Everyone is naked: Privacy on P2P systems

Gonçalo Pestana (goncalo@hashmatter.com), hashmatter

Abstract The ethos of the P2P community and the decentralized web is based on a new paradigm where people and communities can interoperate and collaborate without the need for external stewardship. This new paradigm opens doors for user self-sovereignty, personal data ownership and freedom from central authorities which incentives tend to lock-in their users and to hold and monetize private data in order to maximize their profits. However, P2P and decentralized protocols are complex and collaboration between multiple parties often leaks metadata that can be used to target individuals and communities. While centralized systems disclose user's behaviour and social graph to one centralized entity, naive decentralized systems potentially disclose that information to everyone in the network. While it is clear that current systems are not focusing on user privacy, failing to deliver it will render decentralized systems unusable and unattractive for mass adoption or as a viable alternative to centralized systems.

1. Introduction

In this article, we'll discuss one of the hardest challenges of decentralized web and P2P systems: privacy. We'll focus on current systems and applications built or relying on P2P networks, focusing on the rather interesting tension between scalability, performance, discoverability (or collaboration, for short) and privacy of the network users. We will demonstrate how current systems are a threat for user privacy and briefly discuss how recent research has been trying to solve current open problems while taking into considerations the tradeoffs between privacy and usability. Technology is a political tool [crypto-moral], so we finish by outlining the importance for building technology that protects users privacy by design and why P2P and decentralized systems must deliver on privacy for it to have a net positive impact on our society.

P2P, distributed systems, encryption, metadata, privacy: Throughout the article we will use concepts that often overlap and mean different things depending on the context. We define a distributed system as a system of entities that collaborate to achieve a common goal. In a distributed system, there can be a set of central authorities which orchestrate the network. A decentralized network is a subset of distributed systems where there is no centralized entity that controls the behavior of the network or maintains a global state. A P2P system is a system where connected peers behave as both clients and servers by consuming and providing resources to the network. Each network peer implements a set of protocols that enables peers to collaborate with each other. Notice how P2P, decentralized and distributed systems overlap at the edges. We will be focusing especially on P2P and decentralized systems.

Another concept we will discuss throughout the article is metadata. Metadata is often referred as a set of data that describes and gives information about other data. In the context of this article, we define metadata as any type of information (e.g. the fact that a connection occurred between two machines) that may be used to infer behaviour of a node in a P2P network. From now on, when mentioning *privacy* we refer to a metric of how much data and metadata a user leaks when using a system. Or from an attacker perspective, how much user personal information - interactions, social graph, interests - can be inferred by adversaries passively intercepting network requests. Notice that even if the communications are encrypted between peers, disclosing the fact that a communication happened will impact negatively the privacy of the intervenients. Similarly, depending on the network protocol, a peer caching and serving a resource may openly disclose its interest in it.

Threat model (or what we are trying to avoid): From a bird's eye view, we are aiming at building systems that do not leak data or metadata about user behaviour. The potential adversary is local, controls a fraction of the network (usually around 20% of the nodes) and may actively change and delay packets traversing the network. In the context of this article, we are not considering global adversaries with resources to track all activity in a network (e.g. governmental agencies). That's a different ball game and I believe we better start by focusing on the "low hanging fruit": on how to make the systems secure and private against local adversaries.

In a nutshell, interactions between peers in the network should not disclose any information over time about peers' interests and behaviour. Adversaries or curious honest peers should not be able to infer users' behaviour based on their network activity. Let's consider the following cases:

Case 1: User (the *initiator*) wants to access a webpage (a *resource*) which is stored in a distributed network. The *resources* stored in the network are content addressed and replicated by multiple peers (the *providers*). The *initiator* sends the request to a subset of the network peers to learn where to get the *resource* from. The request is routed across the network peers, which use their partial view of the network to resolve which peers are *providers* for the requested *resource*. Once a *provider* is discovered, the *initiator* requests the website from it.

The example above is a high level protocol for routing and discovery in decentralized networks. Since peers in the network collaborate to resolve the *initiator* request, they need to know which *resource* is being resolved. In addition, the *initiator* also learns that the *provider* has most likely accessed (or is the original creator of) the *resource* she requested.

Thus, adversaries or curious honest peers are able to infer the *initiator* and *provider* behaviour and interests based on what content they request and store, respectively.

Case 2: This is a particular instance of the Case 1. This example assumes that users will likely cache personal *resources* on their devices. A personal *resource* (e.g. personal webpage portfolio) is cached and replicated by a set of peers in the network, which map to physical devices such as laptops, mobile devices and servers with an IP address. An adversary may be able to learn which devices and IPs correspond to the owner of the resource by correlating information about which peers are providing a specific *resource*. Over time an adversary will be able to infer what are the IPs a network peer uses, which may disclose location and other personal information.

In real life, how can this be used to target individuals and communities? We will indulge in this question in later sections, but let's take a sneak-peak on how these metadata leaks may affect people in a very tangible way. Imagine a world in which a set of decentralized storages are used to cache and distribute videos. A distributed Youtube service of some sorts: instead of Youtube storing and distributing the videos, we would rely on a network of independent users that collaboratively provide the service. The routing and resource lookup protocol works as described in the Case 1 above. Now imagine a country where its government does not shy away from tracking down and prosecuting its own citizens who have contrarian ideas to that of the current power (spoiler alert for those who haven't realized it yet: they do exist!). In this context, decentralized networks can be used by adversaries to identify and prosecute people who created, stored and distributed content which is deemed as against the government. When the P2P network has location aware routing and content distribution based on peer location, it would be straightforward to track down and identify non conforming individuals in communities. These scenarios are short from being taken from a dystopian future in which freedom is part of the past because of how technology can be used to profile and target individuals and communities.

Notice how the adversary strategy changes when attacking decentralized systems, as compared to centralized systems. In centralized systems, the attacker strategy is to backdoor or enforce a corporation to disclose user behaviour and private information through litigations and subpoenas. In (naive) decentralized systems, the adversary strategy is to be part of the network and to passively record and correlate the network activity. Different paradigms, different strategies, the same problem. Technology shifts power, and thus it needs to be seen as a political tool [crypto-moral].

Next, we are going to see practical examples of how current P2P systems used everyday by thousands of people are leaving its users vulnerable to privacy attacks.

2. The emperor's new clothes

Attacking the pCDN: Content Distribution Networks are systems of distributed servers deployed across the globe that cache online content and distribute it based on the proximity of the content requester [1]. The objectives are to decrease latency of download and streaming by placing the servers where the content is stored closer to the end user. CDNs have become essential for many online applications that require concurrent distribution of high volumes of data to many simultaneous users. As a general rule, the closer the cache servers are to the end user, the lower the latency is, since the bits need to travel less in the wire. The pressure to deliver higher volumes of data at lower latency at an increasing scale has pushed CDNs to expand the number of cache servers in new regions. As end user's hunger for data and impatience keeps growing, it becomes harder for CDNs to keep up with the needs for deploying and maintaining servers at the edges. In order to improve scalability and efficiency in this context, CDNs are using P2P approach to caching and distribution of content, where end users that download resources contribute with caching and distribution of content. The P2P network of peers caching and distribution content is usually called a Peer-assisted Content Delivery Networks (pCDN). The advantages of having this mechanism are clear: end users replace physical servers in the last mile and ensure the network grows and scales organically without the need for deploying and maintaining more servers. Providers offering pCDN services claim 98% of CDN bandwidth savings and improved video latency [peer5] [viblast], when compared to traditional CDN networks.

Peer-assisted CDN networks are hybrid systems - leveraging distributed as well as decentralized networks -, and are a good example of the potential for P2P systems to help building scalable services by having peers collaborating and helping the network by contributing with their resources (disk, CPU, bandwidth). From a user privacy perspective, pCDNs present two distinct challenges. One one hand, the central authority (CDN provider) is able to “see" and log which content users are consuming, when, etc. On the other hand, the peers caching and distributing the content at the edges are also able to “see" what peers in the vicinity want to consume and when. When designed naively, pCDNs bundle the worst of both centralized and decentralized systems in terms of privacy, by disclosing user behaviour not only to a central authority but also to arbitrary network peers. pCDN providers claim to deliver on security and privacy because “all communication are encrypted" [peer5]. However, since the P2P network is not hardened against inference attacks, the claim is simply not true. Let's use Viblast pCDN [viblast] as an example why.

Viblast is a pCDN provider for video distribution which relies on a server-side caching (CDN) and browser technology (WebRTC) to implement a decentralized network to distribute content at the edges. The architecture works as follows: the server stores and distributes videos in the HLS [hls] and MPEG-DASH [mpeg-dash] streaming format. The client initially fetches the video from the server, feeds the stream into a video player and stores the stream data in the browser. The client-side library will register itself as a provider for the content downloaded in a centralized tracker maintained by the server. When a new client requests the same resource, the server decides whether to serve the new client directly or to delegate that function the the providers registered in the tracker. If the server decides to delegate the content delivery to a peer providers, it "introduces" the peers to each other so that the peer streaming can start. We set up the server hosting three videos and create a simple webpage with the client side library installed, from which we can stream the videos hosted by the server and/or peer providers. Our main goal with this demo is to understand whether peer providers can infer behaviour of other users, more specifically interest for a particular video, and obtain personal and identifiable information from the peer requester.

Check the demo video here [everyone-is-naked-repo]. The main takeaway from the demo is that by being a peer provider of a particular resource, we were able to learn the IP address of the clients requesting the resource using widespread networking tools such as Wireshark [wireshark].

[1] Current CDNs do much more than only caching and distribution of online content. Speeding up TLS connections, protecting against DDoS are examples of some other services provided by current CDN solutions. In this article we focus on the caching and distribution content only.

Attack the DHT: Distributed hash tables (DHTs) are overlay P2P networks that rely on a set of peers to implement a key-value storage without a central point of truth and failure. The DHT implements a simple API - *get(content_ID)* and *store(content)* -, allowing network nodes to collaboratively store and request content in the network. One crucial idea behind a DHT is the overlap of peer and content keyspaces, in which a peer with *ID_X* is responsible to store content with *ID_X.* Since network peers have a partial view of the network, this mechanism enables peers to locally decide which of the known peers is more likely to be responsible for storing a given ID. By recursively routing a store/lookup request to the closest

known peer, a request initiator will eventually learn where a given content is stored in the network, if it exists. In sum, DHTs rely on consistent hashing and key-based routing for providing a decentralized key-value store, while network peers implement a collaborative routing protocol to learn which peers are responsible for storing a given content.
The collaborative nature of DHTs results in resilient, scalable and decentralized networks where nodes are not required to maintain a complete view of the network while, simultaneously, not relying on central authorities. These properties make DHTs an important building block for the decentralized web and P2P systems. From a privacy perspective, DHTs are a good example of systems with intrinsic metadata leaks due to the constraint for network peers to maintain only a partial view of the network and to rely on open collaboration to obtain more information about the network. These protocols tend to leak user behaviour information, specially when when privacy preserving mechanisms are not baked into the protocols.

The Interplanetary File System (IPFS) is "is a protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system. [ipfs]". Currently, IPFS relies on a Kademlia DHT protocol [kad] for content routing and discovery. In the IPFS jargon, to locally replicate data is called "pinning" and peers data only upon explicit "pinning" it. If a peer is "responsible" for a set of content IDs (based on the mechanisms described before), it will contain pointers to the peers who are pinning the data, instead of holding the data itself. Thus, we assume that the peers pinning a given content are explicitly interested in the content being cached. \

In the following short demos, we show some examples on how IPFS leaks metadata [n]. To do so, let's set up the stage and give a more practical and tangible perspective to the problem at hand. Let's consider Bob has his own webpage stored in the IPFS network. The webpage has been added to the network and has an unique content ID, which consists of the hash of the webpage. Bob is pinning his webpage in his laptop and mobile phone. In addition, he requested a pinning service (i.e. a server) cache the content, so that the webpage remains online when his devices are not available.

In this context, an attacker can *infer network device information based on pinned content.* The attacker learns about the Bob's device IP addresses by looking up network information (e.g. IP) of the peers which are pinning the webpage.

As Bob moves around and connects his caching devices to other networks, the attacker is able to *infer Bob's location information based on the network device location throughout time.* This can potentially give an attacker a good overview of where Bob is physically over time and infert where Bob's house is, when he's travelling, etc.

Another attack vector consists of an attacker pinning content and inspecting who is requesting it. The attacker can *infer who is interested in a particular content* by simply caching it. This case is similar to pCDN case demonstrated above.

By placing multiple nodes in the network and registering the *lookup* and *store* requests being routed through the network, an attacker can accumulate enough information to *infer who is sharing and consuming what data in the network.* This attack is similar to what companies are currently using for filing lawsuits against BitTorrent users [crawling-dht].

Check the demo videos here [everyone-is-naked-repo]. The main takeaway from the demo is that by being a peer provider of a particular resource, we were able to learn the IP address of the client requesting the resource using widespread networking tools such as Wireshark [wireshark].

[n] We use IPFS DHT as an example due to its stability, available tooling and APIs which makes it very easy and fast to, for example, query a provided of a given content *ID*. Ironically, we use IPFS as an example of metadata leaks in DHTs because of its success and how good it is. The privacy vulnerabilities, though, are in one way or another similar across most of the systems relying on DHTs, such and BitTorrent's MainlineDHT [mainline-dht], Dat [dat], etc.

3. Privacy Enhancing Technology for P2P

The underlying privacy threats in decentralized systems demonstrated above result from the need for collaboration between peers while each of the peers maintain only partial view of the network topology and its resources. From here stems the tension between efficiency and scalability and privacy in P2P systems: while maintaining a partial view of the network and delegating work to other peers helps with scalability and efficiency, it also requires users to disclose interests and personal private information. Current techniques and tools to enhance privacy in P2P networks work only to alleviate that tension by decoupling *user interests* from peer behaviour and providing plausible deniability to users (e.g. "I'm storing or relaying content but I had no idea about it" or "I'm performing this request by delegation. I'm not the request initiator and I don't even know his identity"). However, privacy preserving P2P protocols will by definition require more work and resources and be less scalable and performant than its leaking counterparts. This fact seems unavoidable based on the recent research and privacy preserving engineering.

We will now briefly outline current research on Privacy Enhancing Technology (PETs) that have potential to solve some of the problems described so far [n.]

Onion routing: Onion routing [onion-routing] protocols are based on creating and exchanging messages that are encapsulated in multiple layers of encryption. Each layer can be decrypted only by a specific node (a *relayer*). Upon decrypting a layer of the onion packet, the relayer obtains routing information about the peer that is able to decrypt the next layer of encryption and forwards the remaining packet to it. Once all the layers have been decrypted, the last relay has access to the original message and (hopefully) no information about the origin of the request. Onion routing aims at creating a secure communication channel where each relay only has information about the previous and next relays, but will not be able to link the packet back the origin.

Although onion routing is vulnerable against timing attacks [timing-attacks], local passive adversaries in the threat model considered in this article are unlikely to learn who is the message originator based on network packet flows. However, this assumes a certain level of entropy in the network and circuits with non-colluding relays. Selecting relays anonymously is paramount for the security of onion routing protocols and hard to design in a completely decentralized manner. P2P networks using onion routing (e.g. The Onion Routing [tor]) often rely on centralized directories that keep a list and metrics of relays in the network. However, as the network size and number of relays increase, maintaining a centralized list of up to date relays and its metrics may become a bottleneck for growth and scalability [pir-tor]. In addition, the circuit constructor must trust the authorities that maintain the directory. Private Information Retrieval (PIR) systems have been considered [pir-tor] as a way to allow circuit constructors to query a centralized relay directory without leaking any information to the entities maintaining the directory. Other completely decentralized solutions for selecting a set of healthy relays anonymous with partial view of the network is based on random stochastic walks on the network and random walks on restricted topologies [shadowwalker]. Although some of these approaches are promising - while others were shown insecure -, these solutions assume complex mechanisms and protocols to ensure that relays are trustworthy and behaving, which increase complexity and infrastructure costs. Needless to say that onion routing involves cryptographic processing and redundant packet forwarding which consumes resources from the network. Onion routing is a good example of the rule of thumb that *privacy is not free*. Multiple non-colluding but independent parties must collaborate, invest resources and respect a set of protocols to provide privacy to a third party. It is clear that for such networks to work at scale and in a reasonable set of networks, there has to be an incentive layer for relayers to be motivated to provide resources to the network while behave as expected.

In summary, onion routing is a promising PET against local adversaries on latency sensitive P2P networks, however this approach hasn't been used by many current networks. There are multiple interesting challenges to solve such as relay incentives [onion-routing-incentives], anonymous and secure circuit building under a partial network view [shadowwalker], circuit redundancy and recovery [tap] and privacy metrics and visibility in P2P networks using onion routing..

More PETs: There are other PETs that can be used to improve metadata resistance in P2P networks. I believe that applied cryptography and cryptographic protocols such as Multi-Party Computation and Zero Knowledge primitives could also be used as tools for building privacy preserving P2P networks, specially as mechanisms that could be used to complement other protocols.

Notice that these technologies are nothing more than mechanisms that "shift tradeoffs": while they may provide more privacy, they will inevitably hinder scalability and/or efficiency. Moreover, privacy is hard to measure and very context dependent. The constant tension between those properties underline the tradeoffs in P2P network design when considering privacy. PETs also increase complexity for system designers and developers and makes it harder to reason about the overall network functioning.

^{n} We are limited in terms of space and time for this article. There will be a followup article and talk [privacy-engineering-p2p-talk] which focuses on the privacy preserving engineering and research of P2P networks, where we dive deep into the current research and implementations of PETs for P2P networks, its tradeoffs and open problems.

4. The why and wrap up

This article shows some examples of how and why current P2P and decentralized services are vulnerable to privacy attacks. Given the current state of affairs, why should we bother with P2P technology at all? Why don't we stick with centralized and client-server architectures instead? Ironically, the answer is privacy (oh, and scalability, protection against surveillance and online censorship). Despite the current problems, P2P systems seem to be the only viable solution for online privacy where large companies and central authorities are not incentivize to harvest and use our private information for power grabbing and profit. Decentralized systems are our best chance to build services where our privacy online is respected provided - this being the main point of this article - that P2P systems are designed and implemented with privacy in mind.

Phillip Rogaway starts his seminal work on The Moral Character of Cryptographic Work [crypto-moral] stating that: "Cryptography rearranges power: it configures who can do what, from what. This makes cryptography an inherently political tool, and it confers on the field an intrinsically moral dimension." (by all means go ahead and read the paper if you haven't yet!). In the article, Phillip Rogaway eloquently justifies technology as a political tool, proceeding by showing why the research community should keep this in mind when deciding on what to research and how. Similarly, P2P and *dweb* researchers, system designers and developers must seriously take privacy into consideration if we want to deliver software that impacts positively the Internet and our society. We need to be upfront about the current privacy challenges with our users; We need to understand that context and metadata is relevant when it comes to privacy. We need to defines proper threat models early. We also need to make privacy as the default. But most importantly, we need to do the research and development on metadata resistant protocols, incentive design, privacy usability and to figure out what it takes for us to deliver on a private and secure decentralized web.

References

[crypto-moral] Rogaway, Phillip. "The Moral Character of Cryptographic Work." IACR Cryptology ePrint Archive 2015 (2015): 1162.

[crawling-dht] Scott Wolchok and J. Alex Halderman. 2010. Crawling BitTorrent DHTs for fun and profit. In Proceedings of the 4th USENIX conference on Offensive technologies (WOOT'10). USENIX Association, Berkeley, CA, USA, 1-8.

[onion-routing] Goldschlag, David, Michael Reed, and Paul Syverson. *Onion routing for anonymous and private internet connections*. NAVAL RESEARCH LAB WASHINGTON DC CENTER FOR HIGH ASSURANCE COMPUTING SYSTEMS (CHACS), 1999.

[onion-routing-incentives] Figueiredo, D. R., Shapiro, J. K., & Towsley, D. (2004). Payment-based incentives for anonymous peer-to-peer systems. Computer Science Department, University of Massachusetts.

[shadowwalker] Prateek Mittal and Nikita Borisov. 2009. ShadowWalker: peer-to-peer anonymous communication using redundant structured topologies. In Proceedings of the 16th ACM conference on Computer and communications security (CCS '09). ACM, New York, NY, USA, 161-172. DOI: https://doi.org/10.1145/1653662.1653683

[tap] Zhu, Yingwu and Yiming Hu. “TAP: a novel tunneling approach for anonymity in structured P2P systems.” *International Conference on Parallel Processing, 2004. ICPP 2004.* (2004): 21-28 vol.1.

[pir-tor] Mittal, Prateek, et al. "PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval." USENIX Security Symposium. 2011

[timing-attacks] Feigenbaum, J., Johnson, A., & Syverson, P. (2010, July). Preventing active timing attacks in low-latency anonymous communication. In International Symposium on Privacy Enhancing Technologies Symposium (pp. 166-183). Springer, Berlin, Heidelberg.

[peer5] Peer5 website https://www.peer5.com/cdn

[viblast] Viblast website https://viblast.com/

[mpeg-dash] Dynamic Adaptive Streaming over HTTP https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP

[hls] HTTP Live Streaming https://en.wikipedia.org/wiki/HTTP_Live_Streaming

[wireshark] Wireshark https://en.wikipedia.org/wiki/Wireshark

[ipfs] IPFS website https://ipfs.io/

[kad] Kademlia https://en.wikipedia.org/wiki/Kademlia

[everyone-is-naked-repo] Everyone is naked talk repo https://github.com/gpestana/notes/tree/master/talks/everyone_is_naked

[mainline-dht] Mainline DHT https://en.wikipedia.org/wiki/Mainline_DHT

[dat] dat project website https://datproject.org/