Pollution Attacks Detection in the P2PSP Live Streaming System

Cristóbal Medina-López, L.G. Casado and Vicente González-Ruiz

Abstract This work studies the *pollution attack*: a challenging security-related problem in peer-to-peer streaming platforms. Different variations of this attack and its combinations are addressed. In order to mitigate such attacks, two different strategies have been proposed in the context of the P2PSP live streaming system, a peer-to-peer streaming platform with free access (i.e. it is not necessary to provide an identification and only endpoints are stored). The first prevention strategy is based on the existence of anonymous trusted peers that detect and report attackers, which are finally expelled. The second strategy increases the security level in the overlay at the cost of a higher computation and communication overhead. Both strategies are analyzed theoretically for several attack configurations.

Keywords P2PSP · Content integrity · Live streaming · Pollution attacks · Peer-to-peer networks

1 Introduction

The problem with scalability in client-server based live video streaming systems is well known: the server is forced to upload a copy of the content per client. In a peer-to-peer system the number of copies uploaded by the server is usually orders of magnitude lower than the number of peers. The collection of peers are in charge of sharing the content among them using their upload bandwidth excess. For that reason, the peer-to-peer (P2P) model is an interesting alternative to Client-Server model, specially since clients see how their available upload bandwidth grows year after year.

C. Medina-López (✉) · L.G. Casado · V. González-Ruiz
University of Almería (CeiA3), Almería, Spain
e-mail: cristobalmedina@ual.es

L.G. Casado
e-mail: leo@ual.es

V. González-Ruiz
e-mail: vruiz@ual.es

Á. Herrero et al. (eds.), *International Joint Conference*, Advances in Intelligent Systems and Computing 369, DOI 10.1007/978-3-319-19713-5_34

Because peers obtain most of the content from other peers rather than from the content source, it is easy for an attacker to alter and resend the data packets it receives. Therefore, it is necessary to design defense mechanisms. This paper focuses on fighting against pollution attacks [5], which are those that change the delivered content in some way. Here, two defense strategies are proposed in the context of the P2PSP streaming system [14, 17]. The first one keeps the communication overhead as low as possible but presents security risks that will be discussed. The second strategy mitigates those risks at the price of a higher communication overhead. Moreover, this work presents an analysis of the types of attacks that each strategy can bear or at least mitigate, as well as the best configurations to perform them.

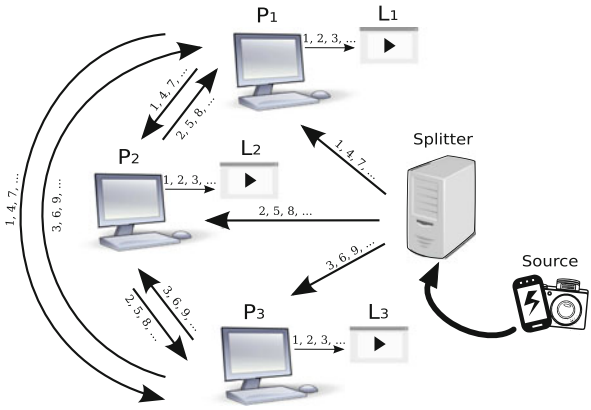

Fig. 1 A P2PSP team. Numbers in arrows refer the transmitted chunk numbers

The rest of this paper is organized as follows. Brief overviews of the P2PSP system and different pollution attacks are shown in Sects. 2 and 3, respectively. Related work is presented in Sect. 4. In Sect. 5 two strategies to prevent pollution and related attacks, introduced in Sect. 3, are presented. Analytical results for both strategies are shown in Sect. 6. The Sect. 7 shows the difficulties to determine who is the attacker, and the last section shows the conclusions and future work.

2 Description of P2PSP

The Peer-To-Peer Straightforward Protocol (P2PSP) is an application-layer protocol designed for real-time broadcasting of media over a peer-to-peer overlay. P2PSP establishes a push-based [10] fully connected mesh scheme [11] where every peer is connected with each other (see Fig. 1). A basic P2PSP network consists of a *splitter* (S) and a collection of *peers* (P_i) named *team*. The splitter receives a media stream from a *source* (O), divides the stream in data chunks of the same size and sends them to the team following a Round-Robin scheme. Next, each peer forwards those chunks received from the splitter to the rest of peers (chunks received from other peers are

not forwarded). Finally, each peer sends the reconstructed stream to a *listener* (L), which is usually a media player.

As a consequence of this simple communication algorithm, the P2PSP is very suitable for execution in nodes with constrained computational resources [13]. Moreover, a peer failing or corrupting its data chunk before forwarding it, produces the loss of the same chunk for the other peers. This feature can be used to efficiently implement error concealment techniques [16], which are less effective when lost chunks are consecutive. For example, if the number of attackers or free riders is 2 on a team with 200 peers, only 1 % of chunks would not be played for some peers (those being attacked or unserved).

3 Pollution Attacks

Content integrity is one of the major issues in most P2P systems when trying to guarantee the quality of service in live-stream transmissions. Pollution attacks [5], also known as stream spoiling, basically consist of a peer or a set of peers modifying the content of the stream. Pollution attack can be done in different ways. Some of them, and other related attacks, are described next:

Persistent attack [5]: This attack consists of doing the highest possible amount of damage in the shortest period of time. Therefore, an attacker poisons every chunk received from the splitter and sends them to the entire team.

On-off attack [7]: In order to improve the *Persistent attack* and to avoid to be quickly expelled from the team, the attacker only poisons some chunks (for instance, 10 % of the total sent to the team). By acting this way, the attacker is resilient to detection systems using trust management methods, since the attacker would be assigned a high enough level of trust by the staying network.

Selective attack: This attack consists of poisoning chunks intended for only one peer or a small subset of peers. As in the previous attack, the reason behind this behavior is to be unnoticed by most of the peers and thus avoid to be expelled if a voting system is used by the overlay.

Collaborative attack [8]: Sometimes a single attacker is not able to produce a big damage; however several attackers may collaborate to produce *Selective* and *On-off* attacks to a large set of peers. By doing so, the amount of information obtained by a pollution detection system about an individual attacker is smaller than in single attacker variants. This is the most difficult attack to deal with.

Hand-wash attack [9]: Those attackers that have been discovered or think that they could have been discovered, leave the team and return to continue the attack with another alias.

Bad-mouth attack [8]: It can be used when peers can complain about others in an attacker detection system. Attackers do bad-mouthing by intentionally blame others regular peers for sending poisoned chunks or not sending chunks, with the intention of expelling them.

4 Related Work

Vulnerability and malicious peer behavior are some of the major issues in P2P overlays. For this reason, these topics have been studied extensively. In the specific scenario of live streaming, the dominant approaches to fight against attacks are: (1) trust management with reputation systems [7, 12, 18], (2) hashing/signature schemes [5] and (3) the use of trusted peers [2], to detect selfish peers, also known as free riders. A selfish peer relies smaller amount of data than it receives, or even no data. Following, a brief overview of previous defense mechanisms is shown:

Trust management: Each peer assigns a trust value for each other peer in the overlay. Moreover, each peer has its own perspective about its trust on the rest. Usually, a peer is expelled from the overlay when its trust value is smaller than a threshold. The main difficulty is how to compute a fair trust value for a peer. The common attack to trust management systems is the *On-off* one. *Trust management* and a method based on direct and indirect trust to fight against *On/off attack* is shown in [8].

Hashing/signature: They are used to detect poisoned chunks at the cost of an additional communication and computation overhead. The most common attack to this defense is the *Bad-mouth* attack. Non-repudiation methods can be a solution, but they still present some drawbacks for P2P networks [3].

Trusted peers: A set of trusted peers to monitor the bandwidth usage by other peers can be deployed [2]. Free rider peers detected by a trusted peer are expelled.

5 Proposals

This section presents two different strategies aiming to mitigate the impact of pollution and related attacks by combining trust management, hashing/signatures and trusted peers. Each strategy defines a set of rules, specially designed for the P2PSP system. The first strategy, denoted by STrPe (Strategy based on Trusted Peers), is a simple approach with a low data overhead in the overall operation of the team. The only difference with respect to an pollution-unaware P2PSP system is an extension of the splitter functionality and the inclusion of anonymous trusted peers (TP) who transparently monitor the behavior of regular peers in the team. Regular peers see a TP as another regular peer.

The second strategy, denoted by STrPe-DS (Strategy based on Trusted Peers and Digital Signatures) is an extension of the first one, where a digital signature of a chunk allows to detect attackers. STrPe-DS generates more data overhead than STrPe but the performance of the defense is greatly improved.

Those attacks detected by STrPe are also detected by STrPe-DS, but not the other way around. Therefore, the most adequate strategy should be used depending on the risk to be assumed. This decision is usually made before the deployment but it is also possible to change it once the P2PSP overlay is running.

5.1 Strategy Based on Trusted Peers (STrPe)

STrPe has been designed to maintain the simplicity characterizing the P2PSP system. By including trusted peers (TPs) into the team, an attacker sending a poisoned chunk to some TP is detected and expelled from the team. For this reason it is important to maintain the anonymity of TPs among regular peers but the splitter. The behavior rules are:

1. Only the splitter knows who are the TPs in the team. It is possible that all peers are TPs except the attacker.
2. Each TP creates a hash for each received chunk, including the chunk number and the endpoint of the source of the chunk. Depending on the computational power available in the TPs, all chunks or a random subset of them may be processed in each round.
3. TPs send the hashes to the splitter, who checks whether the chunks have been altered by comparing them with those hashes calculated when the chunks were delivered to the team. The splitter only listens to TPs for this task.
4. The splitter knows the peer in charge of relaying a given chunk and it knows who altered a chunk when an invalid hash has been received from a TP. Any exposed attacker is expelled from the team by removing it form the list of peers in the splitter (this implies that no more chunks will be send to it). In order to ensure that the attacker is removed from all lists of peers as soon as possible, the splitter sends a expulsion message containing the endpoint of the attacker to all peers in the team.

5.2 Strategy Based on Trusted Peers and Digital Signatures (STrPe-DS)

The STrPe-DS has been designed to mitigate the *Selective attack* and to identify poisoned chunks by using digital signatures. So, any peer is able to know if a chunk was poisoned and/or relayed by a different peer than the one in charge of it. Rules defining STrPe-DS are:

1. A peer receives the public key of the splitter, plus the other necessary information, when a peer joins the team.
2. For each chunk, the splitter sends a message with the chunk, its number (*nChunk*), the destination address (*dst*) and a digital signature $\{chunk, nChunk, dst, S_{priv}(H(chunk + nChunk + dst))\}$, where H is a hash function and S_{priv} is the private key of the splitter.
3. When a peer receives a message, it performs the following steps:
   (a) Check whether *dst* matches the address of the sender. Notice that this action is vulnerable to the well known *Spoofing attack* [1]. The next step is performed only if this one is successful.

(b) Check the correctness of the hash value in the message.

If any of previous checks fails, the sender is removed from the list of peers of the current peer.

4. The splitter periodically requests (and the peers serve) the list of removed peers (since the previous request). In this way, a *Deny of Service (DoS) attack* by sending removed peers to the splitter at high rate is avoided. TPs are the only ones that can send the list of removed peers to the splitter as soon as they are detected.
5. Peers removed by any TP are directly expelled by the splitter (see point 4 in Sect. 5.1).
6. The splitter can decide to expel a peer based on the information received from well-intended or attackers peers. So, a typical approach here is to establish a trust value for each peer depending on several aspects, such as the number of complaints received about a peer during a given period of time, the number of affected peers by a possible attack, the age of the peers in the team, etc. (Fig. 2).

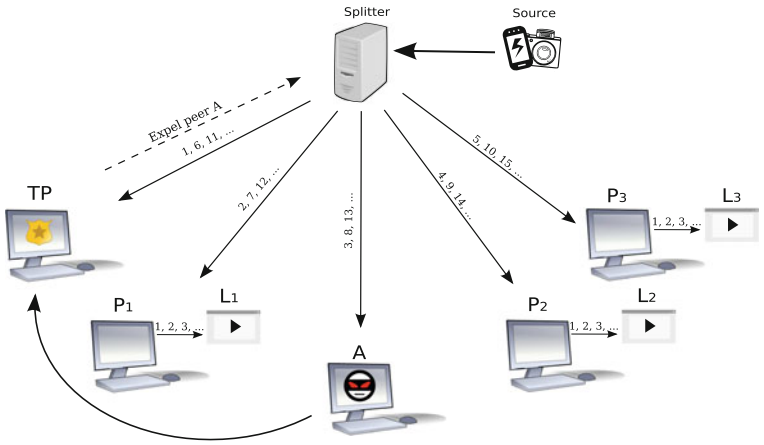

Fig. 2 A P2PSP using STrPe-DS, where a poisoned chunk is received by the TP, which immediately informs to the splitter

Any exchange of information between the splitter and peers should be authenticated by a digital signature in order to avoid *spoofing* attacks but we are interested in free access system where each peer is determined by its endpoint.

6 Analytical Results

Defense mechanisms in P2P overlays can be evaluated in several ways. The preferable is to test the system in a real scenario, but the cost of these experiments can be unaffordable. Another possibility is to analyze a set of simulations with differ-

ent combinations of attack/defense schemes. We will address them in a future work. Here we perform a theoretical study of the advantages and drawbacks of STrPe and STrPe-DS strategies.

6.1 STrPe

Regarding the communication overhead, only the communications between the trusted peers and the splitter suffer it in this strategy. TPs need to send to the splitter a message digest for each received chunk from other peers, this implies a total overhead $O = sH \times Cps \times nTP$, where sH is the size of the hash, nTP is the number of trusted peers and Cps are the chunks per second. For instance, let suppose that a video, whose bit-rate is 1,024 Kbps, is being streamed and that the chunk size is 1,024 bytes. So, at least 125 chunks are being transmitted per second. If hash function generates a digest of 224 bit, the overhead per chunk is 224 bits. Thus, the communication overhead between a trusted peer and the splitter is $125 \times 224 = 28{,}000$ bps $\simeq 27.3$ Kbps. In this example, the total overhead in the team due to the defense strategy would be 27.3 Kbps $\times\, nTP$.

Concerning the probability of detecting an attacker, Fig. 3 shows it as $P = nTP/(N-A)$, where N is the team size and A is the number of attackers. The number of attackers and TPs represents a percentage of the size of the team. In general, the chance to detect an attacker increases with the number of trusted peers in the team.

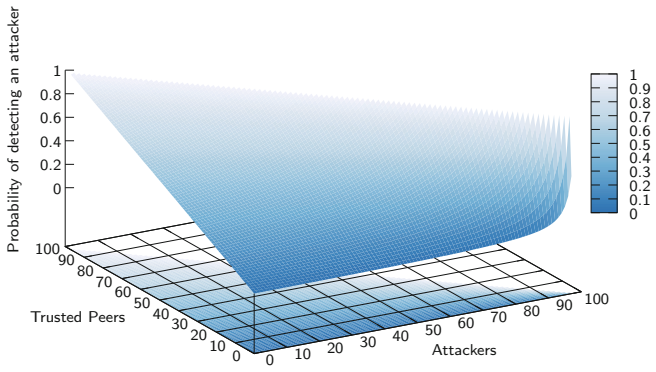

Fig. 3 Probability to detect an attacker depending on the percentages of trusted peers and attackers in the team

A *Persistent* and *On-off attacks* are quickly detected by the splitter. A *Bad-mouth attack* makes little sense in STrPe since only TPs are able to inform about hashes to the splitter. However, the main vulnerability of STrPe is that TPs must remain anonymous. In case of being discovered, the system will not be resilient to *Selective attacks* where attackers do not pollute chunks for TPs. Finally, regular peers do not know if a chunk has been polluted because the chunks are not digitally signed.

6.2 STrPe-DS

Whereas only TPs have communication overhead in previous strategy, the whole team suffer from it in this strategy, because all messages contains not only the chunk but also additional information (see Sect. 5.2). Using the same example of the previous section, if a RSA signature with 1,024 bit key is used (it seems to be fast enough for our purpose, according standards signature times [4]), the overhead per chunk is 1,072 bits (1,024 bits for signature + 48 bits for *dst*). Thus, the communication overhead is $125 \times 1{,}072 = 134{,}000$ bps $\simeq 130.8$ Kbps. However, in this strategy, the total overhead in the team is more difficult to calculate because it depends on the size of the removed peers request made by the splitter, the frequency of such requests, the number of attackers and the number of the trusted peers.

Regarding the detection of attacks, when a *Persistent attack* is carried out over the entire team, it is quickly detected by any TP (see Fig. 2).

A *On-off attack* is meaningless because peers remove the attackers from their lists as soon as they receive an incorrect message, according to the digital signature of the splitter. This attack becomes a *Persistent attack* in a relatively large period of time.

Performing a *Bad-mouth attack* by only one attacker is not usually enough to expel a regular peer in trust based systems. Additionally, the attacker will usually have more complains than any attacked peer and could finally be detected. According to the P2PSP system, a peer A removes a peer B from its list of peers because it does not receive chunks from B or the chunks received are incorrect. This results in B also removes A from its list of peers given that B is not receiving chunks from A anymore. In any case, it is not possible to know who is the attacker.

Expelling a legitimate peer by means of a *Bad-mouth attack* requires a combination with *Selective* and *Collaborative attacks*, and the set of selected peers to be attacked must be small in comparison with the set of attackers. In the absence of TPs, the team can run out of well-intended peers after several repetition of previous combination of attacks. Therefore, not only the rejection-threshold in the splitter or the trust value of peers but also the number of trusted peers in the team are important.

A *Hand-wash attack* is difficult to deal with, if the attacker can guess the value of the rejection-threshold established by the splitter. The rejection-threshold must be established to minimize the damage of the possible attacks and the possibility to expel a polite peer. TPs should perform a kind of *Hand-wash strategy* in order to reduce the probability of be detected.

To the best of our knowledge, there is no effective defense against a combined *Selective*, *Collaborative*, *Bad-mouth* and *Hand-wash attacks* in free access P2P live streaming protocols.

7 Expelling Peers from the Team

Digital signature of chunks allows to each peer receiving a poisoned chunk to detect the attacker, who is removed from its peer list. As mentioned before, a trusted peer receiving a poisoned chunk reports it to the splitter to expel the attacker from the

team. In the StrPe-DS strategy peers can complain to the splitter. The splitter is in charge to decide who is the attacker based on the gathered information. This is a difficult task, because for instance, if five peers are complaining about one peer, it is difficult for the splitter to know whether there are five attackers trying to expel one well-intended peer (*Bad-mouth* attack) or if it is actually an attacker poisoning five well-intended peers. To address this problem there exists two alternatives:

Non-repudiation methods. Most of the current non-repudiation system are based on the existence of a Trusted Third Party (TTP). TTPs are not in consonance with the P2P philosophy because they reduce the distributed computing level in P2P systems. There are some proposals [3] where standard peers act as TTP, but they could be malicious, as well.

There are solutions without TTPs, but they are inefficient since the number of necessary messages is usually high [6]. Additionally, these solutions consider that both parties are interested in the content. This is not the case for P2P streaming systems where the attacker is interested in poisoning the content but not in the content itself. Thus, currently there is not a suitable non-repudiation system allowing the splitter to decide who is the attacker.

Trust-based methods. Due to the absence of a suitable non-repudiation system, this is usually the most used solution. In the StrPe-DS strategy, the splitter gathers all complaints from peers. Based on this and other possible information about peers, the splitter has to establish a decision method in order to determine who will be expelled from the team. As we shown before this is a difficult task.

As future work, we are interested in to study an efficient method for non-repudiation in P2P live streaming systems and a fair trust-based method [15].

8 Conclusions

In this paper two different strategies to mitigate pollution attacks in the P2PSP system are studied: (1) a strategy with low computation and communication overhead, which is designed for resource constrained devices and (2) an extension of the first strategy that increases the defense level against pollution attacks at the expense of introducing a higher overhead. It is necessary to develop a fair trust-based method in order to make the last strategy effective.

There are several problems that must be addressed in order to develop a system against pollution attacks in P2P live streaming networks with free access. A non-repudiation methods without Trusted Third Party can be a good solution but unfortunately the existing proposals are not suitable for P2P live streaming systems. We are also interested in study this topic in a future work.

Acknowledgments This work has been funded by grants from the Spanish Ministry (TIN2012-37483) and Junta de Andalucía (P11-TIC-7176), in part financed by the European Regional Development Fund (ERDF).

References

1. Bellovin, S.M.: Security problems in the TCP/IP protocol suite. SIGCOMM Comput. Commun. Rev. **19**(2), 32–48 (1989)
2. Conner, W., Nahrstedt, K.: Securing peer-to-peer media streaming systems from selfish and malicious behavior. In: Proceedings of the 4th on Middleware Doctoral Symposium, p. 13. ACM (2007)
3. Conrad, M.: Non-repudiation mechanisms for peer-to-peer networks: enabling technology for peer-to-peer economic markets. In: Proceedings of the 2006 ACM CoNEXT Conference, CoNEXT '06, pp. 30:1–30:2. ACM, New York, NY, USA (2006)
4. Dai, W.: Speed Comparison of Popular Crypto Algorithms. http://www.cryptopp.com/benchmarks.html
5. Dhungel, P., Hei, X., Ross, K.W., Saxena, N.: The pollution attack in P2P live video streaming: measurement results and defenses. In: Proceedings of the 2007 Workshop on Peer-to-peer Streaming and IP-TV, pp. 323–328. ACM (2007)
6. Gouda, M.G.: Keynote talk: communication without repudiation: the unanswered question. In: Networked Systems, pp. 1–8. Springer (2014)
7. Hu, Bo., Zhao, H.V.: Pollution-resistant peer-to-peer live streaming using trust management. In: 2009 16th IEEE International Conference on Image Processing (ICIP), pp. 3057–3060, Nov 2009
8. Xin, K., Wu, Y.: A trust-based pollution attack prevention scheme in peer-to-peer streaming networks. Comput. Netw. **72**, 62–73 (2014)
9. Lin, W.S., Zhao, H.V., Liu, K.J.R.: Attack-resistant collaboration in wireless video streaming social networks. In: 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), pp. 1–4. IEEE (2010)
10. Cigno, R.L., Russo, A., Carra, D.: On some fundamental properties of P2P push/pull protocols. In: ICCE 2008 Second International Conference on Communications and Electronics, pp. 67–73. IEEE (2008)
11. Magharei, N., Rejaie, R.: Understanding mesh-based peer-to-peer streaming. In: Proceedings of the 2006 international Workshop on Network and Operating Systems Support for Digital Audio and Video, p. 10. ACM (2006)
12. Marti, S., Garcia-Molina, H.: Identity crisis: anonymity vs reputation in P2P systems. In: Proceedings of Third International Conference on Peer-to-Peer Computing, (P2P 2003). pp. 134–141. IEEE (2003)
13. Medina-López, C., García-Ortiz, J.P., Naranjo, J.A.M., Casado, L.G., González-Ruiz, V.: IPTV using P2PSP and HTML5+WebRTC. In: The Fourth W3C Web and TV Workshop. Munich, Germany, March 2014
14. Medina-López, C., Naranjo, J.A.M., García-Ortiz, J.P., Casado, L.G., González-Ruiz, V.: Execution of the P2PSP protocol in parallel environments. In: Guillermo Botella y Alberto A. Del Barrio Garcia, editor, Actas XXIV Jornadas de Paralelismo (http://www.congresocedi.es/images/site/actas/ActasParalelismo.pdf), pp. 216–221. Madrid, Septiembre 2013
15. Liu, Y., Yang, S., Guo, L., Chen, W., Guo, L.: A distributed trust-based reputation model in p2p system. In: 2007 Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2007, vol. 1, pp. 294–299, July 2007
16. Salama, P., Shroff, N.B., Coyle, E.J., Delp, E.J.: Error concealment techniques for encoded video streams. In: International Conference on Image Processing, vol. 1, pp. 9–9. IEEE Computer Society (1995)
17. P2PSP Team. Peer to Peer Straightforward Protocol. http://p2psp.org/en/p2psp-protocol
18. Vieira, A.B., Campos, S., Almeida, J.: Fighting attacks in P2P live streaming. simpler is better. In: IEEE INFOCOM Workshops 2009, pp. 1–2. IEEE (2009)