alpha$, a sequence of length k shall be correct if $(1-p)^{\frac{k+1}{2}} \geq 1-\alpha$. The probability p that satisfies this inequality is given by Equation 3.4.

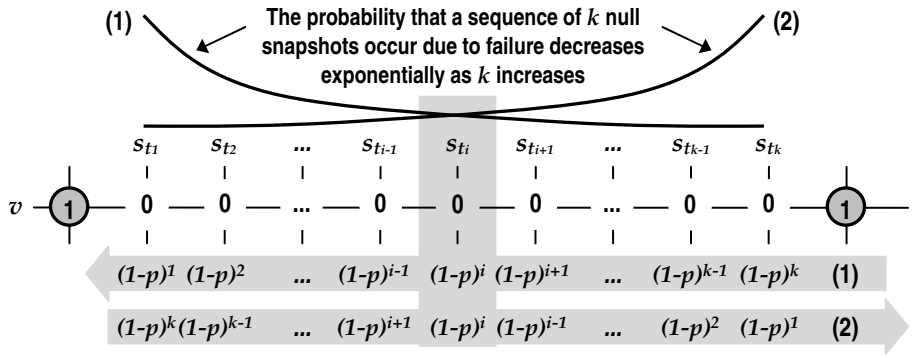


Figure 3.5: Conditional probability.

$$p \leq 1-(1-\alpha)^{\frac{2}{k+1}} \tag{3.4}$$

The failure probability for any sequences of snapshots is then computed through a weighted sum of the probability for each sequences of length k. In this case, we are interested in the lowest probability possible for these sequences. Building from Equations 3.3 and 3.4, we have the non-linear system below:

$$\begin{aligned} &\text{minimize } \hat{p} \\ &\text{subject to } \hat{p} \geq \sum_{x=1}^{n} P[X=x] \cdot \left(1-(1-\alpha)^{\frac{1}{x+1}}\right) \\ &\qquad\qquad \hat{p} \leq \sum_{y=1}^{m} P[Y=y] \cdot \left(1-(1-\alpha)^{\frac{2}{y+1}}\right) \end{aligned}$$

In the non-linear system above, $P[K=k]$ is the fraction (normalized) of sequences of length k, n is the length of the longest sequence of positive snapshots, and m the length of the longest sequence of null ones; these values are derived from the snapshot set under analysis. The value of $\hat{p}$ that satisfies the non-linear system above is the failure probability of the snapshot set being processed. Note that this non-linear system cannot be solved if

there is no value of $\hat{p}$ that can satisfy both constraints at the same time. In this case, there is no sufficient data for obtaining a proper value for $\hat{p}$.

It is important to emphasize that our methodology does not require calibration or prior training with ground-truth data in order to be effective. The only requirement is that the snapshot set submitted as input contain sufficient data so that the non-linear system can be solved and a failure probability can be estimated. Apart from this aspect, it is important to determine a value for α that achieve a proper balance between improved accuracy in the resulting snapshot set and occurrence of false positives. In Section 3.2.4 we present a sensitivity analysis regarding this input parameter.

Correcting the snapshot set

To improve the accuracy of a snapshot set $\mathcal{S}$, we basically apply Proposition 2, using the value of $\hat{p}$ found, to the sequences of null snapshots for a given user v_i. To understand this process, suppose a snapshot set $\mathcal{S}$ given as input. Based on the methodology described in the previous subsection, we propose the steps described in Algorithm 2 to improve $\mathcal{S}$.

The first step in this process is obtaining the tabulated frequencies of sequence lengths of positive and null snapshots from the input snapshot set $\mathcal{S}$ (lines 2 and 3 in Algorithm 2). To illustrate, from the snapshot set shown in Figure 3.6(a), the frequency of positive snapshots of length 2 is 4 ($pos\_st.hist\,[2] = 4$), and the frequency of null snapshots of length 5 is 1 ($pos\_st.hist\,[5] = 1$).

The subsequent steps (lines 4 and 5) consist in obtaining the total number of sequences of positive and null samples in the snapshot set $\mathcal{S}$. In the example from Figure 3.6(a), these numbers are *pos_st.total* = 7 and *nul_st.total* = 4 (note that a null sequence of snapshots is defined as one which is bounded by two positive ones). The longest length of a positive snapshot and null one are also obtained from the snapshot set $\mathcal{S}$ (lines 6 and 7). From the example of Figure 3.6(a), we have *pos_st.max_len* = 7 and *nul_st.max_len* = 5.

The next step in Algorithm 2 is to estimate the failure probability for the sequences of positive ($\hat{p_{pos}}$, lines 8-12) and null ($\hat{p_{nul}}$, lines 13-17) snapshots, and then find a solution to the non-linear system presented in the previous subsection (lines 18-21). Note that if the non-linear system cannot be solved (because of the constraints to $\hat{p}$ described in the previous subsection), then the algorithm returns without modifying the snapshot set given as input (line 19).

In case the system can be solved, the subsequent step is to determine which is the longest length *len* for a sequence of null snapshots so that it can be regarded as a failure (line 22). Note that the value of *len* is obtained from an estimated, average failure probability $\hat{p}$ computed for the entire snapshot set $\mathcal{S}$. As a result, some false positives may occur; in the following section we evaluate the proportion of false positives in the resulting snapshot sets. Then, all the sequences of null snapshots that have length less or equal to *len* are turned into positive snapshots (lines 22 - 27).

To illustrate the process above, consider again the snapshot set from Figure 3.6(a). Assuming that a value of $\hat{p} = 0.26$ was obtained using $\alpha = 0.95$, we have *len* = 2. Therefore, only sequences of null snapshots of length 2 or less can be turned into positive samples in that snapshot set; it means the sequence $s_{8...9}$ for user v_2. As a consequence, user v_2, who had two online sessions in the old snapshot set, will be reported as having just one online session, beginning in s_1 and ending in s_{14}. This new setting in fact corresponds to reality, according to the scenario illustrated in the beginning of Section 3.2.

Finally, after the false negative samples for a given user v have been identified and corrected (admitting an error $1 - \alpha$), the modified snapshot set $\mathcal{S}$ is returned (line 28).

Algorithm 2 Algorithm for correcting snapshot sets

procedure CORRECTSNAPSHOTSET($\mathcal{S}, \alpha$) ▷ *input set and error factor*
  pos_st.hist[] ← tabulated freq. of sequences of positive snapshots from $\mathcal{S}$
  nul_st.hist[] ← tabulated freq. of sequences of null snapshots from $\mathcal{S}$
  pos_st.total ← total number of sequences of positive snapshots from $\mathcal{S}$
  nul_st.total ← total number of sequences of null snapshots from $\mathcal{S}$
  pos_st.max_len ← length of the longest sequence of positive snapshots from $\mathcal{S}$
  nul_st.max_len ← length of the longest sequence of null snapshots from $\mathcal{S}$
  $\hat{p_{pos}} \leftarrow 0$
  for $i \leftarrow 1 \ldots$ pos_st.max_len **do**

    $\hat{p_{pos}} \leftarrow \hat{p_{pos}} + \left(\text{pos\_st.hist}[i] \cdot \left(1 - (1-\alpha)^{\frac{1}{i+1}}\right)\right)$
  end for
  $\hat{p_{nul}} \leftarrow 0$
  for $j \leftarrow 1 \ldots$ nul_st.max_len **do**

    $\hat{p_{nul}} \leftarrow \hat{p_{nul}} + \left(\text{nul\_st.hist}[j] \cdot \left(1 - (1-\alpha)^{\frac{2}{j+1}}\right)\right)$
  end for
  if $\hat{p_{pos}} > \hat{p_{nul}}$ **then**
    return *null* ▷ *failure probability cannot be estimated*
  else
    $\hat{p} \leftarrow \hat{p_{pos}}$
    len $\leftarrow \left\lfloor \frac{\ln(1-\alpha)}{\ln(\hat{p})} \right\rfloor$
    for all sequences $s_{v,1\ldots n} \in \mathcal{S}$ of null snapshots for a given user v_i **do**
      if length $(s_{v,1\ldots n}) \leq$ len **then** ▷ *sequence is likely to be a failure*
        turn $s_{v,1\ldots n}$ into a sequence of positive snapshots and replace in $\mathcal{S}$
      end if
    end for
    return $\mathcal{S}$ ▷ *modified set of snapshots*
  end if
end procedure

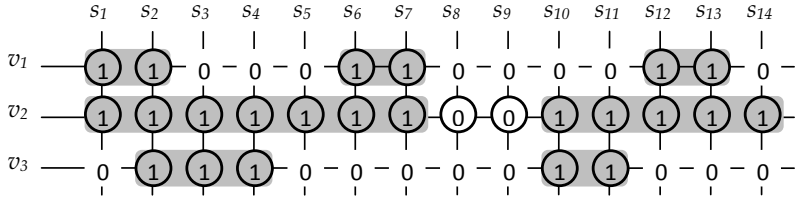
3.2.4 Evaluation

In our evaluation, we attempted to answer three research questions. First, how effective of our methodology is in accurately correcting missing snapshots? Second, what is the incidence of false positives in the correction process? Finally, what are the scenarios for which snapshot set correction using our methodology is more/less appropriate? To answer them, we carried out a set of experiments considering scenarios with varying snapshot sets and parameter setting. Next we discuss the most prominent results achieved.

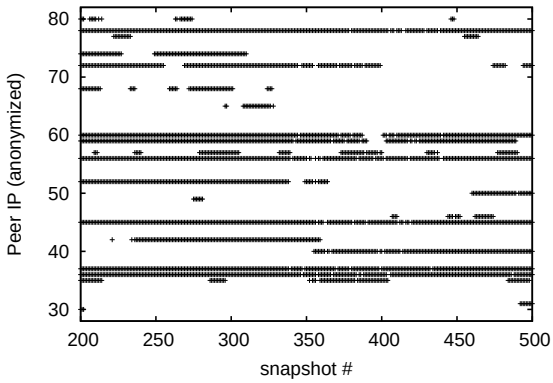
Ground-truth dataset

We considered snapshot sets taken from various swarms of a BitTorrent file sharing community[2]. In summary, we used 50 instrumented clients ($|V'| = 50$) for taking a total

[2]The dataset considered in this evaluation is available for download at `http://www.inf.ufrgs.br/~wlccordeiro/idmgmt/`



(a) example of snapshot set



(b) snapshot set captured using 50 crawlers

Figure 3.6: Example of a snapshot set, and real-life set of snapshots from a BitTorrent file sharing community.

of 2,880 snapshots of the users' population in the system, in a period of 30 days (starting on June 15th, 18:00 UTC 2013); the interval between snapshots was 15 minutes. The average number of users per snapshot in the largest swarm was 3,035, with a maximum of 5,887 and minimum of 1,636 users. The standard deviation was 794.78. That swarm was maintained by 4 trackers. The number of $|V'| = 50$ was obtained after the methodology proposed by Hoãÿfeld et al. (HOÃŸFELD et al., 2011). A partial view of the snapshot set for this swarm is provided in Figure 3.6(b).

Evaluation metrics

Recall from Section 3.2.1 that a snapshot s_t is formed from the union of the partial snapshots collected by each instrumented client $v_i \in V'$. Therefore, in the remainder of this section we denote as $\mathcal{S}_{|V'|}$ the snapshot set captured using $n = |V'|$ instrumented clients. We also denote as $\mathcal{S}'_{|V'|,\alpha}$ the snapshot set corrected using our methodology, using $\alpha = x$, and $\mathcal{S}_{gt}$ the snapshot set used as baseline (ground-truth) for measuring the performance of our methodology. For the sake of evaluation, we define the following metrics:

- **Number of missing positive snapshots (ω).** Recall from Section 3.2.1 that $M_{|V|,|\mathcal{S}|}$ is the binary matrix built from $\mathcal{S}$. Let $P(\mathcal{S}_{|V'|}) = \{m_{i,j} | m_{i,j} = 1\}$ be the set of positive snapshots from a snapshot set $\mathcal{S}_{|V'|}$. Then, we can define the number of missing positive snapshots as

$$\omega = \left| P(\mathcal{S}_{gt}) \setminus P(\mathcal{S}_{|V'|}) \right| \tag{3.5}$$

  In other words, it is the number of positive snapshots from $\mathcal{S}_{gt}$ (ground-truth) that were initially null in $\mathcal{S}_{|V'|}$.

- **Number of modified snapshots (ϕ).** This metric accounts for those positive snapshots in $\mathcal{S}'_{|V'|,\alpha}$ that were initially null in set $\mathcal{S}_{|V'|}$

$$\phi = \left| P(\mathcal{S}'_{|V'|,\alpha}) \setminus P(\mathcal{S}_{|V'|}) \right| \tag{3.6}$$

- **Number of accurately corrected snapshots (τ).** This metric indicates the number of positive snapshots in $\mathcal{S}'_{|V'|,\alpha}$ that, initially null in set $\mathcal{S}_{|V'|}$, match a positive snapshot in $\mathcal{S}_{gt}$ (ground-truth). Formally, we have

$$\tau = \left| \left(P(\mathcal{S}_{gt}) \setminus P(\mathcal{S}_{|V'|}) \right) \cap P(\mathcal{S}'_{|V'|,\alpha}) \right| \tag{3.7}$$

  The possible values for this metric range in the interval $[0, \omega]$. The value $\tau = \omega$ is a global optimum; it means that all missing snapshots were accurately corrected.

- **Number of false positive corrections (ψ).** A false positive correction refers to a snapshot s_t that was erroneously corrected for some user v_i. In other words, v_i was offline when s_t was taken, but our methodology detected s_t as being a false null one. Formally, we have

$$\psi = \left| P(\mathcal{S}'_{|V'|,\alpha}) \setminus P(\mathcal{S}_{gt}) \right| \tag{3.8}$$

  The possible values for this metric range in the interval $[0, \phi]$. The value $\psi = 0$ is a global optimum; it means that no snapshot correction done was a false positive one.

Effectiveness of our methodology

To assess the effectiveness of our approach, we attempted to recreate the snapshot set $\mathcal{S}_{50}$ (our ground-truth), taking as input various snapshot sets $\mathcal{S}_{|V'|}$. For the sake of illustration, we focus on the results achieved considering a partial view of $\mathcal{S}_{50}$, shown in Figure 3.3(c). In this evaluation, we adopted $\alpha = 0.95$.

Table 3.1 shows a summary of the results achieved. In this table, field *Set* indicates the snapshot set considered (and number of instrumented clients used to capture it). Fields $p_{\hat{\text{pos}}}$ and $p_{\hat{\text{nul}}}$ indicate the average failure probability for positive and null sequences of snapshots, respectively. *Correct?* indicates whether it was possible to correct the snapshot set in question. Field *Length* contains the largest length of null snapshots that were considered for correction. Finally, fields *Metric* τ/ω and *Metric* ψ/ϕ present the results obtained for each snapshot set.

Accurately corrected snapshots *Metric* τ/ω. Observe from Table 3.1 that our methodology achieved high degree of accuracy in correcting those snapshot sets obtained with fewer instrumented clients. The ratio of accurately corrected snapshots ranged from 65% (for $\mathcal{S}_2$, i.e. the snapshot set built using 2 crawlers) to 87% (for $\mathcal{S}_{10}$, i.e. the snapshot set built using 10 crawlers). The highest accuracy measured in our experiments was 90%, for snapshot set $\mathcal{S}_{15}$. Note that it was not possible to correct $\mathcal{S}_1$, since $p_{\hat{pos}} > p_{\hat{nul}}$ (as discussed in Section 3.2.3).

Observe also that, in a first moment, the proportion of accurately corrected snapshots increases to a certain threshold, as the number of crawlers used to build the snapshot set grows higher. Then, this proportion starts decreasing, as the number of crawlers approaches the necessary for building a ground-truth dataset. The reason is as follows. In the first moment, the number of possible corrections is extremely high, since there are

Table 3.1: Snapshot sets submitted for correction, using $\alpha = 0.95$.

Set	$p\hat{pos}$	$p\hat{nul}$	Correct?	Length	Metric τ/ω		Metric ψ/ϕ	
					Values	Proportion	Values	Proportion
$\mathcal{S}_1$	0.750257	0.656683	No	–	–	–	–	–
$\mathcal{S}_2$	0.720646	0.754073	Yes	9	$^{1461}/_{2242}$	.6516	$^{31}/_{1492}$	.0207
$\mathcal{S}_3$	0.692409	0.817299	Yes	8	$^{1442}/_{1850}$	.7794	$^{44}/_{1486}$	.0296
$\mathcal{S}_4$	0.664327	0.835471	Yes	7	$^{1258}/_{1617}$	.7779	$^{53}/_{1311}$	.0404
$\mathcal{S}_5$	0.643048	0.849721	Yes	6	$^{1107}/_{1433}$	.7725	$^{53}/_{1160}$	.0456
$\mathcal{S}_6$	0.623862	0.863142	Yes	6	$^{1011}/_{1238}$	.8166	$^{61}/_{1072}$	.0569
$\mathcal{S}_7$	0.590806	0.868512	Yes	5	$^{871}/_{1023}$	.8514	$^{64}/_{935}$	.0684
$\mathcal{S}_8$	0.555304	0.875651	Yes	5	$^{717}/_{814}$	.8808	$^{75}/_{792}$	.0946
$\mathcal{S}_9$	0.543981	0.877639	Yes	4	$^{636}/_{741}$	.8582	$^{76}/_{712}$	.1067
$\mathcal{S}_{10}$	0.522767	0.870914	Yes	4	$^{540}/_{618}$	.8737	$^{80}/_{620}$	.1290
$\mathcal{S}_{15}$	0.473238	0.858767	Yes	4	$^{324}/_{360}$	.9000	$^{82}/_{406}$	.2019
$\mathcal{S}_{20}$	0.449136	0.842509	Yes	3	$^{193}/_{226}$	.8539	$^{81}/_{274}$	.2956
$\mathcal{S}_{25}$	0.420627	0.835212	Yes	3	$^{140}/_{159}$	.8805	$^{83}/_{223}$	.3721
$\mathcal{S}_{30}$	0.391245	0.814221	Yes	3	$^{70}/_{85}$	.8235	$^{83}/_{153}$	.5424
$\mathcal{S}_{35}$	0.337525	0.781824	Yes	2	$^{29}/_{35}$	.8285	$^{79}/_{108}$	.7314
$\mathcal{S}_{40}$	0.320752	0.764094	Yes	2	$^{14}/_{17}$	.8235	$^{79}/_{93}$	.8494
$\mathcal{S}_{45}$	0.302146	0.752152	Yes	2	$^{3}/_{5}$	.6000	$^{79}/_{82}$	.9634

very few crawlers being used for building the snapshot set. However, in various cases there might have fewer snapshots than necessary to perform most of these corrections. In the second moment, as the number of crawlers used to build the snapshot set increases, the opportunities for corrections decreases, thus decreasing the number of snapshots that our solution accurately corrects.

The main conclusion that can be drawn from the discussion above is that our methodology achieved better accuracy precisely for those snapshot sets captured with fewer resources. In other words, it enables one to capture more accurate snapshot sets using significantly less resources than the estimated using the methodology proposed by Hoãÿfeld et al. (HOÃŸFELD et al., 2011).

False positive corrections *Metric* ψ/ϕ. One can see from Table 3.1 that the incidence of false positives is significantly low, in absolute terms. In case of for $\mathcal{S}_{10}$, the occurrence of false positives accounted for 12%, whereas for $\mathcal{S}_2$ the ratio was 2% only. Similarly to the evaluation of accurately corrected snapshots, note that the ratio of false positives decreases significantly as the snapshot set resolution decreases. In summary, our methodology has shown to be more effective precisely in those scenarios where the snapshot set was captured with fewer resources.

Snapshot set correction threshold. This is an important aspect to consider when using our methodology. As shown in Table 3.1, it provided minor improvement to $\mathcal{S}_{45}$, at the expense of some false positives. Next we discuss an approach for assessing those scenarios which can better benefit from it.

As discussed in Section 3.2.3, using fewer instrumented clients increases the probability that some online user is missed. Considering $L_{max} = 200$ and initial swarm population of 4,000 users (from our snapshot set capture process discussed earlier in this

Table 3.2: Influence of parameter α in the correction of $\mathcal{S}_5$.

Set	$p_{\hat{pos}}$	$p_{\hat{nul}}$	Length	Metric τ/ω		Metric ψ/ϕ	
				Values	Proportion	Values	Proportion
$\alpha = 0.10$	0.038356	0.077806	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.05$	0.018886	0.038773	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.20$	0.079275	0.156736	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.30$	0.123293	0.237008	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.40$	0.171157	0.318917	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.50$	0.223962	0.402891	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.60$	0.283424	0.489587	0	$^{0}/_{1433}$	.0000	–	–
$\alpha = 0.70$	0.352509	0.580117	1	$^{383}/_{1433}$	.2672	$^{8}/_{391}$	.0204
$\alpha = 0.80$	0.437208	0.676659	1	$^{383}/_{1433}$	.2672	$^{8}/_{391}$	.0204
$\alpha = 0.85$	0.489514	0.728632	2	$^{656}/_{1433}$	.4577	$^{17}/_{673}$	.0252
$\alpha = 0.90$	0.553874	0.784924	3	$^{856}/_{1433}$	.5973	$^{27}/_{883}$	.0305
$\alpha = 0.95$	0.643048	0.849721	6	$^{1107}/_{1433}$	.7725	$^{53}/_{1160}$	.0456
$\alpha = 0.99$	0.779671	0.922288	18	$^{1333}/_{1433}$	.9302	$^{144}/_{1477}$	.0974

section), we can estimate a lower bound for the failure probability using Equation 3.2: $p = 1 - \frac{\min(200 \cdot |V'|, 4000)}{4000}$. For $\mathcal{S}_2$ we have $p = 0.90$, whereas for $\mathcal{S}_{45}$ we have $p = 0.0$.

One can use Equation 3.2 to asses whether our methodology can (significantly) improve a given snapshot set. In summary, one can simply compare the measured failure probability $p_{\hat{pos}}$ with the lower bound p. In case $p \gg p_{\hat{pos}}$, then the snapshot set is more suitable for correction with our methodology. This is because the probability that $\mathcal{S}_{|V'|}$ might have missed some online users is higher than our methodology has estimated. As a result, more accurate results can be achieved during the correction process than would happen if $p \ll p_{\hat{pos}}$.

To illustrate, consider $\mathcal{S}_2$. From Table 3.1, we have $p_{\hat{pos}} \approx 0.72$, smaller than its lower bound $p = 0.90$, and thus indicating a candidate for highly accurate correction. Conversely, consider $\mathcal{S}_{45}$. For this snapshot set we have $p_{\hat{pos}} \approx 0.30$, significantly higher than its lower bound $p = 0.0$. Therefore, this snapshot set might benefit significantly less from our methodology (which in fact did, as one can see from Table 3.1).

Sensitivity analysis

Table 3.2 shows the results from a sensitivity analysis of α taking $\mathcal{S}_5$ as basis. Observe that α regulates the conservativeness of our methodology in estimating $p_{\hat{pos}}$, and thus defining the length of the sequences of null snapshots that shall be corrected.

Lower values of α lead to more conservative estimates of $p_{\hat{pos}}$. Therefore, fewer sequences of null snapshots will become candidate for correction. Note that for $\alpha = 0.60$ (and lower), the length of sequences of null snapshots that would be corrected is 0. This is because the probability that a single snapshot has failed (0.28) is smaller than the error $1 - \alpha$ tolerated for not correcting a snapshot (in this case, $1 - 0.60 = 0.40$).

Conversely, higher values of α make our methodology more effective in accurately correcting failed snapshots, at the expense of some false positive corrections. For example, in case of $\alpha = 0.70$ (and higher), the measured failure probability (0.35) is higher

Table 3.3: Summarized statistics of the studied snapshot sets.

Set	Min.	1st. Quartile	Median	Mean	Std. Dev.	3rd. Quartile	Max.
Number of Sessions per User							
$\mathcal{S}_5$	1	1	3	10.97	33.88	7	671
$\mathcal{S}'_5$	1	1	2	4.95	13.45	4	303
$\mathcal{S}_{50}$	1	1	2	5.36	14.83	4	552
Duration of Sessions per User (minutes)							
$\mathcal{S}_5$	15	15	15	28.28	25.49	30	825
$\mathcal{S}'_5$	15	15	45	86.03	156.54	90	6,660
$\mathcal{S}_{50}$	15	15	45	102.9	257.62	105	24,750

than the error allowed ($1 - 0.70 = 0.30$).

In summary, a proper setting for α only takes in consideration the conservativeness one has for estimating the failure probability of the studied snapshot set. In other words, defining α only requires that one assesses the desired trade-off between accurate corrections and occurrence of false negatives.

Analysis of trace properties

As part of our evaluation, we analyzed aspects such as users' recurrence (i.e. number of arrivals per user) and duration of online sessions in our dataset. In summary, we attempted to answer the following research question: what impact our methodology causes to the corrected traces? To answer it, we considered a complete view of the snapshot set studied in the previous section, with resolutions $\mathcal{S}_5$ and $\mathcal{S}_{50}$ (ground-truth).

Table 3.3 shows a statistical summary for the snapshot sets considered: $\mathcal{S}_5$, $\mathcal{S}'_5$ (corrected using $\alpha = 0.85$, for the sake of conservativeness), and $\mathcal{S}_{50}$. Observe that the corrected snapshot set is more similar to the ground-truth dataset, in contrast to the original snapshot set $\mathcal{S}_5$. This can be observed, for example, by comparing the average number of sessions per user ("Mean") and its standard deviation ("Std. Dev.").

An interesting aspect to note is that the average number of sessions dropped by a factor of 2 (approximately), comparing snapshot $\mathcal{S}_5$ and $\mathcal{S}_{50}$, and that the average of sessions increased by a factor of 3.5 (approximately). This observation can be explained by the fact that correcting a null snapshot (i.e. turning it into a positive one) basically "merges" two sessions into one. Therefore, the number of sessions is decreased by one, whereas the duration of the new session becomes the sum of the duration of the merged sessions plus the length of the correction (which varies from 1 to $\lfloor {}^{\ln(1-\alpha)}/_{\ln(\hat{p})} \rfloor$).

Figure 3.7 shows a partial view of the cumulative distribution function for the users' recurrence and duration of online sessions. Observe that the curves for $\mathcal{S}'_5$ and the ground-truth are roughly the same. This evidences that our methodology can provide accurate corrections, and therefore can deliver more precise snapshot sets, even in scenarios where a very few amount of resources were used for their collection. The possibility of capturing reasonably accurate snapshot sets using just a few amount of resources represents a significant improvement over state-of-the-art solutions, which require an extremely high number of crawlers to capture accurate snapshots sets (HOÃŸFELD et al., 2011; ZHANG et al., 2011).

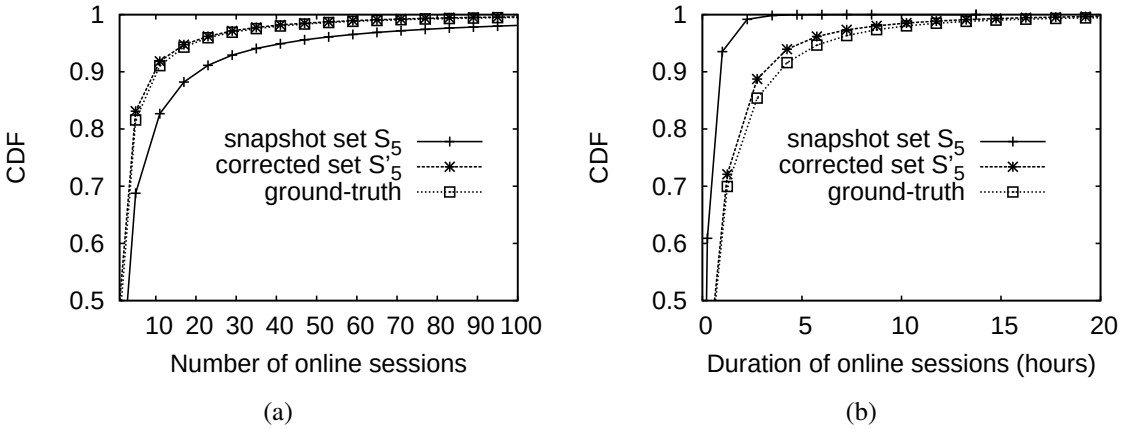


Figure 3.7: Analysis of users' recurrence (left) and session duration (right).

From a Chi-Square Goodness of Fit test, we observed that the users' recurrence in the system followed an exponential distribution. The rate parameter was $\lambda_{\mathcal{S}_5} = 9.97$ (for $\mathcal{S}_5$), $\lambda_{\mathcal{S}'_5} = 3.96$ ($\mathcal{S}'_5$), and $\lambda_{\mathcal{S}_{50}} = 4.37$ ($\mathcal{S}_{50}$). As for the duration of online sessions, it also followed an exponential distribution, with rates $\lambda_{\mathcal{S}_5} = 13.3$, $\lambda_{\mathcal{S}'_5} = 71$, and $\lambda_{\mathcal{S}_{50}} = 87.9$. Note that the rates for the corrected snapshot set $\mathcal{S}'_5$ and the ground-truth dataset are more similar, compared to the original snapshot set $\mathcal{S}_5$. In summary, these results provide important evidence that our methodology is able to correct traces with good accuracy and, more importantly, without deforming its main properties.

3.3 Assessing our hypothesis

The methodology presented in the previous section was used as a key component to help assessing the validity of our hypothesis. In addition to the traces of users' usage sessions obtained from Technische Universiteit Delf P2P Trace Archive (ZHANG et al., 2012), we also used traces collected with TorrentU (MANSILHA et al., 2011) and processed using the methodology described earlier. In the end, we obtained a vast set of traces of identity requests having diverse characteristics.

In order to understand the profile of identity requests patterns, in this thesis we explore three key concepts: *time of arrival of identity requests*, *time between arrivals of identity requests*, and *recurrence*. The former two are intuitive, being based on concepts largely explored in the queuing theory realm.

In the context of this thesis, we define *recurrence* as the number of online sessions observed for some specific user in the system. An online session, in turn, is defined as the period of time (of arbitrary duration) between a user arrival and departure in the system. Note that online sessions have finite duration by definition. It means that any user arrival in the system will be followed, after some arbitrary time, by a departure event from that same user. To illustrate, suppose that a user arrived four times during some period of observation. In this case, there were four online sessions; the recurrence for that user, in that period of observation, was four.

Based on the concepts above, and supported by an analysis of traces of users' usage sessions, we formulated the hypothesis defended in this thesis and, more importantly, assessed the potentialities of the research direction initially envisaged. In this section we discuss the results of the analysis carried out.

Table 3.4: Excerpt of the BitTorrent log file employed in the analysis.

1	2579139627397792460|3111354312011519843|2006-01-01 00:32:49|2006-01-02 18:18:49|0.5981862991410045|LEECHER|no|0.0
2	-1976250767827230296|-3161433101217960484|2006-01-01 00:33:01|2006-01-02 18:19:01|0.21703508372128355|LEECHER|no|0.0
3	-5948056174658895913|2855973049255676852|2006-01-01 00:37:00|2006-01-02 18:18:00|0.1325415990077356|LEECHER|no|0.0
4	-5276874893991588532|-2553693720321829928|2006-01-01 00:38:22|2006-01-02 18:19:22|0.11622309767807151|LEECHER|no|0.0
5	1985783098817489872|-2553693720321829928|2006-01-01 00:38:31|2006-01-02 18:19:31|0.11705653824459557|LEECHER|no|0.0
6	-5948056174658895913|7056542205054267382|2006-01-01 00:44:00|2006-01-02 18:18:00|0.20494416938186288|LEECHER|no|0.0
7	-8035607309117631139|3691911114549011734|2006-01-01 00:44:12|2006-01-02 22:54:51|0.16915404929047936|LEECHER|no|32.27539
8	1703653772333122919|-1891502691723097807|2006-01-01 00:46:30|2006-01-02 18:19:30|2.9296278965678595|LEECHER|no|0.0

3.3.1 Characterization of the considered traces

For the sake of clarity and legibility, in this thesis we focus on five of the traces we obtained both from public repositores and using the snapshot-based approach. Next we characterize these traces.

Traces from P2P Trace Archive

From the files of users' participation in torrent swarms analyzed we extracted traces that rebuild identity requests events in that swarm. The traces we considered were already anonymized, for the sake of privacy; as a consequence, no location information (e.g., IP address prefixes) was available. Each line in the log files considered (an excerpt is illustrated in Table 3.4) represents a session in the swarm, which is identified by the torrent ID (1^{st} field of the log line, anonymized). The log also reports in each line, among other information, the user participating in the session (identified by a hash of its IP address, in the 2^{nd} field), and the session duration (the difference between the end and begin timestamps, which are located in the 3^{rd} and 4^{th} fields, respectively).

The methodology employed to extract the trace of identity requests from the information contained in this log is described as follows. First, observe that there may exist one or more overlapping sessions for a given user. These overlapping sessions may indicate that user participate in multiple swarms during the usage of his/her BitTorrent client. This is the case, for example, of lines 4 and 5 from the trace excerpt shown in Table 3.4. There is a high probability that these multiple, overlapping sessions (when associated to a same IP address) actually refer to a single BitTorrent client used to download (or upload) multiple files. This is so because BitTorrent clients generate locally an identity when they are started, and use this same id to join any swarm until the client is terminated. Based on this assumption, the resulting sequence of identity requests is obtained by merging the overlapping sessions associated to a same IP address. To illustrate, lines 4 and 5 in Table 3.4 refer to a single identity request event, performed on 2006-01-01 00:38:22. By employing this methodology, we processed and generated three distinct traces, whose relevant characteristics for the evaluation presented hereafter are shown in Table 3.5.

Note that the traces considered were already anonymized. Therefore, it was not possible to assess that they contained records of a Sybil attacks. For this reason, we assume that identity requests in these traces are presumably legitimate. It is important to emphasize that this is a conservative assumption, and therefore it helps us to design a solution that minimizes any side effects to legitimate users.

Table 3.5: Characteristics of part of the traces used in our analysis.

	Trace 1	Trace 2	Trace 3
First identity request date/time	Sun Jan 1 00:00:47 2006	Wed Feb 1 00:00:00 2006	Wed Mar 1 00:01:34 2006
Last identity request date/time	Sat Jan 7 20:53:09 2006	Mon Feb 6 14:48:08 2006	Tue Mar 7 23:59:49 2006
Total number of identity requests	203,060	738,587	545,134
Total number of sources of identity requests	44,066	50,512	47,968
Trace duration (hours)	164.87	134.80	167.97
Average number of identity requests per minute	20.52	91.31	54.09

Traces collected using TorrentU

TorrentU was employed to capture two traces of users' usage sessions from a BitTorrent file sharing community. Both traces were captured using an average of 50 crawlers. One of the traces (referred to as "Trace 4") was captured following the methodology of Hoãÿfeld et al. (HOÃŸFELD et al., 2011), described in the previous section. As a consequence, it does not require correction. The other trace (referred to as "Trace 5") focuses on larger BitTorrent swarms. Therefore, it was corrected using our snapshot correction methodology, using $\alpha = 0.85$; the measured failure probabilities were $\hat{p_{pos}} = 0.563326$ and $\hat{p_{nul}} = 0.579387$, and the correction length was 3. A brief description of the traces is provided in Table 3.6.

Table 3.6: Characteristics of part of the traces used in our analysis.

	Trace 4	Trace 5
First identity request date/time	Sat 15 Jun 15:00:00 2013	Wed 09 Oct 21:00:00 2013
Last identity request date/time	Sat 22 Jun 15:00:00 2013	Wed 16 Oct 21:00:00 2013
Total number of identity requests	3,315,363	7,426,316
Total number of sources of identity requests	1,320,074	2,194,519
Trace duration (hours)	168	168
Average number of identity requests per minute	328.905	736.737
Number of swarms monitored	60	5
Average number of users per swarm	870.4	5,010
Minimum and maximum number of online users (all swarms)	42,052 and 73,613	14,862 and 57,702
Average number of online users (all swarms)	52,220	25,050
Standard deviation of online users (all swarms)	7,000.794	6,040.359

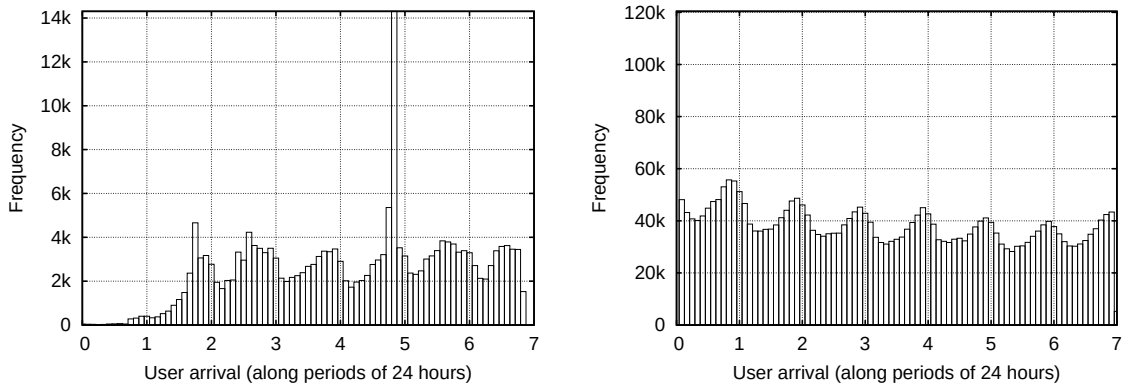
It is important to emphasize that ethical standards for measurement were considered when obtaining the traces used in this research. More specifically, the traces have been anonymized to make it impossible any sort of user identification and/or tracking. We also assume that these traces contain presumably legitimate activity (with regard to identity requests) only, since no evaluation of peer identity versus IP address of the user was carried out.

3.3.2 Analysis

We start our analysis recalling our hypothesis that "one can separate presumably legitimate identity requests from those potentially malicious by observing their source of origin and users' identity request patterns". Exploring this hypothesis requires the understanding of users' identity request behaviors in the context of large-scale distributed systems.

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 the traces available (ZHANG et al., 2012). However, the research community has not considered another important aspect from these traces, which is directly related to the aforementioned hypothesis mentioned: the pattern of users' participation in swarms. In the remainder of this section we concentrate our analysis on this aspect.

The roadmap of our characterization work basically consisted in evaluating three important aspects related to users' participation in swarms: their arrival in the swarm, time between arrivals, and frequency of arrivals (also referred to as "recurrence" in the remainder of this thesis). The goal was to obtain any useful insight that could be used to limit attackers in their process of obtaining counterfeit identities.

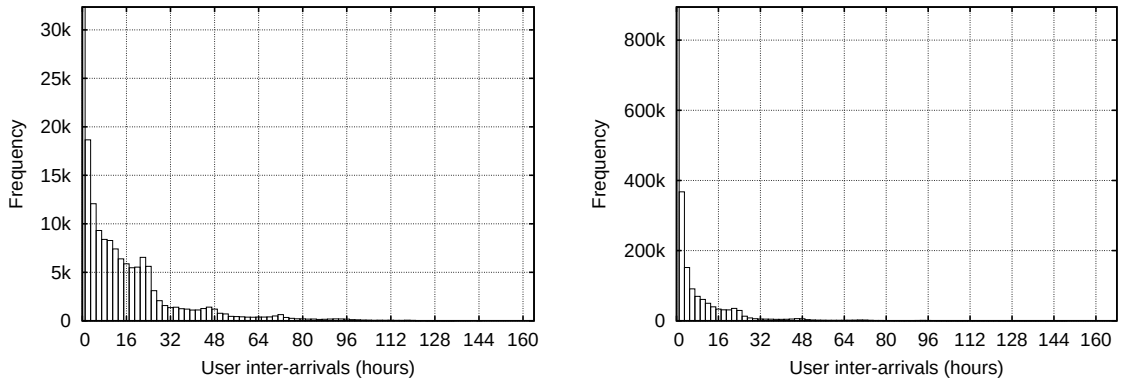


(a) Trace 1 (b) Trace 5

Figure 3.8: Distribution of users' arrivals during the period of the traces.

Our investigation began focusing on the users' arrival distribution, in the period of each of the traces considered. Figure 3.8 shows a histogram of the users' arrival distribution. In this histogram, each bin indicates the frequency of arrivals that occurred in a period of two hours. The bins between two consecutive numbers contain the frequency of arrivals that occurred in each day, between 00:00 and 23:59; for example, the bins between 0 and 1 capture the arrivals that occurred during the first day of the trace.

One can see in Figure 3.8(a) (Trace 1) that the BitTorrent community presented a consistent behavior over the week, apart from a few access peaks (for example by the end of the 5th day). A higher frequency of arrivals occurred during daytime, in comparison to past midnight. These observations also hold for Trace 5, illustrated in Figure 3.8(b). As one can see, there is no visual indication that the users' arrival in the system follows a particular distribution. In fact, we used the Chi-Square Goodness of Fit test to identify the distribution of users' arrival in the analyzed traces. However, the correspondent p-values calculated did not allow the acceptance of the null hypothesis for any of the tested distributions. In summary, no useful insight (for the purposes of the research reported in this thesis) could be drawn from the traces analyzed.



(a) Trace 1 (b) Trace 5

Figure 3.9: Distribution of the time between arrivals.

We moved along to an analysis of the time between arrivals of each user in the system. Recall that the information contained in the trace is limited, as it does not report the user identity, but the IP address used by the BitTorrent client (which was anonymized *a priori*). Therefore, we considered as an inter-arrival time (or time between arrivals) the period between consecutive arrivals of a same IP address. Note that an arrival of a same IP address may not necessarily represent the arrival of the same user, e.g. in case of networks using NAT, or due to Internet Service Providers (ISPs) having a limited number of IP addresses. In either case, this is a limitation that a candidate design for identity management should be robust against.

Figure 3.9 shows a histogram of users' inter-arrival distribution seen for Traces 1 and 5. One can visually see from Figure 3.9(a) that the distribution between inter-arrivals resembles an exponential one. In the case of Figure 3.9(b), it resembles a power-law. Regardless of the actual shape of the distribution, one can see that a large fraction of the arrivals occurred within a short interval from each other, with a very sparse number of inter-arrivals being extremely longer.

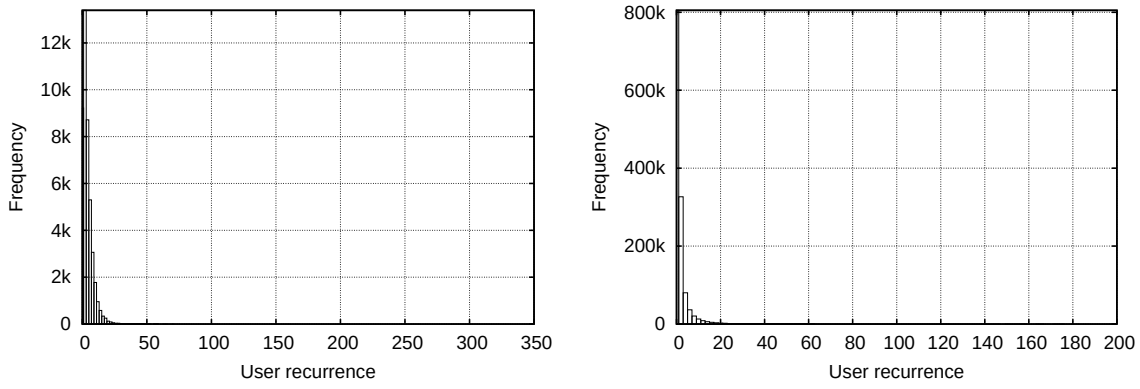
Table 3.7: Statistical summary of the users' inter-arrivals (in seconds), for each of the traces considered.

Traces	Minimum	1st decile	Mean	Standard Deviation	Median	9th decile	Maximum
Trace 1	1	2,820	56,220	66,601.04	34,830	139,336	590,900
Trace 2	1	581.0	16,030	39,690.62	662	53,614	480,300
Trace 3	1	591	22,360	47,874.41	1,656	72,116	588,800
Trace 4	1,800	3,600	26,830	50,443.8	8,100	72,900	604,800
Trace 5	3,600	4,500	20,940	42,589.56	9,000	45,900	603,900

Table 3.7 presents a statistical summary of the inter-arrival times, for each of the traces considered. One can see that the difference between the minimum and maximum inter-arrival times is very large for each of the traces. However, as one can confirm in Figure 3.9, a large fraction of the inter-arrival times was short; the medians (or 5th decile) for Trace 1 was 34,830 seconds; for Trace 2 it was only 662 seconds. The mean of inter-arrival times was also relatively short (when compared to the duration of the trace):

16,030 (≈4 hours) for Trace 2; and 56,220 seconds (≈16 hours) for Trace 1.

From the analysis briefly described above, one can see a strong indicative that a number of users left and re-joined the system within relatively short time intervals. Considering our conservative assumption that users' activities reported in these traces are legitimate (i.e., there was no ongoing Sybil attack), one important conclusion we can draw is that regardless of the design choice, it should enable users to rejoin the system with a certain frequency without being penalized.



(a) Trace 1 (b) Trace 5

Figure 3.10: Distribution of the users' recurrences.

The last aspect we focused in our analysis was the users' recurrence in the system. Because of the limited information available in the traces studied (as previously discussed in the analysis of inter-arrival times), we considered the frequency in which IP addresses were seen joining the swarm. Figure 3.10 shows the frequency of users' recurrence for Traces 1 and 5. Observe from Figure 3.10(a) that the vast majority of users joined the system very few times in the period of one week. Observe also that the distributions of recurrences have the shape of an exponential/power-law; in fact, this observation was confirmed for the several other traces analyzed in the context of this research.

Table 3.8: 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

Table 3.8 presents a statistical summary of the frequency of recurrences. One can see in this table that the range of recurrences is large in each of the traces. For example, the minimum and maximum recurrence observed for a single user in Trace 1 was 1 and 273. However, observe that at least 90% of users joined the system no more than 9 times. In the case of 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 much more frequently, some interesting patterns were observed. One of the users, for example, joined the system every 5 minutes on average. To illustrate, consider the plot shown in Figure 3.11. In this figure we show the arrivals of five users that had the highest recurrences in Trace 2, throughout the period of that trace. For the sake of comparison, we also included in this plot the arrival times of a user having recurrence equals to the median (28). One can clearly see that these users left and re-joined the swarm several times; in ≈99% of times the inter-arrival occurred within 5 minutes, for each of them.

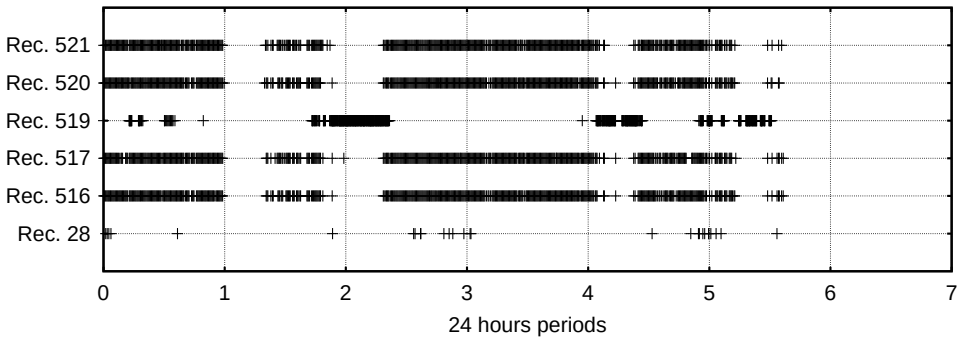


Figure 3.11: Arrival times of five users with highest recurrences, from Trace 2.

Although it is not possible to draw any conclusions about the nature of these users, their behavior is consistent with robots that monitor the network to collect statistical information, or to identify those users that disseminate copyrighted content illegally (MANSILHA et al., 2011). In spite of the users presenting anomalous behavior, an important conclusion that can be drawn from the analysis described above is that the majority of users tend to join the swarm in a relatively low frequency during a given period; this is a key observation we explore in our design for identity management.

3.4 Summary

The analysis of traces of identity requests represented a fundamental building block for the research reported in this thesis. A number of traces from BitTorrent file sharing communities, available from a public repository (ZHANG et al., 2012), were considered in this analysis. However, due to the lack of more recent traces available in that repository, we also considered collecting traces from a BitTorrent community, using an instance of the TorrentU framework (MANSILHA et al., 2011). The approach we used for the trace collection was a snapshot-based one, in which snapshots from the users currently online in the system are taken at regular time intervals. From these snapshots, we then re-created the dynamics of users' usage sessions in the system.

Because snapshots may fail to capture some online users, we also devised a probabilistic-based methodology for snapshot correction. Our methodology has shown to be effective, achieving a high degree of precision (ranging from 65% to 90% for the evaluated scenarios) in snapshot correction, with a low percentage of false positive corrections. More importantly, our methodology generated traces with consistent properties (e.g. users' recurrence and duration of online sessions) when compared to the ground-truth dataset. The importance of these results is underscored by the fact that capturing accurate traces requires a large fraction of resources (HOÃŸFELD et al., 2011; ZHANG

et al., 2011). In this context, our methodology represents a significant contribution to the research community, by enabling one to capture traces with similar degree of accuracy but using far less resources.

With regard to the analysis of the traces of identity requests (both from the public repository and also the ones we collected using TorrentU), it enabled us to understand the profile of users' access patterns in a representative subset of large-scale distributed systems. From a detailed analysis of the users' arrival, time between arrivals, and recurrence, we have observed that (*i*) a number of users left and re-joined the system within relatively short time intervals, with others presenting larger time between arrivals, and (*ii*) a large fraction of users appeared in the system in a relatively lower frequency, whereas a small proportion of them presented some anomalous arrival behavior.

The second observation in special allows us to conclude that the majority of users tend to access the system in a relatively low frequency during a given period. Assuming that attackers present a different behavior than legitimate users, requesting a significantly higher number of identities from a limited number of sources, keeping track of sources recurrence becomes then a promising approach to limit the dissemination of counterfeit identities. The findings described above serve as an important foundation for building our design for identity management, which will be presented in the following chapters.

4 CONCEPTUAL SOLUTION

The analysis of traces described in the previous chapter has revealed that a large fraction of users tend to join BitTorrent swarms less frequently, and that some users often re-join the system within shorter time intervals. The natural step in our research is to understand how these findings can be explored towards the design of a solution for limiting the spread of fake accounts in large-scale, distributed systems. In this thesis we describe our solution in an incremental fashion, by first introducing our conceptual design for identity management. In this chapter we also cover the possible strategies an attacker can use to subvert our solution, along with their trade-offs[1].

Organization. The remainder of this chapter is organized as follows. In Section 4.1 we present the conceptual design that forms the basis for our contributions to the state-of-the-art. In this section the concept of sources is described in detail, and also the protocol we consider for a generic identity management process. In Section 4.2 we discuss the possible strategies an attacker can use to subvert our solution for identity management and control the number of identities he desires. In Section 4.3 we enumerate a list of considerations on the conceptual design and attack model. Finally, in Section 4.4 we close the chapter with a summary.

4.1 Overview of the identity maintenance process

Our solution is built upon the notion that a user has to dedicate a fraction of resources (processing cycles, memory, storage, time, etc.) to obtain and renew identities in large-scale distributed systems. The major goal of our solution is to minimize the number of counterfeit identities that an attacker can obtain and control. The secondary goal is to reduce as much as possible the fraction of resources demanded to this end. In this context, our solution makes the following considerations about the target system:

- the amount of resources available to users, 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 a significant fraction of the identities in the 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 higher number of requests;

[1]This chapter is based on the following publication: Weverton Luis da Costa Cordeiro, Flávio Roberto Santos, Marinho Pilla Barcellos, Luciano Paschoal Gaspary. *Make it Green and Useful: Reshaping Puzzles for Identity Management in Large-scale Distributed Systems*. In: 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium.

- 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 his/her requests so that they appear to originate from various, distinct ones. For this reason, our solution requires that identities be periodically renewed, in order to make it even more prohibitive for an attacker to accumulate them.

Our solution is suitable for systems with various degrees of centralization (purely centralized, with a number of distributed servers, and structured networks). Examples include many P2P networks (e.g., closed BitTorrent communities) and other distributed computing systems (e.g., for collaborative intrusion detection). It is not suitable, however, for purely decentralized systems (e.g., P2P networks such as Freenet), where there is no central service providing any sort of coordination to participating users. In the remainder of this thesis we discuss how our proposed design exploits the characteristics enumerated above to limit the number of fake accounts an attacker can control.

To aid the presentation of our solution, 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 his/her identity. The definition of a source accommodates 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. In the remainder of this section we described in detail the concept of sources of identity requests (Section 4.1.1) and the protocol for identity management (Section 4.1.2).

4.1.1 Dissecting sources of identity requests

The concept of *source of identity requests*, fundamental building block upon which our design is built, is defined as an aggregation of one or more users (either legitimate, malicious, or both), located in a specific portion of the network, from which identity requests originate. Therefore, the exact meaning of "a *source* requests and obtains identities" is "user(s), from a certain *source*, request(s) and obtain(s) identities". Figure 4.1 (left) illustrates this concept, also highlighting the interactions between sources of identity requests and the bootstrap service (entity that assigns identities to users interested in joining some system).

Following a traditional process of identity creation/assignment, when a user becomes interested in joining the system, it issues an *identity request* to the bootstrap service (flow 1 in Figure 4.1). In the absence of a mechanism to control the assignment of peer identities, the bootstrap service replies to that request assigning an identity to the user (flow 2), so that he/she may use it to join the system (flow 3). In our solution, identity requests are associated to a source according to user location. In Figure 4.1, a source is represented as a "cloud", and the users associated to a given source are located within its respective cloud. Before granting an identity, the bootstrap service requires the user to accomplish some task (e.g., solve a puzzle), whose complexity depends on the behavior observed for that source (as discussed in the following section). To prevent tampering, the messages exchanged between the user and the bootstrap service, as well as granted identities, must be digitally signed by the bootstrap service (using its pair of public/private keys).

There are two general strategies that can be used (either combined or separately) to delineate the portion of the network regarded as a single source. The first one is viewing

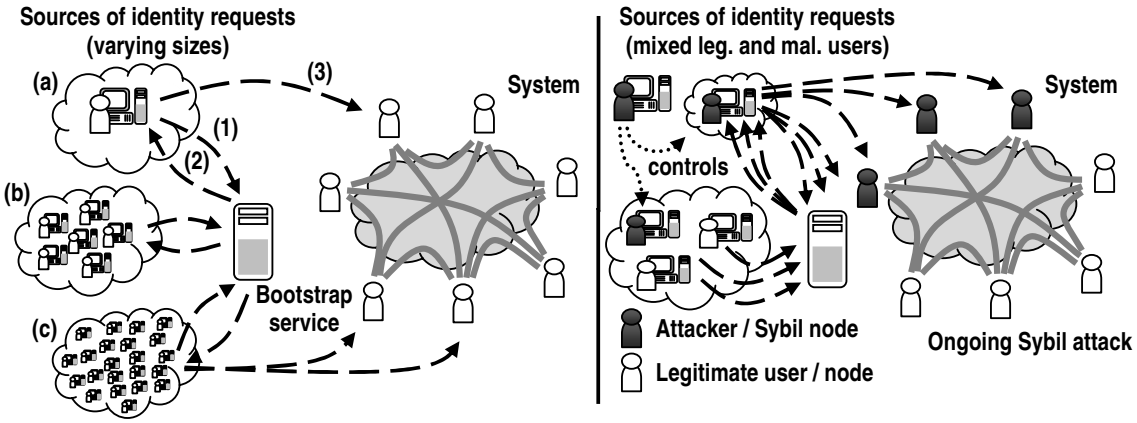


Figure 4.1: Characterization of sources of identity requests (left), and launch of a Sybil attack onto the system (right).

an IP address, a sub-network, or a combination of non-contiguous sub-networks as a single source. The second main strategy considers the use of a network coordinate system (e.g., Vivaldi (DABEK et al., 2004) and Veracity (SHERR; BLAZE; LOO, 2009)) for distinguishing requests coming from certain regions, cities, states or even countries. One could also combine both strategies to distinguish a large number of users behind NAT (or behind a small number of IP addresses); conversely, it may be useful to use network coordinates to identify a single attacker using a large number of IP addresses. The combination of both strategies is out of scope of this thesis, being envisaged as future research on the topic. However, we present in Chapter 6 an evaluation of the effectiveness of our solution against a coordinated attack originated from various sources, and also its impact to various users located in a same source.

The design decision of using IP addresses to materialize the concept of sources is a well established one, and was first used by Liang et al. (LIANG; NAOUMOV; ROSS, 2005). In that work, sources are defined as an aggregation of various users, located in specific IP ranges, who upload contents (multimedia files, documents, software, etc.) to peer-to-peer file sharing networks. Similarly, our concept of sources of identity requests could be materialized as an aggregation of various users, behind specific IP ranges, who request identities to the bootstrap service.

Observe also that sources may vary in *granularity* and *composition*, regardless of the strategy used to delineate them. Figure 4.1 (left) shows that the granularity (i.e., number of users associated to a source) may vary. Such a granularity is strongly influenced by the effectiveness of the strategy employed (along with their technical limitations) to delineate sources. It may range from a single user, e.g., a user's workstation (entity a), to a group of several users, e.g., a local network (entity b) or an entire autonomous system (entity c). Figure 4.1 (right) shows, in turn, that sources may be heterogeneous, i.e., may be shared between legitimate users and attackers. In fact, malicious requests may also originate from sources to which legitimate users are associated. Our solution performs satisfactorily regardless of the source granularity and composition, as shown in the analysis presented in Chapter 6.

4.1.2 Proposed identity life-cycle and supporting protocol

In our design, we assume that identity I contains the following information: $I = \langle i, t, v, e, \theta, s \rangle$. 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 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 Chapter 6).

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 chapter). 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 Figure 4.2. The conditions that trigger each of the transitions depicted in this figure are described in the following paragraphs.

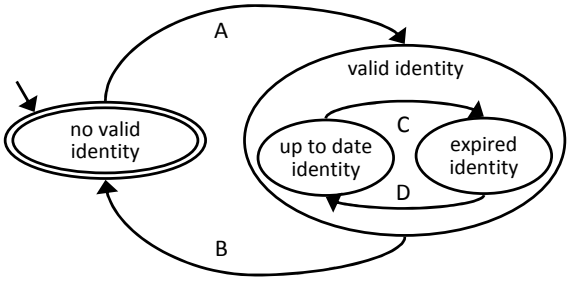


Figure 4.2: Possible states in an identity lifecycle.

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 $I\langle e \rangle \leftarrow T + E$.

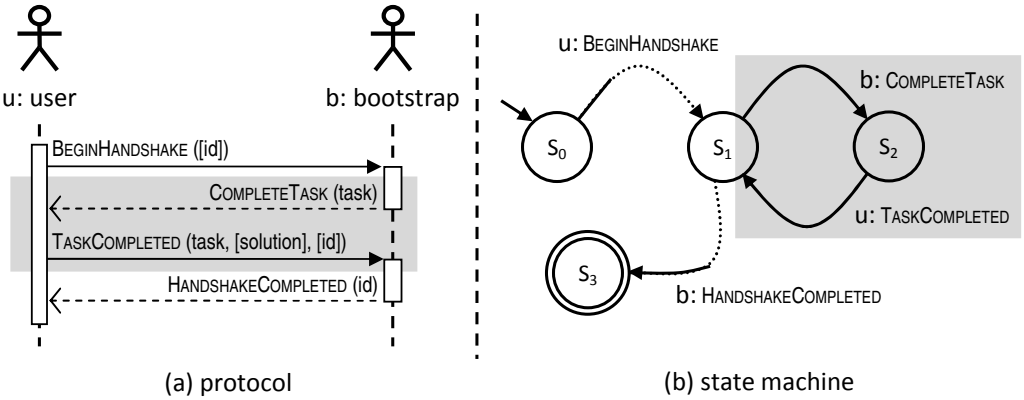


Figure 4.3: Proposed extension for an identity management protocol (left), and its respective state machine (right).

Figure 4.3(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. It is out of scope of this thesis, however, to

specify which security techniques or cryptographic algorithms should be used to this end. Figure 4.3(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, generic identity management scheme.

The initial state of a user is "no valid identity" (either because the user had never joined the system, or his/her identity is not valid anymore). Transition A (arrow A in Figure 4.2) takes place when the user contacts the bootstrap service to request a new identity. This transition takes place when the user issues a BEGINHANDSHAKE message (first arrow in the sequence diagram of Figure 4.3(a)), without parameters. The bootstrap replies with 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 (for example, solving some puzzle). Later in Chapter 6 we discuss the type of tasks we considered in the scope of this thesis.

After completing the task, the user 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/she has a valid, unexpired identity. It is important to observe from Figure 4.3(b) that the protocol enables as many iterations COMPLETETASK, TASKCOMPLETED as the bootstrap see fit. In Chapter 6 we describe how this aspect is explored in our solution for identity management.

Depending on the system nature, users might be required to renew their identities periodically, so as to keep contacting each other. We envisage a similar process for identity renewal, following the same protocol depicted in Figure 4.3(a). The difference in this case is that the messages exchanged between user and the bootstrap entity now contain the identity to be renewed (parameter *id*). Observe that the bootstrap entity is the sole responsible for assigning tasks to users requesting identities, and validating the task solution returned in response. This responsibility could be decentralized to some degree; however, it is important that the entities in charge be trusted (to prevent tampering). Observe also that task assignment should occur every time some user becomes interested in obtaining or renewing identities.

In case the user does not renew its identity and $I\langle e\rangle$ is passed, transition C (Figure 4.2) 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 he/she had never been in the system.

4.2 Attack model

We considered in our research that the attacker will dedicate as much resources as he possesses to subvert our solution and obtain as much fake accounts as possible. The evaluation of our solution is then centered on the number of identities the attacker is able to control over time, given an amount of resources in his/her hands.

Considering the conceptual solution described in the previous section, there are three basic strategies an attacker can use to obtain fake accounts. The first strategy consists of

using his/her own workstation to request identities one by one (Section 4.2.1). The second strategy, in turn, comprises hiring some service for processing the tasks assigned by the bootstrap entity (Section 4.2.2). Finally, the third strategy encompasses the distribution of identity requests to zombie workstations around the globe (Section 4.2.3). Each of these strategies are described in detail next.

4.2.1 Obtaining fake accounts using a single workstation

The use of a single station is a trivial strategy the attacker can use to obtain fake accounts. In the context of our solution, it corresponds to using a single source. This strategy can be easily implemented and requires a minimal amount of resources. However, it is only effective if the identity management in place does not impose any requirement, such as solving a puzzle. Figure 4.4 illustrates the basic idea behind this strategy.

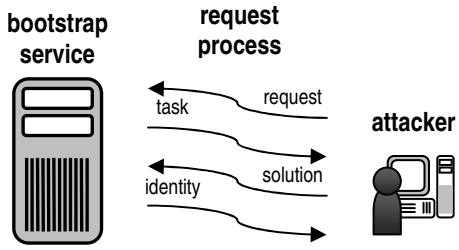


Figure 4.4: Attacker using a single workstation to obtain fake accounts.

Although being the cheapest one, observe that the use of a single source with just a single entity processing the tasks assigned by the bootstrap entity poses a constraint in the rate in which fake accounts can be obtained. In the following sections we evaluate the performance of this strategy in terms of measured trust score of the source, and number of fake accounts obtained over time.

4.2.2 Combining a single workstation with outsourcing of task processing

This strategy represents a more elaborated attack one can launch against our solution. In this strategy, only one source is used to contact the bootstrap entity and obtain fake accounts. However, the processing of the assigned tasks is fully outsourced. There are a number of possibilities to materialize the outsourcing, which also depends on the nature of the task. Figure 4.5 provides a simple illustration of the idea behind this strategy.

Suppose that the tasks assigned by the bootstrap entity are cryptographic puzzles to be solved. In this case, the attacker can use a cluster of high-performance computers to solve them. The higher the number of CPUs in this cluster, the more puzzles the attacker will be able to solve in parallel. Another possibility is hiring a cloud-based processing service; Amazon Cloud (AMAZON.COM, 2012) offers a pay-as-you-go service for as low as US$ 0.06/hour for *Standard On-Demand Instances*, and US$ 0.145/hour (Medium) and US$ 0.580/hour (Extra Large) for *High-CPU On-Demand Instances* service, all of these in a non-commitment basis. Cheaper prices can be obtained for 1-year and 2-year commitment contracts.

In case CAPTCHAs are used, the attacker can hire a CAPTCHA-solving service for as low as US$ 0.5/1,000 (MOTOYAMA et al., 2010). There are various providers that offer such services, which vary on quality (proportion of CAPTCHAs accurately solved), speed, and price. Some providers offer this service for as much as US$ 20/1,000, being able to solve multilingual CAPTCHAs with high precision and speed.

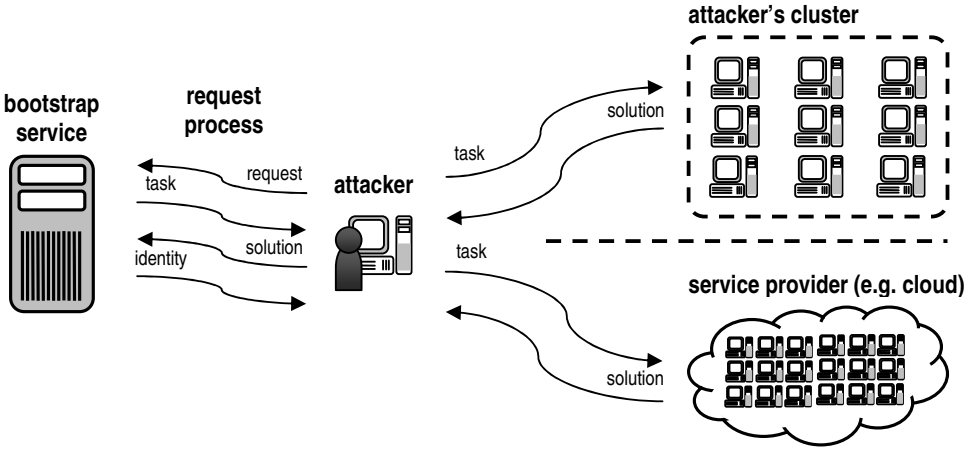


Figure 4.5: Outsourcing the processing of tasks assigned by the bootstrap entity.

The major advantage of this attack strategy is the increased power available for processing tasks (which comes at a cost, one should note). However, simply outsourcing puzzle resolution may also become ineffective. Recall that our scheme relies on measurement of trust score values for each of the sources from which identity requests depart (this aspect will be discussed in detail in the following chapter). Therefore, with a multitude of requests departing from a single source, its value of trust score will become significantly low. Such a poor value of trust score ultimately results in more complex tasks being assigned to future identity requests, therefore undermining the extra power the attacker dedicates for obtaining fake accounts.

4.2.3 Using a fully-fledged botnet to distribute identity requests

This strategy, illustrated in Figure 4.6, is the most effective one against our solution. One possible deployment is to outsource the identity request (and processing of tasks) to a botnet. This service is available in the black market, and research shows that the prices vary depending on the usage purpose and number of infected computers (PRINCE, 2013). Prices can be as high as US$ 535 for a five hours rent. The attacker can also contract consulting services to build his/her own botnet; prices vary between US$ 350 and US$ 400 per botnet. The consulting services can also be charged depending on the size of the desired botnet, with prices around US$ 500 per 1,000 zombie computers. Building a botnet having zombie workstations from North America, European Union, and Australia is offered a premium service; it is more expensive than building a botnet having zombie computers from regions such as Asia and Eastern Europe only.

Quite often the infected computers have low processing power available (either because they have users' processes running, or because they have several other malware installed and concurrently consuming the CPU power). In this case, the attacker can outsource only the identity request process to the botnet; the processing of tasks could be outsourced considering the strategy described in the previous section. This aspect results in a higher deployment cost for the attack, but has the advantage of increasing the attacker's power of quickly obtaining fake accounts.

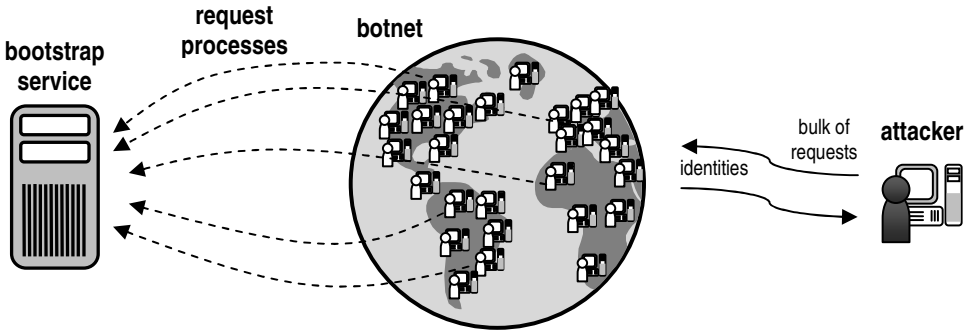


Figure 4.6: Using a botnet to obtain fake accounts.

4.3 Considerations

In this section we present some considerations regarding the conceptual solution and attack model discussed in this chapter. We organize this discussion following the same structure in which this chapter was organized.

Conceptual solution for identity management. It is important to highlight that the deployment of our conceptual solution in large-scale distributed systems is expected to require minimal changes to entities that compose the network. Considering for example its instantiation in a peer-to-peer system such as BitTorrent (COHEN, 2003), we envisage the need to slightly modify user agents and trackers so as to include the following interactions: (*i*) agent requests an identity to the *tracker*; (*ii*) *tracker* demands from the requesting agent the completion of a task; (*iii*) agent replies to the *tracker* with the solution to the proposed task; (*iv*) *tracker* checks the validity of the solution and, if correct, issues the identity to the agent. From this moment on, user agent and *tracker* interact using the in-place BitTorrent protocol. In this scenario, the most significant modifications would be restricted to the *tracker* – in such a way that it keeps information about sources' recurrence rates, compute their respective trust scores, and defines the complexity of the task to be accomplished as a function of the current trust score.

With regard to sources of identity requests, note that our design relies on its concept, not on the strategies that could materialize it – in part due to the limitations of existing ones. It is to overcome these limitations that we derive a trust score for each source found in the network, and adjust the cost per request based on that score (which varies upwards and downwards dynamically, one should observe). In other words, we rely on the sources trust score because sources themselves cannot be reliably tracked and may change over time. As research in the field of network geo-location evolves, our design would easily accommodate and benefit from achieved advances.

Now focusing on verification of identities, observe that our scheme does not require strict clock synchronization between participating entities. However, users must have their clock adjusted so that they can assess the validity of identities. Note that this identity verification is similar to the verification of certificates for those websites using secure HTTP. It is also important to mention that we attempted to come up with a protocol design that minimizes any changes to existing identity management schemes. Considering an actual deployment of our proposal as a protocol extension, the exchange of messages COMPLETETASK and TASKCOMPLETED should occur before the exchange of messages that implement the actual identity assignment.

Attack model. The discussion of the required budget to launch an attack against our solution also involves another important aspect: the profit of the attacker with each iden-

tity he obtains. As we will show in the following chapters, the attacker has a certain monetary cost per identity he obtains (which varies according to the scenario). However, one cannot currently compare these costs with the monetary profit the attacker can obtain with each identity. In this context, we argue that our major contribution is raising significantly the costs associated to deploying such an attack, making it less profitable or even not interesting from the attacker's point of view. More importantly, we argue (based on the evaluation to be presented in the following chapters) that our solution increases significantly the cost per counterfeit identity in comparison to existing proposals.

4.4 Summary

In this chapter we discussed two important concepts related to our solution for identity management in large-scale, distributed systems: the conceptual solution that forms the basis of our design for identity management, and the possible strategies an attacker can use to subvert our solution.

As part of our design, in this chapter we also introduced a conceptual solution to enable the use of adaptive puzzles as a controlling factor to Sybils. We discussed the lifecycle we envisage for identities and the protocol to support the operations of identity creation and renewal. The idea is that a bootstrap entity will keep track of the number of identities assigned to a given source during a specific interval, and define the price to be paid for that request taking into account the behavior of other sources in that same period.

Observe that our solution does not make attacks impossible. Instead, it increases the price an attacker has to pay in order to obtain his/her fake accounts. In this chapter we discussed the three basic strategies the attacker can use towards this end: using a single station to obtain fake accounts, outsourcing task processing, and using a botnet for identity requests. Each strategy has its advantages and drawbacks. As we will see later in this thesis, each of them leads to a different performance in the process of obtaining fake accounts.

5 TRUST SCORES FOR IDENTITY MANAGEMENT

The research reported in this thesis is built on the notion that users have to dedicate a fraction of resources to obtain identities in large-scale distributed systems, as a strategy to protect these systems from the dissemination of fake accounts. As an important step towards dealing appropriately with presumably legitimate identity requests and those potentially malicious, in this chapter we propose the concept of trust score. *Trust score* is a reputation index that establishes the likeliness a given identity request is part of an ongoing attack attempting to create fake accounts[1]. With such an index, those approaches based on proof of work can balance the price (in terms of fraction of resources) per identity requested, by using the values of trust score to parameterize the complexity of the tasks assigned by the bootstrap entity. By doing so, these solutions can finally cause less burden to presumably legitimate users, and be even more severe with attackers.

The concept of trust score has been evaluated by means of a large set of experiments. We have used the traces studied in the scope of this thesis (described in Chapter 3) to reproduce the true dynamics of identity requests. In addition, attacks have been designed to analyze the robustness of the proposed concept. The results achieved show that our proposal is able to assign lower values of trust score to those requests that are likely to be malicious, whereas presumably legitimate ones are largely regarded as trustworthy.

Organization. The remainder of this chapter is organized as follows. In Section 5.1 we present the concept of trust score and the mathematical formulation supporting it. In Section 5.2 we describe our evaluation and major findings. In Section 5.3 we enumerate some considerations on the proposed concept. Finally, in Section 5.4 we close the chapter with a summary.

5.1 The proposed model

In the scope of this thesis, we define *trust score* (θ) as 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. The trust score index assumes values in the interval $(0, 1)$: on one extreme, values close to 1 denote a high trust on the legitimacy of (the) user(s) associated to the i*-th* source; on the other, values close to 0 indicate high distrust, i.e., a high probability that there exists an attacker behind that source, who is launching a Sybil attack.

The computation of the value of trust score for some identity request originated from

[1]This chapter is based on the following publication: Cordeiro, W., Santos, F., Mauch, G., Barcellos, M., Gaspary, L.: *Securing P2P Systems from Sybil Attacks through Adaptive Identity Management*. In: 7th International Conference on Network and Service Management (CNSM 2011), 2011, Paris, France. Mini-conference Proceedings, 2011.

a certain source involves solving three main sub-problems, namely: (*i*) characterize the behavior of sources (or the behavior of users associated to them); (*ii*) translating the observed behaviors into values of trust score; and (*iii*) deal with the dynamics of users' behavior in the calculation of the trust score. Each of these sub-problems is addressed in the following sections.

5.1.1 Employing recurrence metrics to characterize behaviors

In order to characterize the behavior of sources of identity requests, it is important to keep track of the number of identities already granted to the users associated to a given source. In the context of this work, this number is defined as $\phi_i(t)$ for the *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 the bootstrap entity granted to users associated to some specific source i, in the last Δt units of time. The latter corresponds to the average number of identities that sources have obtained from the bootstrap entity in the same period. Observe that the computation of these metrics is straightforward, as the bootstrap has direct access to information regarding identity requests.

The network recurrence metric $\Phi(t)$ is computed using the simple mean of the values of sources' recurrence, according to Equation 5.1. 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.

$$\Phi(t) = \begin{cases} 1 & \text{, if } n = 0 \\ \dfrac{1}{n} \times \displaystyle\sum_{i=1}^{n} \Delta\phi_i(t) & \text{, if } n \geq 1 \end{cases} \tag{5.1}$$

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

5.1.2 Calculating trust scores from observed behaviors

The Sybil attack is often characterized by the launch of a multitude of fake identity requests to the system bootstrap entity, so that the attacker controls a significantly large number of identities in the system. This behavior leads to the increase in the recurrence

observed for the source(s) associated to the attacker, as illustrated in Figure 4.1 right (in Section 4.1.1). Conversely, it is expected that the sources associated to legitimate users obtain fewer identities. Therefore, the underlying idea to filter out presumably legitimate identity requests from those potentially malicious is to compare the sources recurrence to the average recurrence of the network.

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. For example, $\rho(t) = -1$ means that, at instant t, $\Delta\phi_i(t)$ is 50% lower than $\Phi(t)$ (i.e., $\Phi(t)$ is 100% higher than $\Delta\phi_i(t)$). Likewise, a positive $\rho_i(t)$ expresses how higher is the recurrence of the i*-th* source. For instance, if $\rho_i(t) = 2$, then $\Delta\phi_i(t)$ is 200% higher than $\Phi(t)$. Equation 5.2 provides the value of $\rho_i(t)$.

$$\rho_i(t) = \begin{cases} 1 - \dfrac{\Phi(t)}{\Delta\phi_i(t)} & \text{, if } \Delta\phi_i(t) \leq \Phi(t) \\ \dfrac{\Delta\phi_i(t)}{\Phi(t)} - 1 & \text{, if } \Delta\phi_i(t) > \Phi(t) \end{cases} \tag{5.2}$$

The relationship between the recurrences of the source and the network ($\rho_i(t)$) is used to compute the trust score of the i*-th* source ($\theta_i(t)$). This score is calculated at instant t according to Equation 5.3, and assumes values in the interval $(0, 1)$. As discussed earlier in this chapter, values close to 1 denote a high trust on the legitimacy of (the) user(s) associated to the i*-th* source, whereas values close to 0 indicate high distrust.

In Equation 5.3, the constant 0.5 defines the mean trust score, which is incremented or decremented by 0.5 (therefore resulting in a trust score that ranges from 0 to 1) depending on $\rho_i(t)$. The function $\arctan(x)$ is used to map any possible value of $\rho_i(t)$, in the interval $(-\infty, +\infty)$, into a value in the interval $(-\frac{\pi}{2}, +\frac{\pi}{2})$. The term π in Equation 5.3 normalizes the result of the $\arctan$ function, to the interval $(-0.5, 0.5)$ (see **Property 1** next). The metric $\rho_i(t)$ is raised to the power of 3 in order to create an interval around $\rho_i(t) = 0$ for which the trust score varies minimally (see **Property 2**). Finally, the term $\rho_i(t)^3$ is weighted by the current recurrence of the network ($\Phi(t)$) to regulate the conservativeness of trust scores in face of the current network behavior (see **Property 3**).

$$\theta_i(t) = 0.5 - \frac{\arctan(\Phi(t) \times \rho_i(t)^3)}{\pi} \tag{5.3}$$

Figure 5.1 presents five different configurations that illustrate how the trust score obtained for the i*-th* source varies as a function of $\rho_i(t)$. In each of these configurations, the current recurrence of the network $\Phi(t)$ controls how conservative the concept of trust scores should be regarding the current perception of the recurrence of a source. Note that the values used in this plot are arbitrarily chosen for the sake of illustration; their meaning depends on the interval chosen. The plot reveals three important properties that Equation 5.3 holds. These are enumerated below.

Property 1. *The function that maps values of $\rho_i(t)$ into trust scores is asymptotic in 0 and 1; therefore, for $\rho_i(t) \to -\infty$ or $\rho_i(t) \to +\infty$, there is always a corresponding value of trust score.*

Property 2. *The trust score varies minimally for values of $\rho_i(t)$ close or equal to 0; this corresponds to a situation in which the i-th source behaves similarly or equals to the network average $\Phi(t)$.*

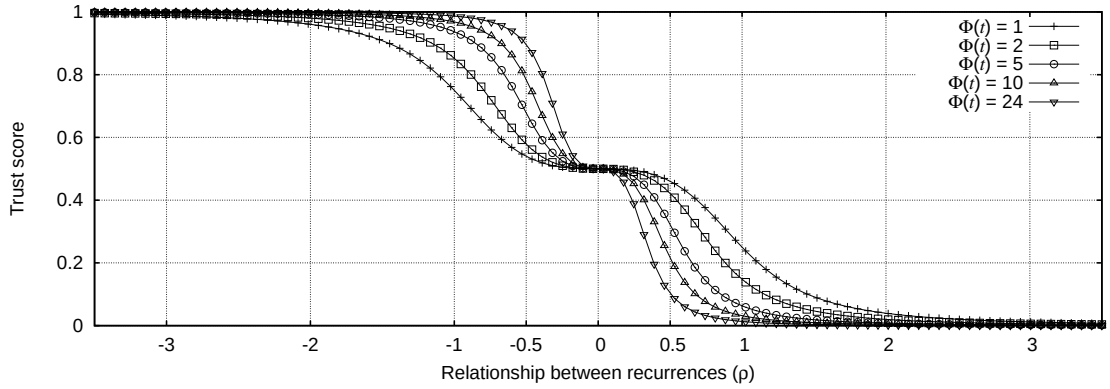


Figure 5.1: Curves of Equation 5.3 for the calculation of the source trust score, considering different values of network recurrence ($\Phi(t)$).

Property 3. *The computation of trust scores becomes more conservative as the current network average $\Phi(t)$ increases, and less conservative otherwise.*

Observe that **Property 2** allows a certain degree of tolerance in the evaluation of source behavior. To illustrate, consider the curve for $\Phi(t) = 1$ shown in Figure 5.1; variations of $\rho_i(t)$ lying in the interval $[-0.5, 0.5]$ indicate a source behavior similar to the pattern observed in the network, and thus have minimal impact. Behaviors that deviate significantly from this interval, however, will be assigned lower (or higher) trust scores.

Now focusing on **Property 3**, it enables weighting appropriately the value of $\rho_i(t)$ taking into account the current network behavior. To illustrate, consider the case of $\rho_i(t) = 0.5$ in Figure 5.1. Regardless of the current network recurrence, this value of $\rho_i(t)$ indicates that the i*-th* source has obtained 50% more identities than the average of all sources. Considering as an example the case of $\Phi(t) = 2$, the currently active sources in the network have requested 2 identities in the interval Δt on average, whereas the i*-th* source has obtained 50% more identities in the same interval, i.e., 3 identities; as a consequence, a trust score of approximately 0.4 is assigned to subsequent identity requests originating from this source. In the case of $\Phi(t) = 24$, however, the same value of $\rho_i(t) = 0.5$ means that the i*-th* source obtained 36 identities in the interval Δt (50% more than the average of 24 identities requested by the currently active sources). Although the proportion of identities obtained (in the interval Δt) in regard to the average of the network remained the same (50%), the surplus of identities obtained in the latter case ($36 - 24 = 12$ identities) was much higher than in the former one ($3 - 2 = 1$ identity only); as a consequence, the bootstrap entity becomes more conservative, and establishes a trust score of 0.1 to future identity requests originating from this source.

5.1.3 Dealing with the dynamics of users' behavior

Peer autonomy is an important characteristic of peer-to-peer networks and has several implications. Peers may arbitrarily join and leave the network at any moment, or become unavailable. In general, such observation holds for most online systems and large-scale distributed systems (such as Skype, Facebook, Digg, BitTorrent, among others). One possible effect of such dynamics is the variation in the behavior of both individual sources and the network as a whole. Next, we discuss how the concept of trust scores deals with the dynamics of observed behaviors.

The trust score provided by Equation 5.3 is instantaneous (at some time t). This

variable is limited in the sense it does accommodate fluctuations in the recurrence of a source (which could originate identity requests in bursts). Thus, a source with historically lower recurrence should be entitled to demand more identities for some time without being penalized. To accomplish this, a smoothing factor is used to properly represent the trust score of a given source in light of its past behavior.

The smoothed trust score, defined as $\theta'_i(t)$ for the i-*th* source in instant t, is calculated as shown in Equation 5.4. The smoothing factor β determines the weight of present behavior in the calculation of the smoothed trust score, assuming values in the interval $(0, 1]$. Thus, values of β close to 0 assign a high weight to the historical behavior of the source under consideration, and vice-versa. In the special case in which $\beta = 1$, the current trust score (as calculated through Equation 5.3) is fully considered, and the historical behavior, totally ignored. In Equation 5.4, $\theta'_i(t')$ refers to the last computed value of smoothed trust score.

$$\theta'_i(t) = \begin{cases} \theta_i(t) & \text{, if } \theta'_i(t) \text{ was never computed} \\ \beta \times \theta_i(t) + (1 - \beta) \times \theta'_i(t') & \text{, otherwise} \end{cases} \tag{5.4}$$

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

5.2 Evaluation

In order to evaluate the effectiveness of using the trust score model as a resource to measure the reputation of identity requests, we implemented a bootstrap entity for use in a simulation environment. In summary, the implemented entity aggregates the functionalities of management of identity requests from users interested in joining some system.

In this evaluation, we attempted to answer the first two research questions posed in the introduction: (i) how effective would a mechanism based on the hypothesis that *one can filter potentially malicious requests* be in detecting them? and (ii) what is the overhead that such a mechanism would cause to legitimate users? To this end, we carried out experiments considering scenarios with and without attack, using real-world traces of identity requests. In the remainder of this section we describe the details of the environment considered in our evaluation (Section 5.2.1), and our major findings (Sections 5.2.2 and 5.2.3).

It is important to emphasize that we only measure the values of trust scores of identity requests in this evaluation. In other words, our goal is to evaluate the concept of trust scores considering a variety of scenarios, with and without attack. The use of measured values of trust scores as a mean to actively hamper the dissemination of fake accounts is discussed in the following chapter.

Table 5.1: Summary of parameters involved in the evaluation of the concept of trust scores.

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

5.2.1 Simulation environment

To evaluate the concept of trust scores, we developed a discrete event simulator. In our simulator, the arrival of an identity request, and also each interaction between the bootstrap entity and the user that follows it, are regarded as events. The global clock is used mainly for event dispatch and to keep track of identity grants that should be gradually discarded (an effect of the sliding window). The simulator also keeps a queue of events, which is dynamically ordered considering the timestamp in which events will occur in the simulation. The global clock always evolves to the timestamp of the next event in the queue. The ending condition in our simulator is the timestamp, in the trace file, of the last arrival of a legitimate identity request.

The parameters involved in the evaluation, summarized in Table 5.1, are separated in two classes: *Environment-related parameters* characterize the large-scale distributed system under consideration; and *Input parameters*, those that may be adjusted to regulate the effectiveness of the concept of trust scores in properly detecting potentially malicious requests.

For the sake of this evaluation, we considered the set of traces described in Section 3.3.1. For assessing the resilience of the concept of trust scores in scenarios in which there is an ongoing attack, we focused on the first three traces described in that section. It is important to emphasize that we focus on these traces because they represent more challenging scenarios for the concept of trust scores. The other traces discussed in Section 3.3 (Traces 4 and 5) represent scenarios in which presumably legitimate users deviate significantly less from the average behavior of the network, and thus would be (i) assigned comparatively higher values of trust score, and (ii) easily isolated from malicious requests. Conversely, in the case of Traces 1, 2, and 3, there is a significant fraction of users who recur even more frequently to obtain identities than the sources in hands of the attacker.

In our evaluation, we consider a peer-to-peer network using a weak identity scheme. It means that users obtain identities upon joining the network. To analyze scenarios in which the system is under attack, we injected artificially generated malicious identity requests. In these scenarios, an attacker, provided with some amount of resources, launches his/her attacks in an attempt to obtain (and thus control) as much identities as possible. For the sake of this evaluation, we bound the number of fake accounts the attacker requests to 1/3

of legitimate identities. This number was chosen since it exceeds the proportion of fake accounts that any Sybil-resilient solution tolerates (YU et al., 2006, 2008).

Three scenarios were evaluated: without attack, with $M_u = 1\%$, and with $M_u = 10\%$ malicious sources. The second scenario ($M_u = 1\%$) corresponds to an attacker with limited resources to forge various distinct locations from which identity requests depart. In the third scenario ($M_u = 10\%$), the attacker has a botnet at his/her service and uses it to launch the attack. When doing so, the attacker is able to increase the speed in which he/she obtains identities in the system. Further, he/she may also alter the "perception of normality" in the network: a higher number of malicious sources behaving similarly in the network tend to change the perception of what is, effectively, the behavior of the majority of sources. Table 5.2 describes these attack scenarios in more detail.

Table 5.2: Characteristics of the scenarios evaluated for each trace.

	Trace 1			Trace 2			Trace 3		
M_r	104,606			380,484			280,826		
Interval between requests (sec)	5.67			1.27			2.15		
Attack scenarios	No attack	1%	10%	No attack	1%	10%	No attack	1%	10%
M_u	-	440	4,406	-	505	5,051	-	479	4,796
Requests per source (avg)	-	237.74	23.74	-	753.43	75.32	-	586.27	58.55
Interval between requests per source	-	41.6 min	6.94 h	-	10.7 min	1.78 h	-	17.2 min	2.86 h

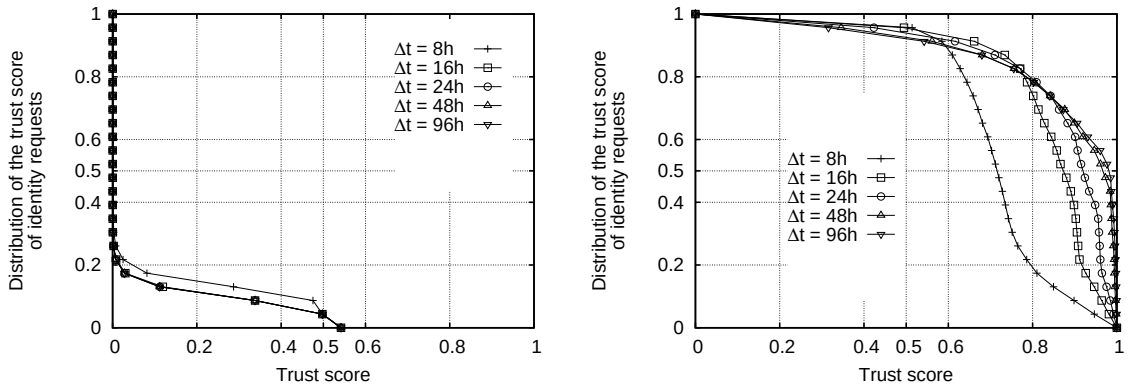
As a first step in our evaluation, we analyzed the sensitivity of trust scores considering various parameter setting (Section 5.2.2). Once we assessed a methodology to determine the most appropriated values for the input parameters, we analyzed the performance of trust scores considering the traces of identity requests (Section 5.2.3). Next we describe the results achieved.

5.2.2 Sensitivity analysis

In this section, we evaluate how well the concept of trust scores captures those potentially malicious identity requests considering various distinct parameter settings. More specifically, we evaluate the values of trust score assigned to identity requests considering different sizes for the sliding window, and various values for the smoothing factor. For the sake of clarity, we focus on the results achieved with Trace 1 (described in Section 3.3.1).

Sliding window Δt

Here we evaluate how the time interval Δt influences the performance of trust scores in detecting potentially malicious identity requests using various distinct locations (sources). For requesting fake accounts, the attacker uses 440 sources ($M_u = 440$, i.e. 1% of the number of legitimate sources; see Table 5.2), and uses them to originate his/her $M_r = 104,606$ requests; each of these sources originate around 237 requests only, one every 41 minutes approximately ($164.87 \cdot 60 \cdot \frac{440}{104,606}$). We consider such a request rate since it represents the best choice for the attacker – given the design of trust scores (we prove this aspect formally in the following chapter). Otherwise, some malicious requests would receive even lower trust scores, should the attacker choose a different balance of requests per source per unit of time (rather than evenly dividing requests among sources



(a) malicious (b) legitimate

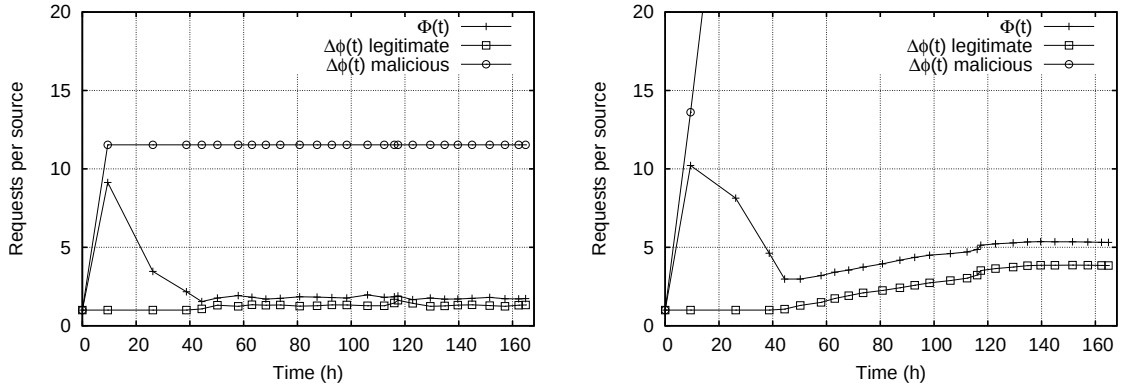
Figure 5.2: CCDF of the trust score of requests performed by legitimate users and attacker, using Trace 1 as basis, under different sizes for the sliding window Δt.

and throughout time). For this analysis, we consider the following values of Δt: 8, 16, 24, 48, and 96 hours. The smoothing factor is initially set to $\beta = 0.125$ (next we provide an analysis for this parameter).

Figure 5.2(a) shows that malicious requests were significantly affected, being assigned extremely low values of trust scores for a majority of them, regardless of the values for Δt considered. For example, using $\Delta t = 8$ hours only 20.63% of malicious requests were assigned a value of trust score higher or equal to 0.01; in other words, over than 79.37% of requests were assigned extremely poor values of trust scores. Only 13.38% of malicious requests received values of trust score higher or equal to 0.1, and less than 3.79% were assigned values of trust scores of 0.5 or higher. No malicious requests were assigned a score higher than 0.6. Such low trust scores may be explained by the recurrence of each malicious source, which is comparatively higher than the average recurrence of the network.

Observe also from Figure 5.2(a) that the higher the duration of the sliding window, the more restrictive the concept of trust scores becomes for the attacker. For example, an increasing of the sliding window from $\Delta t = 8$ to $\Delta t = 16$ decreases from 11.7% to 7.75% the proportion of requests that were assigned values of trust score of 0.5 or higher. The reason is that a larger duration for the time interval Δt makes a higher fraction of the history of sources to be considered when calculating their recurrences $\Delta\phi(t)$ (as discussed in Section 5.1.1). Such an increase in their recurrences has two effects. First, the average recurrence of the network $\Phi(t)$ also increases, and the bootstrap entity becomes more conservative when computing trust scores to those requests coming from sources having higher $\Delta\phi(t)$ (see **Property 3** in Section 5.1.2). Observe for example in Figure 5.3 that the average recurrence of the network (curve "$\Phi(t)$") changes substantially, when comparing the scenarios using $\Delta t = 8$ and $\Delta t = 96$.

The second effect is that the value of $\Delta\phi(t)$ for those sources in hands of the attacker, already high for the case of $\Delta t = 8$, becomes even higher when using $\Delta t = 96$. From Figure 5.3(a), observe that the average recurrence of malicious sources is approximately 11.6 identities per hour for $\Delta t = 8$ (curve "$\Delta\phi(t)$ malicious"); this number increases to approximately 138 per hour for $\Delta t = 96$ (curve partially omitted for the sake of legibility). As a consequence of such an increase in the recurrence of malicious sources, even lower trust scores are assigned to future requests originating from them. The main con-



(a) $\Delta t = 8$ (b) $\Delta t = 96$

Figure 5.3: Average number of requests per source overtime, considering different durations for the sliding window.

clusion we can derive from these results is that attacks for creating a bulk of fake accounts can be easily tracked and largely mitigated.

After evaluating the effectiveness of the concept of trust scores in detecting potentially malicious identity requests, we now focus on the measured values of trust scores for those requests originated from (presumably) legitimate sources. From the curves shown in Figure 5.2(b), it is clear that the measured values were overall significantly high in each of the scenarios, regardless of the interval Δt used. The majority of identity requests from legitimate sources received a value of trust score higher or equal to 0.5 (above 92% in all cases).

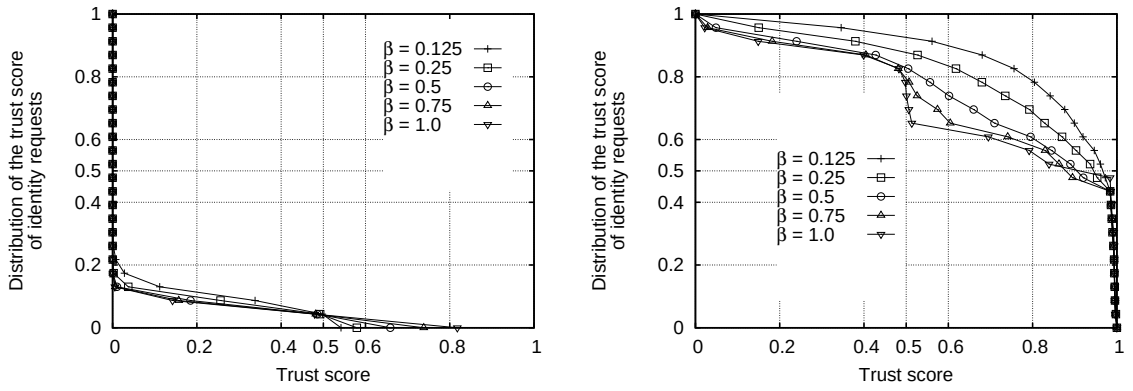
One can clearly see from Figure 5.2(b) that increasing the size of the sliding window improves substantially the trust score of legitimate requests. This is because the average recurrence of legitimate users, although increased because of higher sizes of sliding window, becomes comparatively lower than the average recurrence of the network. This aspect can be observed comparing the distance between curves "$\Phi(t)$" and "$\Delta\phi(t)$ legitimate" in Figures 5.3(a) and 5.3(b).

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

Smoothing factor β

Now we focus on the effect that varying values for the smoothing factor β causes to the identity requests of these sources. We concentrate on the following values of β: 0.125, 0.25, 0.5, 0.75, and 1. For this set of experiments, we use $\Delta t = 48$. The other parameters remain unchanged.

Figure 5.4 indicates the trust score achieved by requests launched to the bootstrap entity, for each of the considered values of smoothing factor. Focusing on Figure 5.4(a),



(a) malicious (b) legitimate

Figure 5.4: Trust score of requests performed by legitimate users and attacker, under different values for the smooting factor β.

observe that increased values of β makes a larger fraction of malicious requests be assigned lower values of trust score. For example, less than 3.8% of malicious requests received a value of trust score higher or equal to 0.5, when using a smoothing factor of $\beta = 0.125$; this proportion decreased to 2.2% when using a smoothing factor of $\beta = 1.0$. With regard to legitimate users, Figure 5.4(b) evidences that lower values for β decreases the overhead caused to legitimate requests. In spite of this, the proportion of requests assigned with a trust score higher or equal to 0.5 was above 76% in all scenarios, and above 92% in the particular case of $\beta = 0.125$.

In principle, the higher the value of β, the more restrictive the concept of trust scores becomes for both legitimate and malicious requests. To understand this, first recall (from Section 5.1.3) that higher values of β assign higher weight to the instantaneous trust score $\theta_i(t)$ (upon the calculation of the smoothed trust score $\theta'_i(t)$); conversely, lower values of β assign higher weight to the historical behavior of $\theta_i(t)$. Recall also that both $\Delta\phi_i(t)$ and $\Phi(t)$ may vary significantly through time, either because of newly granted identities or because of gradual disposals of older grants from the sliding window Δt, thus making the instantaneous trust score $\theta_i(t)$ to experience high fluctuations. Therefore, the higher values of β make the smoothed trust score to reflect even more the fluctuations seen in the instantaneous trust score, which ultimately results in lower trust scores (and thus puzzles of higher complexity) being eventually assigned to identity requests. This observation holds mainly for the case of the attacker, who has a limited set of resources and requests a proportionally higher number of identities. In contrast, lower values for β reduce this fluctuation of $\theta'_i(t)$. Therefore, identity requests are assigned with lower trust scores (and thus puzzles of higher complexity) only if the source from which they originate maintains the discrepant behavior of requesting a higher number of identities (in regard to the network) for a longer period.

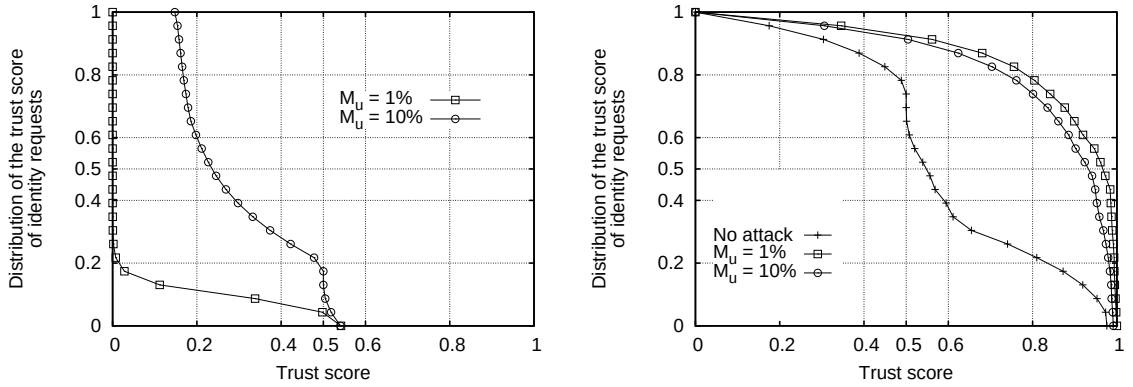
The definition of an appropriate value for β is extremely subjective, and basically represents a trade-off between tolerance with those presumably legitimate users (or sources of identity requests) who suddenly make a burst of requests, and conservativeness with those sources potentially involved with an ongoing attack. In other words, should the system maintainer decide in favor of tolerance to those presumably legitimate users facing transient network failures (and thus occasionally making a burst of identity requests), then using lower values for β is more convenient. Conversely, should the system main-

tainer decide to be rigorous with potential attackers (regardless of the negative impact to presumably legitimate users), then higher values for β become more appropriate.

Given the discussion above, we chose a value of $\beta = 0.125$ (along with $\Delta t = 48$) for the analyses that follows, as it makes the concept of trust scores more robust to sources that continuously request more identities than the network average, whereas allows a legitimate user to request more identities during a transient failure (e.g., unstable network connectivity) without being penalized for that.

5.2.3 Performance analysis

Having evaluated analyzed the effectiveness of the concept of trust scores under the influence of various parameter setting, we now focus on its performance in a real life scenario. In this section we consider the three attack scenarios described in Table 5.2. The parameters we adopt for the concept of trust scores are $\Delta t = 48$ and $\beta = 0.125$, values chosen according to the sensitivity analysis presented earlier.



(a) malicious (b) legitimate

Figure 5.5: Trust score of requests performed by legitimate users and attacker, for each of the scenarios evaluated with Trace 1.

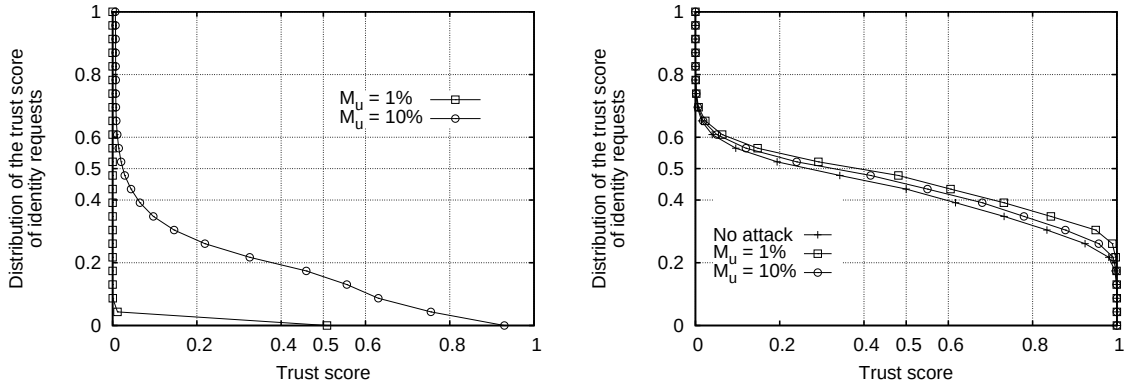
Figures 5.5, 5.6, and 5.7 show the results obtained for each of the studied scenarios, for Traces 1, 2, and 3, respectively. Recall that we concentrate on these traces since they represent a more challenging scenario for the concept of trust scores (since a large fraction of users recur even more frequently to obtain identities than the sources of the attacker).

Observe from Figure 5.5 that the bootstrap entity assigned significantly lower values of trust scores to those identity requests coming from malicious sources, whereas minimally penalizing presumably legitimate ones (in scenarios "$M_u = 1\%$" and "$M_u = 10\%$"). Even in the extreme scenario of "$M_u = 10\%$", over than 56% of legitimate requests were assigned a value of trust score higher or equal to 0.9; conversely, over than 80% of malicious requests were assigned a value of trust score lower or equal to 0.5. The effectiveness of the concept of trust scores in properly separating malicious requests from those presumably legitimate becomes even more evident in the scenario where the attacker is constrained in resources ("$M_u = 1\%$").

Observe also that the overhead caused to presumably legitimate requests is minimal. In the scenario where the attacker uses the most of resources to tamper with the concept of trust scores, less than 2% of those requests were assigned a value of trust score lower or equal to 0.1; less than 3% of them received a value lower or equal to 0.2, and less than 5% received a value lower or equal to 0.3.

Table 5.3: Statistical summary of the results obtained for Trace 1.

Scenarios		Min.	1st decile	Median	Mean	Std. Dev.	9th decile	Max.
no attack	Leg.	0.0	0.3332	0.5495	0.5928	0.2234766	0.9499	0.9762
	Mal.	–	–	–	–	–	–	–
$M_u = 1\%$	Leg.	0.0	0.6045	0.9671	0.8694	0.2040771	0.9959	1.0
	Mal.	0.0	0.0	0.0	0.05399	0.1386912	0.25206	0.542
$M_u = 10\%$	Leg.	0.0	0.54737	0.9319	0.8386	0.2096832	0.98730	0.9916
	Mal.	0.1477	0.1578	0.2423	0.2966	0.1376637	0.5024	0.5406



(a) malicious (b) legitimate

Figure 5.6: Trust score of requests performed by legitimate users and attacker, for each of the scenarios evaluated with Trace 2.

An important aspect to be discussed regarding Figure 5.5 is the percentage of approximately 25% of requests having trust score lower than 0.5. Even though the sources contained in the trace are presumably legitimate (i.e., have not launched a Sybil attack), there are cases in which sources may originate identity requests in a higher frequency than the network average. In spite of this, less than 5% of legitimate identity requests received values of trust scores lower or equal to 0.2.

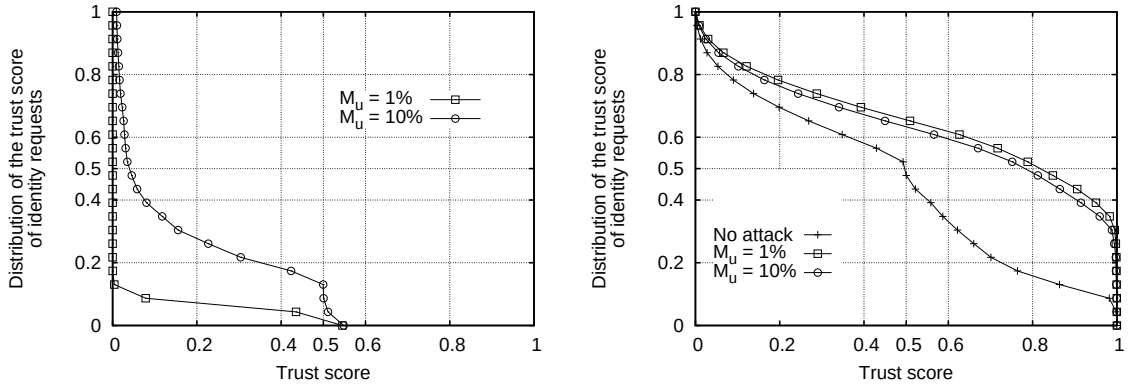
Another important discussion is related to the improvement in the trust score of presumably legitimate requests, in those scenarios with attack. This phenomenon is observed because malicious requests increase the average recurrence of the network, thus making legitimate ones to appear less suspicious. Therefore, as long as malicious requests do not depart from those sources shared by legitimate users, legitimate requests are not hampered by the occurrence of attacks. In the following chapter we present an analysis of scenarios where both legitimate users and attacker share sources of identity requests.

Table 5.3 presents an statistical summary of the results shown in Figure 5.5. Observe that the mean value of trust score assigned to malicious requests is significantly low (at most 0.2966, in the scenario where the attacker dedicates more resources). In contrast, the mean value of trust score assigned to requests of presumably legitimate users was significantly higher (at least 0.5928, in the scenario without attack).

The results achieved for Traces 2 and 3, depicted in Figures 5.6 and 5.7 (and summarized in Tables 5.4 and 5.5), confirm the positive results achieved. The attacker was

Table 5.4: Statistical summary of the results obtained for Trace 2.

Scenarios		Min.	1st decile	Median	Mean	Std. Dev.	9th decile	Max.
no attack	Leg.	0.0	0.0	0.2635	0.4248	0.4262424	0.9994	0.9999
	Mal.	–	–	–	–	–	–	–
$M_u = 1\%$	Leg.	0.0	0.0	0.3867	0.4616	0.4377551	0.9999	1.0000
	Mal.	0.0	0.0	0.0	0.007776	0.05034481	0.0001	0.508600
$M_u = 10\%$	Leg.	0.0	0.0	0.3250	0.4430	0.4312081	0.9997	1.0
	Mal.	0.0064	0.0071	0.0245	0.1690	0.2495451	0.59920	0.9296



(a) malicious (b) legitimate

Figure 5.7: Trust score of requests performed by legitimate users and attacker, for each of the scenarios evaluated with Trace 3.

severely penalized in both scenarios, with approximately 73% of malicious requests receiving a trust score lower or equal to 0.2 for Trace 2, and 72% for Trace 3. The mean value of trust score the attacker was able to achieved was also significantly low: at most 0.169 for Trace 2, and 0.1494 for Trace 3.

Now focusing on the overhead to presumably legitimate requests, again it was kept to a minimum. In the scenario without attack, over than 43% of requests received a value of trust score higher or equal to 0.5 for Trace 2, and over than 51% for Trace 3. In the presence of attack, the overhead decreases even further. Focusing on scenario "$M_u = 10\%$", approximately 30% of requests were assigned a value of trust score higher or equal to 0.9 for Trace 2, and approximately 40% for Trace 3.

The results achieved with these traces provide important evidence to answer the research questions enumerated earlier in this section. First, the use of trust scores was effective in filtering malicious requests from presumably legitimate ones: malicious requests were penalized with significantly lower values of trust score, whereas presumably legitimate ones were assigned significantly higher values. The results also show that the attacker needs to dedicate a large amount of resources, in terms of distributed sources, to outsmart the scheme of classification and aggregation of requests per source of origin. Second, the use of trust scores has minimally penalized legitimate users in those scenarios without attack, with a small fraction of them receiving lower values of trust score.

Two additional, important conclusions can also be drawn from the results presented

Table 5.5: Statistical summary of the results obtained for Trace 3.

Scenarios		Min.	1st decile	Median	Mean	Std. Dev.	9th decile	Max.
no attack	Leg.	0.0	0.01570	0.5000	0.4486	0.319038	0.95480	0.9999
	Mal.	–	–	–	–	–	–	–
$M_u = 1\%$	Leg.	0.0	0.0389	0.8199	0.6490	0.3783124	0.9997	1.0
	Mal.	0.0	0.0	0.0	0.03273	0.111368	0.0377	0.54540
$M_u = 10\%$	Leg.	0.0	0.0319	0.7834	0.6263	0.3817145	0.9992	0.9999
	Mal.	0.0095	0.0116	0.0409	0.1494	0.1840468	0.5001	0.5484

earlier. The first is that the concept of trust scores reacts adequately to increases in the sources recurrences, severely penalizing those sources that request identities in a frequency higher than the network average. Recall for example that in scenario "$M_u = 10\%$" for Trace 2, each source requests approximately 23 identities; this number increases to approximately 237 identities in scenario "$M_u = 1\%$" (see Table 5.2). The second conclusion is that the concept of trust scores forces sources to "behave adequately" – i.e., make use of the bootstrap entity harmonically in comparison to other sources – in order not to be penalized with lower values of trust score for each identity requested.

5.3 Considerations

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

Now focusing on the scalability of an actual deployment, observe that multiple bootstrap entities could be instantiated to distribute the load of identity requests. To this end, we envisage the same strategies proposed by Rowaihy et al. (ROWAIHY et al., 2007) to instantiate multiple certification entities in their mechanism based on static puzzles. For example, the bootstrap entity could be replicated (holding the same public/private key pair) across multiple hosts. In this scheme, DNS redirection could be used to balance the load of identity requests among these hosts. Alternatively, a *master certification authority* could be designed to certificate multiple bootstrap entities. In this scenario, a technique based on two random choices (MITZENMACHER; RICHA; SITARAMAN, 2001) could be used to balance the load of identity requests among them.

5.4 Summary

The Sybil attack consists on the indiscriminate creation of counterfeit identities by an attacker. An effective approach to tackle this attack consists in establishing computational puzzles to be solved prior to granting new identities. Solutions based on this approach have the potential to slow down the assignment of identities to malicious users, but unfortunately may affect normal users as well. Assuming computational puzzles of similar complexity, attackers having access to high performance computing hardware might be able to solve them orders of magnitude faster than legitimate users. Consequently, attackers may obtain a larger number of identities. However, simply increasing the complexity of puzzles would hamper the admission of legitimate peers to the network.

To address this problem, in this chapter we proposed the concept of trust scores as an approach to price identity requests. The key idea behind this concept is to estimate a reputation index for each identity request, calculated as a proportion of the number of identities already granted to (the) user(s) associated to the source from which the request departed, in regard to the average of identities granted to users associated to other sources. The higher the frequency (the) user(s) associated to a source obtain(s) identities, the lower the trust score of that source. The measured trust score can then be used to estimate the price (e.g., in terms of computing resources) to be paid before granting a request.

The experiments carried out showed that potential attackers, who launched an indiscriminate number of identity requests, were assigned significantly lower values of trust scores. Conversely, legitimate users were in general minimally penalized, having received higher values of trust score. When computing lower trust scores to sources having higher recurrences, (malicious) users associated to these sources will have to cope with higher prices per request. In contrast, users associated to presumably legitimate sources (and that made fewer use of the bootstrap service to request new identities) will pay less per identity (given the higher values of trust scores determined for the great majority of these sources in the system).

It is important to emphasize that our concept can be used with any model to price identity requests. In the scope of the research described in this thesis, we have concentrated on a proof of work strategy, and adopted computational puzzles to this end. In the following chapter we discuss how we have combined the concept of trust scores with computational puzzles in order to come up with a solution that assigns puzzles of adaptive complexity to users requesting identities.

6 PRICING REQUESTS WITH 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. In this chapter, we discuss how we can take advantage of the concept of trust score and combine it with proof of work strategies as a strategy to limit the spread of Sybils[1]. In contrast to existing proof of work approaches, namely computational puzzles of fixed complexity, we take advantage of the perceived reputation of identity requests (computed considering the trust score model described in the previous chapter) to adaptively determine the complexity of the puzzles users must solve.

There are two other important issues related to computational puzzles that we also address in this chapter. First, puzzle-solving incur considerable energy consumption, which increases proportionally to the system popularity and the interest of attackers in controlling counterfeit identities. Second, users waste computing resources when solving puzzles (i.e., no useful information is actually processed during puzzle resolution). To tackle these issues, we build on adaptive puzzles and combine them with waiting time to introduce a *green* design for lightweight, long-term identity management. Our design minimally penalizes presumably legitimate users with easier-to-solve puzzles, and 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 philosophy of the ReCAPTCHA project, which aims to keep robots away from websites and helps digitizing books (AHN et al., 2008).

To assess the effectiveness of adaptive puzzles, we carried out an extensive evaluation analytically, by means of simulation, and experiments using PlanetLab. We considered various parameter and environment settings, and scenarios with and without attack, using synthetic and real traces of identity requests. An in-depth analysis of the results achieved evidenced that potential attackers have to face comparatively more complex puzzles, and thus the rate of identity assignment to malicious users is substantially reduced. More

[1]This chapter is based on the following publications:

- Weverton Luis da Costa Cordeiro, Flávio Roberto Santos, Gustavo Huff Mauch, Marinho Pilla Barcellos, Luciano Paschoal Gaspary. Identity Management based on Adaptive Puzzles to Protect P2P Systems from Sybil Attacks. *Elsevier Computer Networks (COMNET)*, 2012.
- Weverton Luis da Costa Cordeiro, Flávio Roberto Santos, Marinho Pilla Barcellos, Luciano Paschoal Gaspary. *Make it Green and Useful: Reshaping Puzzles for Identity Management in Large-scale Distributed Systems*. In: 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium.

importantly, the results show that the overall energy consumption is comparatively lower than in existing puzzle-based identity management schemes.

Organization. The remainder of this chapter is organized as follows. In Section 6.1 we introduce the notion of adaptive puzzles, and discuss how the trust score model is used to materialize it. In Section 6.2 we present our approach for making adaptive puzzles green and useful. We present the results achieved with simulation and experimentation using PlanetLab in Section 6.3, and enumerate a list of considerations on the proposed concept in Section 6.5. In Section 6.4 we evaluate analytically the robustness of adaptive puzzles against tampering by an attacker, and present the results of an analysis of our solution scaled to millions of users. Finally, in Section 6.6 we close the chapter with a summary.

6.1 From trust scores to adaptive puzzles

The idea behind the use of computational puzzles is that legitimate users are able to prove their good intentions with the system, by compromising a fraction of their resources. In contrast, attackers interested in creating multiple identities are forced to spend a large fraction of their time processing puzzles and, therefore, consuming resources. This reduces their power to assume large number of identities.

Puzzles have been used for a long time for identity management in large-scale distributed systems (CASTRO et al., 2002; BORISOV, 2006; ROWAIHY et al., 2007). However, existing proposals do not distinguish between requests from legitimate users and those originated by attackers. Since both are subject to the payment of the same (computational) price for each requested identity, these proposals may lose effectiveness when the computational resources of attackers outweigh those of legitimate users. If computational puzzles are of similar complexity, an attacker with access to high performance computing hardware might be able to solve them orders of magnitude faster than legitimate users. Consequently, an attacker may obtain a larger number of identities. However, simply increasing the complexity of puzzles would hamper the admission of legitimate users to the system.

In this thesis, we use the trust score calculated for a source of identity requests to determine the complexity of the puzzles to be solved by users associated to that source. The mapping between trust and puzzle complexity is given by an abstract function $\gamma : \Theta \rightarrow \mathbb{N}^*$, 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 value of the trust score $\theta'_i(t) \in \Theta$ is mapped to a computational puzzle having exponential complexity, equivalent to $O(2^{\gamma}_i(t))$.

An example of mapping function for computing γ is given in Equation 6.1; note that the puzzle complexity is defined based on a maximum possible complexity Γ. The resulting value, $\gamma_i(t)$, can then be used to assess the difficulty of the puzzle. In this equation, the constant 1 defines the minimum possible complexity.

$$\gamma_i(t) = \lfloor \Gamma \times (1 - \theta'_i(t)) + 1 \rfloor \tag{6.1}$$

To illustrate, consider the computational puzzle presented by Douceur in (DOUCEUR, 2002): 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. The time required to solve the proposed puzzle is proportional to $2^{\gamma-1}$, and the time to assert the validity of the solution is constant. Any feasible puzzle

can be employed with our solution. There are examples in the related literature, such as (DOUCEUR, 2002; BORISOV, 2006; ROWAIHY et al., 2007), so there is no need to invent a new one.

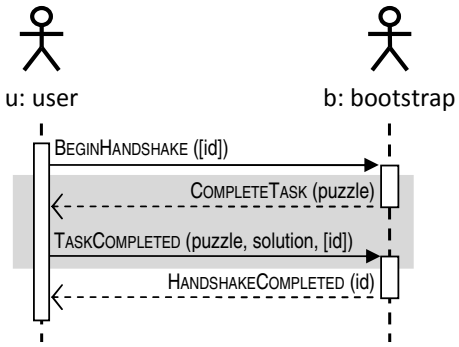


Figure 6.1: Instance of the protocol proposed in Chapter 4 for the case of adaptive puzzles.

Figure 6.1 illustrates the message exchange for obtaining identities prior to the resolution of an adaptive puzzle. As discussed in Chapter 4, the process of requesting an identity begins when the user issues a BEGINHANDSHAKE message. The bootstrap entity then replies with a COMPLETETASK message, informing the puzzle that must be solved before the identity assignment process continues. Once the puzzle is solved, the user replies to the bootstrap entity with a TASKCOMPLETED message, informing the puzzle solution. This process is similar for the case of an identity renewal.

For requesting a new identity, we propose that the user solve a puzzle estimated considering a differentiated, higher value of maximum puzzle complexity, $\Gamma = \Gamma_{req}$. In order to renew an identity, the user should solve a puzzle considering a lower value of maximum puzzle 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, $\Gamma_{renew} < \Gamma_{reval} < \Gamma_{req}$.

6.2 Making puzzles green and useful

In the following subsections we describe our proposal for reshaping traditional puzzles, in order to make them green and useful but without degrading their effectiveness in limiting the spread of fake accounts.

6.2.1 Towards green puzzles

The effectiveness of cryptographic puzzles in limiting the spread of fake accounts comes from the fact that solving them is ultimately a time-consuming task. Assigning puzzles that take one second to be solved will not stop attackers; in order to keep them away, it is important to assign puzzles that take longer to be solved (but not extremely longer, as legitimate users are also affected).

The more time users spend solving puzzles, the more energy is consumed, however. This problem is aggravated with the increase in popularity of the system (and with the interest of attackers in creating fake accounts in that system). In the current context of growing concern with rational usage of the available natural resources, the investigation for “green puzzles” (which demand less resources, but remain effective in limiting fake accounts) becomes imperative.

We propose reducing the average complexity of assigned puzzles, and complement them with "wait time". In order to understand the concept, suppose that an effective puzzle complexity to keep attackers away should result in a resolution time of five minutes. In our approach, instead of assigning such puzzle, we assign one that takes one minute; as a complement, once the user solves the puzzle, we ask him/her to wait four more minutes to obtain an identity. The trust score is used for estimating both the puzzle complexity and the wait period, and the procedure for computing this value depends on the process currently taking place.

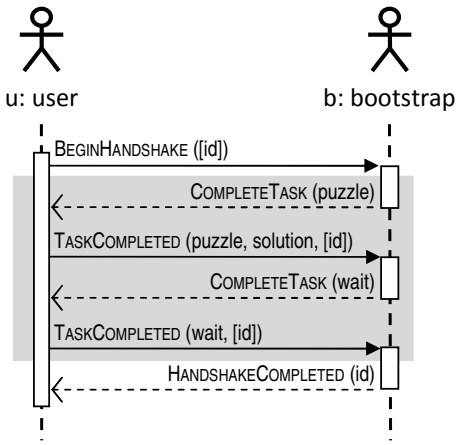


Figure 6.2: Instance of the protocol proposed in Chapter 4 for the case of green and useful puzzles.

The protocol for identity request/renewal in the case green and useful puzzles are employed is illustrated in Figure 6.2. Note that, compared to the case of adaptive puzzles, users now have to interact twice with the bootstrap entity: the first one is for solving a puzzle, whereas the second one is for obeying the wait time. In this case, the protocol requires the exchange of a pair of COMPLETETASK and TASKCOMPLETED messages. Upon request of a new identity, the bootstrap entity sends to the user a COMPLETETASK message, informing the puzzle to be solved. Once the user solves the puzzle, he/she replies to the bootstrap entity with a TASKCOMPLETED message. The bootstrap then estimates the wait time the user must obey prior to obtaining the identity, and afterwards replies with another COMPLETETASK message (informing the estimated wait time). As soon as the wait time is over, the user can reply to the bootstrap with a TASKCOMPLETED message. From this point on, the actual identity assignment can be carried out.

Defining the puzzle complexity

The strategy we envisage for defining the puzzle complexity depends on the process currently taking place, which can be either an *identity request* or *renewal*. Each of these are described in the detail next.

Identity request process. In this process, we use as input the value of smoothed trust score $\theta'_i(t)$ computed considering the model described in Section 5.1 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 $\theta'_i(t) \in \Theta$ is mapped to a computational puzzle having complexity equivalent to $O(2^\gamma)$.

An example of mapping function for computing γ is given in Equation 6.2; note that

the puzzle complexity is defined based on a maximum possible complexity Γ. The resulting value, $\gamma_i(t)$, can then be used to assess the difficulty of the puzzle.

$$\gamma_i(t) = \lfloor \Gamma \cdot (1 - \theta'_i(t)) \rfloor + 1 \tag{6.2}$$

In an identity request process, the value of $\gamma_i(t)$ is estimated considering a differentiated, higher value of maximum puzzle complexity, $\Gamma = \Gamma_{req}$. The value of smoothed trust score $\theta'_i(t)$ is saved in the identity, for later use during its renewal.

Identity renewal process. In this process, the value of $\theta'_i(t)$ used to estimate the puzzle complexity is computed based on the trust score saved in the identity, $I\langle\theta\rangle$, according to Equation 6.4. Once the renewal process is complete, the bootstrap must save $\theta'_i(t)$ in the identity ($I\langle\theta\rangle \leftarrow \theta'_i(t)$), for use during future renewal processes.

$$\theta'_i(t) = \beta \cdot 1 + (1 - \beta) \cdot I\langle\theta\rangle \tag{6.3}$$

To renew an identity, the user must solve a puzzle considering a lower value of maximum puzzle 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, $\Gamma_{renew} < \Gamma_{reval} < \Gamma_{req}$.

It is important to emphasize that Equation 6.4 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 for identity renewal). Other equations can be used, given that they provide incentives for renewing identities.

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 $\theta'_i(t)$. The design we consider for computing ω is given in Equation 6.4. In this function, Ω represents the maximum factor for the waiting time.

$$\omega_i(t) = \Omega \cdot (1 - \theta'_i(t)) \tag{6.4}$$

6.2.2 Towards useful puzzles

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 (DOUCEUR, 2002; BORISOV, 2006; ROWAIHY et al., 2007). An important characteristic of such puzzles is that their processing does not result in actual useful information. For example, consider the one proposed by Douceur (DOUCEUR, 2002): given a 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 $\gamma_i(t_k)$ least significant bits are 0. The actual result of the puzzle is, *de facto*, useless. 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 any identity request or renew 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 Equation 6.1), then $|\mathcal{J}| = 2^4 = 16$.

6.3 Evaluation

In this section we evaluate the effectiveness and robustness of adaptive puzzles in limiting the dissemination of fake accounts. We initially focus our evaluation on the notion of adaptive puzzles (Section 6.3.1). Then, we assess the feasibility and effectiveness of reshaping adaptive puzzles to make them green and useful (Section 6.3.2).

In this evaluation, we focus on the following research questions (the first two already posed in the introduction): (i) how effective would a mechanism based on the hypothesis that *one can filter potentially malicious requests* be in detecting them? (ii) what is the overhead that such a mechanism would cause to legitimate users? and (iii) what is the monetary cost of launching the attack (in terms of acquiring the resources necessary for deploying it)? Next we discuss the evaluation carried out, and our major conclusions.

6.3.1 Adaptive puzzles

We start our evaluation by assessing the effectiveness and potential benefits of using adaptive puzzles as an approach to limit the dissemination of fake accounts. We first describe the simulation environment used in the evaluation (simulation parameters, puzzle model, evaluation metrics). Then, we focus on sensitivity analysis of our solution, considering synthetic traces of identity requests. Finally, we present the results achieved considering real life traces of identity requests. For the sake of evaluation, we define the following evaluation metrics:

- **Proportion of fake accounts created (P_m).** This metric indicates the proportion of fake accounts successfully created by an attacker, compared to a baseline scenario. The possible values for this metric are in the range $[0..1]$. The value $P_m = 0$ is a global optimum; it means that no fake account was created when the considered approach (which can be our solution, or any other proposal in the literature) is used.

- **Proportion of legitimate accounts created (P_l).** This metric indicates the proportion of legitimate accounts successfully created, compared to a baseline scenario. The possible values for this metric are in the range $[0..1]$. A value of $P_l = 1$ is a global optimum; it means that the considered approach (which can be our solution, or any other proposal in the literature) has blocked no legitimate request, during the period of evaluation.

- **Proportion of easier puzzles assigned to an attacker (E_m).** This metric indicates the proportion of malicious identity requests that were assigned a puzzle of lower complexity, compared to a baseline scenario. The possible values for this metric are in the range $[0..1]$. A value of $E_l = 0$ is a global optimum; it means that all puzzles assigned to the attacker were more complex if compared to the baseline scenario.

- **Proportion of easier puzzles assigned to legitimate users (E_l).** This metric indicates the proportion of legitimate identity requests that were assigned a puzzle of lower complexity, compared to a baseline scenario. The possible values for this

metric are in the range $[0..1]$. A value of $E_l = 1$ is a global optimum; it means that legitimate users received easier to solve puzzles if compared to the baseline scenario.

- **Fairness[2] in the assignment of puzzles (F).** Recall that the goal of using adaptive puzzles is ensuring that legitimate are minimally penalized with easier to solve puzzles per identity requested, whereas comparatively more complex puzzles are assigned to attackers. This metric measures the fairness of our solution, i.e. the proportion of puzzles of a certain complexity assigned to legitimate users, compared to those assigned to an attacker. Formally, we have:

$$F = \frac{1}{n-1} \cdot \sum_{i=0}^{n-1} i \cdot (m_i - l_i) \tag{6.5}$$

  In this equation, n is the number of classes of complexity of assigned puzzles (being 0 the index of the least complex puzzle, and $n-1$ the index of the most complex one); m_i is the proportion of puzzles of complexity of the i-th class assigned to attackers; and l_i is the proportion of puzzles of complexity of the i-th class assigned to legitimate users. The possible values for this metric range in the interval $[-1, +1]$ (normalized). A value of $F = +1$ is a global optimum, meaning that our solution assigned only the most complex puzzles to the attacker, and only the least complex ones to legitimate users.

Simulation environment

To evaluate the feasibility of using adaptive puzzles in limiting the spread of fake accounts, we extended the bootstrap entity and the simulation environment for implementing the dynamics of puzzle assignment and resolution. In summary, the extended entity aggregates the functionalities of managing identity requests from users interested in joining some generic system, assignment of puzzles for each identity request, validation of puzzle solutions received, and granting (or denial) of requests (according to the correctness of received solutions). Similarly to the evaluation presented in the previous chapter, the scope of our simulation is a peer-to-peer file sharing network (due to trace availability to use as input for the simulation).

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

In order to model the delay caused by solving computational puzzles, we considered the puzzle presented by Douceur in (DOUCEUR, 2002) and described in Section 6.1. Further, for the sake of simplicity, we considered that a puzzle having complexity $\gamma_i(t_k)$

[2]Observe that we use the term *pricing* to refer to the computational effort some entity (legitimate user or attacker) must dedicate to obtain a single identity. The term *fairness*, in contrast, is used as an indicative that our solution imposes to potentially attackers a higher price per identity requested, when compared to the price per identity request for presumably legitimate users.

Table 6.1: Summary of parameters involved in the evaluation of the notion of adaptive puzzles (novel parameters specific for this evaluation are highlighted in gray).

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

takes $2^6 + 2^{\gamma_i(t_k)-1}$ seconds to be solved in a powerful but standard, off-the-shelf computing hardware (which has a normalized computing power of 1, for reference); a computer two times faster takes half of this time to solve the same puzzle.

Sensitivity analysis

In this section we evaluate how well the proposed solution ameliorates the negative impact of a Sybil attack considering various distinct settings. The parameters involved in the evaluation are summarized in Table 6.1. Note that the parameter set presented in this table extends from the set of parameters involved in the evaluation of the concept of trust scores, shown in Section 5.2.1.

In our analysis, we evaluate the dynamics of identity granting considering (*i*) different sizes for the sliding window; (*ii*) various values for the smoothing factor; (*iii*) sharing of legitimate sources with the attacker; (*iv*) different sizes of sources; (*v*) increasing computing power of the attacker; and (*vi*) varying proportions of sources in control of the attacker. The attack strategies considered in this analysis are based on the discussion presented earlier in Section 4.2.

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

We assumed an exponential distribution ($\lambda_{cp} \approx 0.003$) for the *computing power* of users' workstations, ranging from 0.1 to 2.5 times the computing power of the standard, off-the-shelf hardware used as reference (an evaluation considering varying distributions for users' computing power is provided next). The amount of time required to solve a puzzle is then obtained by dividing the total number of complexity units (of the assigned puzzle) by the computing power of the user's station.

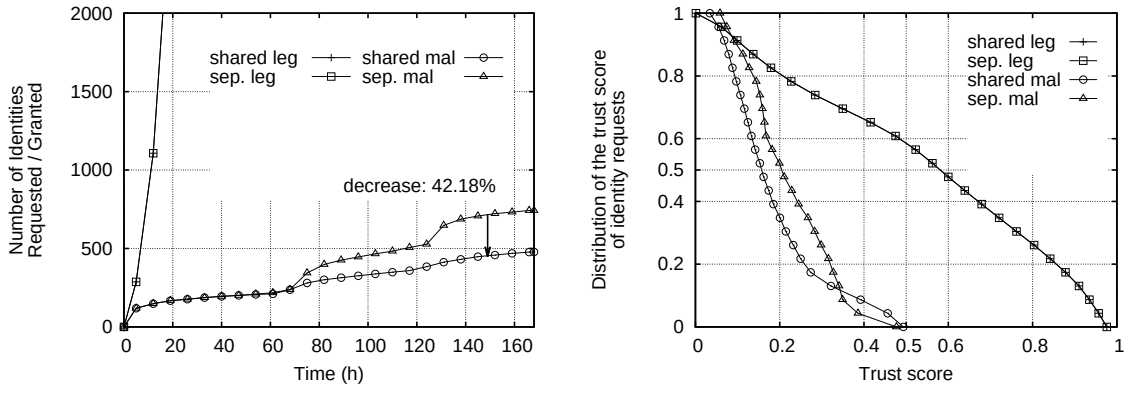
Based on this model, an identity request performed by a user associated to a source i having $\theta'_i(t) = 0.5$ would be assigned a puzzle whose goal is to obtain two numbers x and z so that the concatenation $x|y|z$, after processed by a secure hash function, leads to a number where the $\lfloor 18 \times (1 - 0.5) + 1 \rfloor = 10$ least significant bits are 0 (note that $\Gamma = 18$

in this model). This puzzle has complexity of $2^6 + 2^{10-1} = 576$ units. Considering the reference computing power, this puzzle takes 576 seconds to be solved; in the case of a computing device two times faster, the same puzzle takes 288 seconds.

Sharing legitimate sources with an attacker. In this analysis, we compare the performance of our solution when malicious requests depart from sources shared with legitimate users, in contrast to a scenario in which the attacker uses his/her own sources to request identities. In practice, the first scenario could correspond to an attacker behind networks using NAT, in which other legitimate users can also be found. Conversely, the second scenario corresponds to an attacker using its own set of valid IP addresses to request the fake (counterfeit) identities. As the goal of the attacker is controlling $1/3$ of the identities in the system, the total number of counterfeit identities to be requested is set to 82,425 ($M_r = 82,425$).

In this analysis, the $L_u = 160,000$ legitimate users are evenly distributed along 10,000 sources ($S = 10,000$); this makes a total 16 legitimate users per source ($L_s = 16$). The number of legitimate requests per source follows an exponential distribution, with $\lambda_r \approx 0.0634$ and bounded between 16 (making at least one request per legitimate user) and 128 (making eight requests per user). The *first arrival* of an identity request of each source is normally distributed during the simulation period, with $\mu_f = 302,400$ (half of the week) and $\sigma_f = 100,800$; the *time interval* between requests of each source ranges from 1 minute to 2 hours, and follows an exponential distribution (with $\lambda_i \approx 9.94 \times 10^{-4}$). It is important to highlight that the choice for these distributions (exponential for the number of requests per source, normal for the first arrival of a user's request, and exponential for the time between requests) was based on an analysis of the traces presented in Section 3.3.1.

For requesting the counterfeit identities, the attacker employs 10 high-performance computers ($M_c = 10$), each equipped with a hardware 2.5 times faster than the reference one (the most powerful hardware considered in our evaluation). The attacker controls 10 of the legitimate sources ($M_u = 10$), and uses them to originate the $M_r = 82,425$ identity requests; each of these sources originate around 8,242 malicious requests, one every 73.3 seconds approximately ($\frac{604,800}{82,425} \times 10$). In the first scenario (shared sources), there are $S = 10,000$ sources in the system (10 out of them are used for both legitimate and malicious requests); in the latter scenario (separated sources), there is a total of $S = 10,010$ sources. Finally, we adopt the following parameter setting for our solution: $\Delta t = 48$, and $\beta = 0.125$. Recall that these values were derived from the sensitivity analysis described in Section 5.2.



(a) Identities granted

(b) Trust score of identity requests

Figure 6.3: Effectiveness of our solution considering sharing of sources.

Table 6.2: Characteristics of the sources in each of the evaluated scenarios.

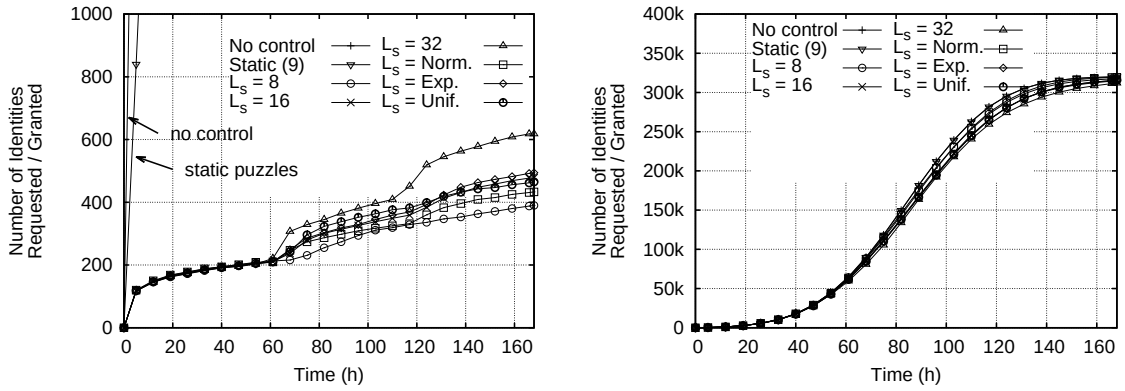
Scenario	Number of Sources	Distribution of sources size	Users per source	Distribution of sources requests	Requests per source
$L_s = 8$	$S = 20,000$	Fixed	8	Exponential ($\lambda \approx 0.03168$)	between 32 and 256
$L_s = 16$	$S = 10,000$	Fixed	16	Exponential ($\lambda \approx 0.06337$)	between 16 and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

Figure 6.3 shows that sharing sources with legitimate users clearly degrades the effectiveness of the attack. In the scenario in which the attacker uses his/her own sources (curve "sep. mal"), 742 counterfeit identities are created. This number is reduced in 35.57% (to 478 identities) when the malicious requests depart from sources shared with legitimate users (curve "shared mal"). Although shared sources also impact the legitimate users associated to them, this impact is restricted, as legitimate users request very few identities each: only 0.001% of identity requests (4 identities) were not granted when sources are shared. From the experiment above, only 10 sources (out of 10,000) were shared. Given that there is an average of 16 legitimate users per source (as discussed in the sensitivity analysis of the time window Δt), only 160 users (out of 160,000) – or 320 identity requests (out of 320,000) – were directly affected. Focusing on the trust score of identity requests, observe from Figure 6.3(b) that the score of legitimate requests remains virtually unchanged, whereas malicious requests are largely affected (even though a few requests obtain better values of trust score).

Size of sources (L_s). As discussed in Section 4.1.1, a varying number users may be associated to a single source of identity requests – therefore resulting in sources of varied sizes. This is the case of legitimate users behind networks using NAT, when the network addressing scheme is used to delineate sources; and also the case of users located in a same geographical location (e.g., same building), when network coordinate systems are used to distinguish the sources. In order to evaluate the impact of such a situation (various users behind a same source) on the performance of the proposed solution, we carried out several experiments considering different sizes of sources, and also different distributions for sources sizes. Here we discuss the most prominent ones.

For this analysis, we considered six scenarios: three in which the size of sources is fixed, and other three in which it varies following a given distribution. Table 6.2 presents in more detail the characteristics of each scenario. For the sake of evaluation, in this evaluation (and also in the following ones) malicious requests depart from sources shared with legitimate users. The choice for an exponential distribution for the number of requests per source (column "Distribution of sources requests" in Table 6.2) is based on the analysis of traces presented in Section 3.3.1. For the sake of comparison, we also show results obtained with "no control" (i.e., where no puzzles are assigned prior to granting identities, and with "static puzzles" (BORISOV, 2006). For the mechanism based on static puzzles, we consider a complexity of $\Gamma = 9$ units; it takes between 3 and 85 minutes to solve, depending on the computer capacity.

Figure 6.4(a) shows that the higher the average size of sources is, the better for the attacker. In the scenario "$L_s = 8$" (8 legitimate users behind each source), the attacker



(a) Fake identities created

(b) Legitimate identities granted

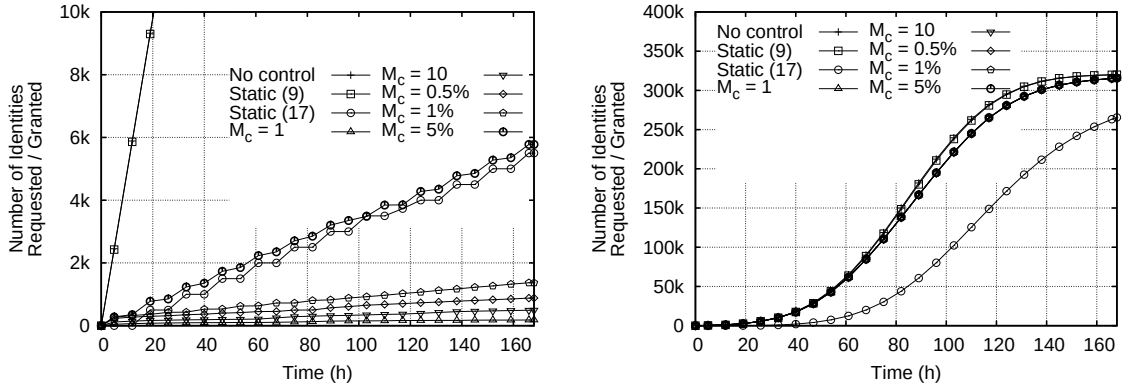
Figure 6.4: Effectiveness of the proposed solution under varying source sizes.

was able to obtain 390 identities only. This number increased to 618 (or 58.46%) in the scenario "$L_s = 32$". This is because of the higher average of source recurrences ($\Delta\phi_i(t)$, for the i-th source), which in turn increase the network recurrence rate $\Phi(t)$; as a consequence, the sources in hands of the attacker become less suspicious (as the relationship between source and network recurrence rate $\rho_i(t)$ decreases), and puzzles of lower complexity are assigned to requests coming from them. It is important to highlight that, despite such increase, the attacker remained far from achieving the desired number of identities ($M_r = 82,425$). With regard to the scenarios in which the size of sources followed a distribution, the results achieved were in general similar to the scenario "$L_s = 16$". As for the impact to the requests of legitimate users, again it remained marginal – as one can see from Figure 6.4(b). These results, although not exhaustive, suggest that the proposed solution should perform consistently regardless of the arrangement of users among sources in the system.

Increasing the computing power of the attacker (M_c). In this analysis, we measure the effectiveness of our solution when the attacker increases the computing power available to solve the assigned puzzles. The number of sources the attacker uses to request identities remains the same ($M_u = 10$). An analysis considering an increasing number of sources in hands of the attacker is shown next. For the sake of this evaluation, we consider the same number of sources, distribution of sizes of sources, and distribution of recurrence of sources considered in the scenario "$L_s = Exp.$" (from the previous analysis). The choice for this scenario is supported on a recent study by Maier, Schneider, and Feldmann (MAIER; SCHNEIDER; FELDMANN, 2011), which suggests that the number of unique end-hosts behind networks using NAT – in residential DSL lines of a major European ISP – follows an exponential distribution.

The total number of counterfeit identities to be requested is again 82,425 ($M_r = 82,425$). For this attack, we consider the availability to the attacker of the following number of high-performance computers (M_c): 1, 10, 0.5% (with regard to the number of sources S, i.e., 50 computers), 1% (100 computers), and 5% (500 computers). For the sake of comparison, we also show results obtained with "no control" (i.e., where no puzzles are assigned prior to granting identities, and with "static puzzles" (BORISOV, 2006). For the latter, we consider two scenarios: one with puzzles of complexity $\Gamma = 9$, and another with puzzles of $\Gamma = 17$ units (which take between 14 and 364 hours to solve). In both cases, the computing power of the attacker is the highest considered ($M_c = 5\%$

high-performance computers). The parameter setting for our solution remains $\Delta t = 48$ and $\beta = 0.125$.



(a) Fake identities created

(b) Legitimate identities granted

Figure 6.5: Legitimate and malicious identity requests granted, considering a varying computing power of the attacker.

Figure 6.5(a) shows that the more computing power the attacker has, the higher number of identities it obtains from the system. However, even when 500 high-performance computers are used (curve "$M_c = 5\%$"), the attacker is not able to repeat the same performance as in the "No control" scenario, and only 5,598 counterfeit identities are created (a reduction of 93.2%). In the scenario in which the attacker is extremely limited in resources ($M_c = 1$), only 184 identities are created during the period of 168 hours. The reason for such a poor attack performance is that the recurrence of the sources in hands of the attacker becomes extremely high. Recall that the attacker controls (in this experiment) only $M_u = 10$ sources; consequently, the average number of malicious requests per source is $\frac{82{,}425}{10} \approx 8,242$. On one hand, with a very limited computing power available (e.g., $M_c = 1$ high-performance computer), the rate in which the attacker solves the assigned puzzles is slower; therefore, the trust score of subsequent malicious requests does not drop significantly. On the other hand, with a higher computing power available (e.g., $M_c = 5\%$, or 500 high-performance computers), puzzles are solved more quickly, in a first moment. The recurrence of the sources controlled by the attacker then becomes extremely higher (than the average network recurrence rate). As a consequence, even more complex puzzles will be assigned to future malicious requests – therefore undermining the attackers' ability to keep solving puzzles quickly.

There are two important observations regarding the results shown in Figure 6.5. First, the gain obtained by the attacker – when our solution is used – increases almost linearly with the number of high-performance computers employed in the attack. As shown in Figure 6.5(a), with $M_c = 0.5\%$ the attacker is able to obtain 881 identities. This number increases to 1,369 identities with $M_c = 1\%$, and to 5,778 with $M_c = 5\%$. This behavior indicates that the attacker, in order to improve the performance of the attack, must dedicate a directly proportional amount of resources to solve the assigned puzzles. In other words, higher success rates come at even higher (and increasing) costs for the attacker.

The second observation is that the mechanism based on static puzzles is completely taken over (i.e., becomes ineffective) when easier-to-solve puzzles are assigned to requests (curve "Static (9)" in Figure 6.5(a)). Conversely, it only presents a similar performance to our solution for the worst-case scenario ($M_c = 5\%$), when an extremely

difficult puzzle – with 2^{17} complexity units – is used (curve "Static (17)"). Observe from Figure 6.5(b), however, that such a puzzle imposes a very high cost for legitimate users as well – and 54,469 legitimate requests are not granted an identity within the period of 168 hours. In the case of the proposed solution, the identity requests of legitimate users experience a marginal impact, despite the computing power the attacker has available. This observation highlights the major trade-off involved with the use of static puzzles to mitigate fake accounts, between effectiveness (in preventing the indiscriminate creation of counterfeit identities) and flexibility (to legitimate users).

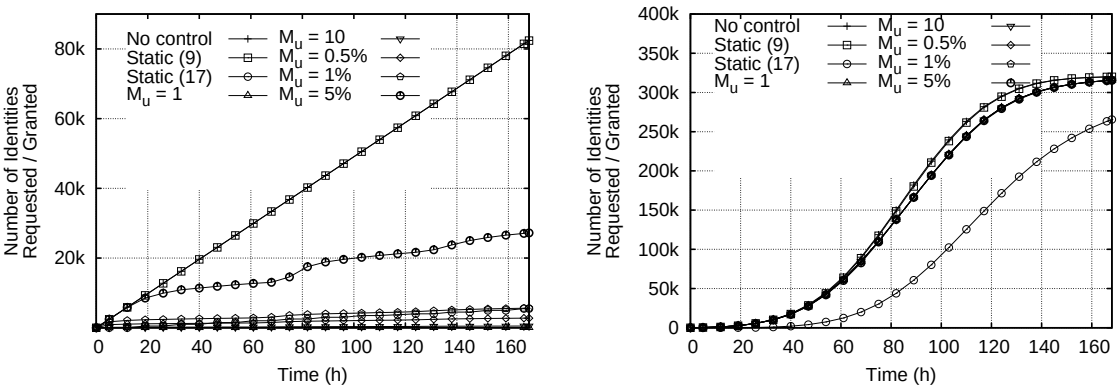
In summary, from the discussion presented above, two main conclusions may be drawn. First, simply increasing the computing power dedicated for the attack is not sufficient for circumventing the proposed solution, as the higher recurrence of malicious sources becomes a bottleneck for the success of the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

Varying proportions of malicious sources (M_u). In this analysis, we evaluate the performance of the proposed solution in a different scenario: instead of solely increasing the computing power available (using a high-performance cluster), the attacker controls a botnet and uses its machines to request identities and process the associated puzzles. The total number of counterfeit identities to be requested is again 82,425 ($M_r = 82,425$). For this attack, we consider the availability to the attacker of a botnet of high-performance computers, each associated to its own source, having the following sizes (M_u): 1, 10, 0.5% (with regard to the number of sources S, i.e., 50 zombie machines), 1%, and 5%. It is important to keep in mind that $M_u = 5\%$ is a very extreme case, in which still makes sense (for the attacker) launching a Sybil attack. Should the attacker have a higher number of workstations connected to the Internet, he/she would already have outnumbered legitimate users connected to the system, and launching a Sybil attack would not be necessary.

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

In the case of $M_u = 0.5\%$ (a botnet of 50 zombie machines), around 1,648 malicious requests originate from each machine/source, i.e., a request every 366 seconds, per source. Analogously, in the case of $M_u = 1\%$, around 824 requests are performed from each machine/source in the botnet; a request every 733 seconds, per source. In the extreme case of $M_u = 5\%$, 164 requests are performed per source, one every $3,665$ seconds (1 hour). Recall that the lower the number of identities requested by a machine and the higher the interval between requests, the better the trust score of requests coming from that machine, and consequently the lower the complexity of the puzzle to be solved upon each request. All other properties of the analysis remain the same.

Figure 6.6(a) shows that the attacker is able to create only 53 counterfeit identities when using only one machine (curve "$M_u = 1$"). As the number of zombie machines used in the attack increases, so increases the number of counterfeit identities created. With $M_u = 10$, 494 identities are created. This number increases linearly with the amount of resources dedicated to the attack (as seen in the previous analysis): 2,760 identities with



(a) Fake identities created

(b) Legitimate identities granted

Figure 6.6: Effectiveness of our solution with varying proportions of malicious sources.

50 machines ("$M_u = 0.5\%$"), 5,624 identities with 100 machines ("$M_u = 1\%$"), and 27,209 with 500 machines ("$M_u = 5\%$"). In spite of the increased performance, however, the attacker does not achieve the same success as in the "No control" scenario. In the extreme case where 500 zombie machines are used (curve "$M_u = 5\%$"), the attacker only achieves 33% of its goal during 168 hours. Now focusing on the overhead caused to legitimate users, Figure 6.6(b) shows that it remained stable and minimal. These results not only evidence that our solution is able to limit the creation of counterfeit identities, but also shows its better performance (specially under extreme conditions, for which launching Sybil attacks still makes sense for the attacker) when compared to the mechanism based on static puzzles.

Evaluation using real life traces of identity requests

Having evaluated our solution using a synthetically generated trace, and analyzed its behavior under the influence of various environment and parameter settings, we now focus on its performance in a real life scenario. For the sake of comparison, we also show results obtained with static puzzles (BORISOV, 2006) and when control is absent (no control). Next we cover the settings employed for the evaluation and the results achieved.

Evaluation settings. In this evaluation, we consider the three traces of identity requests characterized in Section 3.3.1. For each of the traces used, three attack scenarios were evaluated: without attack, with $M_u = 1\%$, and with $M_u = 10\%$ malicious sources. The second scenario ($M_u = 1\%$) corresponds to an attacker with limited computing power to solving puzzles and possessing various distinct locations from which identity requests depart. In the third scenario ($M_u = 10\%$), the attacker has a botnet at his/her service and uses it to launch the attack. Note here that we increased even further the amount of resources in hands of the attacker, in contrast to the scenario evaluated in the sensitivity analysis of varying proportion of malicious sources. We assume the worst case scenario for the botnet, in which all *zombie* machines forming it are equipped with the most powerful hardware considered in our experiments (i.e., 2.5 times higher than of the off-the-shelf, reference hardware). Table 6.3 describes these scenarios in more detail. For the computing power of legitimate users, we analyze cases in which it follows an Exponential ($\lambda_{cp} \approx 0.003$) and Gaussian ($\mu_{cp} = 1.2$ and $\sigma_{cp} \approx 0.4$). In both cases, it ranges from 0.1 to 2.5 times the computing power of the reference hardware. We consider such settings since they reflect the distribution of users' computing power as reported in

the literature (NET MARKET SHARE, 2013; CPU BENCHMARK, 2013; ENIGMA @ HOME, 2013). It is important to emphasize that we use a continous distribution in order to model slight variations in the computing power available at some user's workstation, because of processes running in parallel in the host operating system.

Table 6.3: Characteristics of the scenarios evaluated for each trace.

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

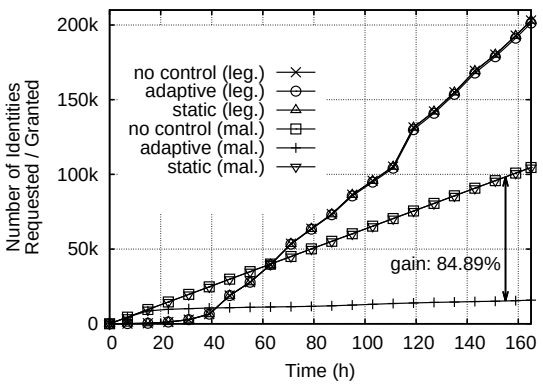
The specific parameters of each solution were defined $\Delta t = 48$ and $\beta = 0.125$. The mechanism based on static puzzles (BORISOV, 2006) uses a puzzle with $\Gamma = 9$ complexity units, which takes between 5 minutes and 2 hours (approximately) to solve.

Efficacy of our solution and overhead to legitimate users. Figures 6.7, 6.8, and 6.9 show the results obtained for each of the compared solutions, for Traces 1, 2, and 3, respectively. For the sake of clarity and space constraints, only the results obtained for the second attack scenario ($M_u = 1\%$) are shown in these figures.

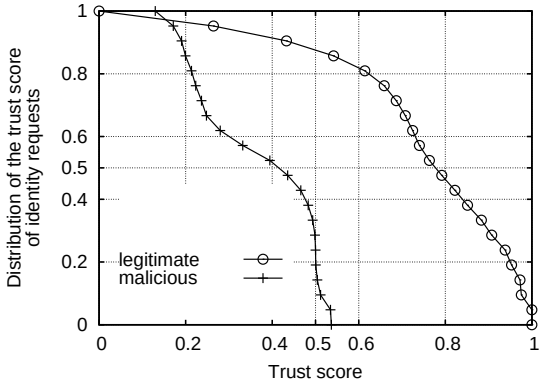
Observe from Figure 6.7(a) that our solution outperforms both static puzzles (BORISOV, 2006) and the absence of control in limiting the creation of counterfeit identities. The proposed solution (curve "adaptive (mal.)" in Figure 6.7(a)) reduced in 84.9% the number of counterfeit identities granted, in comparison to the "no control" scenario (curve "no control (mal.)"). This represented an effective gain of 84.89% over the scenario where static puzzles were used (curve "static (mal.)"); the use of such puzzles reduced marginally the number of granted counterfeit identities (0.05% only). Such a performance of the static puzzles is because the time required to solve them (five minutes, in the case of the attacker) was overall smaller than the interval between identity requests. In regard to the overhead caused to legitimate users, observe from Figure 6.7(a) that the curve "adaptive (leg.)" (and also "static (leg.)") overlaps with the curve "no control (leg.)", thus indicating that the imposed overhead was negligible.

We now focus on the trust scores assigned by the proposed solution. In Figure 6.7(b), the close the curves are from $x = 1$ and $y = 1$ axes, the higher the proportion of identity requests that received higher trust scores. Thus, it is clear that a higher proportion of legitimate identity requests received better trust scores, than malicious requests. Looking at these results in more detail, one may see that over 43% of legitimate requests were assigned a value of trust score higher or equal to 0.8. This proportion increases to 87% if we also consider those requests assigned with a trust score higher or equal to 0.5. This represents in fact that 87% of identity requests coming from legitimate users were not considered suspicious, and consequently received puzzles of lower complexity. Conversely, 78% of malicious requests were assigned a value of trust score lower or equal to 0.5.

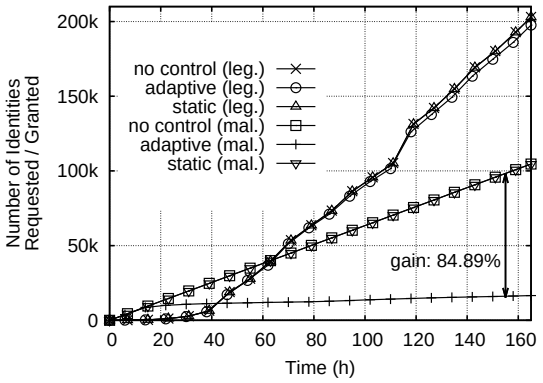
Observe that the results are marginally affected by the distribution of users' computing



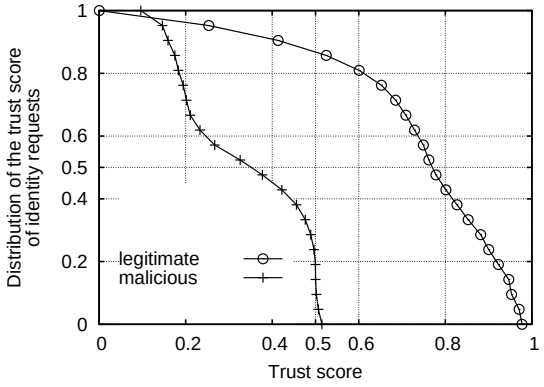
(a) Identities granted (Exponential)



(b) Trust score of identity requests (Exponential)



(c) Identities granted (Gaussian)



(d) Trust score of identity requests (Gaussian)

Figure 6.7: Legitimate and malicious identities granted, and trust scores assigned (when the proposed solution is used), for each of the scenarios evaluated with Trace 1, considering an Exponential and a Gaussian distribution for users' computing power.

power considered. Comparing for example the number of identities granted to legitimate users and the attacker (Figures 6.7(a) and 6.7(c)), the difference observed is marginal. These results evidence that our solution is able to perform satisfactorily regardless of the environment settings in place.

Table 6.4 presents a summary of the results achieved using Trace 1 as input, for each scenario evaluated. The table shows, per scenario evaluated, statistics for both *legitimate* and *malicious requests*. The statistics include: (*i*) number of identities granted (field *granted*), (*ii*) proportion of identities granted using as baseline the "no control" scenario (fields P_l and P_m, respectively), and (*iii*) number and proportion of legitimate and malicious identity requests that were assigned a puzzle of lower complexity, using the *static puzzles* mechanism (BORISOV, 2006) as baseline (fields E_l and E_m, respectively). For the sake of clarity, the results achieved with our solution are highlighted in gray.

Observe from Table 6.4 that our solution effectively prevented the creation of counterfeit identities across the network, in both scenarios with attack ($M_u = 1\%$ and $M_u = 10\%$), whereas imposing a marginal overhead to legitimate requests. In contrast, the mechanism based on static puzzles did not cause any impact to malicious requests, and the attacker easily succeeded in obtaining 34% of identities in the network. Even in the worst case scenario ($M_u = 10\%$), our solution prevented 8.53% of malicious requests from being granted a (counterfeit) identity. As previously discussed in the sensitivity analysis

Table 6.4: Results obtained for each of the scenarios evaluated, for Trace 1.

Scenario	Legitimate Requests				Malicious Requests				F
	granted	P_l	E_l (num)	E_l (%)	granted	P_m	E_m (num)	E_m (%)	
No attack, Static	202,889	0.99	-	-	-	-	-	-	-
No attack, Adaptive	200,869	0.98	161,064	79,32	-	-	-	-	-
1%, Static	202,889	0.99	-	-	104,554	0.99	-	-	0
1%, Adaptive	201,251	0.99	178,960	88,13	15,796	0.15	5,837	36.72	0.4211
10%, Static	202,889	0.99	-	-	104,554	0.99	-	-	0
10%, Adaptive	201,545	0.99	186,593	91,89	95,678	0.91	22,694	22.69	0.5817

of varying proportion of malicious sources, note that $M_u = 10\%$ is a very extreme case, in which still makes sense (for the attacker) launching a Sybil attack. Furthermore, the relative "success" comes at a very high cost to the attacker, who either has to control a large number of high-performance, *zombie* workstations in the Internet, or possess a large number of high-performance computers dedicated to solving puzzles.

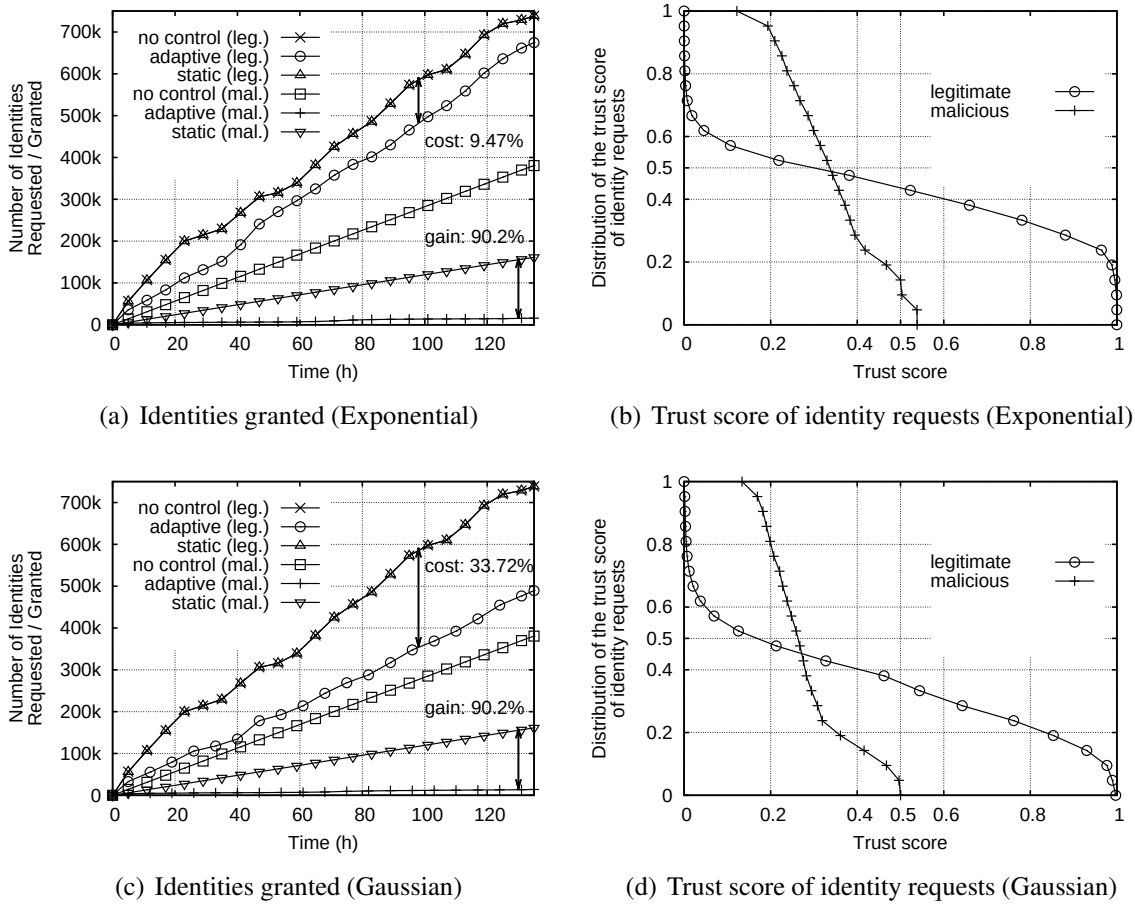
Another important metric for measuring the efficiency of our solution, presented in Table 6.4, is the number of puzzles of lower complexity – than the complexity of those used for the "static puzzles" mechanism – assigned to identity requests (E_l and E_m). Ideally, E_l should be as high as possible, and E_m as low as possible. Note that the proportion of legitimate requests that received a puzzle of lower complexity ranged from 79.32% (no attack scenario) to 91.89% (scenario $M_u = 10\%$). Conversely, the proportion of malicious requests that received a puzzle of higher complexity ranged from 36.72% to 22.69%. These results evidence that our solution not only prevented the spread of counterfeit identities across the network, but also distinguished satisfactorily between identity requests originated from correct users and attackers, and dealt with them accordingly.

Finally, observe that our solution achieves a considerable fairness in the assignment of puzzles to legitimate users and the attacker. For scenario $M_c = 1\%$, the use of adaptive puzzles achieves a fairness of $F = 0.4211$, whereas for scenario $M_c = 10\%$ the fairness improves to $F = 0.5817$.

The results obtained for Trace 2 and Trace 3, depicted in Figures 6.8 and 6.9, confirm the positive results our solution achieved, in contrast to existing solutions. In the case of Trace 2 (Figure 6.8(a)), while the attacker was able to obtain 161,190 counterfeit identities (over 135 hours) when static puzzles were used, the use of our solution reduced this number to 15,801 identities; this represents a gain of approximately 90.2% over the results achieved with static puzzles, and a reduction of 95.84% in the number of counterfeit identities that would be granted (380,483) if no control was imposed. For Trace 3 (Figure 6.9(a)), the effective gain our solution provided was of 95.69%.

In regard to the overhead caused to the requests of legitimate users (cost), although higher than observed in the case of Trace 1, both were comparatively low. For Trace 2 (Figure 6.8(b)), only 9.47% of the identity requests coming from legitimate users were not granted (considering the exponential distribution for users' computing power), during the period of 135 hours; for Trace 3, the measured overhead was 13.48% in the same scenario (Figure 6.9(b)).

Now concentrating on the trust scores assigned to requests, Figure 6.8(b) shows that approximately 40% of legitimate ones were assigned a value of trust score higher or equal



(a) Identities granted (Exponential)

(b) Trust score of identity requests (Exponential)

(c) Identities granted (Gaussian)

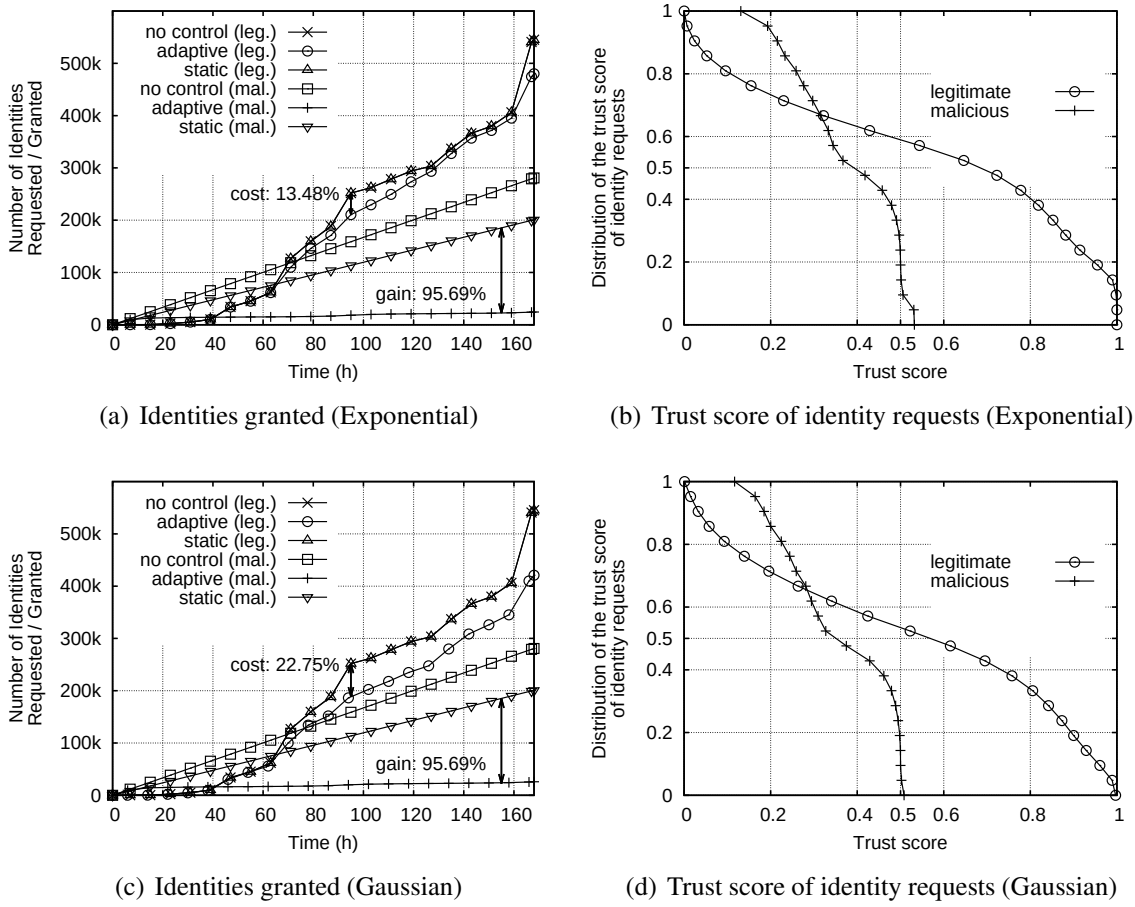
(d) Trust score of identity requests (Gaussian)

Figure 6.8: Legitimate and malicious identities granted, and trust scores assigned (when the proposed solution is used), for each of the scenarios evaluated with Trace 2, considering an Exponential and a Gaussian distribution for users' computing power.

to 0.6. No malicious request was assigned such a value of trust score, and only 10% of malicious requests were assigned a value of trust score higher or equal to 0.5. Figure 6.9(b), in turn, shows that approximately 60% of legitimate requests were assigned with a value of trust score higher or equal to 0.5, whereas approximately 86.19% of malicious ones were assigned values lower or equal to 0.5.

Tables 6.5 and 6.6 clearly evidence that our solution outperforms the static-puzzle-based mechanism even in the worst case scenario, for both Trace 2 and Trace 3, respectively. While static puzzles were completely ineffective, our solution allowed the attacker to control at most 22.41% of identities in the system, in the case of Trace 2, and only 8.02% for Trace 3. Further, our solution assigned puzzles of lower complexity to at least 40% of legitimate requests (over than 50%, in the case of Trace 3); less than 25% of malicious requests (up to 20% in the worst case scenarios) received such a puzzle. More importantly, observe in these tables that our solution achieves some considerable fairness, being as high as $F = 0.3629$ in the case of Trace 3.

Monetary analysis of the attack. We finally focus on the economic impact our solution causes to the attacker. In this analysis, we take into account the prices for hiring the services required to launch each of the attacks considered in this evaluation, as shown in Section 4.2. It is important to emphasize that the prices considered are the cheapest found in the specialized literature.



(a) Identities granted (Exponential)

(b) Trust score of identity requests (Exponential)

(c) Identities granted (Gaussian)

(d) Trust score of identity requests (Gaussian)

Figure 6.9: Legitimate and malicious identities granted, and trust scores assigned (when the proposed solution is used), for each of the scenarios evaluated with Trace 3, considering an Exponential and a Gaussian distribution for users' computing power.

Table 6.7 provides a brief economic analysis for each of the attack scenarios considered, when the identity management scheme uses puzzles to control the assignment of identities. The price per processing unit in the cluster we considered in the analysis is US$ 0.06/hour, and the price of the botnet is US$ 0.50/zombie computer. Values fields "cluster" and "botnet" are displayed in units hired. Field "total" contains the budget of the attack in each scenario considered (displayed in US dollars). Field "US$/id" contains the cost per identity effectively obtained by the attacker.

In the absence of any protection mechanism ("no control" scenarios), the attack is costless. This is because no cluster is required to solve puzzles, and no botnet is necessary to trick the identity management scheme. However, one can see from the "static" and "adaptive" schemes that the attack poses a significant cost to the attacker (as high as US$ 50,741 in the case of Trace 3), even in the case commodity services are hired to deploy it. Our solution makes the attack even more expensive, as the attacker is required to hire zombie computers to trick the concept of sources of identity requests. Observe that the increased cost to the attacker comes with the additional benefit that presumably legitimate users are less harmed by the identity management scheme (in their request for identities).

An important index to analyze the data shown in Table 6.7 is the total cost of the attack per identity effectively obtained (field "US$/id"). One can see that our solution increases

Table 6.5: Results obtained for each of the scenarios evaluated, for Trace 2.

Scenario	Legitimate Requests				Malicious Requests				F
	granted	P_l	E_l (num)	E_l (%)	granted	P_m	E_m (num)	E_m (%)	
No attack, Static	738,443	0.99	-	-	-	-	-	-	-
No attack, Adaptive	674,426	0.91	311,827	42.22	-	-	-	-	-
1%, Static	738,443	0.99	-	-	161,190	0.42	-	-	0
1%, Adaptive	674,544	0.91	313,263	42.41	15,801	0.04	2,190	13.77	0.1029
10%, Static	738,443	0.99	-	-	380,252	0.99	-	-	0
10%, Adaptive	676,244	0.91	320,520	43.40	195,336	0.51	38,398	19.17	0.1664

Table 6.6: Results obtained for each of the scenarios evaluated, for Trace 3.

Scenario	Legitimate Requests				Malicious Requests				F
	granted	P_l	E_l (num)	E_l (%)	granted	P_m	E_m (num)	E_m (%)	
No attack, Static	544,921	0.99	-	-	-	-	-	-	-
No attack, Adaptive	478,694	0.87	286,173	52.50	-	-	-	-	-
1%, Static	544,921	0.99	-	-	200,804	0.71	-	-	0
1%, Adaptive	480,149	0.88	315,409	57.86	8,636	0.03	5,837	23.89	0.1864
10%, Static	544,921	0.99	-	-	280,689	0.99	-	-	0
10%, Adaptive	484,519	0.88	336,373	61.70	42,228	0.15	22,694	12.09	0.3629

significantly the monetary cost of each counterfeit identity the attacker effectively obtains. In the case of Trace 3, such an increase was as high as 598% (from US$ 0.1722 per identity obtained to US$ 1.2015).

There is no precise estimate on the profit the attacker (or third parties) can make with fake accounts. According to one study by Barracuda Labs, cited by The New York Times (PERLROTH, 2013), the average price for 1,000 twitter followers ranges around US$ 18 in the black market; prices can be as low as US$ 11 depending on the reseller (SCOTT, 2013; LEMOS, 2013). There are also dealers offering a bulk of 100 Gmail accounts for US$ 10 (DOE, 2013). Observe that 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 by far surpassed by the cost of obtaining a single fake account that our solution imposes (US$ 1.2015).

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

Table 6.7: Economic analysis of the attack, for each of the considered traces.

Scenario	Trace 1				Trace 2				Trace 3			
	cluster	botnet	total	US$/id	cluster	botnet	total	US$/id	cluster	botnet	total	US$/id
1%, Static	100	0	989	0.0094	100	0	808	0.0050	100	0	1,008	0.0050
1%, Adaptive	100	440	1,209	0.0765	100	505	1,061	0.0671	100	479	1,247	0.1443
10%, Static	4,406	0	43,585	0.4168	5,051	0	40,852	0.1074	4,796	0	48,343	0.1722
10%, Adaptive	4,406	4,406	45,788	0.4785	5,051	5,051	43,377	0.2220	4,796	4,796	50,741	1.2015

6.3.2 Green and useful puzzles

In this section we present and discuss the results achieved with an evaluation of the notion of green and useful adaptive puzzles. Our evaluation comprises both simulation and experimentation in the PlanetLab environment. For the sake of this evaluation, we define the additional evaluation metrics:

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

$$D = 1 - \frac{\min(\text{savings (}\textit{baseline}\text{)}, \text{savings (}\textit{solution}\text{))}}{\text{savings (}\textit{baseline}\text{)}} \tag{6.6}$$

  In the equation above, savings (*solution*) is the total energy consumption caused by our solution, and savings (*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.

- **Proportion of useful puzzles (U).** This metric indicates the ratio of assigned puzzles that were real world production jobs, compared to the total number of puzzles assigned by another mechanism regarded a baseline. Note that test jobs (i.e. those issued to assert that the attacker is not tampering by faking job results) are not considered by this metric. Formally, we have

$$U = \frac{\min(r \cdot \sum_{i=0}^{n} |\mathcal{J}_i|, \sum_{j=0}^{m} \gamma_j)}{\sum_{j=0}^{m} \gamma_j} \tag{6.7}$$

  In the equation above, r is the ratio of test jobs issue per puzzle assigned (recall that the result of part of the jobs sent as puzzles are known in advance, to difficult tampering); n is the total number of puzzles issued by our solution, and $|\mathcal{J}|_i$ is the complexity of the i-th puzzle; m is the total number of puzzles issued by the baseline solution, and γ_i is the complexity of the j-th puzzle. Possible values for this metric range in the interval $[0..1]$. A value of $U = 1$ is a global optimum; it means our solution provided the highest possible reuse of puzzles, when compared to the baseline mechanism.

Simulation

With this evaluation, besides assessing the impact of our solution to legitimate users and attackers (and also the monetary costs involved), we focus on an additional research question: what are the potential energy savings that the use of green puzzles can provide? To answer these questions, we have evaluated scenarios with and without attack, considering the following solutions: without control, based on static puzzles (as proposed by Rowaihy et al. (ROWAIHY et al., 2007)), and our solution. To evaluate them through simulation, the following aspects had to be observed: (*i*) the behavior of legitimate users, (*ii*) their computing power, (*iii*) the complexity of puzzles, and (*iv*) the goal of the attacker (strategies and resources available). Each of these aspects is discussed next.

Characteristics of the simulation environment. The simulation has 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 choice for this distribution is supported on a recent study by Maier, Schneider, and Feldmann (MAIER; SCHNEIDER; FELDMANN, 2011), which suggests that the number of unique end-hosts behind networks using NAT – in residential DSL lines of a major European ISP – follows an exponential distribution. The arrival 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 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.

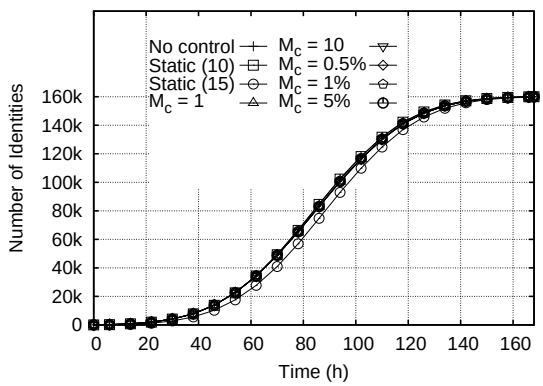
To model the delay incurred from puzzle-solving, we consider that a puzzle of complexity $\gamma_i(t_k)$ takes $2^6 + 2^{\gamma_i(t_k)-1}$ seconds to be solved using the standard, off-the-shelf hardware; a computer twice as fast takes half of that time. As for the waiting time, we consider that a factor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

To obtain the fake accounts we consider two scenarios for the attacker (inspired in the attack model shown in Section 4.2): one in which the attacker increases the computing power (using a cluster of high-performance computers, i.e., 2.5 times faster than the reference hardware), or increase both the computing power and the number of distinct sources from which the malicious requests depart (using a botnet of high-performance computers).

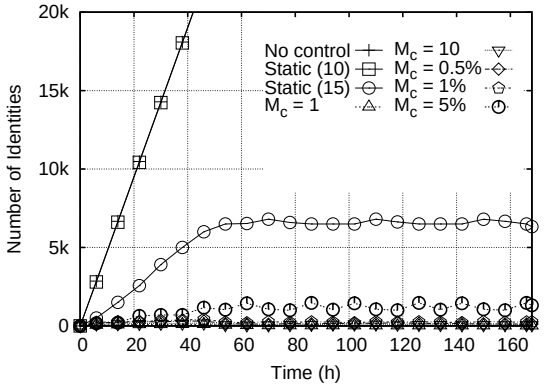
The other parameters in our evaluation are defined as follows. For green and useful puzzles, $\Delta t = 48$ hours and $\beta = 0.125$. We use $\Gamma_{req} = 15$ (as maximum possible complexity when the user does not have a valid identity), $\Gamma_{reval} = 14$ (when the user has a valid but expired 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 mechanism based on static puzzles, 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). We also compare the performance of green and useful puzzles solution with the adaptive puzzles one; we set the maximum value of puzzle complexity as $\Gamma = 17$, $\Delta t = 48$ hours, and $\beta = 0.125$.

Effectiveness in mitigating fake accounts. Figure 6.10 shows the results achieved when the attacker increases the computing power as an strategy to control more counterfeit identities. The scenarios considered are one in which the attacker has $M_u = 10$

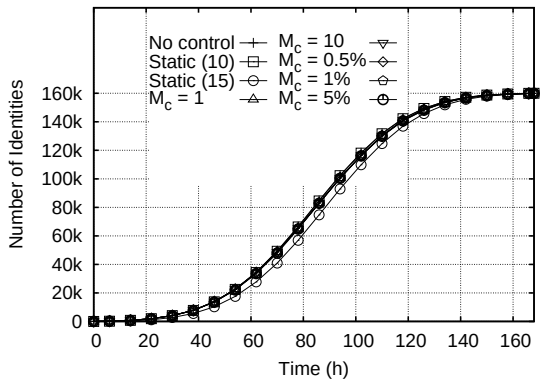
sources in his/her control (Figures 6.10(a) and 6.10(b)), and another in which he/she controls $M_u = 500$ sources (Figures 6.10(c) and 6.10(d)). The number of high-performance computers available for the attack is defined proportionally to the number of legitimate sources considered: $M_c = 1$ computer; $M_c = 10$; $M_c = 0.5\%$ (50 computers); $M_c = 1\%$ (100 computers); and $M_c = 5\%$ (500 computers). In the attack scenarios where static puzzles are used, the number of high-performance available for the attack is $M_c = 5\%$. For the sake of legibility, we omit the results obtained with adaptive puzzles from these figures.



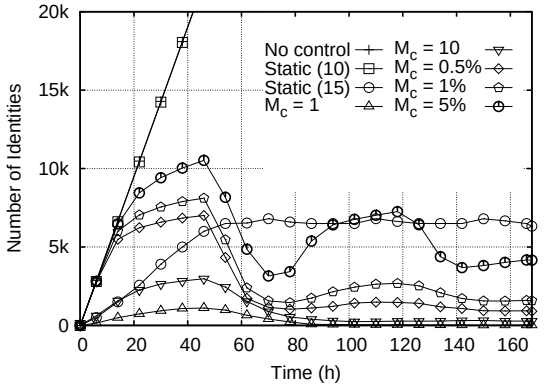
(a) Legitimate, $M_u = 10$



(b) Fake, $M_u = 10$



(c) Legitimate, $M_u = 5\%$

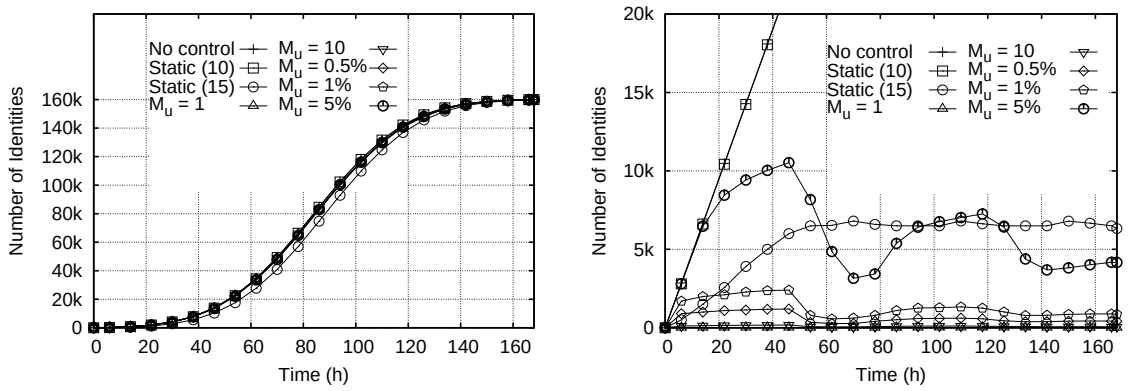


(d) Fake, $M_u = 5\%$

Figure 6.10: Number of legitimate and fake accounts, in the scenario where the attacker increases his/her computing power to solve the puzzles.

One can see in Figure 6.10(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 (ROWAIHY et al., 2007) 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 79% the number of fake accounts he/she can control, comparing to the scenario "Static (15)" (from 6,237 valid identities to 1,309). As for the overhead imposed to legitimate users, Figure 6.10(a) shows that it was negligible. Observe, however, that the use of static puzzles with complexity $\Gamma = 15$ imposed a non-negligible overhead to legitimate users.

Figure 6.10(d) evidences that increasing the computing power available is the only way the attacker can circumvent our solution; even so, he/she is not able to control more



(a) Legitimate (b) Fake

Figure 6.11: Number of legitimate and fake accounts in the *botnet* scenario.

Table 6.8: Puzzle resolution times (seconds).

Scenario		Mean	Std. Dev.	Median	9th decile
Adaptive puzzles	legitimate users, "$M_c = 10$"	3,033	8,173.412	84	8,848.0
	attacker, "$M_c = 10$"	11,950	10,264.92	6,582	26,242.0
	legitimate users, "$M_c = 5\%$"	3,830	10,890.83	80	13,474
	attacker, "$M_c = 5\%$"	11,160	13,382.31	6,582	26,242
Green and useful puzzles	legitimate 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
	legitimate 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	legitimate users, "$\Gamma = 10$"	497.6	198.5	455	596
	attacker, "$\Gamma = 10$"	412	0	412	412
	legitimate users, "$\Gamma = 15$"	15,825.4	6,032.2	14,466	19,003
	attacker, "$\Gamma = 15$"	13,110	0	13,110	13,110

identities than would happen in the case of static puzzles with complexity $\Gamma = 15$; the gain with our solution was of 34.2% when compared to static puzzles (from 6,237 identities to 4,161). Figure 6.10(c) shows that legitimate users remain unaffected. In the case of adaptive puzzles, legitimate users obtain 159,683 identities and the attacker, only 51 identities during the period of evaluation for $M_c = 10$; in the scenario where $M_c = 5\%$, 159,597 legitimate identities are created, and 25,038 fake ones.

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 he/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. Figure 6.11 shows that he/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

Table 6.9: Puzzle complexity and energy consumption (estimates).

Puzzle	Resolution	Consumption	Adaptive Puzzles				Green Puzzles			
complexity	(seconds)	(*joules*)	# leg.	KJ leg.	# mal.	KJ mal.	# leg.	KJ leg.	# mal.	KJ mal.
0	65	78.97	70,944	5,602	0	0	75,179	5,937	0	0
1	66	80.19	17,927	1,437	0	0	20,229	1,622	0	0
2	68	82.62	14,962	1,236	0	0	18,026	1,489	0	0
3	72	87.48	14,206	1,242	0	0	16,489	1,442	0	0
4	80	97.20	13,928	1,353	0	0	16,900	1,642	0	0
5	96	116.64	14,214	1,657	0	0	18,321	2,136	0	0
6	128	155.52	15,134	2,353	0	0	20,838	3,240	261	40
7	192	233.28	16,494	3,847	0	0	21,781	5,081	4,477	1,044
8	320	388.80	18,630	7,243	3,291	1,279	17,807	6,923	1,253	487
9	576	699.84	16,334	11,431	1,785	1,249	16,658	11,657	2,081	1,456
10	1,088	1,321.92	14,229	18,809	819	1,082	16,644	22,002	2,871	3,795
11	2,112	2,566.08	13,716	35,196	1,136	2,915	17,953	46,068	3,599	9,235
12	4,160	5,054.40	13,983	70,675	2,158	10,907	19,127	96,675	4,464	22,562
13	8,256	10,031.04	14,972	150,184	2,617	26,251	17,722	177,770	3,704	37,154
14	16,448	19,984.32	15,697	313,693	3,365	67,247	5,926	118,427	959	19,164
15	32,832	39,890.88	16,121	643,080	4,458	177,833	0	0	0	0
16	65,600	79,704.00	13,573	1,081,822	4,585	365,442	0	0	0	0
17	131,136	159,330.24	4,625	736,902	1,324	210,953	0	0	0	0
Total			319,689	3,087,772	25,538	865,161	319,600	502,117	23,669	94,941

sustainable; by the end of the evaluation, he/she has only a few identities more than he/she would have if static puzzles were used. Again, the overhead to legitimate users was minimal.

Table 6.8 presents the average time that legitimate users and 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 the case of green and useful puzzles, legitimate users are assigned easier-to-solve ones (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 assigned puzzles (6,224 seconds in the same scenario). More importantly, 90% of legitimate users took at most 1,751 seconds to solve the assigned puzzles; for the attacker, this time was 6,582 seconds.

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). These results evidence that adaptive puzzles and wait time represent a promising direction to tackle fake accounts.

Energy efficiency. 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[3] 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

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

observed that the consumption is constant and equals to 1.215 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 solution.

Table 6.9 gives an overview of the energy consumption caused by green puzzles for the scenario $M_u = 5\%$ depicted in Figure 6.11. 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*. For the sake of comparison, we also include the results achieved with adaptive puzzles.

The total energy consumption (summing up legitimate users and the attacker) caused by green puzzles (597 MJ) is significantly lower compared to the energy consumption measured in the case of adaptive puzzles (3,952 MJ); the savings ratio is $D = \frac{\min(3952,597)}{3952} = 0.84$. More importantly, it 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 legitimate users, and 23,174 assigned to the attacker). This represents a savings ratio of $D = \frac{\min(15530,597)}{15530} = 0.96$, and a difference of 14,945 MJ (4.15 MWh), or 28.25% of the annual energy consumption *per capita* in Brazil (IBGE, 2012). These results not only emphasize the need for "green" puzzles, but also highlight the potentialities of using waiting time to materialize them.

Monetary analysis of the attack. Next we present a budgetary analysis of the attack. Table 6.10 presents a summary of the analysis, for the scenarios related to our solution. Values fields "cluster" and "botnet" are displayed in units hired. Field "total" contains the budget of the attack in each scenario considered (displayed in US dollars). Field "US$/id" contains the cost per identity obtained by the attacker.

Table 6.10: Economic analysis of the attack scenarios evaluated with our solution.

Scenario	cluster	$M_u = 10$			$M_u = 5\%$		
		identities	total	US$/id	identities	total	US$/id
$M_c = 1$	1	6	15	2.50	17	260	15.2941
$M_c = 10$	10	63	105	1.66	278	350	1.2589
$M_c = 0.5\%$	50	119	509	4.27	902	754	0.8359
$M_c = 1\%$	100	268	1,013	3.77	1,581	1,258	0.7956
$M_c = 5\%$	500	1,309	5,045	3.85	4,161	5,290	1.2713

In the scenarios where static puzzles were used, the number of cluster instances hired was 500, at the price of US$ 0.06/hour each. In total, the budget for the attack was US$ 5,040. In the case of $\Gamma = 10$, the cost per identity obtained was US$ 0.0630, whereas in the case of $\Gamma = 15$, the cost per id was US$ 0.7965.

One can clearly see that our solution makes it extremely expensive for an attacker to obtain identities; in order to obtain the highest profit possible (in terms of identities obtained), the attacker must pay US$ 5,290 (which makes a total of US$ 1.2713 per identity). These results show that although the attacker is still able to obtain counterfeit identities, he/she has to dedicate a large fraction of budgetary resources to do so. More importantly, the cost per identity our solution imposes to attacker overcomes the expected profit one can make with fake accounts (US$ 0.10, as discussed in the previous section).

Assuming an attack scenario in which the attacker has a ratio of one zombie computer

in the botnet for two identities he/she wants to control, the required budget would be $\frac{80{,}000}{2} \times \text{US\$}\,0.50 = \text{US\$}\,20,000$. Note that this is a scenario in which the attacker is able to subvert our solution with marginal computing power for the puzzle-solving process (therefore, not being required to hire a cluster for that purpose). Although the cost per identity is only US$ 0.25, the required budget is significantly higher, thus proving our argument that the attacker in fact has to dedicate a large fraction of resources to launch a successful attack.

Experimental evaluation using PlanetLab

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

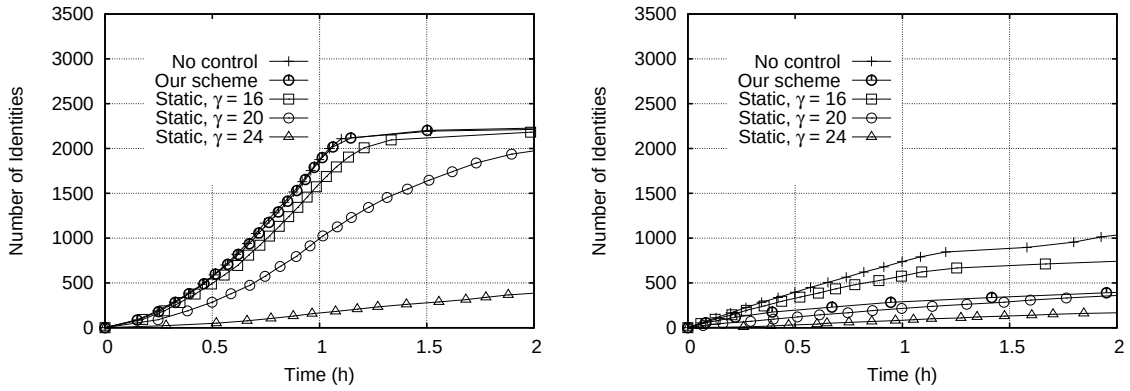
In this evaluation we consider 240 legitimate sources and 20 malicious ones. The legitimate users request 2,400 identities during one hour. The first request of each user is uniformly distributed during this period; their recurrence follows an exponential distribution, varying from 1 to 15. The interval between arrivals is also exponentially distributed, between 1 and 10 minutes. The attacker requests 1,200 identities ($1/3$ of the requests of 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) 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 solution 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 $2^{\gamma_i(t)-1}$. For the mechanism based on static puzzles, we considered the one proposed by Douceur (DOUCEUR, 2002) (discussed in Section 6.1).

The other parameters were defined as follows. For our solution, $\Delta t = 48$ 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.

Figure 6.12 shows that the dynamic of identity assignments to legitimate users with the proposed solution (curve "Our solution") 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 solution reduced significantly the number of fake accounts he/she created (compared to the scenario without control).

The estimates of energy consumption obtained also indicate the efficacy of our solution. 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 solution led to only 13.39 KJ. It represents 22.81% ($D_{\gamma_{pl}=16} = 0.7718$), 2.41% ($D_{\gamma_{pl}=20} = 0.9749$), and 1.66% ($D_{\gamma_{pl}=24} = 0.9833$) of the estimated consumption with static puzzles.



(a) Legitimate (b) Malicious

Figure 6.12: Results achieved with the PlanetLab environment.

Table 6.11: Number of jobs assigned per puzzle complexity.

Puzzle complexity	Number of jobs	Useful Puzzles # assigned	Useful Puzzles total jobs	Puzzle complexity	Number of jobs	Useful Puzzles # assigned	Useful Puzzles total jobs
1	2	911	1,822	10	1,024	195	199,680
2	4	130	520	11	2,048	181	370,688
3	8	125	1,000	12	4,96	77	315,392
4	16	137	2,192	13	8,192	57	466,944
5	32	142	4,544	14	16,384	44	720,896
6	64	132	8,448	15	32,768	41	1,343,488
7	128	124	15,872	16	65,536	47	3,080,192
8	256	100	25,600	17	131,072	64	8,388,608
9	512	113	57,856	18	262,144	37	9,699,328

Total	# assigned	total jobs
	2,657	24,703,070

Finally, we evaluate how useful our solution can be, compared to traditional puzzles. Observe from Table 6.11 that the total number of simulation jobs assigned to users requesting identities (both legitimate users and the attacker) was 24,703,070. In the case of static puzzles with $\gamma_{pl} = 16$, $\gamma_{pl} = 20$, and $\gamma_{pl} = 24$, these numbers were 191,889,408 (2928 puzzles), 2,451,570,688 (2338 puzzles), and 8,992,587,776 (536 puzzles) respectively. Considering the metric of proportion of useful puzzles assigned compared to static ones, and assuming $r = 0.5$ (i.e. only half of the assigned jobs were not test jobs), we have $U_{\gamma_{pl}=16} = 0.064$, $U_{\gamma_{pl}=20} = 0.005$, and $U_{\gamma_{pl}=24} = 0.001$. It is important to observe that, although the proportion of useful simulation jobs issued is small compared to the number of puzzles assigned by the baseline mechanism, our solution is the only that actually assigns such useful jobs.

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.

6.4 Analytical modeling and analysis

Having assessed the effectiveness of our solution considering traces of identity requests (synthetically generated and real ones), in this section we evaluate our solution by means of analytical modeling. Here we focus on the following research questions: (i) is the proposed solution able to scale in environments comprised of millions of users, over long periods of time? And (ii) would an attacker, being aware of the inner-working of the mechanism, be able to subvert it? If so, under which conditions? To answer these questions, we present in the following subsections a mathematical model derived from our conceptual strategy (Subsection 6.4.1), and discuss the strategies that an attacker can use to tamper with our design for identity management (Subsection 6.4.2). We close this section with an analysis, extrapolating to millions of users and in a period of one year, whose goal is to assert that our design clearly limits the power of an attacker to obtain a widespread number of counterfeit identities, whereas imposes marginal overhead to legitimate users (Subsection 6.4.3).

6.4.1 Model

We assume that legitimate requests arrive following some distribution $f_{leg}(t;\ldots)$, whereas malicious ones arrive following another distribution $f_{mal}(t;\ldots)$. Note that either distribution can be Gaussian, Exponential, Uniform, or Fixed. In this case, we can derive as $\lambda_{leg}(t) = S_{leg}^{-1} \cdot f_{leg}(t;\ldots)$ the arrival rate of legitimate requests, and $\lambda_{mal}(t) = S_{mal}^{-1} \cdot f_{mal}(t;\ldots)$ for malicious requests. In these equations, S_{leg} and S_{mal} represent the number of legitimate and malicious sources, respectively.

The analysis in our model evolves in rounds, where each round has duration Δt. In this case, the number of identity requests within a given round i ($\forall i > 0$) is given by Equation 6.9. In this equation, k can be either leg or mal, and R_k is the number of identity requests of type k. For the sake of analysis, we assume that the attacker aims to obtain (and control) $1/3$ of identities in the network. Furthermore, in this model legitimate users and the attacker do not share sources. In the following section we discuss those scenarios in which legitimate users share sources with the attacker.

$$\Lambda_k(i) = R_k \int_{(i-1)\cdot\Delta t}^{i\cdot\Delta t} \lambda_k(t)\,\mathrm{d}t \tag{6.8}$$

To model identity renewal in our model, we define a function $\Lambda'_k(i)$ (computed following Equation 6.9), which is the sum of the number of users that arrived for the first time in the network (defined by $\Lambda_k(i)$), and the number of identities that expired in the previous round and thus must be renewed (see Equation 6.16), $\Psi_k(i-1)$. For simplicity, we assume here that both legitimate users and attackers renew their identities once expired.

$$\Lambda'_k(i) = \frac{\Psi_k(i-1)}{S_k} + \Lambda_k(i) \tag{6.9}$$

The rate in which identities are granted per second (service rate), defined for a given round i, is given by Equation 6.10 (for $k \in \{leg, mal\}$). In this equation, P_k is the average computing power of the hardware 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 other variables in this equation have the same meaning as discussed in Section 5.1. The values of $\gamma_k(i)$ and $\omega_k(i)$ are computed according to Equations 6.1 and 6.4, respectively.

$$\mu_k(i) = P_k \cdot \frac{1}{2^{\gamma} + 2^{\gamma_k(i)} + 2^{\omega} + 2^{\omega_k(i)}} \tag{6.10}$$

For example, assume that a given legitimate user has an average computing power equivalent to the capacity of a standard, off-the-shelf hardware considered as reference (i.e. $P_k = 1$). Assume also the following parameter setting: $\gamma = 4$, $\omega = 0$, $\Gamma = 10$, and $\Omega = 16$. For a measured value of trust score equals to $\theta_k(i) = 0.5$, the rate in that user can obtain identities is $\mu_k(i) \approx 0.00075$ (a). If the trust score is $\theta_k(i) = 0.9$, we have $\mu_k(i) \approx 0.00096$ (b). With a computing power of $P_k = 2$ (twice faster than the standard, off-the-shelf hardware), we have $\mu_k(i) \approx 0.00148$ when $\theta_k(i) = 0.5$ (c), and $\mu_k(i) \approx 0.00194$ when $\theta_k(i) = 0.9$ (d). Note that $\mu_k(i)$ increases with higher values of trust scores and also with the increasing user's computing power. Conversely, it decreases with higher values of γ, ω, Γ, and Ω.

Based on the service rate defined above, the maximum number of identities that one can obtain in each round can be computed using Equation 6.11.

$$M_k(i) = \Delta t \cdot \mu_k(i) \tag{6.11}$$

Considering a size of sliding window $\Delta t = 28,800$ (eigth hours), in the four cases illustrated in the previous paragraphs we have the following number of identities the user can obtain: 21.6 (a), 27.65 (b), 42.63 (c), and 55.87 (d).

An important input for defining the service rate (as shown in Equation 6.10) is the average trust score of the sources of identity requests (either legitimate or malicious). In order to estimate it, we first need to estimate the average recurrence of sources, and the network recurrence rate. The former is given by Equation 6.12, whereas the latter is given in Equation 6.14.

$$\Delta\hat{\phi_k}(i) = \min(\Lambda'_k(i) + \psi_k(i-1)\,,\, M_k(i-1)) \tag{6.12}$$

The rationale for computing $\Delta\hat{\phi_k}(i)$ is the following. In each round i, $\Lambda_k(i)$ new identity requests arrive in the system. However, the number of identities effectively granted for some (or several) round(s) can be smaller ($\Lambda_k(i) > M_k(i)$). In practice, it means that the users whose requests were not yet granted are either solving the assigned puzzle or respecting the waiting period. As a consequence, there will be a number of identity requests that will not be granted an identity in that round, and therefore will retry in the next round. Therefore, for computing $\Delta\hat{\phi_k}(i)$ in the i-th round, we take the minimum between the number of identity requests that arrived in that round $\Lambda_k(i)$ plus the number of requests that arrived in the previous rounds, but have not yet received an identity $\psi_k(i-1)$, and the number of requests that were granted an identity in the previous round $M_k(i-1)$. Observe that the mechanism for identity management in our framework is recurrent; the service rate in the next round depends on the measured arrival rate in the previous one.

$$\psi_k(i) = \Lambda'_k(i) + \psi_k(i-1) - \Delta\hat{\phi_k}(i) \tag{6.13}$$

Similarly, the number of identity requests that have not been granted an identity and therefore will retry in the next round ($\psi_k(i)$, given in Equation 6.13) is defined as the difference between the number of identities that arrived and the estimated value of $\Delta\phi_k(i)$.

The network recurrence rate (Equation 6.14) is given by the minimum between 1 (in case there is no request yet granted an identity) and the average of legitimate and malicious identity requests that were granted an identity. 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 or attackers have obtained identities during that round), it returns 1.

$$\hat{\Phi(i)} = \max\left(1\,,\, \frac{1}{\max\left(1, |\mathcal{K}|\right)} \cdot \sum_{k \in \mathcal{K}} \Delta\hat{\phi_k(i} - 1)\right) \tag{6.14}$$

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 the size of the sliding window Δt, i.e. $\lfloor {}^{T}/_{\Delta t} \rfloor$) is given by $G_k(i)$, as shown in Equation 6.15. In this equation, v is the number of rounds in which an identity is valid, and is given by the maximum validity $v = {}^{V}/_{\Delta t}$.

$$G_k(i) = \left\lfloor S_k \cdot \sum_{j=i-v}^{i} \hat{\Delta\phi_k}(j) \right\rfloor \tag{6.15}$$

It is important to keep track of the number of identities that expired, so that they re-enter the system as new identity requests. This number, denoted as $\Psi_k(i)$ for the i-th round, and computed according to Equation 6.16, is defined as the difference between the total number of identity 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 an identity.

$$\Psi_k(i) = S_k \cdot \sum_{j=0}^{i} \Lambda_k(j) - (G_k(i) + \psi_k(i) \cdot S_k) \tag{6.16}$$

6.4.2 Analysis of possible attack strategies

We use the model described earlier to answer the following research question posed in the introduction: would an attacker, being aware of the inner-working of the mechanism, be able to subvert it? If so, under which conditions? To answer this question, in this analysis we assume that the goal of the attacker is to obtain (and control) a minimum of $1/3$ of identities in the network.

Given such goal, there are two strategies that an attacker can attempt to achieve it more quickly: augment its computing power (to solve the puzzles faster), and increase the number of sources under his/her control (to perform requests with higher values of trust scores, and thus receive puzzles of lower complexity). While the former strategy is already known in the literature, the latter is specific to our design.

We argue that an attacker, in possession of a finite amount of computational resources and controlling a certain number of sources of identity requests, must evenly divide the number of identities to be requested among the sources in his/her control, and request identities at a fixed rate, to maximize the profit of the attack. Based on the previous statement, and taking into account the mathematical design of our design, we derive the following theorems:

Theorem 3. *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 evenly divide the S_{mal} requests among the R_{mal} sources.*

Proof. Consider $\gamma = \omega = 0$, and $P_{mal} = 1$ (i.e. the attacker has an average computing power per source equals to the computing power of a standard, off-the-shelf hardware

used as reference). In order to obtain puzzles of lower complexity from the bootstrap service, the value of $\theta(t)$ of the sources in hands of the attacker 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 counterfeit 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 to our scheme. Suppose that, by doing this, the recurrence of sources in hands of the attacker becomes equal to $\Phi(t)$ (i.e. $\Delta\phi_i(t) = \Phi(t)$, where i is the index of a source in hands of the attacker). As a consequence, we have $\theta_i(t) = 0.5$. From Equation 6.10, the attacker's capability to solve puzzles is then $\mu = {}^{1}\!/_{(2^{0.5\cdot\Gamma} + 2^{0.5\cdot\Omega} + 2)}$.

To show that evenly dividing all requests to be performed among all sources is the best strategy for the attacker, suppose a different, generic attack strategy in which counterfeit requests are *not* evenly divided. Therefore, for some source(s), its recurrence will be larger than $\Phi(t) = \frac{r}{u}$. Conversely, for the rest it 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 total number of identities which remain to be requested is $k \cdot x$, so that all r identity requests are performed. These requests must be originated from the remaining $u - k$ sources. If the remaining requests are evenly distributed among them, 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 the average recurrence of the network $\Phi(t) = \frac{r}{u}$; 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 $\mu'_2 \ll \mu \ll \mu'_1$. One may see 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 puzzles has an exponential growth rate. This proof shows that any attack strategy that does not evenly divide requests among sources will result in even more complex puzzles being assigned to requests. □

Theorem 4. *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. Let Δt be the sliding window considered for computing the recurrence of a source $\Delta\phi(t)$ and of the network $\Phi(t)$. Consider also that the time period begins in T_0 and ends in T_f. For the sake of simplicity, we consider that the attacker controls one single source of identity requests. The results presented here, however, can be straightforwardly generalized for the case of an attacker having multiple sources.

In order to obtain puzzles of lower complexity from the bootstrap service, the value of $\theta(t)$ of the source in hands of the attacker 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 the same value of the measured network recurrence rate, $\Phi(t)$. To achieve this, and at the same time perform r counterfeit 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 his/her source low within the sliding window Δt, and

thus appear less suspicious to our scheme. Suppose that, by doing this, the recurrence of sources in hands of the attacker becomes equal to $\Phi(t)$. From Equation 6.10, the attacker's capability to solve puzzles is then $\mu = {}^{1}/_{(2^{0.5\cdot\Gamma} + 2^{0.5\cdot\Omega} + 2)}$.

To show that evenly dividing all requests to be performed along T is the best strategy for the attacker, suppose a different, generic attack strategy in which counterfeit requests are *not* evenly divided. 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 total number of identities which remain to be requested is x, so that all r identity requests are performed. These requests must be performed in the remaining $T_f - T_i$ period of time. If the remaining requests are evenly distributed among them, 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 $\mu'_2 \ll \mu \ll \mu'_1$. One may see 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 shows that any attack strategy that does not evenly divide requests throughout the time will result in even more complex puzzles being assigned to requests. □

Theorem 5. *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 his/her number of identity requests will be significantly larger (in compliance with the goal of obtaining fake accounts). □

From the discussion and the theorems presented above, we conclude the following about the strategy that maximizes the attacker's profit: (i) the identity requests to be performed must be evenly divided among the sources in control of the attacker; (ii) these requests must be uniformly spread during the period of the attack; and (iii) the attacker must avoid sources that are shared with other users.

6.4.3 Dynamics of identity assignment

We employed the mathematical model described earlier to analyze how our mechanism would behave in extreme conditions (e.g. millions of users, intense workload, etc.), and also to verify the results achieved with simulation and experimentation (to be discussed next). In this subsection we cover some of the results achieved.

Figure 6.13 presents the results of an evaluation to understand how our mechanism would behave with a very large number of users, during a long period of time. In this analysis, we considered a number of 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 a number of 1.8 million identities in the same period (in order to make $1/3$ of the identities in the system). We considered the following distributions for the arrival of legitimate users: fixed rate, Gaussian, and Exponential. As for the arrival of malicious identity requests, we considered a fixed distribution (which is in line with the conclusions drawn in the previous subsection). The other parameter setting for the analysis are: $P_{leg} = P_{mal} = 1$; $\Delta t = 48$ hours; $\beta = 0.125$; $\gamma = 6$; $\Gamma = 13$; $\omega = 0$; and $\Omega = 17$. The identities assigned are valid for a period of four days, or twice the duration of the sliding window.

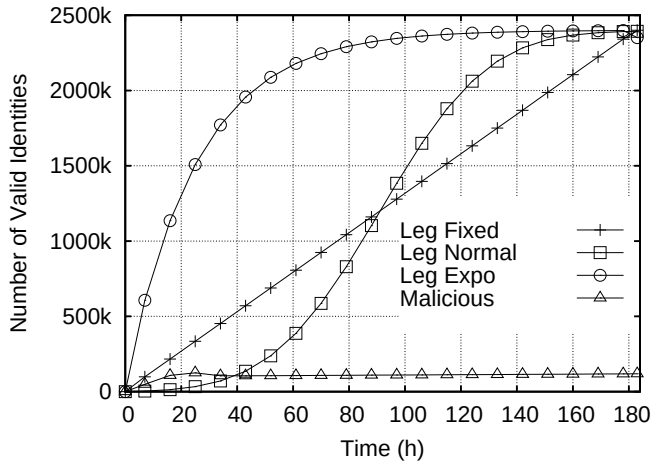


Figure 6.13: Number of valid identities for legitimate and malicous users.

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 severely constrained in resources, and thus was able to achieve only 6% of the initial 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, far less than the initial goal of controlling $1/3$ (or $\approx$ 34%) of valid identities.

6.5 Considerations

In this section, we present some considerations regarding the use of adaptive, green, and useful puzzles. With respect to the validity of a puzzle, an attacker may explore a period of time in which the source he/she is associated to achieves a high trust score, and request a potentially high number (e.g., millions) of identities. Because the trust score varies only with the number of identities *granted* (rather than the number of identities *requested*), the attacker would obtain millions of easy-to-solve puzzles in this scenario.

The attacker would then solve the received puzzles and obtain millions of identities at a significantly lower cost (in contrast to the case in which a new identity request comes only after the puzzle of the previous request is solved).

To prevent this attack, puzzles may carry a validity timestamp, defined based on the time it is expected to be solved using a standard, off-the-shelf hardware. Once this timestamp is expired, the puzzle is no longer valid, and its solution may be refused by the bootstrap service. To remediate the case in which low capable machines do not timely solve the puzzles, another puzzle may be assigned, having a validity timestamp that is double the time it is expected to be solved using a standard hardware (and so on).

There are also three important considerations regarding the adoption of green and useful puzzles in the context of identity management. With regard to using wait as a complement to puzzles, a possible attack 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 TASKCOMPLETED message sent for indicating the wait is finished); if the difference between these values exceeds a treshold $\Delta\theta$ (i.e., $\theta'_i(t) - \theta'_i(t') \geq \Delta\theta$), 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 $\theta'_i(t)$ (the trust score of his/her request) decreased more than $\Delta\theta$ (maximum tolerable) since the request initiated.

Now focusing on the use of jobs as an alternative to cryptographic puzzles, 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 (AHN et al., 2008)). In this case, the attacker will not be able to distinguish which are "test jobs" and "real jobs"; the attacker will have to solve all of them correctly to avoid the risk of faking the result for a test job, and having his/her request process terminated by the bootstrap.

Finally, it is important to emphasize that both our solution and those based on traditional puzzles must verify puzzle solution for each identity requested. Therefore, assuming that each solution receives a same number of identity requests, energy consumption due to puzzle verification will be similar. In other words, the energy consumption caused by puzzle solution verification is approximately the same regardless of the puzzle-based approached used to limit fake accounts. This observation allows us to claim that the energy required for puzzle verification does not affect the green aspect of our solution, when compared to existing puzzle-based approaches.

6.6 Summary

The use of computational puzzles to limit the spread of fake accounts has been hampered 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 neither take advantage of the discrepant behavior observed between legitimate users and attackers as an approach to weight the complexity of assigned puzzles. To address these limitations, in this chapter we proposed the use of adaptive puzzles as a controlling factor to Sybil attacks.

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

In this chapter we also proposed a novel, lightweight solution for long-term identity management, based on adaptive puzzles, waiting time, and massive distributed computing. Our proposal enables the assignment of puzzles that consume resources in an efficient and useful manner. The results achieved with green and useful puzzles have shown that it is possible to force potential attackers to pay substantially higher costs for each identity; legitimate users received more easier-to-solve puzzles than the attacker, and took 52.9% 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 77.1% when compared to static puzzles (ROWAIHY et al., 2007)). 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. 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.

7 CONCLUSIONS

The goals of this chapter are threefold: summarize the thesis, present the conclusions, and discuss future work. First, the list of incremental steps to support the hypothesis in this thesis is presented. Second, the main contributions are summarized and discussed. Third, and last, prospective directions for future research are shown.

7.1 Summary of contributions

In this thesis, we proposed a framework for adaptively pricing identity requests as an approach to limit the spread of counterfeit identities (Sybils). Our framework is based on the hypothesis that **"the sources in hands of attackers launch a significantly higher number of identity requests than those associated to presumably legitimate users."** Based on it, we formulated a model that derives values of trust scores based on the frequency that sources of identity requests obtain identities, in comparison to the average number of identities already granted to other sources in the network. The model offers support for the concept of adaptive puzzles – a proof of work strategy that limits the spread of fake accounts in the network by making it extremely expensive for potential attackers to obtain a single identity, while keeping the price to be paid per identity significantly lower for legitimate users. In our research, we also reshaped traditional cryptographic puzzles in order to propose a lightweight design for *green* and *useful* identity management; green because cryptographic puzzles should consume as lower resources (e.g., energy) as possible for their resolution; and useful because puzzle solutions should have value for some external production system. The four main contributions of this thesis and a brief summary follow.

First contribution. A trust score model for pricing identity requests.

Summary. This contribution forms the basis of our framework for adaptively pricing identity requests (MAUCH et al., 2010; CORDEIRO et al., 2011). An extensive evaluation has shown the potentialities of using values of trust score as a solution to measure the reputation of identity requests, and thus price legitimate and malicious identity requests accordingly; presumably legitimate requests received significantly higher values of trust score, whereas a vast majority of the malicious request received lower values, even in scenarios where the attacker had a large fraction of resource for launching the attack. These observations showed that the proposed trust score model not only has potential to mitigate the dissemination of fake accounts, but also required a significant effort from the attacker to manipulate it.

Second contribution. The use of adaptive (computational) puzzles.

Summary. The concept of trust score introduced in the context of this thesis enables an important advance in the use of proof of work to limit fake accounts – from computational puzzles of fixed complexity to adaptive ones (CORDEIRO et al., 2012). Presumably legitimate users, whose requests receive higher values of trust score, are benefited with easier-to-solve puzzles; conversely, potential attackers are penalized with more complex puzzles for each identity request made to the bootstrap service. The evaluation has shown that attackers become severely constrained in resources, and thus cannot keep up the rate in which they create fake accounts considering both a scenario without control, and also scenarios in which puzzles of fixed complexity are used.

Third contribution. The notion of *green* and *useful* computational puzzles.

Summary. Traditional puzzles consume a non-negligible amount of resources – more specifically, energy; when taking into account the energy consumed by all users interested in joining a system, the values become significant. More importantly, these resources are ultimately wasted, given that no useful information is processed when solving traditional cryptographic puzzles. In the context of this thesis, we have addressed this problem by proposing green and useful puzzles – which take less energy for their resolution without losing effectiveness, and that enable the processing of useful data as a challenge for obtaining access to a service (in this case, identities) (CORDEIRO et al., 2012, 2013). An extensive evaluation has shown that green puzzles are as effective as traditional ones, with the benefit of saving significant amounts of energy. From experimentation in the PlanetLab environment, we also verified the technical feasibility of making puzzles useful – using a software that emulates a small simulation experiment.

Fourth contribution. A methodology for correcting traces of users' usage sessions.

Summary. The comprehension of users' behavior is paramount for evaluating improvements to networked, distributed systems. To this end, several strategies have been proposed to obtain traces based on the capture of usage information, which can then serve for evaluation purposes. One main strategy consists of taking snapshots of online users. In spite of its popularity, related proposals have fallen short in ensuring accuracy of obtained data. Due to system-specific limitations, users may fail to appear in some snapshots, although online. To bridge this gap, we proposed a methodology to correct ill-collected snapshots and build more accurate traces from them (CORDEIRO et al., 2014). In summary, we estimate the probability that a given snapshot is missing some users. The snapshot is corrected if the probability exceeds a given threshold. An evaluation using ground-truth data assessed the effectiveness of our methodology, and provided important evidence that estimating failure probability is a feasible approach for improving trace accuracy.

Given the aforementioned evolution of this thesis, highlighting the contributions and lessons learned, the next section closes the thesis with a general discussion about our achievements and presents prospective directions for future research.

7.2 Final remarks

The work presented in this thesis investigated the problem of dissemination of fake accounts under a different perspective. Previous solutions attempted to limit their existence in the network using certification authorities, behavior analysis (this is the case for example of reputation systems), or proof of work, to cite a few strategies. In this thesis, we proposed a different direction to tackle fake accounts, namely the observation of profile of request patterns of sources of identity requests. Using this strategy, we have shown that it is possible to filter out presumably legitimate requests from those potentially malicious.

In conclusion, the overall work presented in this thesis underscored the importance of adopting security measures to mitigate malicious behavior in large-scale distributed systems. In the absence of countermeasure mechanisms, even a small proportion of fake accounts is able to subvert systems such as file sharing communities (SANTOS et al., 2010). The introduction of a framework that adaptively prices identity requests and renewals in this thesis demonstrated the efficacy of the concept of trust score as an index for measuring the likeliness an identity request is part of an ongoing Sybil attack. The simple, yet powerful, trust score model thwarted attacks *en masse* and protected the system against the widespread dissemination of fake accounts.

In spite of the progresses reported in the thesis, promising opportunities for future research remain. The most prominent one is the instantiation of our framework for limiting the dissemination of fake identities in online social networks. The basic idea is to create an admission process in which newly created accounts are regarded as "verified" only after a number of users already in the system (i.e., which already have verified accounts) have vouched for (i.e., "introduced") them. Our investigation will consider the hypothesis that the majority of potential users already have real-life friends registered in the network. Therefore, a user creating an account must indicate a number of other users having already verified accounts to vouch his/her own identity.

REFERENCES

ABERER, K.; DATTA, A.; HAUSWIRTH, M. A decentralized public key infrastructure for customer-to customer e-commerce. **Intl. Journal of Business Process Integration and Management**, [S.l.], v.X, n.X, p.26–33, 2005.

AHN, L. von et al. reCAPTCHA: human-based character recognition via web security measures. **Science**, [S.l.], v.321, n.5895, p.1465–1468, 2008.

Alexa Internet, Inc. **Alexa toolbar**. Disponível em: <http://www.alexa.com/toolbar>. Acesso em: junho 2012.

AMAZON.COM. **Amazon EC2 Pricing**. Disponível em: <http://aws.amazon.com/ec2/pricing/>. Acesso em: setembro 2012.

ANGWIN, J.; SINGER-VINE, J. Selling You on Facebook. **Wall Street Journal**, [S.l.], April 2012. Disponível em: <http://online.wsj.com/article/SB10001424052702303302504577327744009046230.html>. Acesso em: junho 2012.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: a survey. **Computer Networks**, [S.l.], v.54, n.15, p.2787 – 2805, 2010.

BITSOUP.ORG. **Bitsoup.org – The Number One Site for your Torrent Appetite**. Disponível em: <http://bitsoup.org/>. Acesso em: fevereiro 2010.

BLOND, S. L. et al. Spying the World from your Laptop. In: USENIX CONFERENCE ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET'10), 3. **Proceedings...** [S.l.: s.n.], 2010. p.1–8.

BORISOV, N. Computational Puzzles as Sybil Defenses. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2006), 6. **Proceedings...** [S.l.: s.n.], 2006. p.171–176.

CAO, Q. et al. Aiding the detection of fake accounts in large scale social online services. In: USENIX CONFERENCE ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI 2012), Berkeley, CA, USA. **Proceedings...** USENIX Association, 2012. p.15–15.

CASTRO, M. et al. Secure Routing for Structured Peer-to-Peer Overlay Networks. In: USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION (OSDI 2002), 5. **Proceedings...** [S.l.: s.n.], 2002. p.299–314.

CBC BOOKS. **The true story behind Sybil and her multiple personalities**. Disponível em: <http://www.cbc.ca/books/2011/12/sybil-exposed.html>. Acesso em: junho 2013.

COHEN, B. **Incentives Build Robustness in BitTorrent**. Disponvel em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.1911>. Acesso em: janeiro 2013.

CORDEIRO, W. et al. Securing P2P Systems from Sybil Attacks through Adaptive Identity Management. In: INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT (CNSM 2011), 7. **Proceedings...** [S.l.: s.n.], 2011. p.1–6.

CORDEIRO, W. et al. Identity management based on adaptive puzzles to protect P2P systems from Sybil attacks. **Computer Networks**, [S.l.], v.56, n.11, p.2569 – 2589, 2012.

CORDEIRO, W. et al. Segurança Verde: usando desafios com espera adaptativa para conter sybils em redes par-a-par. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E DE SISTEMAS DISTRIBUÍDOS (SBRC 2012), 30. **Anais...** [S.l.: s.n.], 2012. p.17–30.

CORDEIRO, W. et al. Make it Green and Useful: reshaping puzzles for identity management in large-scale distributed systems. In: IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM 2013). **Proceedings...** [S.l.: s.n.], 2013. p.400 –407.

CORDEIRO, W. et al. Were You There? Bridging the Gap to Unveil Users' Online Sessions in Networked, Distributed Systems. In: CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2014) (SUBMITTED), 33. **Proceedings...** IEEE, 2014.

COSTA, C.; ALMEIDA, J. Reputation Systems for Fighting Pollution in Peer-to-Peer File Sharing Systems. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2007), 7., Galway, Ireland. **Proceedings...** IEEE Press, 2007. p.53–60.

CPU BENCHMARK. **PassMark CPU Benchmarks - Common CPU's**. Disponível em: <http://www.cpubenchmark.net/common_cpus.html>. Acesso em: agosto 2013.

CUEVAS, R. et al. Is content publishing in BitTorrent altruistic or profit-driven? In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES (CO-NEXT '10), 6. **Proceedings...** [S.l.: s.n.], 2010.

DABEK, F. et al. Vivaldi: a decentralized network coordinate system. **SIGCOMM Computer Communications Review**, New York, NY, USA, v.34, n.4, p.15–26, October 2004.

DANEZIS, G.; MITTAL, P. SybilInfer: detecting sybil nodes using social networks. In: NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM (NDSS 2009), San Diego, California, USA. **Proceedings...** The Internet Society, 2009.

DOE, J. **Buy Gmail Accounts**. Disponível em: <http://buygmailaccounts.org/>. Acesso em: janeiro 2013.

DOUCEUR, J. R. The Sybil Attack. In: INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS 2002), 1. **Proceedings...** [S.l.: s.n.], 2002. p.251–260.

DRUSCHEL, P.; ROWSTRON, A. I. T. PAST: a large-scale, persistent peer-to-peer storage utility. In: WORKSHOP ON HOT TOPICS IN OPERATING SYSTEMS (HOTOS 2011), 8. **Proceedings...** IEEE Computer Society, 2001. p.75–80.

EDWARDS, J. Facebook Targets 76 Million Fake Users In War On Bogus Accounts. **Business Insider**, [S.l.], March 2013. Disponível em: <http://www.businessinsider.com/facebook-targets-76-million-fake-users-in-war-on-bogus-accounts-2013-2>. Acesso em: abril 2013.

ELLISON, C. Establishing Identity Without Certification Authorities. In: USENIX SECURITY SYMPOSIUM, 6. **Proceedings...** [S.l.: s.n.], 1996. p.67–76.

ENGLE, M.; KHAN, J. I. **Vulnerabilities of P2P Systems and a Critical Look at their Solutions**. Technical Report 2006-11-01, Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University. Disponível em: <http://www.medianet.kent.edu/techreports/TR2006-11-01-p2pvuln-EK.pdf>. Acesso em: agosto 2012.

ENIGMA @ HOME. **CPU / OS stats sorted by total credits granted**. Disponível em: <http://www.enigmaathome.net/cpu_os.php>. Acesso em: agosto 2013.

FELDMAN, M. et al. Free-riding and whitewashing in peer-to-peer systems. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.24, n.5, p.1010–1019, 2006.

FIEGE, U.; FIAT, A.; SHAMIR, A. Zero Knowledge Proofs of Identity. In: ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING (STOC '87), 9., New York, NY, USA. **Proceedings...** ACM, 1987. p.210–217.

GASPARY, L. P.; BARCELLOS, M. P. Fundamentos, Tecnologias e Tendências rumo a Redes P2P Seguras. **Atualizações em Informática**, Rio de Janeiro, Brazil, p.187–244, 2006.

GAYLE, D. YouTube cancels billions of music industry video views after finding they were fake or 'dead'. **Mail Online (Daily Mail)**, [S.l.], December 2012. Disponível em: <http://www.dailymail.co.uk/sciencetech/article-2254181/YouTube-wipes-billions-video-views-finding-faked-music-industry.html>. Acesso em: fevereiro 2013.

GUO, L. et al. A performance study of BitTorrent-like peer-to-peer systems. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.25, n.1, p.155–169, 2007.

HANSEN, M.; SCHWARTZ, A.; COOPER, A. Privacy and Identity Management. **Security Privacy, IEEE**, [S.l.], v.6, n.2, p.38–45, 2008.

HARDIN, G. The Tragedy of the Commons. **Science**, [S.l.], v.162, n.3859, p.1243–1248, 1968.

HOÃŸFELD, T. et al. Characterization of BitTorrent swarms and their distribution in the Internet. **Computer Networks**, [S.l.], v.55, n.5, p.1197–1215, 2011.

IBGE. **Sustainable Development Indexes 2012 (in Portuguese)**. Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/noticia_visualiza.php?id_noticia=2161>. Acesso em: abril 2012.

JETTER, O.; DINGER, J.; HARTENSTEIN, H. Quantitative Analysis of the Sybil Attack and Effective Sybil Resistance in Peer-to-Peer Systems. In: INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC 2010), Cape Town, South Africa. **Proceedings...** [S.l.: s.n.], 2010. p.1 –6.

JØSANG, A. Robustness of Trust and Reputation Systems: does it matter? In: IFIP WG 11.11 INTERNATIONAL CONFERENCE ON TRUST MANAGEMENT (IFIPTM 2012), 6. **Proceedings...** Springer, 2012. p.253–262. (IFIP Advances in Information and Communication Technology, v.374).

JØSANG, A.; ISMAIL, R.; BOYD, C. A survey of trust and reputation systems for online service provision. **Decision Support Systems**, [S.l.], v.43, n.2, p.618 – 644, 2007. Emerging Issues in Collaborative Commerce.

KAMVAR, S. D.; SCHLOSSER, M. T.; GARCIA-MOLINA, H. The Eigentrust algorithm for reputation management in P2P networks. In: WORLD WIDE WEB (WWW '03), 12., New York, NY, USA. **Proceedings...** ACM Press, 2003. p.640–651.

KARAKAYA, M.; KORPEOGLU, I.; ULUSOY, O. Free Riding in Peer-to-Peer Networks. **Internet Computing, IEEE**, [S.l.], v.13, n.2, p.92–98, 2009.

KOBZA, J.; JACOBSON, S.; VAUGHAN, D. A Survey of the Coupon Collectors Problem with Random Sample Sizes. **Methodology and Computing in Applied Probability**, [S.l.], v.9, n.4, p.573–584, 2007.

KUMAR, R. et al. Fluid Modeling of Pollution Proliferation in P2P Networks. **ACM SIGMetrics Performance Evaluation Review**, [S.l.], v.34, n.1, p.335–346, 2006.

LAMPROPOULOS, K.; DENAZIS, S. Identity management directions in future internet. **Communications Magazine, IEEE**, [S.l.], v.49, n.12, p.74 –83, december 2011.

LEE, U. et al. Understanding Pollution Dynamics in P2P File Sharing. In: INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS 2006), 5. **Proceedings...** [S.l.: s.n.], 2006.

LEMOS, R. Market for Fake Social Network Accounts Still Booming. **eWEEK**, [S.l.], July 2013. Disponível em: <http://www.eweek.com/security/market-for-fake-social-network-accounts-still-booming/>. Acesso em: outubro 2013.

LEVIN, D. et al. TrInc: small trusted hardware for large distributed systems. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI), 6. **Proceedings...** [S.l.: s.n.], 2009.

LIANG, J. et al. Pollution in P2P File Sharing Systems. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2005), 24., Miami, Florida, USA. **Proceedings...** [S.l.: s.n.], 2005. p.1174–1185.

LIANG, J.; NAOUMOV, N.; ROSS, K. W. Efficient blacklisting and pollution-level estimation in p2p file-sharing systems. In: ASIAN INTERNET ENGINEERING CONFERENCE ON TECHNOLOGIES FOR ADVANCED HETEROGENEOUS NETWORKS (AINTEC'05), 1., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2005. p.1–21.

LIANG, J.; NAOUMOV, N.; ROSS, K. W. The Index Poisoning Attack in P2P File-Sharing Systems. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2006), 25., Barcelona, Catalunya, Spain. **Proceedings...** [S.l.: s.n.], 2006. p.1–12.

LIN, E. et al. SPoIM: a close look at pollution attacks in p2p live streaming. In: INTERNATIONAL WORKSHOP ON QUALITY OF SERVICE (IWQOS 2010), 18. **Proceedings...** [S.l.: s.n.], 2010. p.1–9.

MAIER, G.; SCHNEIDER, F.; FELDMANN, A. NAT Usage in Residential Broadband Networks. In: SPRING, N.; RILEY, G. (Ed.). **Passive and Active Measurement**. [S.l.]: Springer Berlin / Heidelberg, 2011. p.32–41. (Lecture Notes in Computer Science, v.6579).

MANSILHA, R. B. et al. Observing the BitTorrent Universe Through Telescopes. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2011., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2011.

MARTI, S.; GARCIA-MOLINA, H. Taxonomy of trust: categorizing {P2P} reputation systems. **Computer Networks**, [S.l.], v.50, n.4, p.472 – 484, 2006.

MAUCH, G. H. et al. Dois Pesos, Duas Medidas: gerenciamento de identidades orientado a desafios adaptativos para contenção de sybils. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E DE SISTEMAS DISTRIBUÍDOS (SBRC 2010), 28. **Anais...** [S.l.: s.n.], 2010. p.17–30.

MITZENMACHER, M.; RICHA, A.; SITARAMAN, R. The Power of Two Random Choices: a survey of techniques and results. In: RAJASEKARAN, S. et al. (Ed.). **Handbook of Randomized Computing**. [S.l.]: Kluwer Academic Publishers, 2001. v.1, p.255–312.

MOHAISEN, A.; YUN, A.; KIM, Y. Measuring the mixing time of social graphs. In: CONFERENCE ON INTERNET MEASUREMENT, 10., New York, NY, USA. **Proceedings...** ACM, 2010. p.383–389.

MORSELLI, R. et al. KeyChains: a decentralized public-key infrastructure. **Technical Reports from UMIACS**, [S.l.], March 2006. Disponível em: <http://hdl.handle.net/1903/3332>. Acesso em: fevereiro 2011.

MOTOYAMA, M. et al. Re: captchas: understanding captcha-solving services in an economic context. In: USENIX CONFERENCE ON SECURITY, 19., Berkeley, CA, USA. **Proceedings...** USENIX Association, 2010. p.28–28. (USENIX Security'10).

NET MARKET SHARE. **Operating System Market Share**. Disponível em: <http://www.netmarketshare.com/operating-system-market-share.aspx>. Acesso em: agosto 2013.

ORAM, A. (Ed.). **Peer-to-Peer: harnessing the power of disruptive technologies**. 1.ed. Sebastopol CA, USA: O'Reilly & Associates, Inc., 2001.

PERLROTH, N. Fake Twitter Followers Become Multimillion-Dollar Business. **The New York Times**, [S.l.], April 2013. Disponível em: <http://bits.blogs.nytimes.com/2013/04/05/fake-twitter-followers-becomes-multimillion-dollar-business/>. Acesso em: outubro 2013.

PONTES, F.; BRASILEIRO, F.; ANDRADE, N. Sobre Calotes e Múltiplas Personalidades no BitTorrent. In: SIMPOSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUIDOS (SBRC 2007), 25., Belem, PA, Brasil. **Anais...** [S.l.: s.n.], 2007.

POUWELSE, J. et al. The bittorrent p2p file-sharing system: measurements and analysis. In: PEER-TO-PEER SYSTEMS (IPTPS'05), 4. **Proceedings...** [S.l.: s.n.], 2005. v.3640 / 2005, p.205–216.

PRINCE, B. Botnet Business Continues to Thrive: fortinet. **eWeek**, [S.l.], March 2013. Disponível em: <http://www.eweek.com/security/botnet-business-continues-to-thrive-fortinet/>. Acesso em: agosto 2013.

RATNASAMY, S. et al. A scalable content-addressable network. In: ACM SIGCOMM 2001 CONFERENCE ON DATA COMMUNICATION (SIGCOMM '01), New York, NY, USA. **Proceedings...** ACM, 2001. p.161–172.

ROWAIHY, H. et al. Limiting Sybil Attacks in Structured P2P Networks. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2007), 26., Anchorage, Alaska , USA. **Proceedings...** [S.l.: s.n.], 2007. p.2596–2600.

SANTOS, F. R. et al. Choking Polluters in BitTorrent File Sharing Communities. In: IFIP/IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (NOMS 2010), 12. **Proceedings...** [S.l.: s.n.], 2010. p.559 – 566.

SANTOS, F. R. et al. Funnel: choking polluters in bittorrent file sharing communities. **IEEE Transactions on Network and Service Management**, [S.l.], v.8, n.4, p.310 –321, december 2011.

SANTOS, F. R. et al. Beyond pollution and taste: a tag-based strategy to increase download quality in p2p file sharing systems. **Computer Communications**, [S.l.], v.36, n.2, p.191 – 202, 2013.

SCHREIBER, F. R. (Ed.). **Sybil**. New York, USA: Warner Books, 1974.

SCOTT, C. Market for Fake Twitter Accounts Is Booming. **Social Times**, [S.l.], July 2013. Disponível em: <http://socialtimes.com/market-for-fake-twitter-accounts-is-booming_b131197>. Acesso em: outubro 2013.

SHERR, M.; BLAZE, M.; LOO, B. T. Veracity: practical secure network coordinates via vote-based agreements. In: USENIX ANNUAL CONFERENCE (USENIX '09). **Proceedings...** [S.l.: s.n.], 2009.

SINGH, A. et al. Eclipse Attacks on Overlay Networks: threats and defenses. In: CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2006), 25., Barcelona, Catalunya, Spain. **Proceedings...** [S.l.: s.n.], 2006. p.1–12.

SpamHaus. **The SpamHaus Project**. Disponível em: <http://www.spamhaus.org/>. Acesso em: janeiro 2013.

STEINER, M.; EN-NAJJARY, T.; BIERSACK, E. W. Long term study of peer behavior in the KAD DHT. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.17, n.5, p.1371–1384, oct 2009.

STOICA, I. et al. Chord: a scalable peer-to-peer lookup service for internet applications. In: ACM SIGCOMM 2001 CONFERENCE ON DATA COMMUNICATION (SIGCOMM '01), San Diego, California. **Proceedings...** [S.l.: s.n.], 2001.

TIMPANARO, J. P. et al. BitTorrent's Mainline DHT Security Assessment. In: IFIP INTERNATIONAL CONFERENCE ON NEW TECHNOLOGIES, MOBILITY AND SECURITY (NTMS 2011), 4., Paris, France. **Proceedings...** IEEE, 2011.

TRACY, K. Identity management systems. **Potentials, IEEE**, [S.l.], v.27, n.6, p.34–37, 2008.

TRAN, N. et al. Optimal Sybil-resilient Node Admission Control. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2011), 30., Shangai, China. **Proceedings...** [S.l.: s.n.], 2011. p.3218 –3226.

TRUSTED COMPUTING GROUP. **Trusted Platform Module**. Disponível em: <http://www.trustedcomputinggroup.org/developers/trusted_platform_module/>. Acesso em: fevereiro 2013.

VIEIRA, A. B. et al. SimplyRep: a simple and effective reputation system to fight pollution in p2p live streaming. **Computer Networks**, [S.l.], v.57, n.4, p.1019 – 1036, 2013.

WALLACH, D. S. A survey of peer-to-peer security issues. In: MEXT-NSF-JSPS INTERNATIONAL CONFERENCE ON SOFTWARE SECURITY: THEORIES AND SYSTEMS, 2002., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2003. p.42–57. (ISSS'02).

WALSH, K.; SIRER, E. G. Experience With A Distributed Object Reputation System for Peer-to-Peer Filesharing. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN & IMPLEMENTATION (NSDI 2006), 3., San Jose, California, USA. **Proceedings...** USENIX, 2006. p.1–14.

XUE, W. et al. DHTrust: a robust and distributed reputation system for trusted peer-to-peer networks. **Concurrency Computat.: Pract. Exper.**, [S.l.], v.24, p.1037–1051, 2012.

YANG, Z. et al. Uncovering Social Network Sybils in the Wild. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT CONFERENCE (IMC'11), New York, NY, USA. **Proceedings...** ACM, 2011. p.259–268.

YAO, Z. et al. Node Isolation Model and Age-Based Neighbor Selection in Unstructured P2P Networks. **IEEE/ACM Transactions on Networking**, [S.l.], v.17, n.1, 2009.

YOSHIDA, M.; NAKAO, A. A Resource-Efficient Method for Crawling Swarm Information in Multiple BitTorrent Networks. In: INTERNATIONAL SYMPOSIUM ON AUTONOMOUS DECENTRALIZED SYSTEMS (ISADS 2011), 10. **Proceedings...** [S.l.: s.n.], 2011. p.497–502.

YOSHIDA, M.; NAKAO, A. Measuring BitTorrent swarms beyond reach. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2011). **Proceedings...** [S.l.: s.n.], 2011. p.220–229.

YU, H. et al. SybilGuard: defending against sybil attacks via social networks. In: ACM SIGCOMM 2006 CONFERENCE ON DATA COMMUNICATION (SIGCOMM '06), New York, NY, USA. **Proceedings...** ACM Press, 2006. p.267–278.

YU, H. et al. SybilLimit: a near-optimal social network defense against sybil attacks. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY. **Proceedings...** IEEE Computer Society, 2008. p.3–17.

ZHANG, B. et al. **The Peer-to-Peer Trace Archive**. Disponível em: <http://p2pta.ewi.tudelft.nl/pmwiki/?n=Main.Home>. Acesso em: abril 2012.

ZHANG, C. et al. BitTorrent darknets. In: INFORMATION COMMUNICATIONS (INFOCOM 2010), 29. **Proceedings...** [S.l.: s.n.], 2010.

ZHANG, C. et al. Unraveling the BitTorrent Ecosystem. **IEEE Transactions on Parallel and Distributed Systems**, Piscataway, NJ, USA, v.22, n.7, p.1164–1177, July 2011.