P3LS: Plausible Deniability for Practical Privacy-Preserving Live Streaming

1st Jérémie Decouchant
SnT, University of Luxembourg
Luxembourg, Luxembourg
jeremie.decouchant@uni.lu

2nd Antoine Boutet
Univ Lyon, INSA Lyon, Inria, CITI
F-69621 Villeurbanne, France
antoine.boutet@insa-lyon.fr

3rd Jiangshan Yu
Monash University
Melbourne, Australia
jiangshan.yu@monash.edu

4th Paulo Esteves-Verissimo
SnT, University of Luxembourg
Luxembourg, Luxembourg
paulo.verissimo@uni.lu

***Abstract*—Video consumption is one of the most popular Internet activities worldwide. The emergence of sharing videos directly recorded with smartphones raises important privacy concerns. In this paper we propose P3LS, the first practical privacy-preserving peer-to-peer live streaming system. To protect the privacy of its users, P3LS relies on k-anonymity when users subscribe to streams, and on plausible deniability for the dissemination of video streams. Specifically, plausible deniability during the dissemination phase ensures that an adversary is never able to distinguish a user's stream of interest from the fake streams from a statistical analysis (i.e., using an analysis of variance). We exhaustively evaluate P3LS and show that adversaries are not able to identify the real stream of a user with very high confidence. Moreover, P3LS consumes 30% less bandwidth than the standard k-anonymity approach where nodes fully contribute to the dissemination of k streams.**

I. Introduction

Platforms that allow users to share video streams rapidly became a prominent component of the Web ecosystem (e.g., 300 hours of video are uploaded to YouTube every minute and almost 5 thousand million videos are watched each day [1]). In addition, the widespread adoption of continuously connected smartphones caused more and more users to consume and share videos directly recorded with their smartphones during their daily lives (e.g., Snapchat). The emergence of this new pervasive usage considerably exposes the personal life of users and raises serious privacy concerns.

To sustain the requirements of high video consumption demand, many technologies and systems have been developed to serve and optimize video streaming. For instance, adaptive streaming over HTTP allows a client video player to dynamically adjust the video bitrate as a function of the network condition and CPU usage [2], and Content Delivery Networks (CDNs) cache videos close to their consumption sites [3]. However, the high bitrates of the broadcast streams and the quality of service requirements of video applications make centralized solutions costly. To reduce this cost, peer-to-peer (P2P) or partially decentralized approaches have emerged as key architectural solutions. Peer-to-peer (P2P) is an application architecture paradigm that consists in distributing a service among a set of nodes. P2P architectures can be made highly resilient to failures and bring a cheaper scalability compared to centralized architectures [4]. P2P-based applications have been successfully developed to support various services (e.g., file sharing [5], video streaming [6], pub-sub [7]).

During a P2P video streaming session, a special node - the *source* - streams a video and at the same time generates chunks of this video, called *updates*. These chunks are then sent, as they are generated, to nodes that subscribed to this stream through either structured, tree or mesh-based, or unstructured gossip-based dissemination. These approaches build overlay networks gathering nodes interested by the same video stream and willing to participate in the dissemination of video chunks by exchanging them. After it has been emitted by the source, a chunk of a video stream is exchanged between peers during a predetermined duration. After this duration has elapsed, it can then be played by the peers' video applications.

Privacy has attracted a lot of attention lately and measures have appeared to protect users against invasive tracking or massive data collection of personal information, not only of technological kind but also legislative (c.f., Europe's GDPR [8]). However, so far, no system has been proposed to ensure user privacy in video streaming systems despite the widespread adoption and use of these applications. While relying on P2P systems should decrease privacy leaks by avoiding the centralization of the information, in practice these systems exhibit an increased threat surface in the form of privacy threats from other nodes if information related to what users are watching becomes public. Several works have highlighted [9] or measured [10], [11] to which extent P2P protocols leak information about their users.

The subscription phase in a video streaming system can be either explicit or implicit. With explicit subscription, each stream is publicly associated with an overlay of nodes. Partial or full membership lists of these overlays are publicly advertised, which allows new nodes to directly join an overlay.

This work is partially supported by the University of Luxembourg - SnT and by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128.

However, evaluating this membership relation also allows adversaries to infer the stream a node is watching, since through this list a node is directly linked with a stream it consumes. With implicit subscription, each node maintains a profile of interests (e.g., its last consumed items) which is used to dynamically discover and build an overlay of nodes sharing similar interests. However, as these profiles are exchanged between nodes, the interests of users can easily be discovered. Moreover, in both cases, videos watched by a user can be inferred by other nodes during their interactions (i.e., from the streams they received and disseminated).

To protect the interests of a user from being inferred, several privacy-preserving schemes have been proposed at different levels. Firstly, several works have leveraged anonymity to improve user privacy [12], [13]. However, in the context of video stream system, relying on anonymizing overlays introduces a substantial performance overhead. For example, communications between two nodes using Tor [14] go through five relay nodes, which introduces a significant delay. In addition, a video streaming system running on Tor would require nodes to dedicate a six fold increased communication bandwidth to video streaming compared to what non-privacy preserving solutions demand. In addition, the Tor project reports a total network capacity of 300Gbps and a current utilization of 120Gbps[1], which suggests that a large scale use of Tor for video streaming will likely lead to a collapse of the network. Other anonymity approaches, like Dissent [15] and RAC [16], would only worsen performance. Secondly, a node can hide its interests by subscribing in several membership lists and relaying streams in several network overlays. By doing so, the node benefits from k-anonymity in the subscription phase by making its real interest indistinguishable from fake ones. However, this solution significantly increases bandwidth consumption since a node receives k streams when it is interested in only one.

Dissemination protocols introducing randomness have been proposed in order to ensure plausible deniability [17]. Randomness prevents an adversary from determining without doubts the interests of a user, when observing the exchanged chunks during dissemination. These approaches rely on a difficult to control trade-off between privacy and utility. In the context of video streaming, this utility is captured by the relevance of the video stream received and the quality of service. Moreover, all the proposed solutions provide no information to their users, explaining the impact of the protection mechanism on the overhead of the system. This lack of transparency makes it difficult for users to control and parameterize the protection mechanisms according to their expectations. We believe that bringing more transparency to users about their privacy and utility trade-offs will improve the adoption of privacy-preserving systems.

In this paper, we therefore present P3LS, the first practical privacy-preserving live streaming system that enforces a user-defined lower bound on their privacy. P3LS preserves the privacy of its users by protecting both the subscription phase and the dissemination phase. We employ three core techniques to achieve this goal. Firstly, P3LS protects the stream subscription of users through k-anonymity by hiding their real subscription among fake ones. Secondly, to avoid revealing the stream watched by users through their dissemination patterns, P3LS relies on a novel dissemination protocol ensuring plausible deniability. This dissemination protocol propagates multiple streams (to multiple nodes) through different dissemination preferences that respect privacy properties controlled by an analysis of variance. Lastly, P3LS leverages random sampling to request multiple streams through a push-based gossip protocol from a dynamically and randomly selected set of nodes.

P3LS enables nodes to control their dissemination preferences, while enforcing their ultimate privacy guarantee, which allows a node to control the amount of bandwidth spent to obfuscate its real subscription. P3LS is the first protocol to enforce plausible deniability against any set of nodes, potentially controlled by a risk-averse honest-but-curious adversary, in the sense that it is not able to statistically infer which stream a user is more interested in.

We implemented and exhaustively experimented P3LS using simulations with up to 50,000 nodes. We show that P3LS is able to generate a large spectrum of privacy and performance overhead combinations providing a large choice of utility and privacy trade-offs to users. We were then able to show that the privacy property we enforce protects peers against adversaries controlling any number of nodes in the system: an adversary cannot avoid a high level of false positives and false negatives, when trying to identify the stream of interest of a user. Finally, P3LS is superior to the classical strategy that consists in subscribing to k streams instead of one, by exhibiting up to 30% better performance, and providing satisfying statistical privacy guarantees.

The remaining of this paper is organized as follows. Section II presents the system and threat models we consider. Section III provides on overview of P3LS. Section IV describes in details P3LS's design. Section V details our experimental setup. Section VI reports the performance evaluation of this solution. Section VII reviews related work while Section VIII concludes this paper.

II. System and Threat Model

In this section, we describe our system model (Section II-A) and the adversary we consider (Section II-B).

A. System model

We consider a fully decentralized system where a set of peers are distributed over the Internet and are willing to cooperate to receive and exchange a digital stream (e.g., videos, music). In particular, each stream is generated by a special node (i.e., the *source*) that generates and periodically sends chunks of this content (also called *updates*) to a subset of interested nodes. Updates are exchanged during *RTE* seconds after their emission by the source. After receiving an

[1]https://metrics.torproject.org/bandwidth.html as of Dec 6, 2018.

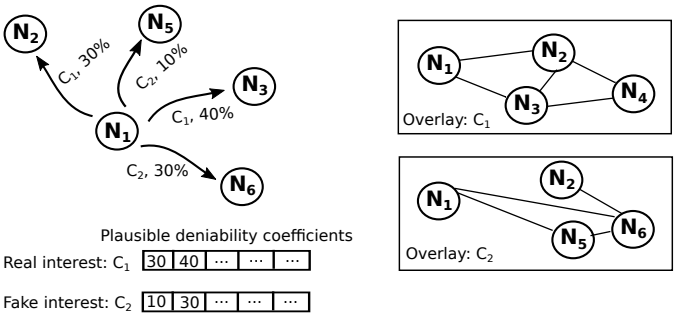

Fig. 1. Overview of the system for a node N_1 using k=2.

update, nodes can forward it during t seconds to other nodes, according to an epidemics based dissemination [18].

We assume that a video is uniquely identified, for example using its source's identifier (i.e., its IP address) associated to a stream ID, a number of updates released per second, and an expected duration. We also assume that videos are not associated to semantics. We assume that users have a unique and publicly known identity (e.g., their IP address) and have access to secure encryption and signature schemes.

For the sake of simplicity, we assume that a tracker service is available, where all sources can register their streams, and which collects a bootstrapping set of nodes interested in those streams. A peer subscribing to a stream is then able to contact this tracker, and obtain the bootstrapping set of nodes. We assume videos with the same duration or used padding to have the same duration. We also assume that each peer is interested in only one stream. We do not aim at protecting the privacy of the sources of the streams, but since there is only one source per stream, they could rely on an anonymous communication system to inject their updates in the associated overlay. Lastly, as the notion of privacy sensitive information can be differently defined, we assume that users are willing to protect their own privacy with different preferences.

B. Adversary model

We consider an *honest-but-curious* [19] adversary which passively collects information about peers. Specifically, the adversary may control a set of peers that correctly follow the protocols, and obtain from them the details of their interactions with other peers. We assume that the adversary is not able to monitor the entire communication network, and in particular the interactions between two peers it does not control. This adversary aims at determining the real interest of users. However, this adversary is risk-averse, means that it avoids to link a user with a stream without being truly convinced about it. This adversary model is inspired by judiciary systems, where convicting an innocent is deemed unacceptable.

III. Protocol Overview

In this section, we provide an overview of P3LS. Throughout this section, we refer to Figure 1 which provides an illustration of this overview.

Once users are aware of the list of streams available in the system (e.g., from a centralized tracker service as described in Section IV), they can subscribe to their actual interest. To avoid revealing this interest, P3LS also subscribes to fake interests by picking at random $k-1$ streams in the list of available streams. In addition, to avoid to leak information from the popularity of the streams, this selection enforces that these k streams have the same distribution of popularity than the whole streams available in the system. Figure 1 illustrates P3LS with $k=2$ for a node N_1 that subscribed to its stream of interest c_1 and to another stream c_2. By relying on gossip protocols, each node builds and maintains an overlay network associated to each subscription. On the right part of Figure 1, node N_1 is shown in the overlays associated to c_1 and c_2. These overlays list nodes that N_1 is aware of and which are able to disseminate the associated streams.

	Null hypothesis H_0 (k-anonymity) is	
Decision	True (k-anonymity)	False (Plausible deniability)
Fail to reject H_0	Confidence $(1-\alpha)$	Type II error (β)
Reject H_0	**Type I error** (α)	**Power** $(1-\beta)$

TABLE I
DECISION TABLE OF A STATISTICAL HYPOTHESIS TEST.

Once the subscription phase is done, the dissemination phase can start. This dissemination also relies on a push-based gossip protocol to request a given proportion of the live chunks of a stream. More precisely, a node willing to receive a specific stream selects at random peers that are part of the associated overlay to request chunks. In Figure 1, node N_1 contacts nodes N_2 and N_3 for content c_1, and nodes N_5 and N_6 for content c_2. To avoid revealing the real interest of the users while ensuring users' privacy, the dissemination has to be carefully performed. Specifically, an adversary should not be able to infer a real interest by analyzing the different requests and, for instance, discovering that a node has significantly more updates of a given content than another. To achieve that, P3LS calibrates requests to ensure the plausible deniability property [20]. This property ensures that an adversary can not be certain to identify the real interest of the users. Practically, P3LS ensures that requests for both the real and fake interests are indistinguishable from a statistical properties point of view.

More precisely, we consider the one-way analysis of variance (ANOVA) as a statistical test to decide whether the observed difference between two sets of requests (i.e., streams requested for both the real and the fake interests) is statistically significant above a predefined confidence level $(1-\alpha)$. The goal of the ANOVA test is to decide whether to reject or not the null hypothesis H_0, which is the hypothesis that the streams are equally requested by a node. In P3LS, we extend this ANOVA test to identify whether the set of requests associated to the real interest is statistically different from all the $k-1$ requests generated by the subscriptions to fake interests. By construction, in P3LS, the ANOVA test runs on vectors of coefficients generated using normal laws.

Table I details the different decisions an adversary could take when studying a peer using either k-anonymity or plausible deniability, and for each scenario names the probability to

associate the peer to either k-anonymity or plausible deniability. Table I uses the standard notations associated to hypothesis testing, which we adapted to our context. For example, α is the probability for the adversary to associate a peer using k-anonymity, i.e., participating equally in all contents, as a node using plausible deniability, which would lead the adversary to associate a user to a stream.

As further described in Section IV, peers generate in advance the percent of streams they will request during any interaction for a given subscription (called vectors of plausible deniable coefficients in the following). In Figure 1 (left part), node N_1 precomputed two vectors, one for content c_1 whose first values are 30 and 40, and one for content c_2 whose first values are 10 and 30. Node N_1 then uses these coefficients during its interactions with the other nodes. However, we leverage ANOVA tests to generate vectors of requested chunks that cannot be statistically distinguished from the others, given an assumed maximum risk α to wrongly reject the null hypothesis. Consequently, according to the privacy policy defined by the user, P3LS is able to control the bandwidth improvement compared to k-anonymity.

IV. System Details

We now describe in details P3LS's design. First, in Section IV-A, we review P3LS's parameters and explain how users can determine their values. We then describe the subscription phase in Section IV-B before detailing in Section IV-C how peers generate the vectors of coefficients used during their interactions with other peers. Lastly, Section IV-D details the dissemination phase.

A. Individual Privacy Settings

P3LS is user-driven. Each user can control the expected level of privacy. To achieve that, users can control two different parameters: i) α: the upper bound of the probability that the adversary identifies a node as a node having a higher interest in one of the k streams (i.e., the maximum type I error in Table I), and ii) k: the number of subscriptions (i.e., including the real and $k-1$ fake ones).

The value of both k and α have an impact on both privacy and performance. A higher k value better protects a user's privacy by making its real interest indistinguishable among more fake ones. However, nodes have to be ready to commit more bandwidth consumption. A larger α value better protects the privacy of nodes by making the vectors of coefficients more statistically similar to each other, therefore making nodes receive more equally in all streams. To balance the utility and privacy trade-off and helping users to define the value of these parameters, users can leverage the perceived quality of service (e.g., a too degraded quality results of an unbalanced trade-off on the utility side).

B. Subscription Phase

Nodes never declare their real interest (i.e., the stream associated to their real interest). Instead, nodes enforce their privacy through k-anonymity on explicit subscriptions to streams. Specifically, they declare being interested on k streams simultaneously: their real interest and $k-1$ fake interests. A node maintains a profile gathering the list of stream IDs associated to its current k subscriptions which therefore will be available to be disseminated to others.

In video streaming, all videos do not receive the same popularity. Indeed, the distribution of human consumption on social media and online streaming platforms is characterized by the well-known long tail [21]. This heterogeneity in term of popularity can give assumption of the real interest of a user if this interest is a popular video and all fake ones are non popular for instance. To avoid that, P3LS adopts a scheme similar to *t*-closeness [22]. Specifically, the selection of the $k-1$ streams ensures that the distribution of the popularity of all k streams follow the same distribution than all streams in the system. For sake of simplicity, we collect the global popularity of each video through the tracker. However, more advanced solution can be considered such as a secure aggregation through a multi-party computation protocol [23]. By periodically contacting the tracker, each node is aware of the actual popularity of each video and can use it to carefully select its fake subscriptions.

P3LS manages overlay networks associated to the subscriptions of the user. More precisely, P3LS relies on gossip protocols to build and maintain multiple overlay networks. The bottom gossip layer implements a random peer sampling (RPS) service [24], [25]. By gossiping random views between nodes, this RPS service ensures the construction of a continuously changing and randomly graph of nodes, which is made of a unique connected component, and the quickly removing of offline nodes. The upper gossip layer, in turn, manages the overlay networks associated to the current k subscriptions of the user (i.e., one overlay per subscription). To manage these overlay networks, nodes exchange both their profile and the profile of nodes that is aware of with random nodes (i.e., part of its RPS) and nodes part of the overlays associated to their subscriptions. By doing so, nodes learn about new other nodes in the system subscribing to the same streams and able to serve chunks in the dissemination phase. These new nodes are used to feed and dynamically maintain the overlay networks. Lastly, after that the dissemination of a stream ended, nodes can remove the associated subscription (i.e., the associated overlay network).

C. Plausibly Deniable Coefficients

Each node taking part in the dissemination of a stream (either associated to a real or a fake interest) is in charge of both requesting chunks and serving requests that it receives. A request contains the ID of the associated stream and a coefficient indicating the expected percent of chunks. To ensure privacy, requests associated to the real interest should not be distinguishable from requests associated to fake interests. To achieve that, P3LS carefully defines a set of coefficients for each of its subscriptions ensuring plausible deniability. More precisely, P3LS ensures that the set of coefficients are not distinguishable from a statistical test based on an one-way

Algorithm 1 *Plausibly deniable vectors*

Input: k: number of streams (including the real interest)
α: maximum type-I error the adversary may take when rejecting the null hypothesis
N: number of coefficients to generate per stream
m_{real}: targeted proportion of the stream of interest to receive
m_{obf}: targeted proportion of the obfuscating streams to receive
σ: variance of the normal laws used to generate the vectors
Uses: ANOVA test returns p value; **instance** anova-pvalue
Normal distribution generator of N coefficients with mean m and standard deviation σ; **instance** Normal(m, σ, N)
Output: $C_{\text{real}} = C[0]$: a vector of N coefficients
$C_{\text{obf}} = C[1], \cdots, C[k-1]$: $k-1$ vectors of N coefficients
$\forall i \in \{1, \cdots, k-1\}$, C_{real} and $C_{\text{obf}}[i]$ are not statistically distinguishable through an ANOVA test with confidence level $(1-\alpha)$

```
1:  int[][] C = int[k][N]
2:  int count = 0
3:  while True do
4:      C_real = C[0] = Normal(m_real/N, σ, N)
5:      for 1 ≤ i ≤ k − 1 do
6:          C_obf[i] = C[i] = Normal(m_obf/N, σ, N)
7:      end for
8:      if anova-pvalue(C[0], C[1], ··· , C[k − 1]) < α then
9:          break
10:     end if
11:     count++
12:     if count mod 10000 == 0 AND m_obf < m_real then
13:         m_obf++
14:         count = 0
15:     end if
16: end while
17: M = max(C[i][j] | 0 ≤ i < k, 0 ≤ j < N)
18: for each C[i][j] do
19:     C[i][j] = max(0, C[i][j]) ∗ M/100
20: end for
21: return (C_real, C_obf)
```

analysis of variance (ANOVA). In other words, the probability that an adversary analyzing coefficients identifies that the means value of one distribution is larger than another is bounded and under control through a privacy parameter.

We use Algorithm 1 to generate vectors of coefficients providing plausible deniability [20]. This algorithm uses normal laws to generate the vector of coefficients, as an ANOVA test assumes. Algorithm 1 uses the user-specified parameters k and α, that we previously described in Section IV-A, and some additional parameters that we set for the user, before generating the vectors of coefficients: N is the number of coefficients to generate per content, m_{real} is the expected sum of the coefficients in the vector associated to the real interest of a peer, m_{obf} is the expected sum of the coefficients in the vectors associated to the obfuscating interests, and σ is the variance of the two normal laws. This algorithm draws random vectors generated using the two normal laws (i.e., in lines 5-8, Algorithm 1), until the ANOVA test is not able to reject the null hypothesis above the specified confidence level $1-\alpha$ (i.e., in lines 10-12).

To expect the means of the generated vectors to be different, and therefore expect some performance gains when using the coefficients in a practical protocol, one would choose $m_{\text{obf}} < m_{\text{real}}$. However, the choice of the values of m_{real} and m_{obf} has an impact on the completion time of the algorithm and on the empirical means of the generated vectors. When m_{real} and m_{obf} are too close to each other, the empirical difference between the means of the vectors is limited. When m_{real} and m_{obf} are too far from each other the algorithm takes longer to finish, and may take an unbounded amount of time. To maintain high expected performance improvement, and a short execution time, after every 10,000 unsuccessful generation of vectors Algorithm 1 decreases the difference between m_{obf} and m_{real}, by incrementing m_{obf} (i.e., in lines 14-18). We measured that generating the coefficients took less than 2 seconds for each of the configurations we studied. Finally, the values in the vectors are normalized between 0 and 100, so that peers can use them to indicate a proportion of a stream it is willing to receive or send (i..e, in lines 21-24).

To summarize, Algorithm 1 allows a peer to obtain a vector for its real interest with a larger mean than the means of the vectors associated to the fake streams. In practice, this will result in the peer being able to mainly receive the stream it is interested in, and save bandwidth by limiting its reception of fake streams. Therefore, plausible deniability offers more flexibility than k-anonymity, since it can be extended to k-anonymity (i.e., when $\alpha = 100$), or reduced to providing no-privacy at all (i.e., when $\alpha = 0$).

All vectors of coefficients a node chooses can be revealed to an adversary since even then the plausible deniability property would stand.

D. Dissemination Phase

Once a vector of coefficients is built for each subscription, nodes can exchange together to request and serve requests of content. P3LS adopts an exchange protocol working by round. Periodically (i.e., at each round), a node changes the list of nodes with which it exchanges during the dissemination. That includes the corresponding nodes associated to the initiated requests, and served nodes.

Peers follow Algorithm 2 to select the nodes they interact with and select coefficients to use during their interactions. This algorithm basically says that for each stream c_i it wants to receive, a peer randomly selects N nodes to contact based on the associated overlay network. Then, for each selected peer, the algorithm selects a random coefficient in the precomputed plausibly deniable vector associated to stream c_i. In addition, a peer never distributes two different coefficients to the same peer. Indeed, nodes keep in memory the coefficient they have exchanged with any other node about a given stream, so that they never reveal more than one of their precomputed coefficient to a given node. This makes it more difficult for an adversary trying to guess the α values that the user used to generate its coefficients.

Basically, Algorithm 2 makes nodes contact a fixed number N of partners per communication round. If a node selects a partner with which it had an interaction in the past it has to reuse the same coefficient (lines 8-9), otherwise it randomly selects one of its coefficient (lines 11-12). The number of partners contacted can be dynamically adapted to increase the

Algorithm 2 Partners selection

Input: N: the number of partners to select per stream
L_i: the membership list associated to stream C_i
S_c: the streams the peer subscribed to
$V[c_i]$: the vectors of precomputed interaction coefficients for each stream subscribed to
$past\_partners\_coeff$: a 2-dimensional map that accumulates the interaction coefficients chosen per partner and per stream
Output: $partners\_coeff$: a 2-dimensional map of interaction coefficients chosen per elected partner and per stream

```
1:  partners_coeff = {}{}
2:  for c_i ∈ S_c do
3:      while partners_coeff[c_i].size() < N do
4:          p = random(L_i)
5:          while p ∈ partners_coeff[c_i] do
6:              p = random(L_i)
7:          end while
8:          if p ∈ past_partners_coeff then
9:              partners_coeff[c_i][p] = past_partners_coeff[c_i][p]
10:         else
11:             partners_coeff[c_i][p] = random(V[c_i])
12:             past_partners_coeff[c_i][p] = partners_coeff[c_i][p]
13:         end if
14:     end while
15: end for
16: return partners_coeff
```

probability of receiving all chunks while minimizing duplicate receptions.

P3LS is highly tolerant to churn. Users can leave, or join, the system at any point in time. When selecting partners to receive updates from, if a peer realizes that one of them is not answering to messages, it can simply select another node to contact, select a coefficient and initiate an exchange with it.

When a node receives a request with a specific coefficient, it selects at random a coefficient available in its plausibly deniable vector and serves the request up to the minimum of each of the two coefficients (i.e., the local and the received one). Using the minimum of the coefficients to determine the percent of chunk to serve is necessary for both privacy and performance reasons. First, it is necessary for privacy reasons since if a node sends more updates than its coefficient indicates, then it may provide additional information to the other node. Second, it is necessary for performance reasons, since it allows a node to limit the number of updates it receives per interaction. Otherwise, a node would not have any control on the amount of duplicate update receptions.

When node B chooses a coefficient for its interaction with node A, we can distinguish two situations: i) if node B exchanged some updates from a stream c_1 with node A in a previous round, then node B has to reuse the same coefficient it used in the past; ii) otherwise, node B has a little more freedom, and can slightly optimize the choice of its coefficient, for example by choosing its coefficient for stream c_1 that is the closest to the one it received.

Upon reception, updates from a stream are inserted into a set of updates to retransmit whose keys are the times at which they were emitted by their source. Updates in such sets are kept ordered according to their reception times by the peer. Peers keep retransmitting the updates they have received in the last t seconds in variable proportion and deterministically. This epidemics model has been intensively studied, e.g., in [18].

In P3LS, peers determine the number of updates they send during an interaction depending on the minimum of the two peers' communicated coefficients, which is multiplied by the number of updates generated by the source per second. More precisely, a peer sends the oldest updates from its set of updates to retransmit. For a given stream, nodes are always able to send the specified number of updates, since updates are retransmitted during *RTE* seconds after their emission by the source. In addition, since updates are randomly transmitted in the system, the updates sent by a node during an interaction do not reveal its preference for a given stream.

First, sending the first received updates does not harm the correct propagation of updates, since they are randomly exchanged among all peers. Second, this mechanism ensures that two nodes requesting updates for stream c_i to a given node cannot learn more than the maximum of the interaction coefficients of the proportion of updates the requested node actually has. In addition, the requested node also controls the interaction coefficients (thanks to the min operation), so that these two interactions cannot leak too much information. This property also avoid nodes to fully receive the content of fake streams, therefore improving their performance over a k-anonymity approach, while receiving all of their stream of interest (Section VI).

Depending on other nodes, this simple interaction process may however create redundant reception of updates. In addition, a node may also fail to receive some updates, if they are not disseminated for long enough in the system, which in turn would force nodes to increase the number of simultaneous interactions they request per round to reach their reception objectives. In the performance evaluation section, we show that in practice nodes can receive all the updates, and that the amount of duplicate receptions incurs a reasonable amount of bandwidth overhead.

V. Experimental Setup

In this section we present the experimental setup we used to evaluate P3LS including the methodology (Section V-A), the metrics (Section V-B) and the system parameters (Section V-C).

A. Methodology

Our experiments are simulation-based. In our simulation, we consider a system gathering 50,000 nodes. Among all these nodes, 10% of them implements P3LS (called P3LS-SET) while the others subscribed to k streams simultaneously and participate equally in each (called K-ANONYMOUS-SET). Nodes randomly subscribed to a variable number of streams k they wish to receive and we consider a live streaming system with $5,000$ different streams available. The popularity of these streams is defined to follow a long tail [21]. Each stream is emitted at a rate of 600kbps and split in 100 updates of 6kb (i.e., all streams have the same size as described Section II).

During an experiment, users subscribe and fully consume multiple streams sequentially. Nodes play the updates in their video application after $RTE = 8$ seconds have elapsed since their release by their source. In addition nodes can forward newly received updates during $t = 2$ seconds after their reception. The considered value for a P3LS round is one second (i.e., nodes contact new partners to request updates every second). For integrity, updates are transmitted along with their source's signature, which is a 2048-bits RSA signature. Lastly, we do not bound the communication links bandwidth.

B. Metrics

We use a set of theoretical and system metrics to assess the performance of P3LS:

1) Theoretical analysis: We theoretically analyzed the expected bandwidth improvement from the plausibly deniable coefficients. The values obtained represent the theoretical improvement bounds one cannot expect to exceed. We also studied the erosion of privacy as an adversary discovers a variable proportion of the coefficients of a node.

2) Privacy: The privacy is measured as the average error-rate prediction of an adversary willing to identify if users favor one stream (i.e., if users are part of the P3LS-SET or part of the K-ANONYMOUS-SET). More specifically, it is the average probability that an adversary wrongly decides that users are part of the K-ANONYMOUS-SET (i.e., α in Table I) or part of the P3LS-SET (i.e., $1-\beta$ in Table I). This privacy metric is the exact opposite of the adversary's success rate metric [26], and has the advantage of being intuitive since a larger adversary missing rate means more privacy for users.

3) Bandwidth consumption: We monitored the bandwidth of nodes both in reception and in emission.

C. Parameters

To generate the vectors of plausible deniable coefficients of P3LS, nodes part of the P3LS-SET use the following initial values: $m_{\text{real}} = 100$, $m_{\text{obf}} = 0$, and $\sigma = 5$. However, as previously written, the value m_{obf} and m_{obf} can be modified to ensure the termination of Algorithm 1. During the experiments, we do not consider dynamic values for parameter N, and consider future work the dynamic modification of P3LS's parameters by a peer.

VI. Performance Evaluation

In this section, we present the experimental evaluation of P3LS over three dimensions: a theoretical analysis of the capacity of P3LS to decrease the bandwidth consumption compared to a pure k-anonymity scheme (Section VI-A), the uncertainty of a adversary about the real interest of the user even if the adversary control a population of nodes (Section VI-B), and a practical evaluation of the bandwidth consumption overhead (Section VI-C). We also analysis the impact of P3LS on the popularity of videos.

A. Plausibly Deniable Coefficients

We first theoretically studied the improvement in term of bandwidth consumption provided by the plausible deniability property implemented in P3LS (i.e., the results of Algorithm 1). To do so, we assume that each coefficient in the plausible deniability vectors would lead to the reception of the corresponding proportion of a stream.

Figure 2 reports the theoretical bandwidth consumption of P3LS compared to receiving only the stream of interest for various parameters. To plot this graph, we study the performance improvement over k-anonymity for $k \in [2, 3, 4, 5, 6, 7]$, $N \in [3, 4, 5, 6, 7, 8, 9, 10, 12]$, and $\alpha \in [0.1, 0.4, 0.8]$. First, one can observe that plausible deniability provides a wide range of performance and adversary missing rate. This large spectrum of values makes users able to define a flexible utility and privacy trade-off. As expected, using a larger α parameter increases the privacy of users. Using larger k and N values increases the theoretical cost of the coefficients generated, since it is less probable to generate more or larger vectors that are plausibly deniable, without modifying the m_{obf} parameter in Algorithm 1.

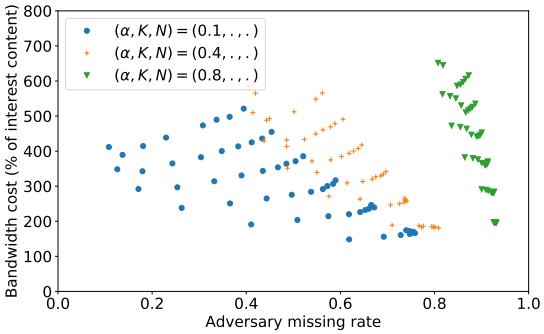

Fig. 2. Adversary missing rate versus theoretical performance overhead over the no-privacy configuration.

Second, we studied the bandwidth improvement of plausible deniability coefficients over k-anonymity coefficients obtained when the means of the two normal laws used in Algorithm 1 are equal, when computed over similar k values. Figure 3 details the theoretical bandwidth decrease over k-anonymity. Using plausible deniable coefficients has therefore the potential to improve performance up to slightly more than 40% depending on the considered parameters.

B. Resiliency Against an Honest-but-Curious Adversary

As written in the system and threat models, P3LS enforces that an adversary controlling a variable number of peers in the system can never identify the interest of a peer above a user-specified confidence level. However, one could wonder whether observing a subset of the coefficients could give away enough information for the adversary to correctly identify the stream of interest of a node. We therefore studied the erosion of privacy when the number of nodes controlled by an honest-but-curious adversary increases. More precisely, we studied the errors an adversary would make if it were to

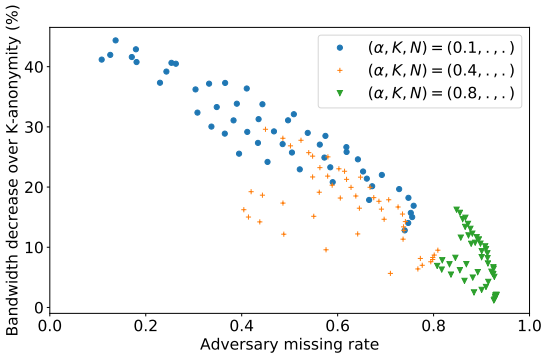

Fig. 3. Privacy vs. theoretical performance improvement over k-anonymity.

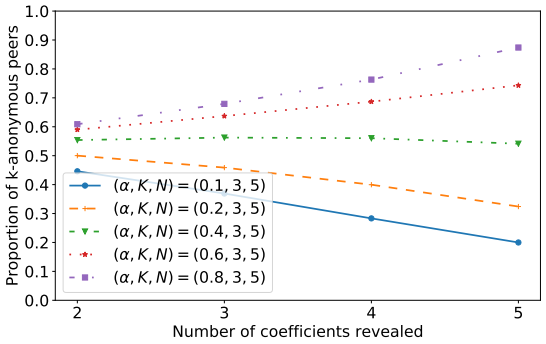

Fig. 4. Proportion of peers using k-anonymity believed to be using plausible deniability (Type I error rate) depending on the number of coefficients revealed.

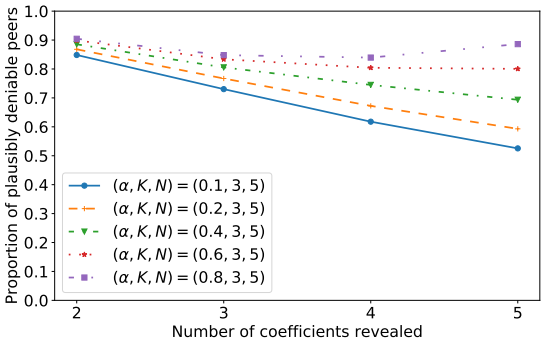

Fig. 5. Proportions of peers using plausible deniability believed to be using k-anonymity (Type II error rate) depending on the number of coefficients revealed.

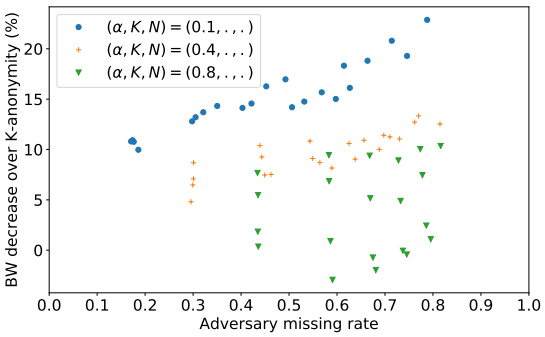

Fig. 6. Adversary missing rate versus performance improvement (bandwidth decrease) over a subscription to k streams.

take a decision based on a given number of the coefficients of all vectors. In this section, we use $(K, N) = (3, 5)$ and $\alpha \in [0.1, 0.2, 0.4, 0.6, 0.8]$.

Figure 4 presents the proportion of peers for which the adversary would reject the null hypothesis depending on the number of coefficients it could observe, even though they would be using k-anonymity. As the adversary observes more coefficients, the proportion of wrong decisions it would take does not follow a clear pattern. It may decrease with small values of α but may also increase with larger values.

Figure 5 presents the proportion of peers that used plausibly deniable coefficients that are identified as such by the adversary. Again this proportion is never equal to 100%, and may decrease or increase depending on the number of coefficients revealed. An adversary that would observe a subset of the coefficients a node has precomputed would therefore not be able to take any insightful decision about whether or not nodes are using plausibly deniable coefficients.

C. Bandwidth Consumption Overhead

Figure 6 depicts the performance improvement observed in the complete P3LS protocol compared to the same version using k-anonymity. It is the practical equivalent of Figure 3, and the difference between those two figures show the impact of deploying privacy-preserving mechanisms in real systems on performance. However, the performance improvement of nodes compared to k-anonymity is still significant. Nodes can expect to save up to 25% of bandwidth compared to k-anonymity. For this figure, we limited the plotted values to those corresponding to α in $[0.1, 0.4, 0.8]$.

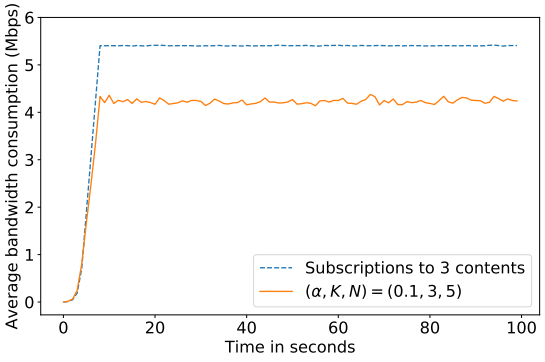

Fig. 7. Evolution of the bandwidth consumption depending on time during an experiment.

Figure 7 shows the bandwidth consumption of peers in the system for $(\alpha, k, N) = (0.1, 3, 5)$. The bandwidth consumption of all nodes increase as a function of the number of nodes they contact. Since most of the nodes use k-anonymity, their download and upload bandwidths are very similar and are represented by a single line. In this example, peers using plausible deniability have a smaller bandwidth consumption, needing around 4.3Mbps, while peers using k-anonymity re-

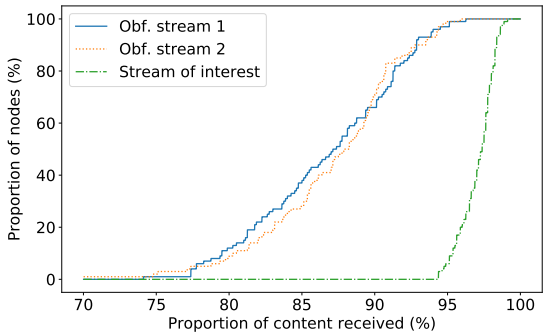

Fig. 8. Cumulative distribution of the proportion of each stream received (1 real interest and 2 obfuscating ones) among nodes $(\alpha, K, N) = (0.1, 3, 5)$.

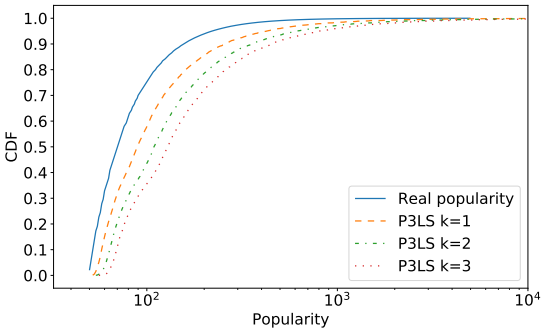

Fig. 9. P3LS artificially increases the popularity of streams.

quire 5.4Mbps.

Lastly, using the same settings, Figure 8 shows the cumulative distribution function of the amount of each multimedia stream that peers using plausible deniability in the system are able to reconstitute. This figure confirms that these nodes can play more than 95% of their stream of interest, while they receive a similar lower amount of the two obfuscating streams (between 75 and 94%).

D. Impact on Popularity

P3LS has an impact on the popularity of videos. Figure 9 depicts the distribution of the real popularity of items in the system as well as the resulting popularity for P3LS with varying k. As mentioned in Section V-A, we follow a power law to define the popularity of each item. Consequently, the distribution of the item popularity follows a long tail, i.e., only a small fraction of the item depicts a high popularity. As shown on the figure, by adding fake subscriptions, our solution artificially increases the popularity of items. This side effect has the advantage to improve the distribution of unpopular contents as more nodes can help in their dissemination.

VII. Related Work

In this section, we present background information on k-anonymity, and we review the related work on privacy, accountability, trust, and plausible deniability in P2P systems.

a) k-anonymity: Relying only on k-anonymity is not enough in some cases. For instance, requesting a database protected by k-anonymity can reveal the sensitive information if all the k selected entries share the same value. This concern has been addressed by the introduction of *l*-diversity [27] which extends k-anonymity by additionally enforcing that at least *l* distinct values are present for each sensitive field for each anonymity group. Then, *t*-closeness [22] is a further extension of *l*-diversity. Instead of just guaranteeing a good representation of sensitive values, this approach enforces that the distribution of every sensitive attribute inside anonymity groups must be the same than the distribution of this attribute in the whole dataset, modulo a threshold *t*.

b) Privacy in P2P: Several works have studied privacy in P2P systems, including privacy-preserving collaborative filtering [28], [29], accountability verifications [30], publish-subscribe systems [7], [31], or content-based routing with Intel SGX [32]. In the context of collaborative filtering, [17] shows how nodes can maintain an obfuscated profile of their interests, and exchange their profiles to route messages to the nodes that are willing to receive them. This approach provides differential privacy to the nodes, but it cannot be applied to video streaming as nodes do not receive all the updates they are theoretically interested in, which would prevent them from visualizing a video stream.

c) Accountability: Several works have described how to modify P2P content dissemination systems to limit the impact of selfish behaviors in peer-to-peer content dissemination systems [33]–[36]. Belenkiy et al. [37] describe how cryptocurrency can help making a file sharing P2P system accountable with privacy, through the use of a central bank. PAG [30] considers the problem of detecting selfish nodes in gossip-based dissemination systems without infringing on the privacy of correct nodes. However, the identity of nodes inside the membership list is not protected, and anyone able to determine which content is propagated would be able to associate a streaming session, and therefore all the nodes in the membership list, with the content disseminated.

d) Trust-based P2P systems: OneSwarm [12] allows users to determine how much trust they place in other peers to protect their interests. This requires manual intervention from the user. Whisper [38] is a middleware that enables group communications with some privacy guarantees, thanks to multi-hops communications. Namely, a node is not able to observe the membership of a group it is not part of, or the content being exchanged in that group. Lu et al. [39] describes a system where nodes rely on a set of trusted collaborators to shield their actions. This category of papers consider that nodes in a group trust each other, which is an assumption we do not make in P3LS.

e) Plausible deniability: Mistrustful P2P [40] relies on erasure coding to avoid advertising users' interests. Chunks of a content are splitted into n fragments. In order to recover a chunk of content, a user then needs to collect at least k of the corresponding n fragments. Nodes then ask for, and transmit, these fragments instead of the original chunks. Mistrustful P2P then organizes interactions between nodes so

that a node can choose to preserve its plausible deniability property against any set of c nodes. However, it is not clear how users can participate in Mistrustful P2P without first joining a membership that is associated to their content of interest. Plausibly deniable systems have also been described for encryption on mobile devices [41]. SwarmScreen [10], in turn, aims at preventing communities of nodes interested in a similar content from being identified. Indeed, nodes sharing interests are much more likely to connect to each other, and an adversary observing the graph of connection between peers could infer the communities. The authors proved this attack to be feasible on BitTorrent traces. To prevent this attack, and depending on their privacy and performance goals, nodes leverage plausible deniability to establish some proportion of random connections to obfuscate the connections graph.

VIII. Conclusion

In this paper, we present P3LS, the first live video streaming protocol to enforce the privacy of its users both at the subscription level, with k-anonymity, and at the dissemination level, with plausible deniability. Compared to k-anonymity, P3LS allows users to tune the level of privacy they want to maintain, and to reduce their bandwidth overhead to up to 30%. On the other hand, plausible deniability in dissemination ensures that an adversary is not able to distinguish the real and fake streams from a statistical analysis (i.e., using an analysis of variance). To do so, P3LS precompute a set of plausibly deniable vectors of coefficients that nodes use in their interactions with other peers to receive content.

Although a user usually watches a single video, considering multiple subscriptions is part of future work.

References

[1] "Youtube usage statistics," https://www.brandwatch.com/blog/39-youtube-stats/.

[2] D. Frey, R. Guerraoui, A.-M. Kermarrec, B. Koldehofe, M. Mogensen, M. Monod, and V. Quéma, "Heterogeneous gossip," in *Middleware*, 2009.

[3] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[4] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec, and R. Patra, "HyRec: Leveraging Browsers for Scalable Recommenders," in *Middleware*, 2014.

[5] "Peertube," https://joinpeertube.org/, accessed: 2018-12-07.

[6] "Pptv," http://pptv.com, accessed: 2018-12-07.

[7] R. Barazzutti, P. Felber, H. Mercier, E. Onica, and E. Riviere, "Efficient and confidentiality-preserving content-based publish/subscribe with prefiltering," *IEEE TDSC*, vol. 14, no. 3, pp. 308–325, 2017.

[8] "General Data Protection Regulation," *Official Journal of the European Union*, vol. L119, pp. 1–88, 2016.

[9] G. Gheorghe, R. L. Cigno, and A. Montresor, "Security and privacy issues in p2p streaming systems: A survey," *Peer-to-Peer Networking and Applications*, vol. 4, no. 2, pp. 75–91, 2011.

[10] D. Choffnes, J. Duch, D. Malmgren, R. Guimera, F. Bustamante, and L. Amaral, "Swarmscreen: Privacy through plausible deniability in p2p systems tech," Technical Report, Northwestern EECS, Tech. Rep., 2009.

[11] S. Le Blond, C. Zhang, A. Legout, K. Ross, and W. Dabbous, "I know where you are and what you are sharing: exploiting p2p communications to invade users' privacy," in *IMC*, 2011.

[12] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving p2p data sharing with oneswarm," in *ACM CCR*, vol. 40, no. 4, 2010, pp. 111–122.

[13] A. Boutet, D. Frey, A. Jégou, A.-M. Kermarrec, and H. B. Ribeiro, "Freerec: an anonymous and distributed personalization architecture," *Computing*, vol. 97, no. 9, pp. 961–980, Sep 2015.

[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Tech. Rep., 2004.

[15] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale." in *OSDI*, 2012.

[16] S. B. Mokhtar, G. Berthou, A. Diarra, V. Quéma, and A. Shoker, "Rac: A freerider-resilient, scalable, anonymous communication protocol," in *ICDCS*, 2013.

[17] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec, "Privacy-preserving distributed collaborative filtering," *Computing*, vol. 98, no. 8, pp. 827–846, 2016.

[18] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié, "Epidemic information dissemination in distributed systems," *Computer*, vol. 37, no. 5, pp. 60–67, 2004.

[19] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[20] V. Bindschaedler, R. Shokri, and C. A. Gunter, "Plausible deniability for privacy-preserving data synthesis," *PVLDB*, vol. 10, no. 5, pp. 481–492, 2017.

[21] G. Ver Steeg and A. Galstyan, "Information transfer in social media," in *WWW*, 2012, pp. 509–518.

[22] N. Li, T. Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," in *ICDE*, 2007, pp. 106–115.

[23] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017, pp. 1175–1191.

[24] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM TOCS*, vol. 25, no. 3, 2007.

[25] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Comput. Netw.*, vol. 53, no. 13, pp. 2340–2359, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.03.008

[26] I. Wagner and D. Eckhoff, "Technical privacy metrics: a systematic survey," *ACM CSUR*, vol. 51, no. 3, p. 57, 2018.

[27] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy Beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 2007.

[28] R. Guerraoui, A.-M. Kermarrec, R. Patra, M. Valiyev, and J. Wang, "I know nothing about you but here is what you might like," in *DSN*, 2017.

[29] A. Boutet, F. De Moor, D. Frey, R. Guerraoui, A. Kermarrec, and A. Rault, "Collaborative filtering under a sybil attack: Similarity metrics do matter!" in *DSN*, 2018.

[30] J. Decouchant, S. B. Mokhtar, A. Petit, and V. Quéma, "Pag: Private and accountable gossip," in *ICDCS*, 2016.

[31] E. Onica, P. Felber, H. Mercier, and E. Rivière, "Confidentiality-preserving publish/subscribe: a survey," *ACM CSUR*, vol. 49, no. 2, p. 27, 2016.

[32] R. Pires, M. Pasin, P. Felber, and C. Fetzer, "Secure content-based routing using intel software guard extensions," in *Middleware*, 2016.

[33] A. Diarra, S. B. Mokhtar, P.-L. Aublin, and V. Quéma, "Fullreview: Practical accountability in presence of selfish nodes," in *SRDS*, 2014.

[34] R. Guerraoui, K. Huguenin, A.-M. Kermarrec, M. Monod, and S. Prusty, "Lifting: lightweight freerider-tracking in gossip," in *Middleware*, 2010.

[35] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar gossip," in *OSDI*, 2006.

[36] S. B. Mokhtar, J. Decouchant, and V. Quéma, "Acting: Accurate freerider tracking in gossip," in *SRDS*, 2014.

[37] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, A. Lysyanskaya, and E. Rachlin, "Making p2p accountable without losing privacy," in *WPES*, 2007.

[38] V. Schiavoni, E. Rivière, and P. Felber, "Whisper: Middleware for confidential communication in large-scale networks," in *ICDCS*, 2011.

[39] Y. Lu, W. Wang, B. Bhargava, and D. Xu, "Trust-based privacy preservation for peer-to-peer data sharing," *IEEE Transactions on SMC*, vol. 36, no. 3, pp. 498–502, 2006.

[40] P. M. da Silva, J. Dias, and M. Ricardo, "Mistrustful p2p: Privacy-preserving file sharing over untrustworthy peer-to-peer networks," in *IFIP Networking*, 2016.

[41] B. Chang, F. Zhang, B. Chen, Y. Li, W.-T. Zhu, Y. Tian, Z. Wang, and A. Ching, "Mobiceal: Towards secure and practical plausibly deniable encryption on mobile devices," in *DSN*, 2018.