Computer Networks 120 (2017) 87–104

Contents lists available at ScienceDirect

Computer Networks

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

Mistrustful P2P: Deterministic privacy-preserving P2P file sharing model to hide user content interests in untrusted peer-to-peer networks

Pedro Moreira da Silva*, Jaime Dias, Manuel Ricardo

INESC TEC, Faculdade de Engenharia, Universidade do Porto Rua Dr. Roberto Frias, 378, Porto 4200-465, Portugal

ARTICLE INFO

Article history:
Received 3 October 2016
Revised 10 March 2017
Accepted 5 April 2017
Available online 6 April 2017

Keywords:
Peer-to-peer
Deterministic privacy
Content interests
Plausible deniability
Legal framework

ABSTRACT

P2P networks endowed individuals with the means to easily and efficiently distribute digital media over the Internet, but user legal liability issues may be raised as they also facilitate the unauthorized distribution and reproduction of copyrighted material. Traditional P2P file sharing systems focus on performance and scalability, disregarding any privacy or legal issues that may arise from their use. Lacking alternatives, and unaware of the privacy issues that arise from relaying traffic of insecure applications, users have adopted anonymity systems for P2P file sharing.

This work aims at hiding user content interests from malicious peers through plausible deniability. The Mistrustful P2P model is built on the concept of mistrusting all the entities participating in the P2P network, hence its name. It provides a deterministic and configurable privacy protection that relies on cover content downloads to hide user content interests, has no trust requirements, and introduces several mechanisms to prevent user legal liability and reduce network overhead while enabling timely content downloads.

We extend previous work on the Mistrustful P2P model by discussing its legal and ethical framework, assessing its feasibility for more use cases, providing a security analysis, comparing it against a traditional P2P file sharing model, and further defining and improving its main mechanisms.

1. Introduction

In the last two decades, the advances in computer technology have drastically changed the media landscape. The widespread availability of digital media and broadband Internet connections enabled consumers to become also producers and distributors of creative work, something that in the past was mostly limited to professional parties. Peer-to-Peer (P2P) networks endowed individuals with the means to easily and efficiently distribute digital media over the Internet, and are extensively used for large-scale file sharing due to their decentralized and scalable nature. Nevertheless, as more information flows through these networks, users are becoming increasingly concerned about their privacy. The P2P architecture enables solutions that enhance privacy, such as Freenet [1], Nymble [2], and Tor [3], but user legal liability issues may be raised as it also facilitates unauthorized distribution and reproduction of copyrighted material [4].

The reasons behind seeking privacy may be various, such as to avoid user profiling, tracking and data mining, to privately download legal contents that may be embarrassing or objectionable from a political, religious or social point-of-view, or to download unethical, illegal or incriminating contents without being subject to any liability. The current state of computer technology enables organizations and individuals to create databases of personal information that were previously impossible to set up, and swap this information, sell it or use it in any other way as a commodity [5]: organizations and individuals are able to collect information about users, combine facts from separate sources, and merge them to create such databases of personal information. Users may feel exposed or embarrassed if it becomes public that they had access to contents such as to help dealing with alcohol and drug abuse, to help dealing with anger management, or considered heretical or profane. Even accessing illegal or incriminating contents may have an ethical motivation because laws are, ideally, drawn from ethics, but that is not always the case: e.g., autocratic regimes consider illegal to share or even access contents that they consider inappropriate or potentially harmful.

Copyright is a legal device that protects the expression of an idea by conferring the creator, for a period of time, exclusive rights

* Corresponding author.
E-mail addresses: pmms@inesctec.pt (P. Silva), jdias@inesctec.pt (J. Dias), mricardo@inesctec.pt (M. Ricardo).

http://dx.doi.org/10.1016/j.comnet.2017.04.005
1389-1286/

to publish, sell and control the reproduction of his work, whom may grant or sell these rights to others. Copyright infringement and piracy are then violations of one of these exclusive rights [6]. However, these rights are granted nationwide, not worldwide: the Berne Convention [7] is not ratified by all countries and only sets the minimum standards for copyright. Therefore, given that P2P users are spread all over the world and a single content sharing may cross many jurisdictions, it is hard to determine the applicable legal framework and the extent of user liability. Even when the legal framework can be clearly defined, it may still be difficult to determine the extent of user liability since the copyright rights may conflict with other interests and rights, being subject to proportionality assessment to balance all the rights and interests at stake [8].

Traditional P2P file sharing systems focus on performance and scalability, disregarding any privacy or legal issues that may arise from their use. These systems take advantage of the large number of interconnected peers[1], and their idle resources, to more efficiently distribute contents at the cost of requiring peers to publicly advertise what they download, making it trivial to identify user content interests. This problem is further aggravated by the fact that peers form interest-based communities, and every single connection presents an opportunity for a malicious peer to passively obtain additional information that may enable the identification of user content interests, with high certainty, by monitoring just a small fraction of the network [9].

Several privacy-preserving P2P systems have been proposed, but, given that there is a trade-off between privacy and performance, they consider different attack models and employ different techniques for privacy preservation. The majority of the solutions employ either techniques to provide anonymity or techniques to provide plausible deniability. Techniques to provide anonymity, such as *onion routing* [10] and *information slicing* [11], are usually stronger but have lower performance. Techniques to provide plausible deniability, such as *request relaying* – peers relay requests to create uncertainty about communicating endpoints – and *content interest disguise* – peers download additional contents to hide their real interests –, are usually weaker but have better performance. Despite their differences, all P2P file sharing systems share one common issue: they require peers to advertise, either fully or partially, what they download. Lacking alternatives, users have adopted anonymity systems for P2P file sharing [12], misunderstanding the privacy guarantees provided by such systems [13], in particular when relaying traffic of insecure applications [14]. Anonymity systems provide a channel to anonymously transmit messages, but the user's identity may be disclosed by the content of that messages, which are the sole responsibility of the application.

The Mistrustful P2P model [15] provides plausible deniability through content interest disguise, as in other privacy-preserving solutions, but it has no trust requirements, prevents user legal liability in case of legitimate usage, and ensures deterministic protection of user content interests against attacks of a size up to a configured level. It is built on the basis of mistrusting all the entities participating in the P2P network, hence its name, and therefore users are not required to establish trust links in order to participate in the content sharing. The Mistrustful P2P model enables each user to set the required trade-off between privacy and performance by configuring, per content, the size of the largest group of colluding peers to be protected against, and the minimum network overhead of content interest disguise (minimum network disguise overhead).

In this paper we extend the work presented in [15] as follows: we discuss the legal and ethical framework to pinpoint the main issues and challenges, and to provide some insight regarding user liability; further assess the feasibility of the Mistrustful P2P model by considering a set of 84 use cases; provide a security analysis for common attacks; compare our model against a traditional P2P file sharing model to better estimate the impact of privacy preservation; further define it to get closer to enable real implementations, while improving its main mechanisms to better adapt to the P2P file sharing dynamics and to the user privacy requirements.

The remainder of this paper is structured as follows. Section 2 depicts the main legal and ethical challenges at stake along with user liability considerations. Section 3 briefly describes traditional P2P file sharing systems. Section 4 characterizes the problem we aim to solve. The related work is presented in Section 5. Section 6 describes the latest version of the Mistrustful P2P model. Sections 7 and 8 provide, respectively, the security analysis and the evaluation of our model. The results and discussion are presented in Section 9. Section 10 draws the main conclusions.

2. Legal and ethical framework

Computer ethics is a field of study that addresses the ethical challenges of computer technology but several lines of thought exist. Therefore, in this section our aim is to describe the ethical challenges that we identified being relevant to our work. We refer the reader to [16] and [17] for a wide and thorough explanation of these challenges. For the legal and privacy dimensions of P2P file sharing, based on the intellectual property law, in particular on the copyright law, we attempt to highlight the key legal aspects to take into consideration on Western countries regarding direct and indirect user liability.

Laws are formally adopted rules that mandate or prohibit a certain behavior, created by the members of a society to balance the individual rights to self-determination against the needs of the society as a whole, and, ideally, are drawn from ethics, which define what is considered right or wrong, i.e., the socially acceptable behaviors. The key difference between laws and ethics is that the former carry the authority of a governing body, usually a nation [5]. In turn, ethics are based on cultural mores, beliefs, values and principles, which reflect the unique existential experiences that we accumulate as individuals as well as societies and, supported on institutions, provide long-term stable rules that are made obvious. Thus, these rules can be seen as refractions of the common world awareness that give rise to different experiences and interpretations: multicultural ethics [18].

The cultural differences, despite some ethical standards being universal (e.g., murder and theft), make it difficult to define what is ethical or not. Studies have shown that the perspective on ethical practices of individuals regarding the use of computer technology differs with their nationality. Asian traditions of collective ownership conflict with Western protection of intellectual property, and many of the ways the former use software is considered software piracy by the latter [5]. These differing perspectives are also evident on the control over the Internet content and on the surveillance made by governments: they radically differ between countries [19]. According to a report [20] from the Reporters Without Borders, the *Great Firewall* of China is getting "taller", the United Kingdom is the "world champion of surveillance", and "NSA[2] symbolizes intelligence services' abuses"; on the other hand, Finland, Netherlands, and Norway are at the top of the World Press Freedom Index 2016 [21].

[1] We use the term *peer* to refer to the network node, and the term *user* to refer to the person.

[2] United States National Security Agency.

The utmost objective of intellectual property law is to promote progress by encouraging and stimulating human intellectual creativity and broad dissemination of its result [22]. Creators are granted exclusive rights as an incentive to continue their works, and, at the same time, the society can benefit from their broad dissemination. However, given that the enforcement of such exclusive rights on the private sphere of users may clash with their privacy rights, the private copying law, when present, introduces an exception to these exclusive rights under some conditions. For private use and for ends that are neither directly nor indirectly commercial, a natural person[3] is allowed to copy copyrighted works on the condition that the rightholders receive fair compensation. Therefore, any media that may be used for the reproduction of copyrighted works by consumers are designated for payment of the private copying levy, which will compensate rightholders for any harm that may be caused [23,24]. The private copying law, when present, differs considerably between countries [25], reflecting the lack of consensus and the complexity of this subject. Moreover, it is not clear that private copying always causes harm – e.g., it may increase the group of fans and enables users to try before buying –, and not all media are used for the reproduction of copyrighted works. We refer the reader to [22,24,26] for further discussion on the subject.

P2P networks connect users all over the world without considering either international borders or culture, thereby a single content sharing may cross many jurisdictions. These networks are usually connoted with copyright infringement because it is estimated that, over the course of a month, 96.3% of users of BitTorrent portals have downloaded at least one infringing content [27]. It is not feasible to enforce the exclusive rights provided by copyright law, if present, in every single jurisdiction while ensuring that they do not conflict with the rights and interests of users. As so, in an attempt to mitigate the impact of copyright infringement, rightholders are now trying to block websites such as The Pirate Bay instead of trying to bring end users to court [28] because, in general, it is more efficient and less onerous to prove that the owners of such websites profit from the copyright infringement. Therefore, such websites are subject to indirect liability as they induce, contribute to or fail to prevent direct copyright infringement. Nevertheless, this does not mean that sharing copyright infringing contents is legal. On the contrary, users may be subject to civil and potentially criminal liability [24]. The extent of such liability depends on the applicable legal framework and on the intent of such acts: a user may also be subject to indirect liability.

The multitude and complexity of copyright laws across the globe make it impossible to define clear boundaries regarding user liability. Still, it is our belief that the users of P2P systems will not be held legally liable for, unknowingly and unwillingly, downloading an illegal content (direct liability) or contribute to its download (indirect liability) if the main motivation to use the P2P system is to share legit contents, and it cannot be proven that the user had access, either in part or fully, to the content data. The user does not benefit directly from the download of an illegal content that he has no access to, and is also unable to determine that the content is unexpectedly illegal before having access to its data. Therefore, it is plausible that he either has been mislead into downloading it or had no intention in assisting a misbehaving peer infringing copyright law.

In sum, the basic ethical imperatives are that a person should not, knowingly and willingly, cooperate in or contribute to the wrongdoing of another, and that the human intellectual creativity needs to be encouraged and stimulated in order to promote progress. The extent of these incentives depends on the cultural background as the well-being of the society may be incentive enough (collective ownership) or further incentives may be required (intellectual property rights). P2P networks introduce new challenges to the private copying levy system, and the legislation on this subject is expected to change in the near future to address them. Users of P2P systems should not be subject to any civil and criminal liability as a result of legitimate usage of the system if the main motivation to use it is to share legit contents, and they have no access to the content data. The intent is important to determine the extent of user liability, especially if his actions directly or indirectly caused provable harm.

[3] In jurisprudence, a natural person is a human being, as opposed to a legally generated juridical person.

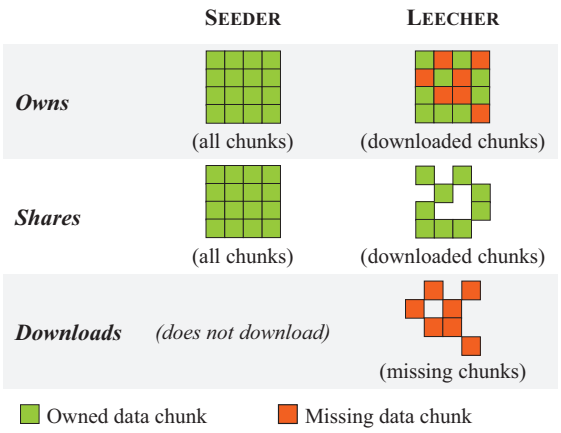

Fig. 1. Peer roles and sharing behavior on traditional P2P systems. A seeder has all data chunks and just shares them; a leecher is a peer still downloading missing data chunks and sharing the data chunks it has already downloaded.

3. Traditional P2P file sharing systems

Traditional P2P file sharing systems focus on performance and scalability, disregarding any privacy and legal issues that may arise from their use. In this section we provide an overview of these systems, and present their main privacy and legal issues. We use BitTorrent as an example given that it is the most prominent of such systems. We start by providing a brief definition of common BitTorrent terminology – *hash, content, chunk, torrent file, peer, seeder, leecher, swarm*, and *tracker* –, describe its content sharing process, and end presenting the main privacy and legal issues.

A *hash* is an alphanumeric string used to identify and to verify the integrity of the data being transferred, usually an SHA-1 digest (hexadecimal string). A *content* is composed of one or more files, identified by the hash of its data, and partitioned into several pieces. A *chunk* is one of those data pieces, and a *torrent file* provides the content's metadata. A *peer* is a node sharing chunks, which, for a given content, can be either a *seeder* – a peer that has all data chunks and is just sharing them – or a *leecher* – a peer still downloading the content and sharing the chunks it has already downloaded. Fig. 1 depicts the peer roles and their sharing behavior. A *swarm* is the set of peers sharing the same content (peers form interest-based groups to share contents), and is usually identified through the hash of the content. A *tracker* is a central node that provides lists of peers in swarms by keeping track of which peers are sharing which contents and their role (seeder or leecher) on each content.

Traditional P2P systems take advantage of the large number of interconnected peers, and their idle resources, to more efficiently distribute contents; their main performance metric is the average download time. E.g., BitTorrent protocol employs several mechanisms for chunk and peer selection, such as rarest first

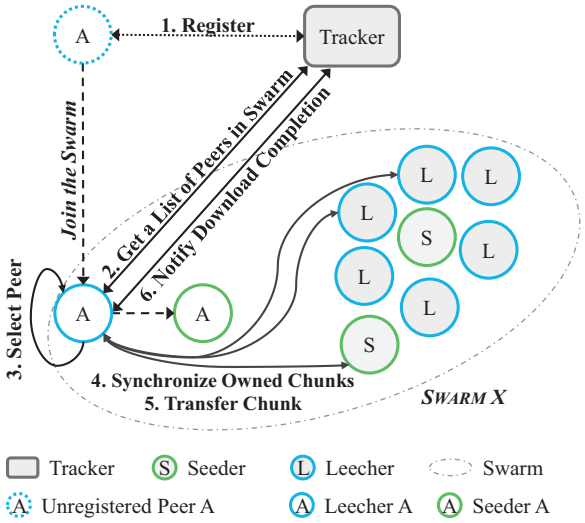

Fig. 2. Overview of the content download process on traditional P2P systems. First, peer A registers at the tracker to join the swarm of content X (swarm X). After registering, the peer starts as a leecher (still downloading) and requests from the tracker a list of peers (a subset) sharing that content. Until download completion, peer A selects peers from that list, synchronizes the chunks it owns with theirs, and transfers those available, if any, that it misses; the list of peers may be updated during content download. Upon download completion, peer A notifies the tracker to be known as a seeder.

mechanism – locally rarest chunks are downloaded first –, and optimistic unchoking – periodically select a random peer – aiming at constantly improving the download bitrate [29]. The distinction between seeders and leechers is used to estimate relative download times (through their ratio), and also to quickly assess the content availability[4], which is assured if at least a single seeder is present. The steps required to download a given content, depicted in Fig. 2, can be summarized as follows: (1) a peer willing to download a given content registers at the tracker and joins the swarm; (2) it requests a list of peers (a subset) in the swarm from the tracker; (3) it selects peers from that list to obtain missing chunks in order to complete its download; (4) peers synchronize the chunks they own and miss to determine if any missing chunks can be transferred; (5) if selected peers own missing chunks, those chunks get transferred; (6) upon download completion, the peer notifies the tracker to be known as a seeder for that content.

The main privacy issues of traditional P2P systems are that they publicly disclose user content interests, and provide a proof that the user is able to access a content in part or entirely; the former discloses an intention while the latter provides a proof of its realization. A peer only downloads the contents that a user is interested in, therefore, by registering at a tracker and joining swarms, the user content interests are being publicly disclosed. Peers provide a proof that the user is able to access a content in part or entirely by advertising which chunks they own, and entirely by notifying the tracker upon download completion.

The main legal issue of traditional P2P systems is that, unknowingly and unwillingly, a user may be held liable for copyright infringement due to illegal content download. If a content is published with a misleading description or if the resource used to obtain the hash of the content is insidious, the user will only become aware of that fact after accessing the content data in part or fully. Nevertheless, for contents that can be accessed in part before fully downloading them, which is usually the case of multimedia contents, the user may be infringing copyright law after downloading a single or a few chunks. The copyright infringement can be trivially proved because peers advertise which chunks they own and notify the tracker upon download completion.

[4] A content is available if it can be fully downloaded.

4. Problem definition

In the context of this work, we define privacy preservation as the concealment of user content interests. We aim at developing a privacy-preserving P2P file sharing model that:

Hides user content interests. The user must be able to hide his content interests from any participant in the system: trackers, regular peers, or groups of colluding peers. Protection against external entities monitoring all the user's traffic, such as ISPs or governments, is out of the scope of this work.

Has no trust requirements. The user must be able to download a content without having to trust anyone. Therefore, the privacy-preserving P2P system must enable content sharing in large groups of untrusted peers (untrusted P2P networks).

Prevents user liability. The user shall not be subject to any liability as a result of legitimate usage of the privacy-preserving P2P system. The user shall be protected against actions of misbehaving nodes, or from the download of contents published with misleading description (misleading contents), which may drive users to unknowingly and unwillingly download unethical or illegal contents.

Enables timely downloads. The user should be able to download a content in due time. The average download bitrate must be within the same order of magnitude of traditional P2P systems that do not preserve the privacy of users.

5. Related work

Several privacy-enhancing P2P systems have been proposed in the literature for a wide range of applications, providing different degrees of privacy to users and employing various techniques, the majority of which provides privacy preservation through anonymity or through plausible deniability. Freenet [1] and Tor [3] are probably the most prominent anonymity solutions for, respectively, anonymous content distribution networks and low-latency anonymity. Despite their merits, anonymity systems tend to introduce more overhead, and their users may be held indirectly liable. For instance, Tor defaults to a path length of four (300% network overhead), which lowers the throughput and increases the average latency [30], and a Tor relay node (core router) may relay traffic of misbehaving peers, which makes the last one on the path (exit node) to appear to be the originator of such traffic. Systems such as OneSwarm [31], which rely on trust links to provide privacy, are not considered given that they are not suitable for untrusted P2P networks. Herein, we depict the privacy-enhancing P2P systems providing plausible deniability and designed specifically for P2P file sharing.

The description provided for each privacy-enhancing P2P system focus on the trust requirements, on the protection against both passive and active attacks aiming at identifying user content interests, and on the potential legal liability of users.

BitBlender [30] provides plausible deniability by introducing relay peers that simply proxy requests on behalf of other peers. Peers willing to act as relay peers can register at a central node called *blender*, and, once requested, will join a P2P swarm in a probabilistic way so that they cannot be distinguished from regular peers. The joining probability of relay peers is defined by the *blender*, when asking registered peers to join a P2P swarm, so that the

Table 1
Comparison of privacy-preserving P2P file sharing systems regarding trust requirements, protection against passive and active attacks aiming at identifying user content interests, and potential user legal liability.

Work	Has No Trust Requirements	Protects Against Attacks		Prevents Legal Liability
		Passive	Active	
BitBlender	✗	✓	✗	✗
SwarmScreen	✓	✓	✗	✗
The BitTorrent Anonymity Marketplace	✓	✓	✓	✗
Petrocco et al.	✗	✓	✓	✗

set of relay peers remains unknown while having the cardinality requested by the tracker. It enables the identification of regular and relay peers through download progress tracking, and provides a trivial proof of content download because peers advertise what they download. Thereby, BitBlender provides protection against passive attacks, but it is vulnerable to active attacks using download progress tracking. It requires users to trust both the tracker and the *blender*. Its users may be subject to legal liability for downloading misleading contents, and for relaying traffic of misbehaving peers.

SwarmScreen [9] provides plausible deniability by obscuring user content interests through content interest disguise. The devised scheme, which consists in "adding a small percentage (between 25% and 50%) of additional random connections that are statistically indistinguishable from natural ones", thwarts guilt-by-association attacks, that is, attacks in which the user content interests can be inferred with high certainty just by classifying peers based on the behavior of the communities they participate in. SwarmScreen presents no trust requirements, but its attack model only considers passive attacks. It is vulnerable to active attacks because contents can be distinguished through download progress tracking. Peers communicate through direct links, but users may be subject to legal liability due to misleading content download.

The BitTorrent Anonymity Marketplace [32] follows SwarmScreen's approach to provide plausible deniability. It does not present any trust requirements, and peers also communicate through direct links. However, in order to protect against both passive and active attacks, all contents are fully downloaded to make them indistinguishable, given that an attacker is able to fully track download progress: peers advertise what they own or miss. The authors define *k-anonymity* as the privacy protection level obtained from fully downloading k contents. Thus, since it introduces high network overhead, it either prevents downloads from timely completing or constrains the achievable level of privacy protection. Users may be subject to legal liability due to the download of misleading contents.

Petrocco et al. [33], following SwarmScreen's approach, proposed a system that aims at hiding user content interests without compromising timely download completion. Their system relies on private swarms, *request relaying*, caching, and partial advertisement of downloaded chunks. As stated by the authors, private swarms are required to ensure a good level of privacy, i.e., to avoid the identification of user content interests through download progress tracking. Yet, to obtain the credentials needed to join a private swarm, peers must trust one or more participants. Also, as only a fraction of the chunks are advertised, it is not clear how a content sharing is bootstrapped with few seeders or how request relay should operate during periods of content unavailability. This system provides protection against passive and active attacks, but its users may be held legally liable due to relay traffic.

Table 1 summarizes and compares the privacy-preserving P2P file sharing systems described above taking into consideration the trust requirements, the privacy protection against passive and active attacks, and the potential legal liability of users. BitBlender and SwarmScreen are unable to hide user content interests from active attacks using download progress tracking. The BitTorrent Anonymity Marketplace and Petrocco et al.'s systems are able to thwart both passive and active attacks. BitTorrent Anonymity Marketplace requires peers to fully download all contents in order to make them indistinguishable, and therefore introduces a considerable network overhead that will either prevent downloads from timely completing or constrain the achievable level of privacy protection. Petrocco et al.'s system requires peers to trust one or more participants in order to reduce network overhead by using partial advertisement of downloaded chunks.

All solutions require peers to advertise, either fully or partially, what they own or miss. An attacker may exploit download progress tracking to obtain a proof that the user is able to access a content either entirely or in part, and to narrow or even void plausible deniability. For unethical or illegal contents that can be accessed in part before fully downloading them, which is usually the case of multimedia contents, an attacker may obtain a proof of such access. As so, the user may be held legally liable for, unknowingly and unwillingly, downloading a single or a few chunks, be it due to traffic relaying or due to misleading content description. Thereby, a legitimate usage of these systems may hold the user liable for copyright infringement.

6. Mistrustful P2P model

In this section we describe the latest version of our model, which is built upon the assumption described in Section 2: legitimate users should not be subject to any legal liability if the system is mainly used to share legit contents and there is no access to the content data (no proof of access). We refer the reader to [15] for a prior version.

This section is structured as follows. Section 6.1 provides an overview of our model and its main building blocks – content interest disguise and mistrustful sharing – to best describe how the problem we aim to solve is addressed. Section 6.2 discusses the peer roles and their sharing behavior, the content sharing process, and the role of mistrustful sharing on it. The attack model considered is defined in Section 6.3. Sections 6.4 through 6.8 characterize the instantiations of each one of the mechanisms used on the evaluation of our model, whose main focus is on its feasibility rather than on its overall performance.

6.1. Overview

The Mistrustful P2P model is built on the concept of mistrusting all the entities participating in the P2P network, hence its name, and therefore users are not required to establish any trust links in order to participate in the content sharing. It relies on two main building blocks – content interest disguise and mistrustful sharing – and aims at hiding user content interests through plausible deniability, in untrusted P2P networks, while overcoming the main limitations of akin P2P file sharing systems. These limitations can be summarized as follows: (1) peers are required to advertise what they download enabling passive attacks; (2) protection against active attacks is only achieved by introducing either trust

requirements or considerable network overhead; (3) the privacy protection against both passive and active attacks is probabilistic; (4) legitimate users may be held legally liable for, unknowingly and unwillingly, downloading or relaying traffic of illegal contents.

Content interest disguise, as in other privacy-preserving P2P systems, hides user content interests through plausible deniability. The user content interests are hidden by downloading both contents that the user is interested in (genuine) and additional contents of no interest to the user (cover), as long as they cannot be distinguished. Registering at a tracker and joining a swarm no longer represents interest in that content, and user content interests can no longer be identified by monitoring just a small fraction of the network [9]. A content interest disguise scheme selects the set of cover contents, how much of each one to download, and may impose constraints to the content sharing in order to hide user content interests. The evaluation of the Mistrustful P2P model is centered on the feasibility of its novelty, the mistrustful sharing building block, and thereby the proposal of a content interest disguise scheme is out of the scope of this work.

The mistrustful sharing building block enables the user to configure the required trade-off between privacy and performance by defining the size c of the largest colluding group to be protected against and the minimum amount m of chunks that are to be downloaded per cover content (minimum network disguise overhead), where $c \leq m < k$ for any content partitioned into k chunks so that there is no proof of full content download (proof of download). Proof of access is then avoided by encoding contents in a way that only enables decoding after full download. The mistrustful sharing building block is composed of two core mechanisms – erasure coding and disclosure constraint –, and three supporting mechanisms – block selection, request backoff, and peer selection. It enables the Mistrustful P2P model to overcome limitations (1) to (4) as follows.

Peers avoid advertising what they download by requesting from other peers random chunks, thus defeating passive attacks of any size; the block download process and the messages exchanged are described in Section 6.2. Attackers have then to engage in the content sharing because the chunks owned or missing are only implicitly disclosed to other peers while sharing. Active attacks, of a size up to c, are defeated by constraining the amount of chunks disclosed to any set of c peers to be, at most, m: an attacker is not able to distinguish cover contents from genuine ones because at least m chunks are downloaded of each content. As so, user content interests are deterministically hidden. For legitimate users, legal liability is prevented by not requiring peers to relay traffic on behalf of other peers, by never fully downloading cover contents (their data is never accessible), and by avoiding both proof of access and proof of download for any attack of a size up to c. The protection of user content interests and the user liability are discussed in more detail in Section 7, which also describes the countermeasures employed against common attacks.

The erasure coding mechanism is the core mechanism used to enable the probability of randomly retrieving a chunk to become significant, and to enable decoding only after full download. Considering a content divided into k chunks, uniformly distributed across the network, the probability of retrieving the last chunk is just $1/k$. The erasure coding mechanism generates a set of n erasure coded chunks (blocks), from a set of k chunks, so that any subset of k' blocks enables to retrieve the content, where $k' = k(1 + \epsilon(k))$ and $\epsilon(k)$ is the erasure coding overhead. If the n blocks are also uniformly distributed across the network, the probability of retrieving the last required block increases to $1 - (k' - 1)/n$. As an example, for $k = 20$, $n = 5 \cdot k = 100$, and $\epsilon(k) = 0$ (optimal erasure code), the probability increases from 5% to 81%.

The disclosure constraint mechanism is the core mechanism used to ensure that no more than m blocks are disclosed (requested or shared) to any set of c peers (largest colluding group considered), and thus prevents proof of download ($m < k$). It also contributes to the reduction of the overhead due to cover downloads because only m blocks need to be downloaded per cover content in order to avoid the identification of genuine downloads among cover downloads. This mechanism enables the disclosure of at least one block to each peer ($m \geq c$), and, on average, m/c blocks can be requested from each peer. Reducing the amount of available block requests by either increasing c or decreasing m may impact the overall performance.

The remaining three mechanisms – block selection, request backoff, and peer selection – are defined to enable the evaluation of our model, and to ensure that contents are timely downloaded. The block selection mechanism determines which block is to be offered to a given requesting peer, affecting the distribution of blocks among peers and therefore the probability of retrieving a useful block (innovative block). The request backoff mechanism determines the delay between block requests aiming at maximizing the amount of useful blocks that can be obtained from the available block requests in the shortest time frame. With the same aim, the peer selection mechanism selects a peer to which a block request will be sent.

The instantiation provided for each mechanism, described in Sections 6.4 through 6.8, is the one used to evaluate our model in Section 8. Due to the large number of variables and factors at play, we consider the impact of those that are expected to change more often – c and m, content size, peer arrival rate, and number of seeders –, and of cover downloads on the average download bitrate and on the download completion ratio. For the sake of clarity and tractability, other factors and variables such as inter-content relations, incentives to share, parallel chunk requests, and Internet connection heterogeneity are not considered. The evaluation aims at demonstrating the feasibility of our model rather than at optimizing its overall performance, i.e., it aims at demonstrating that peers are able to timely download contents without advertising what they download.

In sum, the mistrustful sharing building block reinforces the content interest disguise because the distinction between genuine and cover contents is hardened by not disclosing what peers download or miss, and a larger set of cover contents can be used as they do not need to be fully downloaded. It prevents legitimate users from being held liable due to cover content and misleading content downloads. Cover contents are never fully downloaded to guarantee that the user has never access to their content data. Misleading contents may be fully downloaded, but there is no proof of access or proof of download for any attack of a size up to c.

6.2. Peer roles and content sharing

Peers, per content, can take one of two roles depending on their privacy requirements and the way they contribute to the file sharing: *seeder* – peer having a content that wants to share, and willing to forgo its privacy –, or *commoner* – peer willing to participate in the content sharing if its privacy requirements can be met. A seeder may be the author or a party interested in publishing a content, and therefore does not require the concealment of its content interests. It generates a unique block (erasure coded chunk) for each request it receives, and only refuses to serve block requests if it has no resources available. On the other hand, a commoner does not generate new blocks, only shares them if its interests remain hidden, and only has access to the content data after fully downloading the content. A commoner keeps track of the blocks it shares with other peers both for privacy protection and to avoid offering a block twice to the same peer. It may refuse to serve block requests if it has no useful blocks to offer, due to resource or privacy constraints, or due to content interest disguise strategies. The

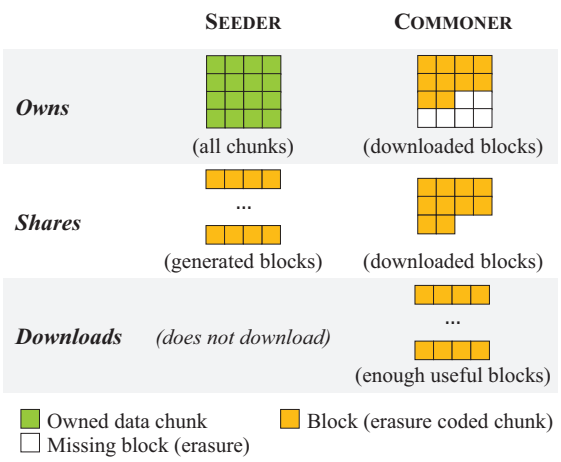

Fig. 3. Peer roles and sharing behavior on the Mistrustful P2P model. A seeder has all data chunks and generates a new block (erasure coded chunk) for each incoming request; a commoner is a peer that may be still downloading enough blocks to reconstruct the content data and shares the blocks it has already downloaded. As so, a commoner never shares data chunks and only has access to the content data after fully downloading the content.

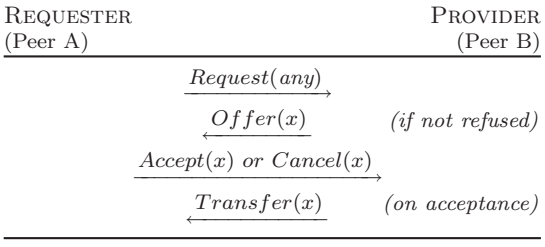

Fig. 4. Messages exchanged during block download process. Peer A requests a random block from peer B, which either refuses the request or offers a block with id *x*. If the block is accepted by peer A, it will get transferred; otherwise, no transfer occurs to avoid unnecessary network overhead.

commoner never discloses the reason behind refusal. Fig. 3 summarizes the peer roles and their sharing behavior.

The block download process on the Mistrustful P2P model differs considerably from the one on other P2P file sharing systems, given that peers do not advertise what they download. This process is summarized in Fig. 4, where peer A is the requester and peer B is the provider, and works as follows. Peer A requests a random block from peer B, i.e., without providing the id of an intended block. Peer B then either refuses the request by simply ignoring it, and thus not disclosing the reason behind refusal, or replies with the id of the block it is willing to share, which must be different from any other that may have been previously disclosed between them (both as requester and as provider). If peer B has offered a block, peer A sends a reply message either accepting the offered block, in order to start its download, or canceling the block request, in order to avoid unnecessary network overhead. Unless the block request has been canceled, peer B sends the block it has offered to peer A. The possible outcomes of a block request are further discussed in Section 6.6.

The content download process of the Mistrustful P2P model, depicted in Fig. 5, consists in the following steps: (1) the peer registers at the tracker to join the swarms of both genuine and cover contents in order to disguise the content interests of the user, without disclosing its role; (2) it requests a list of peers (a subset) in the swarm from the tracker, which is not aware of each peer's role; (3) it selects peers from that list that cope with the user privacy requirements (eligible peers); (4) it requests random blocks from those peers in order to complete its download; (5) if

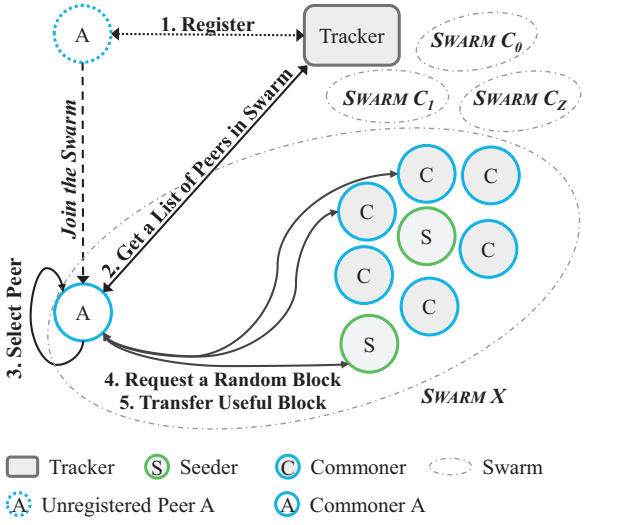

Fig. 5. Overview of the content download process on the Mistrustful P2P model. First, peer A registers at the tracker to join the swarm of content X (swarm X) without disclosing what blocks it owns or misses; it also joins swarms C_0 through C_z to disguise user content interests. Then, it requests a list of peers (a subset) in the swarm from the tracker. Until download completion, and as long as the privacy requirements are met, peer A selects eligible peers in the swarm and requests them random blocks. To prevent unnecessary network overhead, only offered blocks that are useful get transferred; otherwise, the block requests are canceled. The list of peers may be updated during content download, and there is no notification upon download completion.

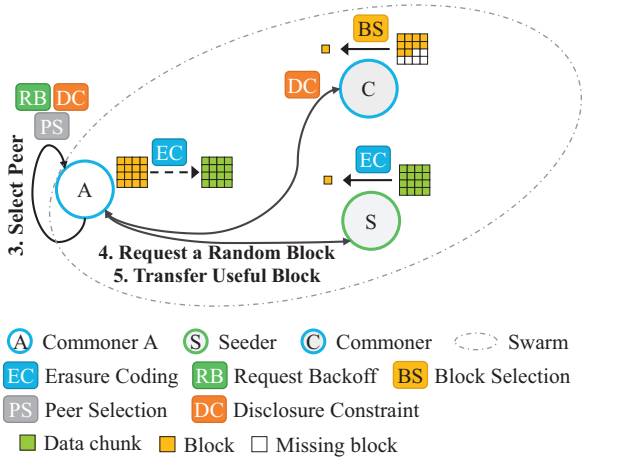

Fig. 6. Mistrustful P2P model mechanisms and their role in the content sharing. The erasure coding mechanism is used by a seeder to generate new blocks, and by a commoner to retrieve the content data after fully downloading a content. The disclosure constraint mechanism determines if a block request can be sent to a peer or accepted by a given commoner. The block selection mechanism determines which block is to be shared, if any; the same block is never offered twice to the same peer. The peer selection mechanism selects an eligible peer to which a block request can be sent. The request backoff mechanism defines the delay between block requests.

the selected peers offer useful blocks, those blocks get transferred; otherwise, the requests are canceled and no transfer occurs. The list of peers may be updated during content download, and steps (3), (4) and (5) are repeated until download completion (genuine contents) or until at least *m* blocks have been transferred (cover contents). There is no notification from the commoner upon download completion.

The scope of action of the mistrustful sharing mechanisms entails mostly steps (3), (4) and (5). Their role on the content sharing is illustrated in Fig. 6. The erasure coding mechanism enables a commoner to retrieve the content data after fully downloading the coded content, and enables a seeder to generate a new block for

each incoming request. The disclosure constraint mechanism determines if a block request can be sent to a peer or accepted by a given commoner. The block selection mechanism is used by the contacted commoner to determine which block is to be offered to the requesting peer, if any. The request backoff mechanism determines the delay between block requests of a commoner. The peer selection mechanism selects the peer to which the block request is sent.

6.3. Attack model

We assume that an attacker might be any entity that participates in the system – a publisher, a tracker, a regular peer, or a group of colluding peers. External entities monitoring all traffic of a peer, such as ISPs or governments, are out of the scope of this work.

Being a participant of the system, we consider that an attacker is able to engage in the content sharing as a commoner or as a seeder, to be a tracker, and to publish contents with misleading description. Also, we consider that an attacker may coordinate a large number of peers that collude with each other (collusion attack) or assume multiple pseudonymous identities (Sybil attack), which is equivalent to a larger colluding group.

The creation of large number of pseudonymous identities (Sybil attack) was first considered by Douceur [34], and is one of the most dangerous attacks that plague P2P networks [35]. Douceur [34] showed that, without trusted identity certification, Sybil attacks are always possible when considering realistic scenarios. In untrusted P2P networks, unlike collusion attacks, Sybil attacks can be mitigated without explicit information either by performing resource testing or by applying recurring costs and fees [35]. We refer the reader to [36] for a survey on existing approaches.

In the context of our work, collusion and Sybil attacks aim at increasing the amount of blocks disclosed by other peers to a single entity or to colluding entities, so that either content download can be proven or user content interests can be determined. The Mistrustful P2P model provides privacy protection against such attacks of a size up to c peers (colluding group), be it colluding peers, Sybil peers, or a combination of both. It requires $c \leq m < k$, and therefore m and k are upper bounds for c. On the one hand, despite increasing the minimum network disguise overhead, m can be increased as needed (up to k) given that it is user configurable. On the other hand, k is defined by the publisher of the content and cannot be changed. Thereby, in order to extend the privacy protection without increasing the size of the largest colluding group considered (c), the user may configure as single entities all the sets of peers that he considers, or suspects, to be colluding or to be Sybil peers.

Peers are identified by their public IP addresses because we consider that IP addresses provide a more flexible resource testing that can be made harder to acquire than other resources such as human time, network bandwidth, computational power or storage capacity, which cannot be related. For privacy protection purposes, a set of peers whose IP addresses are considered to be related can be treated as a single peer (IP address aggregation), or multiple peers sharing a single IP address can be treated individually by considering the (IP, port) pair as the identifier (IP address multiplexing). The user is then able to configure how IP addresses are treated for privacy protection purposes, providing the flexibility to go as low as treating all (IP, port) pairs as unique identities, e.g. all peers behind NAT[5], up to treating all IP addresses of a given entity, colluding entities, city, country or any other set of IP addresses as a single peer. This also enables the user to configure the set of rules that are applied to peers using anonymous systems such as Tor, which can be IP address aggregation rules, IP address multiplexing rules or any combination of both.

[5] Network Address Translation.

6.4. Erasure coding

An erasure code generates a set of n symbols from a set of k symbols, $k < n$, so that any subset of $k(1+\epsilon(k))$ is enough to reconstruct the original information, where $\epsilon(k)$ is the erasure coding overhead. Erasure codes are usually classified according to three orthogonal properties: systematicity, rate fixedness, and coding overhead. An erasure code is systematic if the input symbols are embedded into output symbols, and non-systematic otherwise. If n is static and needs to be known before encoding, the erasure code is fixed-rate. If n can be dynamically increased and the amount of symbols that can be generated does not impose any practical limitation, the erasure code is rateless. Finally, an erasure code is said Maximum Distance Separable (MDS) if any k symbols out of n are enough to reconstruct the original information [$\epsilon(k) = 0$], or non-MDS if additional symbols are required [$\epsilon(k) > 0$]. Non-MDS erasure codes reduce significantly the encoding and decoding time complexity orders by introducing coding overhead.

The erasure coding mechanism supports both MDS and non-MDS rateless erasure codes, but requires the content to be either coded or encrypted in a way that only enables decoding after full download. For evaluation purposes, we consider scenarios in which the MDS erasure codes are more suitable, e.g. those in which the network is the most constrained resource, and thus erasure coding overhead does not represent one additional variable that needs to be considered [$\epsilon(k) = 0$]. We refer the reader to [37] for a rateless MDS construction of Reed-Solomon codes that we developed for our model. These erasure codes are defined over the finite field $\mathbb{F}_{p^2}$ with $\Theta(n \log k)$ encoding and $\min\{n \log n, k \log^2 k\}$ decoding time complexities, where p is a Mersenne prime ($p = 2^q - 1$) and $n \leq 2^{q+1}$. Their performance was evaluated over $\mathbb{F}_{(2^{31}-1)^2}$, so $n \leq 2^{32}$, and does not impose any constraints to the content sharing. They also enable decoding only after full content download.

6.5. Disclosure constraint

The disclosure constraint mechanism enables the user to configure, per content, the required trade-off between privacy and performance by setting the size c of the largest colluding group to be protected against, and the minimum amount m of blocks that need to be downloaded per cover content. The size c of the largest potential attacker is defined by the number of unique peers (unrelated public IP addresses) that are controlled by a single group, either a single attacker or a group of colluding attackers. This mechanism ensures that cover and genuine content downloads cannot be distinguished by tracking their download progress because at most m blocks are disclosed to any set of c peers, and that no malicious peer can prove that a user downloaded a content or had access to its data ($m < k$).

Finding the maximum intersection between the set of blocks disclosed to any set of c peers is an NP-hard problem [38], thus we devised a conservative yet efficient algorithm to evaluate dynamically the number of blocks that can still be shared with a peer. The algorithm is divided into two main functions: one to update the counter of blocks disclosed to a peer (Algorithm 1 – Update Blocks Disclosed), and the other to determine the number of blocks that can still be disclosed to a peer (Algorithm 2 – Blocks to Disclose Left). The variables *commoners* and *blocksDisclosed* are respectively an array sorted by the number of blocks disclosed, and the maximum number of blocks disclosed to any set of c peers.

Algorithm 1 Update Blocks Disclosed.

```
function IncrementBlocksDisclosed(id)
    i ← commoners.getIndex(id)

    if invalidIndex(i) then                          ▷ New.
        commoners.push(id)
        commoners.last.blks ← 1
        i ← commoners.getIndex(id)
    else                                             ▷ Known.
        commoners[i].blks ← commoners[i].blks + 1
        j ← i − 1

        while validIndex(j) do                       ▷ j ≥ 0
            blksI ← commoners[i].blks
            blksJ ← commoners[j].blks

            if blksI > blksJ then                    ▷ Still unsorted.
                swap(commoners[i], commoners[j])
                i ← j
                j ← j − 1
            else                                     ▷ Sorted.
                break
            end if
        end while
    end if

    if i < c then                                    ▷ Changes on top c peers.
        blocksDisclosed ← blocksDisclosed + 1
    end if
end function
```

Algorithm 2 Blocks to Disclose Left.

```
function BlocksDiscloseLeft(id)
    if c > m or m ≥ k then                           ▷ Invalid.
        return 0
    end if

    top ← min(c, commoners.length)                   ▷ Top peers.
    left ← m − blocksDisclosed − (c − top)
    i ← commoners.getIndex(id)

    if invalidIndex(i) then                          ▷ New.
        if commoners.length ≥ c then
            left ← left + commoners[c − 1].blks
        else
            left ← left + 1
        end if
    else                                             ▷ Known.
        if i ≥ c then
            left ← left + commoners[c − 1].blks
            left ← left − commoners[i].blks
        end if
    end if

    return left
end function
```

Algorithm 1 receives as input the *id* of the peer to which one additional block was disclosed. If none has yet been disclosed, a new entry is created; otherwise, the entry is updated and, if needed, some elements are swapped to keep the array sorted. In each case, if the updated entry is on one of the top c positions, the maximum number of blocks disclosed is updated. Algorithm 1 has linear time complexity.

Algorithm 2 also receives as input the *id* of the peer. If the configured privacy requirements are not met (invalid), no blocks can be disclosed to any peer. If they are met, *left* contains the number of blocks that can still be disclosed, ensuring that at least one block can be disclosed to each one of the top c peers; at most, $m - (c - 1)$ blocks can be disclosed to a single peer. *left* needs to be updated if there are already at least c peers and the peer referred by *id* is outside of that set. Algorithm 2 runs in logarithmic time.

6.6. Block selection

The block selection mechanism is used by commoners to determine which block is to be offered to a requesting peer. It plays an important role on how the blocks end up distributed across the network, affecting the probability of peers obtaining useful blocks. This mechanism ensures that no uploaded block is offered twice to the same peer, and determines when requests should be refused. The request refusal may be due to the lack of useful blocks to offer, due to resource or privacy constraints, or due to content interest disguise strategies.

With the aim of balancing the distribution of blocks across the network, each peer attributes a weight to each block it owns. The weight w_i of each block is updated according to the perception of the peer about its availability, which is based on the acceptance or cancellation of the offered blocks. The blocks to offer are picked through a random weighted selection, and recently downloaded blocks start with a weight w_s. When a block is offered, if it is accepted, the weight is updated using an additive factor, w_λ; otherwise, the weight is updated using a multiplicative factor, w_η. Therefore, to ensure that the weight decrease on acceptance is never greater than the one on cancellation, the weight update is given by Eq. (1).

$$w_i = \begin{cases} max\left(1, w_i - w_\mu\right), & \text{if it is accepted} \\ max\left(1, \left\lfloor \frac{w_i}{w_\eta} \right\rfloor\right), & \text{otherwise} \end{cases} \qquad (1)$$

$$\text{where } w_\mu = min\left(w_i - \left\lfloor \frac{w_i}{w_\eta} \right\rfloor, w_\lambda\right)$$

With the Mistrustful P2P model there is no need to suddenly terminate or remove downloads nor to stop sharing because the provided protection does not depend on the time a peer keeps sharing a content, as long as cover and genuine downloads are treated the same way. To evaluate our model, we considered $w_s = 100$, $w_\lambda = 5$, and $w_\eta = 2$ with the goal of favoring more recent blocks.

6.7. Request backoff

The request backoff mechanism determines the delay between block requests to help maximizing the amount of useful blocks that can be obtained from the available block requests in the shortest time frame. The mechanism identifies the set of peers to which block requests can be sent (eligible peers), and determines for how long no block requests should be sent. Therefore, as the former is a direct result of individual peer behavior and the latter depends on the swarm behavior, we define the backoff time as a two-dimensional variable that has per peer and per swarm components. The peer backoff component provides the delay to return a peer to the set of eligible peers; the swarm backoff component provides the delay until the next block request. The actual backoff time is randomly generated within the interval $[0, b]$, where b is the calculated backoff time.

A block request has five possible outcomes: (1) refusal – the request is refused by the contacted peer; (2) cancellation – the request is canceled by the requester (duplicate block); (3) acceptance

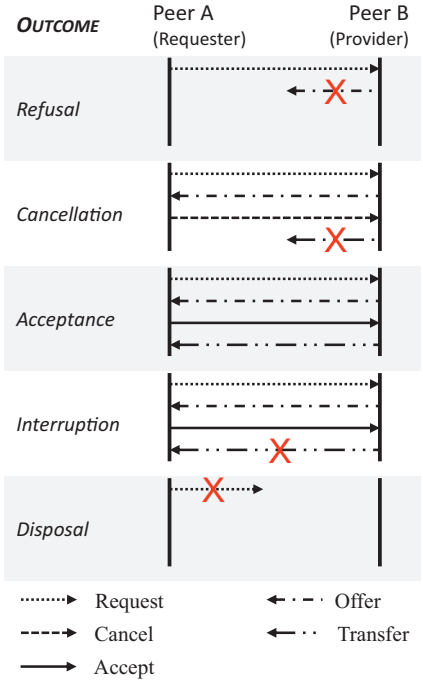

Fig. 7. Possible outcomes of a block request: refusal – the request is refused by peer *B* (no offer); cancellation – the request is canceled by peer *A* (no transfer); acceptance – the request is accepted and the block is transferred from peer *B* to peer *A*; interruption – the request is accepted but the block transfer is interrupted; disposal – no request is sent.

– the request is accepted and a block is downloaded; (4) interruption – the request is accepted but the download is interrupted; (5) disposal – no request is sent due to the lack of eligible peers. Refusal and disposal disclose no information about block ownership, but all the others do. Cancellation and acceptance reveal that both peers already own that block; interruption reveals that the contacted peer owns that block. Fig. 7 illustrates each possible outcome.

Consider u and v to be respectively the number of consecutive refused requests and the number of consecutive requests that were either canceled or interrupted, and u_i and v_i the same variables but for a peer i; by consecutive we mean until a request is accepted. Let μ, α, λ, β, η, and $\bar{\tau}$ be respectively the maximum backoff time, the base linear backoff time, the linear backoff time factor, the base exponential backoff time, the exponential backoff time factor, and the estimated average block transfer time. The backoff time b is then given by Eq. (2).

$$b = min\left(\mu, \alpha + \lambda u + \beta\left(\eta^{v} - 1\right)\right) \tag{2}$$

For block requests that do not disclose block information, u, it is applied a linear increase, λ; for block requests that disclose block information, v, it is applied an exponential increase, η. The backoff time never exceeds μ. The peer and the swarm components are expressed using Eq. (2), but the values of their variables may be different. Thus, the variables of each component are distinguished by a p subscript (peer) and an s subscript (swarm). For the swarm component only, when there are no eligible peers $\left(\forall_i, b_{p_i} > 0\right)$, $b_s = \min\left(\lambda_p, \min\left(b_{p_i}\right)\right)$.

For the sake of clarity and tractability, the evaluation assumes no simultaneous block requests, homogeneous Internet connections, and a single content download. Therefore, in order to evaluate our model, we considered $\alpha_p = 100$ ms, $\beta_p = \bar{\tau}/4$, $\lambda_p = \bar{\tau}/10$, $\eta_p = 2$, and $\mu_p = \frac{k\bar{\tau}c}{m}$ for the peer component. The per peer backoff time is a function of the block transfer time, and is instantiated such that the block requests sent to the same peer are, on average, 50 ms apart and such that, on average, m/c block requests can be sent to each peer during content download. m/c represents the configured protection and $k\bar{\tau}$ the minimum time required to download a given content. For the swarm component, we considered $\alpha_s = 0$, $\beta_s = \bar{\tau}/4\rho$, $\lambda_s = \bar{\tau}/16\rho$, $\eta_p = 2$, and $\mu_s = \bar{\tau}$, where ρ is the number of active peers. The swarm backoff time is a function of both block transfer time and the number of active peers, which can be obtained from the tracker(s). Given that the evaluation assumes no simultaneous block requests, there is no minimum delay required between consecutive requests, which can be, at most, the time required to download a single block.

Table 2
Summary of countermeasures employed to prevent common attacks.

Attack	Countermeasure(s)
DoS	Blacklist misbehaving peers (at application level).
Sybil and Collusion	Avoid block advertisement; employ the disclosure constraint mechanism; treat multiple peers using related IP addresses as pseudonymous identities of a single entity.
Peer Selection	Use multiple unrelated trackers for the same content.
Forgery and Repetition	Peers communicate through direct links.
Pollution	Each block has a checksum signed by the publisher.

6.8. Peer selection

The peer selection mechanism aims at selecting the eligible peer that has the highest probability of providing a useful block in less time. The set of eligible peers is constrained both by the disclosure constraint and request backoff mechanisms. The former provides, for a given content, the set of peers to which no further block requests can be sent to; the latter provides, for the same content, the set of peers that are ineligible for the moment, and when can the next block request be sent to each one of them.

For the sake of clarity and tractability, peers were selected randomly in order to avoid adding an additional factor into the evaluation. Non-uniform selection of peers is expected to impact on how the blocks end up distributed across the network, especially when considering heterogeneous Internet connections: peers with more available bandwidth are able to replicate more rapidly the blocks they own.

7. Security analysis

This section provides a security analysis of the Mistrustful P2P model. We start by describing the common attacks in P2P systems, and present the countermeasures employed. Then, we discuss the identification of user content interests, the proof of full content download, and user legal liability.

7.1. Common attacks and countermeasures

This section describes the common attacks in P2P file sharing and the countermeasures employed, which are summarized in Table 2, focusing on the context of our work. The attack model considered assumes that an attacker can be any entity participating in the system – a publisher, a tracker, a seeder or a commoner – or a group of those entities in collusion. External entities, such as ISPs and governments, are out of the scope of this work.

DoS. A denial-of-service attack aims either at shutting down the entire system or at making one or more contents unavailable. It may target trackers to prevent peers from knowing each other and therefore disable content sharing, or target seeders to prevent new

blocks from being introduced to harden or even preclude content download.

At the application level, DoS attacks are prevented by blacklisting attackers; at the network level, the Mistrustful P2P model presents the same vulnerabilities as any other P2P file sharing system.

Sybil and Collusion. The attacker creates a large number of pseudonymous identities (Sybil) or multiple malicious peers may coordinate their efforts (collusion) to increase the amount of blocks disclosed by other peers to either prove content download or determine user content interests. The size of the attack is defined by the number of unique peers (not known to be related).

As discussed in Section 6.3, both Sybil and collusion attacks of a size up to c are thwarted by not advertising what peers download or miss (avoiding block advertisement), and by employing the disclosure constraint mechanism. Avoiding block advertisement forces attackers to engage in the content sharing in order to gather information about the blocks owned by peers, increasing the resources required to launch such attacks. The disclosure constraint mechanism limits the amount of information that a single colluding group can gather. This protection may be extended by enabling the user to treat multiple peers using the same public IP address, such as those behind NAT, or related public IP addresses, such as those of a single organization, as pseudonymous identities of a single entity.

Peer Selection. Trackers may narrow the advertised lists of peers to increase the amount of block requests sent to malicious peers and thereby increase the amount of blocks disclosed to them.

Peer selection attacks are avoided by registering at multiple unrelated trackers and requesting them lists of peers in a swarm. As so, through comparison, misbehaving trackers can be identified and blacklisted. Nevertheless, this attack does not increase the amount of information that a single colluding group of a size up to c can gather.

Forgery and Repetition. These attacks try to either tamper with the data being transmitted or to retransmit authentic data previously captured in order to achieve their goals.

We assume that attackers, being participants of the system, cannot forge the source IP address of packets (IP address spoofing). As so, they are not able to forge nor repeat packets because peers communicate directly. Even considering IP address spoofing, as long as key exchange and distribution mechanisms are employed, which are out of the scope of this work, the integrity of packets can be ensured and the addition of a sequence number prevents repetition attacks.

Pollution. Malicious peers may share polluted blocks to waste resources and decrease the overall performance by making compliant peers to further share them.

Pollution attacks are thwarted by employing asymmetric cryptography to sign checksums of each block in order to verify its integrity. The publisher may, e.g., distribute the public key and the checksums along with content description or metadata.

7.2. Determine user content interests

The Mistrustful P2P model provides deterministic protection of user content interests against passive attacks of any size, and against active attacks of a size up to c. Passive attacks are defeated by avoiding block advertisement. Active attacks are thwarted by constraining the amount of blocks implicitly disclosed to an attacker of a size up to c to be at most m, and by downloading at least m blocks per cover content. The provided protection is deterministic because contents are only downloaded if the privacy requirements are met. If the size of an attacker exceeds c, then the provided protection may become probabilistic: the attacker may be able to know if the peer fully downloaded a content (genuine), and thus know that the user has interest in that content. However, our model still provides deterministic protection of user content interests against all other attackers of a size up to c.

The identification of genuine and cover contents may only be possible if the size of the actual attacker exceeds the size c of the largest colluding group considered (underestimated attacker). Still, the identification of a content as cover, per se, only enables an attacker to reduce the set of contents that the user may be interested in, given that the user may have no interest in any of them. Thereby, an underestimated attacker still has to prove content download in order to determine user content interests, as long as the actual amount of blocks downloaded per cover content can take any value between m (inclusive) and k (exclusive).

7.3. Prove content download

The protection provided by the Mistrustful P2P model is deterministic as long as the size of the actual attacker does not exceed c. Therefore, herein we discuss the necessary conditions for an underestimated attacker of size $c+\delta$ to be able to prove content download, where δ is the difference between the actual and the configured sizes of the largest attacker.

The disclosure constraint mechanism ensures that, at most, m blocks can be disclosed to any set of c peers. Let σ be the minimum amount of blocks disclosed to any of the top c peers, which can be, at most, m/c (the case in which the same amount of blocks is disclosed to all c peers). Then, given that the amount of blocks that can be disclosed to each one of the δ peers is at most σ, an underestimated attacker may be able to prove content download if, and only if, $m+\delta\cdot\sigma \geq k$. I.e., the provided protection is still deterministic for $0 < \delta < (k-m)/\sigma$.

For $\delta \geq (k-m)/\sigma$, the provided protection becomes probabilistic, as provided by other P2P privacy-preserving systems, and an underestimated attacker may be able to prove content download.

7.4. Legal liability

The multitude and complexity of copyright laws across the globe make it impossible to define clear boundaries regarding user legal liability. Still, the user should not be held legally liable for, unknowingly and unwillingly, downloading an illegal content if the main motivation to use a P2P system is to share legit contents, and it cannot be proven that the user had access to the content data.

The extent of user liability and the strength of plausible deniability depend on the main motivation to use the system, and on the type of contents that are distributed by the P2P file sharing system: if the system is mostly used to distribute illegal content, it is less plausible that the user intended to share legit contents when using it. Therefore, it is important to endow the content interest disguise scheme with the means to distinguish legit from illegal contents in order to ensure that the main motivation to use the system comes mostly, if not completely, from legit content sharing.

A legitimate usage of a P2P system based on our model may only result in the download of an illegal content either due to inadvertently considering an illegal cover content as legal or due to a misleading description. For the first case, given that the user is never granted access to the content data because cover contents are never fully downloaded, the user is not subject to any legal liability. For the second case, as long as actual attacker is not underestimated, the user is also not subject to legal liability because

it cannot be proven that the user had access to the content data. Even if a misleading content download can be proven by an underestimated attacker, it is our belief that, as long as the percentage of such misleading contents remains low, it is plausible that the user may have been driven to unknowingly and unwillingly download that content.

8. Evaluation

The evaluation of our model was conducted through simulation, and, given that simulations are only as good as their models, they were carried out using the ns-3 discrete-event network simulator [39], which provides realistic models of the network stack and its protocols. Still, the simulation of large-scale P2P networks using accurate models generates a very large number of events so the time required to run simulations is large.

In order to be able to simulate P2P content sharing with several thousands of peers using accurate network layer models, we also used the CIDRarchy module[6][40]; a module that we developed for ns-3 that takes advantage of the hierarchical structure of Internet-like network topology to optimize the time taken to perform IP packet forwarding, and therefore without changing the accuracy of the models. CIDRarchy was implemented and evaluated in ns-3 simulator, and the simulation time gains over existing routing protocols can reach over one order of magnitude: e.g., a simulation taking one week to complete may require less than one day to complete when using CIDRarchy. Simulating the application layer also requires modeling, among other functions, peer arrivals and departures, and peer join-participate-depart cycles (sessions), as the content download may span across multiple sessions [41].

The two main goals of the evaluation are to show that peers are able to timely download contents without advertising what they download, and to estimate the impact of privacy preservation on the average download bitrate; it is not our goal to optimize the overall performance of our model. To do so, we simulated the content sharing to evaluate the ratio of peers that are able to complete their downloads and the average download bitrate. The results obtained were compared against those of a traditional P2P file sharing model. As referred in Section 4, a content is considered to have been timely downloaded if the average download bitrate is within the same order of magnitude of traditional P2P file sharing models.

We are interested in evaluating the impact of content size, peer arrival rate, number of seeders, c and m, and cover downloads on the average download bitrate and on the download completion ratio. For the sake of clarity and tractability, we aim at reducing the impact of other variables on the overall performance, including those referred in the literature [41,42] such as session length, peer lifetime (time between first arrival and last departure), downtime, uptime, lingering time (additional time a peer lingers in the system after download completion), and inter-content relations. We consider that peers have homogeneous Internet connections, do not perform simultaneous chunk requests, always attempt to complete the download in a single session, and leave immediately after completing the download (worst case). Also, we consider a single genuine content download in order to enable fair performance comparison with traditional P2P systems, and thus cover downloads are emulated by increasing the peer arrivals and by having those peers to download only a fraction of the content.

The remainder of this section is organized as follows. First, we describe how peer arrivals are generated from real peer arrival traces. Then, we characterize the simulation setup for both P2P models. Lastly, we define the use cases considered.

[6] The source code for CIDRarchy is available at https://gitlab.inesctec.pt/pmms/CIDRarchy.

8.1. Peer arrivals

A content download is usually broken into three stages: *flash crowd, steady-state*, and *end phase* [43,44]. The *flash crowd* is the most demanding stage because there is a sudden burst of peer arrivals, which largely surpass the peer departures. The *steady-state* stage is characterized by an equilibrium between arrivals and departures. The *end phase* stage comprehends the end of life of a content where there are fewer arrivals than departures. Therefore, an ordinary Poisson arrival process is not able to capture the peer arrival dynamics because the mean peer arrival rate changes over time.

In order to ensure that the peer arrival rates are realistic, we gathered the peer arrival traces of several contents and then we used an exponential function to generate the peer inter-arrival times and change the mean arrival rate every 10 min (non-homogeneous Poisson process). The traces were gathered by monitoring a widely used tracker (*open.demonii.com*), and provide the number of first time peer arrivals over 10 min intervals since content publication up to 21 days. We did not use the gathered peer arrival traces directly because trackers, per request, provide a list of, at most, 200 peers currently in a swarm but not the time instant of their arrival. Therefore, we sent a request to the tracker every 400 milliseconds and considered that a peer has arrived for the first time at the time interval it got firstly listed.

8.2. Simulation setup

According to Akamai's State of the Internet Q3 2016 report [45], the global average peak connection speed, which is considered to be more representative of the Internet connection capacity [46], is 37.2 Mb/s. Therefore, we consider a star network topology with a central node mimicking an ISP, and with homogeneous leaf nodes connecting to it through asymmetric links: 30 Mbit/s downlink, 3 Mbit/s uplink, and 1 ms latency (2 ms delay between peers) to avoid any latency issues.

Peers communicate using TCP New Reno with a maximum transmission unit (MTU) of 1500 bytes, maximum segment size (MSS) of 1460 bytes, and with Nagle's algorithm [47] disabled. Peers are provided with a list of all other peers currently in the swarm, request one block at a time, accept one request at a time, always attempt to complete the download in a single session, and leave immediately after completing the download. Version 3.23 of ns-3 was used. Simulations were run for the first 48 h of content sharing because the most demanding stage, *flash crowd*, usually ends within the first 36 h. Thus, the simulation fully encompasses *flash crowd* stage and ends in *steady-state* stage.

Given that we do not consider simultaneous chunk requests for the sake of clarity and tractability, we defined a simplified model of the BitTorrent protocol [48] that achieves an average download bitrate that is, at least, in line with BitTorrent's. The peer roles and their sharing behavior are described in Section 3. The rarest first mechanism is simulated by letting the tracker keep a global list of the number of replicas of each block. Using this list, a peer picks randomly one of the ten rarest blocks (local rarity variation) and selects an available peer owning that block to send a request to. The set of peers cannot be periodically improved using the optimistic unchoking mechanism because there are no simultaneous downloads. As so, if the request fails because there are no peers available to provide that block, we enable a backoff mechanism to prevent excessive protocol overhead. The backoff time increases linearly as a function of average block download time τ, never exceeding 30 s (typical optimistic unchoking period), and is given, in milliseconds, by $b = \min\left(30000, \frac{\tau}{4} \cdot u\right)$, where u is the number of consecutive failed requests. As for the Mistrustful P2P model,

Table 3
Use cases considered for evaluation of the Mistrustful P2P model and their categories. The set of use cases for each category results from using different values for c, m, number of seeders, peer arrival traces – more popular (MP), popular (P), and less popular (LP) –, and content sizes. Over the first 48 h, the total number of peer arrivals for MP, P, and LP traces are respectively 75800, 22700, and 3400 (approximately).

Category	Block Disclosure (m)	Collusion Size (c)	No. of Seeders	Peer Arrival Traces	Content Size (MiB)	No. of Use Cases
Baseline	$k-1$	1, 31	1, 64	*MP*, *P* or *LP*	100, 800	24
Overhead	$k/2$	1, 31	1, 64	*MP*, *P* or *LP*	100, 800	24
Disguise	$k-1$[a]	1, 31	1, 64	*MP*, *P* or *LP*[b]	100, 800	24
Traditional	n.a.	n.a.	1, 64	*MP*, *P* or *LP*	100, 800	12

[a] One third of the peers only downloads and shares half of the content to simulate a cover download.
[b] The mean peer arrival rates have a 50% increase to simulate arrivals due to cover downloads.

the actual backoff time is randomly generated within the interval $[0, b]$.

8.3. Use cases

We consider 84 use cases to evaluate how content size, popularity (overall peer arrival rate), number of seeders, collusion size c, minimum network disguise overhead m, and cover downloads affect average download bitrate and download completion ratio; by use case we mean an evaluation using a distinct set of values for the variables in study.

The use cases are divided into four categories: *baseline, overhead, disguise*, and *traditional*. The *baseline* category represents the less restrictive protection against any colluding group of a size up to c – all but one blocks can be disclosed to c peers ($m = k-1$) –, and is used to estimate by comparison the impact of minimum network disguise overhead and cover content downloads on the sharing of genuine contents. The *overhead* category represents a reduced minimum network disguise overhead in which all peers fully download a genuine content but only disclose half of the blocks to any group of c peers ($m = k/2$). The *disguise* category represents the employment of a content interest disguise scheme where 50% more peers download 50% of the content; therefore, we consider a 50% increase on the mean peer arrival rates (truncated to the nearest integer), and that one third of all peers only downloads 50% of content blocks before leaving. I.e., the number of active peers increases but also the resource usage. The *traditional* category represents traditional P2P systems, in which no privacy-preserving mechanisms are employed.

For the first three categories we define 24 use cases. For the last category we define the same use cases except those for collusion size variants, totaling 12 use cases. We consider the content size to be either 100 MiB or 800 MiB, the number of seeders to be either 1 or 64, and three video traces to compare different degrees of popularity: a more popular (MP), a popular (P), and a less popular (LP) contents. Over the first 48 h, the total number of peer arrivals for MP, P, and LP traces are respectively 75800, 22700, and 3400 (approximately). Except for the last category, collusion size is either 1 or 31. Table 3 summarizes the use cases and their categories.

9. Results and discussion

Herein, we address the two main goals described in the previous section: (1) to show that peers are able to timely download contents without advertising what they download, and (2) to estimate the impact of privacy preservation on the average download bitrate and on the download completion ratio. The main performance metric considered by P2P file sharing users is the average download time or the average download bitrate, which are two sides of the same coin. The remainder of this section is structured as follows. In order to demonstrate the feasibility of block advertisement avoidance, we start by comparing our model against the traditional P2P model for the case of minimum protection, and by detailing the download completion ratio for each single use case. Then, we discuss the results obtained for all *baseline* use cases to assess the impact of collusion size, content size, number of seeders, and content popularity, which are the variables that are expected to change more often. We end by comparing the use cases of *baseline, overhead, disguise* and *traditional* categories to, respectively, evaluate the impact of minimum network disguise overhead, estimate the impact of cover downloads (additional peer arrivals and partial downloads), and to assess the impact of the provided protection as a whole.

Fig. 8 provides a comparison of the average download bitrate achieved by the traditional P2P model and by the Mistrustful P2P model, which is set for minimum protection – *baseline* use cases for single peer attacks – given that traditional P2P systems do not provide any privacy protection. The results obtained by both models are equivalent, despite considering an optimistic model for representing traditional P2P systems, and show that peers are able to timely download contents without advertising what they download. The performance difference is negligible when considering single seeder use cases, and more noticeable for some use cases when considering 64 seeders. The benefit provided by the larger number of seeders seems to be dependent on the ratio between seeders and regular peers because it fades as the number of simultaneous peers increases, be it due to higher peer arrival rate or larger content size that requires peers to stay longer to complete their download. This correlation with the peer arrival rate is more evident for the 800 MiB content using the popular peer arrival trace (center right).

Table 4 provides the number and ratio of downloads completed for all 84 use cases, and shows that peers are able to complete their downloads: the download completion ratio is always above 96%. This ratio is below 100% for all use cases because there is always a set of peers that is unable to complete the download: peers arriving when the time left to end the simulation is less than the time required to complete the download. Thus, the 800 MiB use cases achieve a lower download completion ratio than their 100 MiB counterparts. The download completion ratio is identical for all categories, but the number of downloads is different for the *disguise* category because the peers used to emulate cover downloads (one third of all peers) are not considered as they only download partially the content (50%). The lowest download completion ratio is achieved for the popular (P) peer arrival trace, in particular for 800 MiB content size, due to an increase on the peer arrival rate near the end of the simulation (see Fig. 8).

All *baseline* use cases are depicted in Fig. 9 to assess the impact of collusion size, content size, number of seeders, and content popularity. A collusion size of up to 31 peers requires a peer to contact, at least, 32 unique peers to be able to complete the download. Thereby, when considering 64 seeders, the results are identical for both a collusion of 1 and of 31 because it is guaranteed that there are always enough peers available. However, in the use cases considering a single seeder and a collusion of 31, mainly for smaller and less popular contents as there are less simultaneous peers, the correlation between the average download bitrate and the peer arrival rate is evident. As the number of

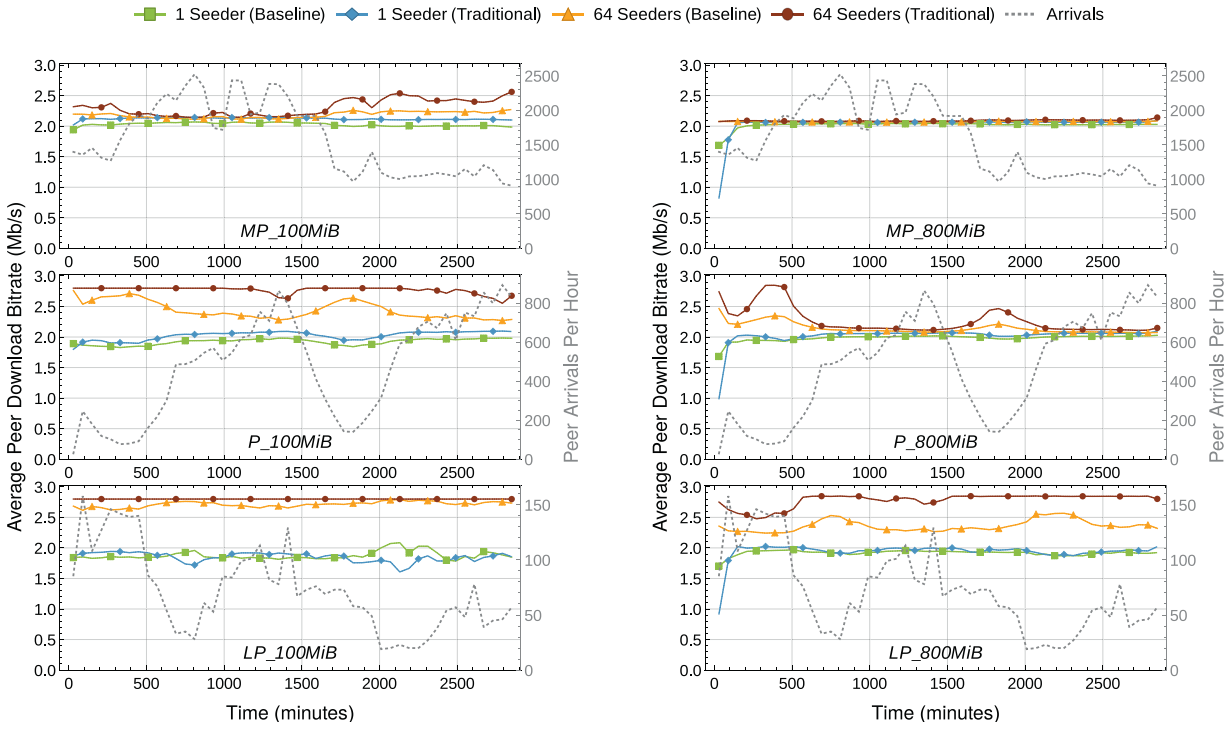

Fig. 8. Average download bitrate over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (*MP*), a popular (*P*), and a less popular (*LP*) peer arrival traces as input (one per row). Each plot depicts four use cases that are a result of using either 1 or 64 seeders, and considering either our model (*baseline* use cases of single peer attacks) or the traditional P2P model. The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

Table 4
Number and ratio of downloads completed for all 84 use cases. The use cases are presented grouped by peer arrival trace – more popular (MP), popular (P), and less popular (LP) –, content size (100 and 800 MiB), and category – *baseline, overhead, disguise* and *traditional* –, for a collusion of either 1 or 31, and either 1 or 64 seeders.

Peer Arrival Trace	Content Size (MiB)	Category	1 Seeder, Collusion of 1	1 Seeder, Collusion of 31	64 Seeders, Collusion of 1	64 Seeders, Collusion of 31
MP	100	*Baseline*	75,777 (99.87%)	75,775 (99.87%)	75,788 (99.88%)	75,788 (99.88%)
		Overhead	75,775 (99.87%)	75,773 (99.86%)	75,788 (99.88%)	75,787 (99.88%)
		Disguise	75,933 (99.84%)	75,935 (99.84%)	75,945 (99.86%)	75,945 (99.86%)
		Traditional	75,785 (99.88%)	–	75,795 (99.89%)	–
	800	*Baseline*	75,030 (98.89%)	75,036 (98.89%)	75,058 (98.92%)	75,057 (98.92%)
		Overhead	75,033 (98.89%)	75,032 (98.89%)	75,054 (98.92%)	75,050 (98.91%)
		Disguise	75,150 (98.81%)	75,163 (98.83%)	75,176 (98.85%)	75,187 (98.86%)
		Traditional	75,052 (98.91%)	–	75,071 (98.94%)	–
P	100	*Baseline*	22,641 (99.46%)	22,641 (99.46%)	22,662 (99.56%)	22,662 (99.56%)
		Overhead	22,643 (99.47%)	22,642 (99.47%)	22,660 (99.55%)	22,664 (99.57%)
		Disguise	22,570 (99.59%)	22,570 (99.59%)	22,580 (99.63%)	22,581 (99.64%)
		Traditional	22,649 (99.50%)	–	22,674 (99.61%)	–
	800	*Baseline*	22,005 (96.67%)	21,995 (96.63%)	22,016 (96.72%)	22,017 (96.72%)
		Overhead	21,999 (96.64%)	22,001 (96.65%)	22,025 (96.76%)	22,021 (96.74%)
		Disguise	21,899 (96.63%)	21,892 (96.60%)	21,913 (96.69%)	21,911 (96.68%)
		Traditional	22,016 (96.72%)	–	22,042 (96.83%)	–
LP	100	*Baseline*	3427 (99.94%)	3379 (98.54%)	3428 (99.97%)	3428 (99.97%)
		Overhead	3426 (99.91%)	3356 (97.87%)	3428 (99.97%)	3428 (99.97%)
		Disguise	3396 (99.97%)	3351 (98.65%)	3396 (99.97%)	3396 (99.97%)
		Traditional	3427 (99.94%)	–	3428 (99.97%)	–
	800	*Baseline*	3374 (98.40%)	3363 (98.08%)	3386 (98.75%)	3384 (98.69%)
		Overhead	3371 (98.31%)	3342 (97.46%)	3384 (98.69%)	3383 (98.66%)
		Disguise	3340 (98.32%)	3339 (98.29%)	3348 (98.56%)	3345 (98.47%)
		Traditional	3374 (98.40%)	–	3393 (98.95%)	–

simultaneous peers increases, be it due to higher peer arrival rate or larger content size, the performance gap between collusion of 1 and collusion of 31 for single seeder use cases fades until it becomes negligible. Thus, to compare the performance of *baseline* category with all others, we focus on single seeder and collusion of 31 use cases as they provide the worst case and enable a better assessment of the impact of each variable. For the sake of clarity over completeness, we only provide plots for the edge use cases: 100 MiB and 800 MiB contents using, respectively, the less popular and the more popular peer arrival traces.

Figs. 10, 11, and 12 provide the results obtained in each use case category – *baseline, overhead, disguise*, and *traditional* – for, respectively, the ratio of block requests sent to seeders out of all block requests, the average download bitrate, and the average ratio

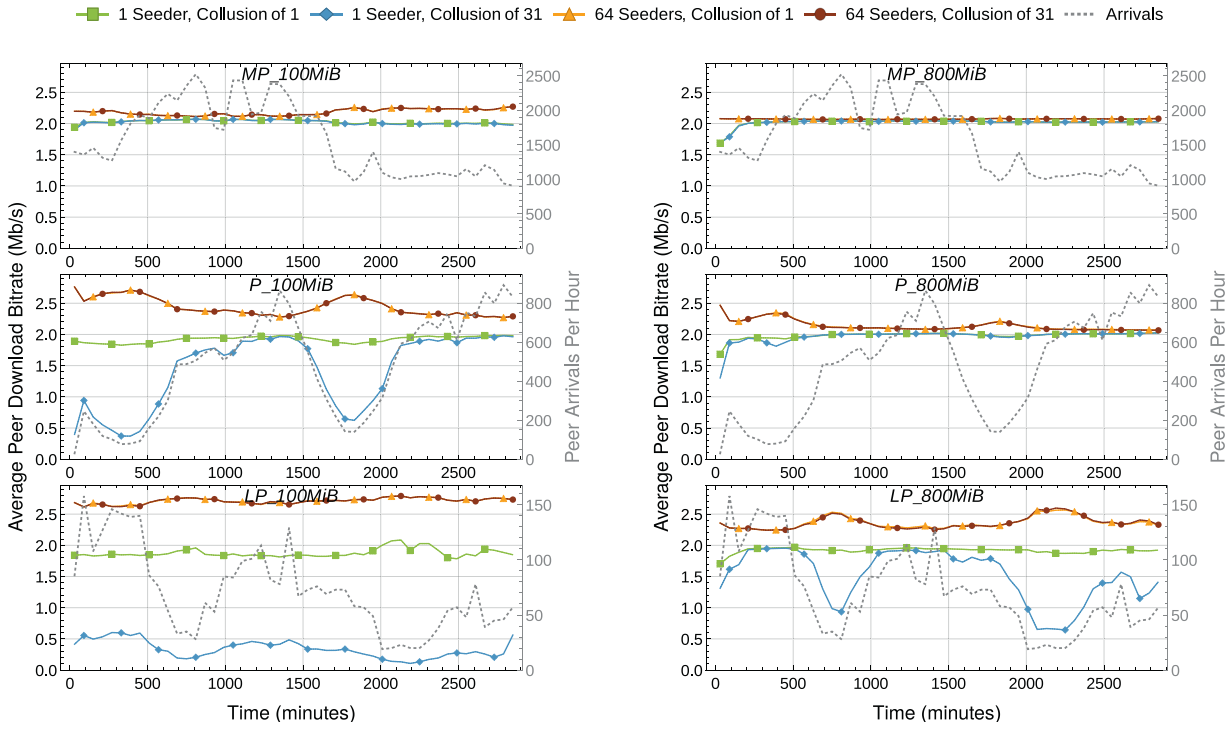

Fig. 9. Average download bitrate over one hour periods for all *baseline* use cases. Contents have either 100 MiB (left) or 800 MiB (right) and use a more popular (*MP*), a popular (*P*), and a less popular (*LP*) peer arrival traces as input (one per row). Each plot depicts four use cases that are a result of using either 1 or 64 seeders, and considering either single peer attacks or collusion attacks of, at most, 31 peers. The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

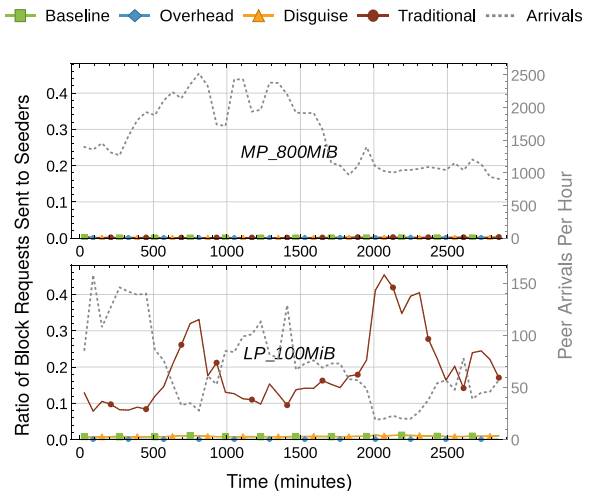

Fig. 10. Average ratio of requests sent to seeders over one hour periods for a 800 MiB (top) and a 100 MiB (bottom) contents using respectively a more popular (*MP*) and a less popular (*LP*) peer arrival traces as input. Each plot depicts each use case category – *baseline, overhead, disguise*, and *traditional* – for a single seeder and collusion attacks of, at most, 31 peers (except *traditional*). The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

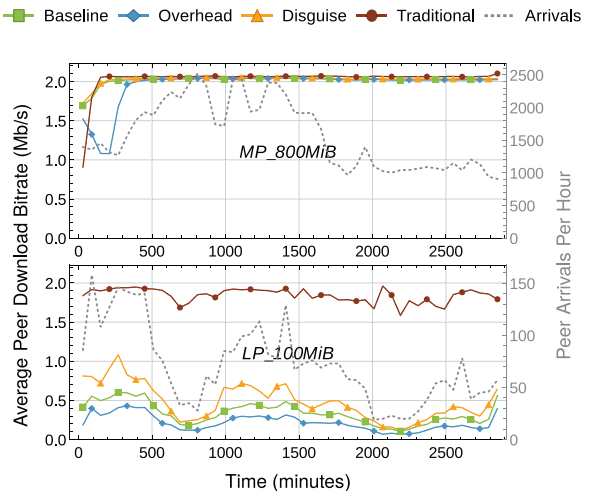

Fig. 11. Average download bitrate over one hour periods for a 800 MiB (top) and a 100 MiB (bottom) contents using, respectively, a more popular (*MP*) and a less popular (*LP*) peer arrival traces as input. Each plot depicts each use case category – *baseline, overhead, disguise*, and *traditional* – for a single seeder and collusion attacks of, at most, 31 peers (except *traditional*). The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

of time spent on backoff (average backoff time ratio). As shown by Fig. 10, the main reason for the performance gap between our model, for all use cases using the less popular peer arrival trace, and the traditional P2P model is the exploitation of the seeder. Unlike the traditional P2P model that favors requests to seeders, our model treats all peers alike and, on average, shares no more than m/c blocks with each individual peer, including the seeder. For this reason, our model presents a low ratio of block requests sent to seeders, which is not subject to significant variations.

In the *overhead* use cases, peers can disclose, at most, 50% of the blocks they download to any set of c peers ($m = k/2$) hardening the probability of retrieving useful blocks, in particular during the *flash crowd* stage because the few useful blocks are being uploaded mostly by seeders. Given that all contents are divided into 64 blocks, increasing the content size augments this effect as seeders will take more time to share the blocks required. This effect is noticeable for the most popular content in Fig. 11 but not for the less popular content because the main bottleneck with the former is the unavailability of blocks and not the peer unavailability as with the latter. As shown by Fig. 12, apart from the beginning, the average backoff time ratio is low throughout the content sharing of the more popular content while it is high throughout the

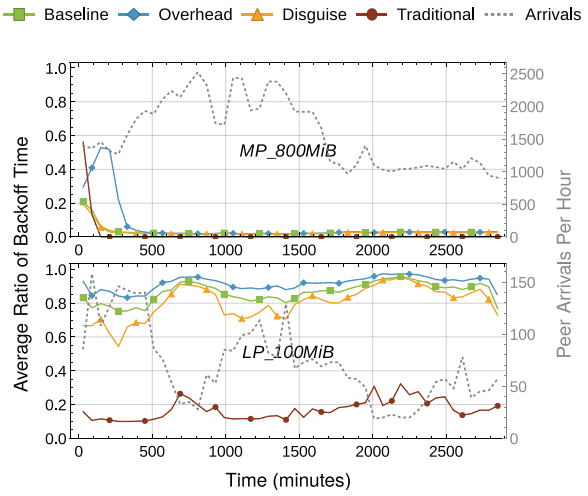

Fig. 12. Average ratio of backoff time over one hour periods for a 800 MiB (top) and a 100 MiB (bottom) contents using, respectively, a more popular (*MP*) and a less popular (*LP*) peer arrival traces as input. Each plot depicts each use case category – *baseline, overhead, disguise*, and *traditional* – for a single seeder and collusion attacks of, at most, 31 peers (except *traditional*). The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

content sharing of the less popular one. Therefore, we can conclude that the impact of minimum network disguise overhead, as for collusion size, depends on the number of seeders and simultaneous peers. Its impact dilutes as the number of simultaneous peers increases.

In the *disguise* use cases, we considered a 50% increase of the peer arrival rate due to cover downloads and that those additional peers only download and share 50% of the blocks required to complete the download. As for all other peers, they leave immediately once they download the content. The results obtained indicate that cover downloads improve the performance of our model on all use cases. Still, this is just an estimation that highly depends on the scheme being used to select cover contents and how much of each to download in order to disguise user content interests.

In the *traditional* use cases, despite considering an optimistic model that has global knowledge of blocks availability/rarity and that is able to take more advantage of seeders, around 20% of time is still spent on backoff when used with less popular contents. This highlights the importance of the number of simultaneous peers on the overall performance, which is amplified by our model when considering protection against larger attackers, given that peers need to wait for other peers to join before being able to complete the download (temporary unavailability of peers).

In sum, the results show that our model is feasible: peers are able to timely download contents without advertising what they download. The performance of the Mistrustful P2P model, when considering minimum protection ($c = 1$, and $m = k - 1$), is close to the one of the optimistic model that represents traditional P2P file sharing systems. The impact of stronger protection depends significantly on the number of simultaneous peers and, after the *flash crowd* stage, becomes negligible for popular contents. For contents with similar size and popularity, the number of simultaneous peers can be increased either by adding more seeders, which also help to improve the distribution and diversity of blocks across the network, or by having peers to download cover contents. Although dependent on the scheme used for content interest disguise, cover downloads are expected to improve the overall performance.

10. Conclusions

The Mistrustful P2P file sharing model hides user content interests through content interest disguise in order to provide plausible deniability to the user. It has no trust requirements to enable content sharing in large groups of untrusted peers, and prevents user liability in case of legitimate usage while enabling timely content downloads. The provided protection enables the user to configure, per content, the required trade-off between privacy and performance by setting the size c of the largest colluding group to be protected against, and the minimum amount m of blocks that need to be downloaded in order to ensure that genuine and cover contents are indistinguishable to any attacker of a size up to c. We discussed Mistrustful P2P model's legal and ethical framework, demonstrated its feasibility for more use cases, provided a security analysis, compared it against a traditional P2P file sharing model, and improved its main mechanisms.

The legal and ethical framework described the main legal and ethical challenges brought by the advances in computer technology, focusing on the legal and privacy dimensions of P2P file sharing. The multitude and complexity of copyright laws across the globe make it impossible to define clear boundaries regarding user liability. Still, it is our belief that the user will not be subject to any legal liability for, unknowingly and unwillingly, downloading an illegal content if the main motivation to use a P2P system is to share legit contents, and it cannot be proven that the user had access to the content data. Thus, any privacy-preserving P2P system should ensure that the main motivation for its adoption is to share legit contents in order to minimize the extent of potential user liability.

We demonstrated the feasibility of the Mistrustful P2P model through simulation, using ns-3, and evaluated the impact of privacy preservation on the average download bitrate, and download completion ratio, considering that peers leave immediately after finishing the download. In the majority of the use cases considered, the average overall download bitrate is close to the one of the traditional P2P model. With the Mistrustful P2P model, peers have no need to suddenly terminate or remove downloads because the provided protection does not depend on the time a peer keeps sharing a content.

The security analysis presented the countermeasures employed against common P2P file sharing attacks, and discussed the identification of user content interests, proof of full content download, and user legal liability. The protection provided by the Mistrustful P2P model is deterministic, and therefore only an underestimated attacker (of a size exceeding c) may be able to determine user content interests, which requires proof of full content download. Nevertheless, the size of an underestimated attacker should be significantly higher than c before it is able to prove content download.

The Mistrustful P2P model reinforces plausible deniability by (1) not advertising what peers own or miss in order to defeat passive attacks of any size, and by (2) constraining the amount of blocks implicitly disclosed to other peers while sharing in order to thwart active attacks of a size up to c. Unlike in other P2P systems, peers are not required to relay traffic on behalf of other peers, and therefore users are also not subject to indirect liability. Proof of access to content data and proof of full content download are avoided through (1), (2), and by encoding contents in a way that only enables decoding after full download, preventing direct legal liability. Attackers have then to engage in content sharing to know which blocks a peer owns, increasing significantly the computational resources required to launch attacks.

The current version of the Mistrustful P2P model, when compared to [15], introduces significant changes mainly to three mechanisms: request backoff, block selection, and disclosure constraint. The request backoff mechanism was updated to be also a function of c, m and the size of the swarm in order to better adapt to different privacy configurations and to the peer dynamics, and to apply a linear or an exponential adjustment depending on the outcome of the block request: linear, if no block is disclosed, and exponential otherwise. The block selection mechanism was improved by using a weighted random selection of blocks, being the weights

updated according to the outcome of the incoming block requests. The purpose of the parameters c and m was clarified on the disclosure constraint mechanism, motivating its renaming.

As future work, we intend to do the following. Further characterize the Mistrustful P2P model by considering more variables such as the number of blocks into which a content is divided. Improve the block selection mechanism by updating the weights based on both incoming and outgoing block requests. Propose a content interest disguise scheme that aims at minimizing the amount of cover traffic required for a given protection level by varying the value of m per cover content while preserving the disguise of the genuine downloads. Evaluate the Mistrustful P2P model for parallel downloads and heterogeneous Internet connections. In such scenarios, evaluate the integration of variables such as the average block transfer time of each peer, the Internet connection bandwidth and latency deviations, and the number of concurrent downloads into the backoff time equations. Provide protection against link monitoring by encrypting communications between peers to extend the attack model in order to also consider ISPs and governments, which requires key exchange and distribution mechanisms. Design an incentive mechanism to stimulate sharing.

Acknowledgments

This work is financed by FEDER through Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme through the Agência Nacional de Inovação (ANI) within the scope of the project no. 3468 (MareCom). This work is also financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project "POCI-01-0145-FEDER-006961", and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013. The first author also thanks FCT for the grant under the fellowship SFRH/BD/69388/2010.

References

[1] The Freenet Project, Inc., FreeNet Project, (https://freenetproject.org/) [Online; accessed Feb. 27, 2017].
[2] P.P. Tsang, A. Kapadia, C. Cornelius, S.W. Smith, Nymble: blocking misbehaving users in anonymizing networks, Dependable Secure Comput. IEEE Trans. 8 (2) (2011) 256–269.
[3] R. Dingledine, N. Mathewson, P. Syverson, Tor: the second-generation onion router, in: Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, in: SSYM'04, 2004.
[4] R.A. Spinello, Intellectual Property: Legal and Moral Challenges of Online File Sharing, John Wiley & Sons, Inc., pp. 553–569.
[5] M. Whitman, H. Mattord, Principles of Information Security, Cengage Learning, pp. 89–116.
[6] J. Clough, Principles of Cybercrime, 1st, Cambridge University Press, New York, NY, USA, 2010.
[7] Berne Convention, (http://www.wipo.int/treaties/en/ip/berne/) [Online; accessed Feb. 27, 2017].
[8] M. Husberg, Blocking injunction requisites - the balancing of rights and other aspects of blocking injunctions towards intermediaries, 2015. Graduate thesis.
[9] D.R. Choffnes, J. Duch, D. Malmgren, R. Guierm, F.E. Bustamante, L. Amaral, SwarmScreen: Privacy Through Plausible Deniability in P2P Systems, Technical Report, Northwestern EECS, 2009.
[10] D. Goldschlag, M. Reed, P. Syverson, Onion routing, Commun. ACM 42 (2) (1999) 39–41.
[11] S. Katti, J. Cohen, D. Katabi, Information slicing: anonymity using unreliable overlays, in: Proceedings of the 4th USENIX Conference on Networked Systems Design and Implementation, in: NSDI'07, 2007.
[12] A. Chaabane, P. Manils, M.A. Kaafar, Digging into anonymous traffic: a deep analysis of the Tor anonymizing network, in: Network and System Security (NSS), 4th International Conference on, 2010, pp. 167–174.
[13] S. Chakravarty, G. Portokalidis, M. Polychronakis, A. Keromytis, Detection and analysis of eavesdropping in anonymous communication networks, Int. J. Inf. Secur. 14 (3) (2015) 205–220.
[14] S. Le Blond, P. Manils, A. Chaabane, M.A. Kaafar, C. Castelluccia, A. Legout, W. Dabbous, One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users, in: Proceedings of the 4th USENIX Conference on Large-scale Exploits and Emergent Threats, in: LEET'11, 2011.
[15] P.M. da Silva, J. Dias, M. Ricardo, Mistrustful P2P: privacy-preserving file sharing over untrustworthy peer-to-peer networks, in: Proceedings of IFIP Networking 2016, in: IFIP Networking '16, 2016, pp. 395–403.
[16] K.E. Himma, H.T. Tavani (Eds.), The Handbook of Information and Computer Ethics, John Wiley & Sons, Inc., 2009.
[17] D.G. Johnson, K. Miller, Computer Ethics: Analyzing Information Technology, Prentice Hall, 2009.
[18] R. Capurro, Intercultural Information Ethics, John Wiley & Sons, Inc., pp. 639–665.
[19] J. Weckert, Y. Al-Saggaf, Regulation and Governance of the Internet, John Wiley & Sons, Inc., pp. 473–495.
[20] R.W. Borders, Enemies of the Internet, 2014a, (http://12mars.rsf.org/wp-content/uploads/EN_RAPPORT_INTERNET_BD.pdf) [Online; accessed Feb. 27, 2017].
[21] R.W. Borders, World press freedom index, 2016b, (https://rsf.org/en/ranking/2016b) [Online; accessed Feb. 27, 2017].
[22] B. Hazucha, Private copying and harm to authors: compensation versus remuneration, 2015, (http://ssrn.com/abstract=2699070).
[23] J. Poort, J.P. Quintais, The levy runs dry: a legal and economic analysis of EU private copying levies, JIPITEC 4 (3) (2013) 205–224.
[24] J.P. Quintais, Private copying and downloading from unlawful sources, IIC 46 (1) (2015) 66–92.
[25] International Survey on Private Copying: Law & Practice 2015, Technical Report, WIPO and Stichting de Thuiskopie, 2016.
[26] A. Vitorino, Recommendations resulting from the mediation on private copying and reprography levies, Technical Report, 2013.
[27] D. Price, Sizing the piracy universe, Technical Report, Net Names, 2013.
[28] P. Savola, The ultimate copyright shopping opportunity – jurisdiction and choice of law in website blocking injunctions, IIC - International Review of Intellectual Property and Competition Law 45 (3) (2014) 287–315.
[29] B. Cohen, Incentives build robustness in bittorrent, in: Workshop on Economics of Peer-to-Peer systems, volume 6, 2003, pp. 68–72.
[30] K. Bauer, D. McCoy, D. Grunwald, D. Sicker, BitBlender: Light-weight anonymity for BitTorrent, in: Proceedings of the Workshop on Applications of Private and Anonymous Communications (AIPACa 2008), ACM, 2008.
[31] T. Isdal, M. Piatek, A. Krishnamurthy, T. Anderson, Privacy-preserving P2P data sharing with OneSwarm, SIGCOMM Comput. Commun. Rev. 41 (4) (2010).
[32] S.J. Nielson, D.S. Wallach, The BitTorrent anonymity marketplace, CoRR abs/1108.2718 (2011).
[33] R. Petrocco, M. Capotă, J. Pouwelse, D.H. Epema, Hiding user content interest while preserving P2P performance, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 501–508.
[34] J.R. Douceur, The Sybil Attack, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 251–260.
[35] Z. Trifa, M. Khemakhem, Mitigation of Sybil attacks in structured P2P overlay networks, in: 2012 Eighth International Conference on Semantics, Knowledge and Grids, 2012, pp. 245–248.
[36] A. Mohaisen, J. Kim, The sybil attacks and defenses: A survey, Smart CR 3 (6) (2013) 480–489.
[37] P.M. da Silva, J. Dias, M. Ricardo, Storm: rateless MDS erasure codes, in: Wireless Internet, Springer, 2015, pp. 153–158.
[38] M.-Z. Shieh, S.-C. Tsai, M.-C. Yang, On the inapproximability of maximum intersection problems, Inf. Process. Lett. 112 (19) (2012) 723–727.
[39] NS-3 Consortium, ns-3 network simulator, 2017, (https://www.nsnam.org/) [Online; accessed Feb. 27, 2017].
[40] P.M. da Silva, J. Dias, M. Ricardo, CIDRarchy: CIDR-based ns-3 routing protocol for large scale network simulation, in: Proceedings of the 8th International Conference on Simulation Tools and Techniques, in: SIMUTools '15, 2015, pp. 267–272.
[41] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, in: IMC '06, ACM, New York, NY, USA, 2006, pp. 189–202.
[42] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, A performance study of BitTorrent-like peer-to-peer systems, IEEE J.Sel. A. Commun. 25 (1) (2007) 155–169.
[43] J. Pouwelse, P. Garbacki, D. Epema, H. Sips, The BitTorrent P2P file-sharing system: measurements and analysis, in: Proceedings of the 4th International Conference on Peer-to-Peer Systems, in: IPTPS'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 205–216.
[44] C. Carbunaru, Y.M. Teo, B. Leong, T. Ho, Modeling flash crowd performance in peer-to-peer file distribution, IEEE Trans. Parallel Distrib. Syst. 25 (10) (2014) 2617–2626.
[45] State of the Internet Q3 2016, Technical Report, Akamai, 2016.
[46] Akamai, State of the Internet metrics: what do they mean?, 2017, (https://blogs.akamai.com/2015/02/state-of-the-internet-metrics-what-do-they-mean-1.html) [Online; accessed Feb. 27, 2017].
[47] J. Nagle, Congestion Control in IP/TCP Internetwork, RFC, 1984.
[48] D. Harrison, Bittorrent protocol specification and enhancement proposals, 2008, (http://www.bittorrent.org/beps/bep_0000.html) [Online; accessed Feb. 27, 2017].

Pedro Moreira da Silva received his MSc degree in Informatics and Computing Engineering from Universidade do Porto, in 2009. He is a researcher at the Centre for Telecommunications and Multimedia (CTM), INESC TEC, focusing on the security and privacy aspects of network communications. He is currently pursuing a PhD degree in Informatics Engineering on the privacy aspects of Peer-to-Peer file sharing.