POLITECNICO DI TORINO
Repository ISTITUZIONALE

Characterization of Band Codes for Pollution-Resilient Peer-to-Peer Video Streaming

Original
Characterization of Band Codes for Pollution-Resilient Peer-to-Peer Video Streaming / Fiandrotti, Attilio; Rossano, Gaeta; Marco, Grangetto. - In: IEEE TRANSACTIONS ON MULTIMEDIA. - ISSN 1520-9210. - STAMPA. - 18:6(2016), pp. 1138-1148. [10.1109/TMM.2016.2535781]

Availability:
This version is available at: 11583/2671708 since: 2017-06-07T15:23:01Z

Publisher:
IEEE

Published
DOI:10.1109/TMM.2016.2535781

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

12 October 2024

Characterization of Band Codes for Pollution-Resilient Peer-to-Peer Video Streaming

Attilio Fiandrotti, *Member, IEEE,* Rossano Gaeta, and Marco Grangetto, *Senior Member, IEEE,*

***Abstract*—We provide a comprehensive characterization of Band Codes (BC) as a resilient-by-design solution to *pollution attacks* in Network Coding (NC) based peer-to-peer live video streaming. Consider one *malicious* node injecting bogus coded packets into the network: the recombinations at the nodes generate an avalanche of novel coded bogus packets. Therefore, the malicious node can cripple the communication by injecting in the network only a handful polluted packets. Pollution attacks are typically addressed by identifying and isolating the malicious nodes from the network. Pollution detection is however not straightforward in NC as the nodes exchange coded packets. Similarly, malicious nodes identification is complicated by the ambiguity between malicious nodes and nodes that have involuntarily relayed polluted packets. This paper addresses pollution attacks through a radically different approach which relies on BC. BC are a family of rateless codes originally designed for controlling the NC decoding complexity in mobile applications. Here we exploit BC for the totally different purpose of recombining the packets at the nodes so to avoid that the pollution propagates by adaptively adjusting the coding parameters. Our streaming experiments show that BC curb the propagation of the pollution and restore the quality of the distributed video stream.**

***Index Terms*—Network coding, peer to peer, pollution attack, measurements, continuity index**

I. Introduction

NOWADAYS, there are many proposals to exploit random Network Coding (NC) to support peer-to-peer (P2P) based cooperative media streaming [1], [2]. It was shown that joining the peculiarities of both techniques yields effective aggregation of users computing and communication capacities, increases the system throughput and solves issues such as the rarest-piece [3], [4]. In NC-based communications, the media is organized in chunks called *generations*, where each generation is further organized in blocks of symbols of identical size. A source node holds the original media content and, for each generation, transmits random linear combinations of the blocks to the peer nodes in the form of coded network packets. The peer nodes receive and buffer the coded packets and periodically exchange with their neighbors linear combinations of the buffered packets drawn at random. When a peer node has collected enough coded packets, it solves a system of equations, decodes the original blocks and recovers the generation.

NC-based architectures are however liable to *pollution attacks* [5], where one or more *malicious nodes* transmit on purpose bogus coded packets to cripple the communication. When a peer node solves the system of equations corresponding to the received packets, it is sufficient that one of the received packets is bogus to make the recovered generation bogus as well. Pollution attacks are particularly treacherous with NC architectures because of the recombinations: if a peer node draws for recombination a bogus packet, also the transmitted packet is bogus, and so the number of bogus packets in the network grows at each recombination. Our previous research [6], [7] showed that a malicious node which injects a few bogus packets can trigger an avalanche of bogus packets thanks to the recombinations, crippling the communication.

Due to the peculiarities of NC, traditional countermeasures such as detection of bogus (*polluted*) packets and identification (and isolation) of its originators fall short in several aspects. Because in NC packet payloads are coded and any payload is in principle admissible, there is no simple way to tell whether a packet is *clean* (i.e., it carries a correct combination of the symbols) or it is polluted and thus it should be discarded. Also, due to the recombinations, an honest node which has relayed a polluted packet may be incorrectly labeled as malicious (*false positive* case). Therefore, to present date, pollution attacks in NC are a largely unsolved issue and the shortcomings of existing approaches prompt the research for innovative solutions.

Paper contribution

This paper proposes a novel approach to the problem of pollution attacks in NC-based networks that leverages a family of low-complexity rateless codes called Band Codes (BC) [8]. BC were originally designed to address the totally orthogonal issue of energy-efficiency, offering adjustable decoding complexity as a function of the *coding window* size. Shortly, random (re)combinations are constrained to a subset of blocks (coded packets) drawn at random from a coding window whose size controls the decoding complexity (and preserves it through the recombinations). This work builds however upon BC in a distinct domain, proposing packet recombination algorithms and schemes which allow to build a network which is resilient by design to pollution attacks. We refer the reader interested in the decoding complexity aspects of BC to [8], where we experiment with real mobile devices showing that BC extend the device operational lifetime with respect to classic NC.

The approach to pollution attacks we propose builds upon and extends the preliminary results we reported in [7]. There, we investigated a large-scale network where malicious and honest nodes cooperate streaming a live video feed. Such scenario is challenging because attacks must be detected and handled within a tight time interval. This work finalizes the analysis of BC as a tool for achieving resilience to pollution attacks in NC, contributing novel analytical results through thorough experimenting.

First, we further the analysis of our probabilistic pollution detection scheme built upon the BC decoding algorithm. Namely, we experiment with different detection strategies, we measure the pollution detection probability and the corresponding gain in video quality. Our BC-based probabilistic pollution detection scheme early quenches the pollution propagation and yields better video quality than a reference scheme which relies on verification with a trusted server at decoding time.
Second and foremost, we experiment with different combinations of BC coding parameters, exploring the tradeoff between resilience to pollution propagation (and thus, video quality), coding overhead and decoding complexity. The coding parameters are adjusted adaptively at streaming time as a reaction to the discovery of a pollution attack, coping with the varying levels of activity of the malicious nodes. Our experiments reveal resilience to pollution attacks and satisfactory video quality with limited impact on the coding efficiency. As a side benefit, our BC-based approach also guarantees controlled decoding complexity thanks to BC. This is a major improvement over our previous approach [6], where we achieved similar pollution resilience without the benefits of controlled decoding complexity.

This paper is organized as follows: Sec. II overviews Band Codes and illustrates the random push protocol for P2P live streaming we developed for evaluating BC characteristics. Sec. III describes the attack model we consider, the pollution detection we propose as well as the strategies to reduce pollution effects once a pollution attack has been detected. Sec. IV and V describe our experimental setup and presents all the results we obtained. Finally, Sec. VI discuss some related literature whereas Sec. VII summarizes contributions of our work highlighting possible future developments.

II. Background

In this section we first overview those Band Codes (BC) aspects instrumental to the goals of this paper [8]: we refer the interested reader to [8] for a thorough description of BC. Next, we briefly describe the key aspects of *ToroStream*, our random-push protocol for P2P live video streaming designed around Band Codes: we refer the interested reader to [9] for a thorough description of the protocol.

A. Band Codes

In NC video streaming the compressed content is subdivided in chunks of approximately the same size called *generations*, where each generation is independently encoded and decoded. Each generation typically encompasses one or more self-decodable units of video (e.g., one or more Groups of Pictures - GOPs). Next, the source subdivides each generation in k blocks of symbols of identical size $(x_1, ..., x_k)$ where each block is approximately the size of a network packet and k is known as *generation size*, as shown in Fig. 1. Every time the source node is granted a *transmission opportunity*, i.e. it is allowed to transmit a packet to the network, it generates a linear combination of the k blocks as $y = \sum_{i=1}^{k} g_i x_i$. In the following we focus on *binary* combinations, i.e. $g_i \in GF(2)$, due to the favorable tradeoff between decoding complexity and coding efficiency; in this case the summation corresponds to a bit-wise XORs of the original blocks. The vector $g = (g_1, ..., g_k)$ is known as *coding vector*, and in the considered binary NC scenario it is such that $g_i \in \{0, 1\}$, i.e. g_i=1 if the i-th block is encoded in a packet, g_i=0 otherwise. The number of elements of g equal to one corresponds to the number of blocks encoded and is referred to as the *degree* of the packet. Eventually, the source transmits to the network a packet $P(g, y)$ that contains the coded payload y prefixed by the relative coding vector g, so that the set of blocks encoded in the payload is known at the receiver.

With BC the blocks to be XORed are randomly drawn in a subset known as *coding window*, i.e. a set of W adjacent blocks, where $W \leq k$ is an input parameter (notice that for $W = k$, BC are equivalent to standard Random NC). In the following, we call the first and last block spanned by a specific coding window the leading and trailing edge of the window and we indicate them as f and l respectively. Fig. 1 shows a generation of k=6 blocks: for a window of size W=4, the leading edge index was drawn at random so that f=2, and consequently $l = f + W - 1 = 5$. Given k and W, the source draws f from an ad-hoc distribution defined in [8] which makes all blocks equally likely to be found into a coded packet. Next, each element of the coding vector within the coding window is independently drawn from a uniform binary distribution so that $\mathcal{P}\{g_i = 1\} = \frac{1}{2}$ if $f \leq i \leq l$, $g_i = 0$ otherwise. Finally, the source generates a coded payload y as described above and transmits packet $P(g, y)$ to the network. In the following, a packet which belongs to a generation of size k and whose coding vector elements are drawn according to a coding window of size W as described above will be indicated as $BC(k, W)$. As we showed in [8], the degree of the packets coded according to the above scheme follows the binomial distribution $\mathcal{B}(W, \frac{1}{2})$, which is the first step towards controlled decoding complexity.

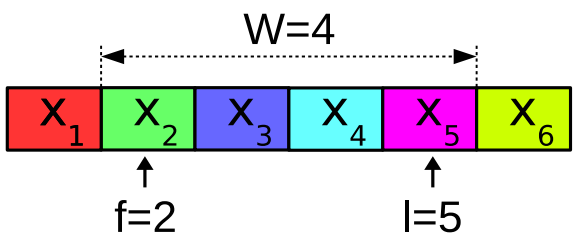

Fig. 1. Generation of size $k = 6$ blocks and coding window of $W = 4$ blocks, with leading and trailing edges $f = 2$ and $l = 5$.

Every node receives BC coded packets from the network and stores them into a separate input buffer for each generation. The received blocks are used for both decoding and recombination to propagate novel coded information in the network.
Let us assume that the input buffer contains q packets $\{P^1, \ldots, P^i, \ldots, P^q\}$, where $P^i = (g^i, y^i)$ and all packets are $BC(k, W)$. The node aims at producing a $P^r(g^r, y^r)$ which is still a $BC(k, W)$. The recombined packet $P^r(g^r, y^r)$ is defined by a linear combination of the coded packets in the input buffer with $g^r = \sum_{i=1}^{q} c_i g^i$ and $y^r = \sum_{i=1}^{q} c_i y^i$.

To guarantee that $P^r(g^r, y^r)$ is still $BC(k, W)$, the node draws the leading edge f_r (and the corresponding trailing edge $l_r = f_r + W - 1$) from one of the $k+W+1$ coding windows according to the same probability function used by the source. Now, let s^i be the first non-null element of the coding vector of the i-th packet in the buffer, i.e. $\forall j < s^i$, $g^i_j = 0$, and let t^i be the last non-null element of the coding vector of the i-th packet in the buffer, i.e. $\forall j > t^i$, $g^i_j = 0$. The node independently draws each $c_i \in \{0, 1\}$ so that $\mathcal{P}\{c_i = 1\} = \frac{1}{2}$ if $f_r \leq s^i \leq t^i \leq l_r$, and $c_i = 0$ otherwise. That is, any packet which falls outside the randomly drawn coding window (f_r, l_r) of size W is excluded from recombination. It turns out that the degree distribution of P^r is still the same binomial distribution $\mathcal{B}(W, \frac{1}{2})$ of the packets in the input buffer, i.e. the packet recombination preserves the degree distribution and thus the decoding complexity with depends on W. Whereas the above BC packet recombination scheme above was conceived to control the decoding complexity, in the following we show that such scheme can be exploited to introduce resilience against pollution propagation by properly adjusting the coding window size W.

To decode the original information each network node processes each received packet $P(g, y)$ via a Gaussian elimination-like algorithm [10] which solves a system of k linear equations $GX = Y$, where G is the $k \times k$ matrix which holds the coding vectors of the received packets, and Y is the $k \times 1$ vector which holds the coded payloads, and X is the $k \times 1$ vector eventually holding the original blocks x_i once recovered. We use the notation G_i to indicate the i-th row of G and $G_{i,j}$ to indicate the j-th element of G_i: if $G_i = 0$, $\forall\, i$, we say that row i-th is *empty* and write $G_i = \emptyset$. Fig. 2 shows an example of the G matrix and Y vector for the case of a toy generation of k=6 blocks.

The decoding algorithm operates in two stages, triangularization and diagonalization, as follows.

The triangularization stage is described in pseudo-code as Alg. 1, which is executed each time the node receives a packet $P(g, y)$. The goal in this stage is to progressively fill the matrix G with linearly independent equations that allow one to solve for the unknown X. Fig. 2 (left) shows the case where the node has already processed 4 linearly independent packets and a fifth packet is received. We indicate with g_s the first element of g such that $g_s = 1$ and $g_i = 0 \forall i < s$. If $G_s = \emptyset$, g is placed in the s-th row of G, y is placed in the s-th row of Y and the algorithm terminates. In standard BC, if G_s is not empty and if $g = G_s$, the coding vector are identical and thus P is a duplicate packet which is hence dropped and the algorithm ends. Otherwise an XOR between g and G_s and an XOR between y and Y_s is executed and the algorithm is iterated until the resulting equation is placed into G or it is recognized as linearly dependent on the row of G, i.e. it represents redundant or duplicate information. For example, in Fig. 2 (left), g is such that s = 3: however, G_3 is not empty and a collision takes place.

In [8], we proposed to always swap (G_s, Y_s) with (g, y) due to the particular recombination scheme proposed there. Conversely, in [10] the swap is taken only if the degree of g after the XOR is lower than the degree of G_s, and so to reduce the decoding complexity. The above algorithm progressively inserts the coding vectors in G that is arranged in an upper-triangular *band* matrix, with band equal to W (hence the name of the BC).

The diagonalization stage takes place once k linearly independent packets have been received, i.e. once the rank of G is equal to k. In detail, matrix G is made diagonal via standard backward-substitution, so that the unknown X can be determined. Fig. 2 (right) shows G and Y after the diagonalization: vector Y contains the recovered blocks, i.e. $Y_i = x_i$.

Note that, due to the random combinations at the source and at the nodes, not all the received packets are innovative, i.e. linearly independent from already received, and practically it takes $k' > k$ packets receptions to recover the generation. In the following, we will define the *code overhead* as $\epsilon_c = \frac{k'-k}{k}$, that represents the bandwidth ratio wasted transmitting non innovative packets. Whereas the BC decoding process was originally designed to minimize the decoding complexity, in the following we show that network nodes can exploit it to detect ongoing pollution attacks with minimal additional complexity, plus we discuss the role of the swap in the code efficiency - pollution resilience tradeoff.

Algorithm 1 Packet decoding, Triangularization

```
receive P(g, y).
while true do
    s ← position of leading one of g
    if G_s = ∅ then
        G_s ← g; Y_s ← y
        return
    else
        if g = G_s
            return
        swap (G_s, g); swap (Y_s, y)
        g ← g ⊕ G_s; y ← y ⊕ Y_s
    end if
end while
```

B. The ToroStream Protocol

The network topology is managed by a centralized tracker, which organizes the nodes in a totally random, non acyclic, overlay. Nodes that wish to become part of the streaming session issue a join request to the tracker, which maintains a directory of the nodes in the overlay (but ignores the actual overlay topology). The tracker replies to join requests with a list of addresses of nodes drawn at random from those in the network: after an handshake, two nodes become neighbors and start exchanging coded packets. Due to the way the tracker manages the network, the peer nodes are arranged as an unstructured, non-acyclic, mesh overlay which simplify the management of the network with respect to tree-like overlays. Random overlays are however more challenging than tree-like structures in term of pollution resilience: in fact, a malicious node can potentially spread the pollution to the whole network and not just to the subset of its children peers, while there

is no easy way to pinpoint the source of the pollution to a single nodes in the tree hierarchy. Periodically every node disconnects from a part of its neighborhood and establishes novel neighborhood relationships with other nodes drawn at random by the tracker to which the node issues a request for novel neighbors. Due to the resulting churning rate, the temporal window which a node can exploit to observe its neighbors and attempt to identify malicious nodes is tight, and hence a resilient-by-design architecture becomes essential. For all these reasons, the ToroStream protocol is a suitable and challenging scenario to stress the pollution resiliency capabilities of our architecture designed around BC.

The nodes exchange BC packets according to a totally random push-based scheduling scheme. At each transmission opportunity the server encodes a packet according to the scheme in Sec. II-A (with $W = k$ at start-up) and uploads it to a random peer in the network. The source seeds coded packets for each generation of the stream for an amount of time corresponding to the generation playout duration and then moves to the generation. The peers join the overlay and buffer the video for a finite amount of time. Periodically each peer transmits a random combination of the buffered packets to a neighbor drawn at random. As already recalled, all the recombined packets exchanged by the nodes are $BC(k, W)$. In the present work, all nodes decode the received packets using the Gaussian-based Alg. 1 enhanced with pollution detection capabilities as in Sec. III-B.

Most of the signaling among peers happens implicitly to save signaling bandwidth. In particular each node needs to let its neighbors know which generations have already been decoded. Because a node buffer typically encompasses a few generations of video, the decoding status of a node can be encoded over a vector of few bits (1 bit per generation): a node embeds such vector in any transmitted packet, which increases the feedback rate provided to its neighbors without additional signaling complexity. Every node also broadcasts a message to its neighbors whenever it detects a polluted packet in its input buffer with the indication of the affected generation. Other explicit signaling messages includes join / leave requests and periodic keep-alive messages addressed to specific neighbors and periodic reports issued to the tracker.

III. Proposed Architecture

This section first models the activity of a malicious node that injects bogus coded packets into the network to cripple the communication. Next, we describe the two cornerstones our pollution-resilient NC scheme, namely i) autonomous probabilistic detection of pollution attacks at the network nodes and, ii) coding strategies for the minimization of the probability that a node relays a polluted packet.

A. Pollution Attack and Propagation Model

We define *pollution attack* the injection of polluted packets into the network by one or more *malicious* nodes with the goal to cripple the communication. The polluted packets payload is not bitwise identical to the linear blocks combination in the coding vector because either the payload, the coding vector or both are altered. Altering the payload is however simpler and safer than altering the coding vector from the malicious node perspective. The coded packets degree in BC follows a known distribution and that may allow to identify polluted packets from their coding vectors. So, we assume that every time a malicious node is granted a transmission opportunity it first produces a valid BC packet as described in Sec.II. Next, with probability p_{poll} the malicious node randomly flips the payload bits prior to transmitting the packet to the network. The polluted packet retains the degree distribution imposed by the source, thus it is not straightforward to tell whether a packet is altered or not. However, because each generation is independently encoded and recovered, pollution does not spread across generations.

B. Detection of Pollution Attacks

Network nodes autonomously attempt to detect polluted packets in their input buffers by exploiting non innovative, i.e. linearly dependent, packets which would normally be discarded. Namely, our scheme performs sanity checks between each received packet and the linear combinations of the previously received packets found in the G matrix and in the Y vector, as described for Alg. 1 in the previous section. The algorithm is executed each time a packet $P(g, y)$ is received, i.e. each time a packet is received there may be at least a chance to spot a pollution attack. Fig. 2 (left) illustrates the case where a node receives a polluted packet $P(g, y)$ for a toy generation of k=6 blocks where the rank of G is equal to 4 (i.e., the node is 2 packets away from recovering the generation). Let s be the index of the leading one of g as defined in Sec. II: the received packet in the figure is such that $s = 3$ and and thus it collides with (G_3, Y_3). In this particular case, we have that $g = G_3$, i.e. packet (g, y) is not only linearly dependent on (G_s, Y_s), but it represents the very same linear combination of the original blocks. Whereas packet (g, y) would normally be discarded because useless to decode, we propose to perform a sanity check on the payloads y and Y_s: if $g = G_3$, then also y and Y_3 should be in agreement, i.e. we verify that for $y = Y_3$. Because the payloads y and Y_3 do not match (indicated in the figure with the use of different colors), one of the packets received so far by the node must be polluted. In the above example, the pollution is detected at the first iteration of the algorithm. The algorithm may however iterate a number of times equal to the rank of G, and at each iteration there may be a chance for pollution detection. This scheme makes possible to detect pollution attacks before a generation has been recovered, enabling to take early countermeasures against pollution propagation. Comparing the payloads y and Y_s does not entail any additional complexity other than a bitwise comparison between two vectors of bits. Such scheme is also suitable for verifying whether a generation has been correctly recovered exploiting *late* packets, i.e. packets which arrive after a generation has been recovered. Fig. 2 (right) illustrates the case where a node has completed recovering a generation. Some packets were however polluted and some of the recovered blocks (blocks x_5 and x_6 in the example) have not been decoded correctly. Let us assume the node later

receives the clean packet $P(g, y)$ that, after an XOR with (G_3, Y_3), collides with (G_6, Y_6). At this point the node checks whether $y = Y_6$, and because the payloads do not match, the pollution has been detected. In the following, we assume that every node decodes the received packets with the pollution-aware version of Alg. 1 and raises a pollution flag whenever a mismatch between payloads is discovered.

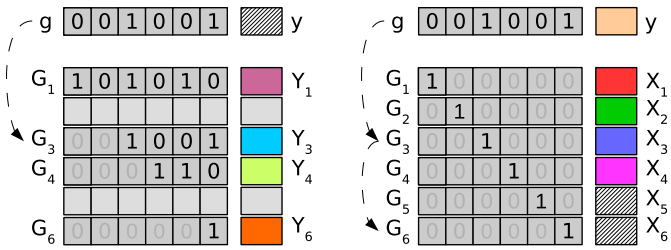

Fig. 2. Example of the G matrix and the Y vector for a toy generation of k=6 blocks. On the left: triangularization stage, two more packets to recover the generation. Right: generation recovered after diagonalization. The figure also shows two example of handling of polluted packets.

C. Adaptive Pollution Propagation Control

A node that detects a polluted packet in its buffer becomes aware of the pollution attack and, as a first countermeasure, stops relaying packets for the polluted generation. However, that is not sufficient to avoid the pollution from spreading over the network because i) the node itself may have buffered polluted packets for other generations without knowing ii) its neighbors may have buffered polluted packets without knowing. As a consequence we devise further actions that the node must undertaken as follows:

i) the node that has detected pollution starts a timer which marks the beginning of the so-called *backoff window*, i.e. a time window t_{back} seconds long. During the backoff window the node adjusts its own coding window W according to the criteria discussed below to reduce the probability to relay polluted packets. The backoff window duration t_{back} contributes to drive the tradeoff between pollution resilience and coding overhead, as we experimentally demonstrated in our preliminary work [7]. Short t_{back} values minimize the impact on coding efficiency, however the node may return to normal operations before the pollution attack is over. Conversely, large t_{back} values are apt to long-lasting pollution attacks, however the coding efficiency may be unnecessarily penalized. In our previous work, we experimentally demonstrated that a backoff window of about t_{back} = 1 s enables a reasonable tradeoff between pollution resilience and coding efficiency. Notice that if the node detects a further pollution attack after the timer has been started, the timer is reset and the backoff window is practically extended. That is, repeated detection of pollution attacks may practically extend the protection over a pollution attack that lasts in time despite a limited t_{back}.

ii) the node that has detected attack broadcasts a warning to its neighbors, suggesting that a pollution attack is going on. A neighbor which receives a pollution warning starts the timer which marks the beginning of its own backoff window. This reaction aims at reducing the propagation of pollution in the neighborhood.

Fig. 3 (left) illustrates the pollution detection and reaction process: each time a node receives a packet, the decoding algorithm may flag a generation as polluted. Therefore, the node starts the timer of its backoff window and broadcasts the pollution warning message to its neighbors.

Each time a node is allocated a transmission opportunity, it checks if its own backoff window timer has been started, as illustrated in Fig. 3 (right). If the backoff window has not been started, the node operates normally, recombining the received packets for some generation from a recombination window of size $W = k$, as in classical NC. Otherwise, the node draws the packets to recombine from a narrower window of size, for example, $W = k/n, n \geq 1$, i.e. $W \leq N$, so to reduce the probability that it relays a polluted packet to its neighbors. The rationale behind recombining from a narrower window as a reaction to a pollution attack is that the narrower W, the fewer packets in the buffer fit into the drawn window of size W and are suitable for recombination, so the lower the chances that any of the polluted packets is drawn for recombination. The W value drives the tradeoff between pollution resilience and coding efficiency of BC. Small W values offer better protection against pollution propagation but will increase the coding overhead. Conversely, large W have minor impact on the coding efficiency but may offer insufficient protection against pollution attacks. Choosing the optimal W is hence constrained by the tradeoff between the desired protection against pollution and the coding overhead the network can tolerate. However, the formulation of the optimal W depends on several factors, among which the number of packets in the buffer, the probability that a buffered packet is polluted and the number of malicious neighbors, and the number of malicious nodes in the network, as the model in [6] shows. Due to the numerous variables and constrains on which the optimal W depends, we leave its analytical formulation out of the scope of this work and in the experimental section we explore the different tradeoffs between pollution resilience and network utilization effectiveness that our BC-based approach enables. straining the propagation of the pollution.

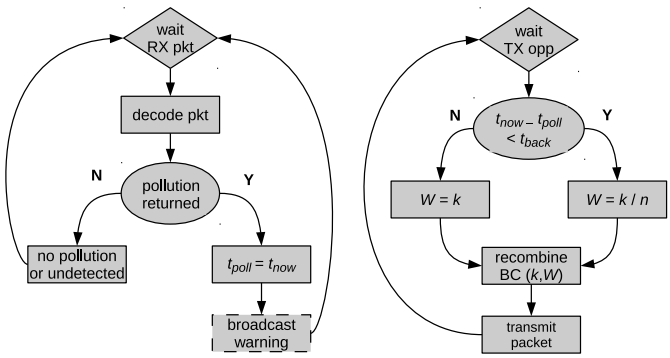

Fig. 3. The proposed architecture based on BC. Left: detection of pollution attacks via Alg.1, start of the backoff timer and warning broadcast. Right: the coding window W is adjusted at each transmission opportunity depending on the time elapsed since the last time a pollution is detected or a warning is received.

D. Age-Based Packet Recombination

Finally, we describe the pollution resilient packet recombination strategy we proposed in [6] that we will use as reference in some of the experiments in Sec. V. In [6] we do not exploit BC, rather we show that packets received earlier are less likely to be polluted and thus are better candidates for recombination. Let us assume that a network node has collected R packets $\{P^1, \ldots, P^R\}$ into its input buffer, so that P^i was received prior to P^{i+1}. When a node is granted a transmission opportunity, each i-th packet in the input buffer is drawn for recombination according to a weighting function $p_r(i)$ decreasing with i. In [6] we show that such scheme represents a simple yet effective countermeasure to the propagation of the pollution, since it reduces the probability that a honest relays polluted packets. The particular weighting function drives the tradeoff between protection against pollution and coding overhead (for example, in [6] we experimented with a negative exponential weighting function). In this work we propose the following linear weighting function:

$$p_r(i) = \frac{1}{\beta}\frac{R-i}{R} + (1-\frac{1}{\beta})\frac{1}{2}. \quad (1)$$

The parameter $\beta \in [1, \infty)$ controls the age sensitivity of the queued packets. For β=1 packets received earlier are drawn for recombination with probability close to 1, whereas packets received later are drawn with a probability close to 0. Conversely, for $\beta = \infty$ we have $\forall i, p_r(i) = \frac{1}{2}$, i.e. each packet is drawn for recombination with identical probability as in classic random NC. It is worth pointing out that the age-based strategy and the strategy based on BC coding window control are completely orthogonal. In other words, it is possible to draw the packets to recombine from a narrower window and then to recombine the packets taking age into account to further increase the resilience to pollution attacks, as we experimentally verify in the following.

Finally, we would like to recall that also the swap step of the decoder (line 10 of Alg.1) plays a role in terms of pollution resilience. Since packets received later on are more likely to be polluted, the swap may replace a clean packet with a polluted packet arrived later. For this reason, in the experimental section we also investigate the possibility to avoid swapping to increase the resilience against pollution.

IV. Experimental Setup

We experiment streaming a 10 minutes video sequence encoded with H.264/AVC at 500 kbit/s to a population of N=1000 peer nodes comprising N_m=20 malicious nodes attempting to prevent the remaining N_h=980 honest nodes ($N_m + N_h = N$) from recovering the video. The JM 18.6 H.264/AVC encoder has been used with a GOP size of 15 frames and IBBP coding scheme; using 30 Hz video frame rate, the selected GOP structure corresponds to independent coding units of duration of 0.5 s. The source seeds the video stream and its output bandwidth is capped at 25 Mbit/s, i.e. the server has bandwidth to serve our 500 kbit/s test video to 50 peer nodes only.

The nodes cooperate to distribute the video stream using the random-push P2P protocol described in section II-B. The neighborhood of each node is constrained to a maximum of N_s=25 peers: periodically, each node drops some of its neighbors at random and creates novel neighborhood relationships to guarantee enough peer churning. Each node joins the network and then buffers the video streams for t_b=5 seconds, i.e. the playback at the nodes is synchronized but lags 5 second behind the server seeding point. Finally, the output bandwidth of the peers is upper bounded to 1 Mbit/s, i.e. each node has enough bandwidth to relay the video to at least another peer.

The pollution attack follows an off-on-off pattern as follows. At time t = 0, the honest nodes join the network in short sequence. After 150 seconds, malicious nodes join the network, mix with the other nodes and start relying packets which are purposely polluted with probability p_{poll} = 0.01, i.e. each packet transmitted by a malicious node is polluted with 1% probability. After 4 minutes of activity, the malicious nodes leave the network, the attack ends and only the honest remain for the rest of the streaming session. We define the 4 minutes interval during which malicious nodes are active as *pollution interval*: because the video is divided in generations which are independently coded, the effect of the pollution attack is constrained within such time interval. While the on-off pattern is instrumental in studying how the network reacts to a the arrival and the departure of the attacker, we would like to point out that our results hold also for other attack patterns from a qualitative perspective.

We experiment with different packet recombination schemes at the network nodes and measure the effect on the pollution attack. The first scheme, *Reference*, is a baseline strategy corresponding to classic Random NC, where the source node encodes random combinations of the original blocks and the network nodes forward random combinations of the received packets. The *Proposed* strategy is our BC-based scheme, which exploits BC to detect ongoing pollution attacks and adaptively reduce the coding window size W when a pollution attack is detected. The backoff window is set to 1 s because our previous experiments showed [7] that this value enable a reasonable tradeoff between protection against pollution attacks and code efficiency.

The pollution attack impact on the video quality is principally measured in terms of Continuity Index (CI), which corresponds, for each generation, to the fraction of nodes which could successfully recover that generation. Namely, a node successfully recovers a generation if it is able to collect enough innovative packets within the generation decoding deadline and none of the packets used for recovery is polluted. We also measure the impact of the attack from a network perspective logging the ratio between the polluted and correct packets flowing across the overlay; we have defined such ratio as pollution overhead ϵ_p.

V. Experimental Results

A. Effects of the Pollution Attack

Fig. 4 shows the CI over the streaming session for the *Reference* NC scheme and for generations of k=50 blocks, as measured at a random sample node in the network. Before

the pollution attack starts, the CI is obviously equal to 1 because all the peer nodes recover the video correctly. When the pollution attack starts, the CI drops to about 41%, i.e. the node recovers correctly only 4 generations out of 10. Finally, when the attack ends, the node is again able to correctly recover the video and the CI returns to 1. The figure also shows the PSNR measured between the original sequence and the sequence as recovered by the same node (each point on the graph corresponds to the PSNR averaged over one generation of frames, lost frames were concealed by frame copy). The *Foreman* sequence (CIF resolution, 30 fps) was used in this experiment. The comparison between the two figures shows that the pollution attack yields a dramatic impairment of the video quality (the impairment consists in frame freezes on the screen which may last several seconds) that yield a PSNR as low as 10 dB.

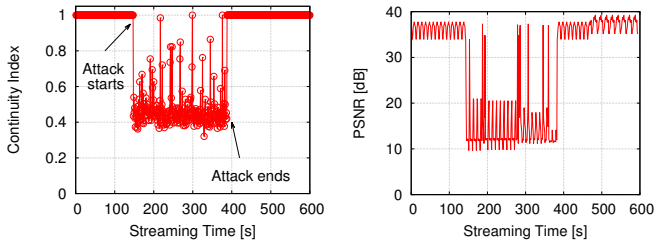

Fig. 4. Continuity Index averaged among all the honest nodes (top) and PSNR measured at a single honest node (bottom) for the *Reference* scheme: the large drop in visual quality is due to the pollution attack.

B. Pollution Detection Strategy

Next, we investigate the performance of our proposed pollution detection scheme. We recall that each node performs a consistency check between coding vectors and payloads each time a packet is received: a mismatch between coding vectors and coded payloads reveals a pollution attack. We consider different pollution detection strategies. In particular, we defined *Early* detection the one based only on packets received before the generation decoding and *Late* detection the opposite case when only the packets received after decoding are checked for pollution. We use the label *Early+Late* when detection exploits all packets. Finally, we termed as *Oracle*, an ideal detection mechanism that verifies the checksum of each recovered generation with a trusted server. The performance of the pollution detection strategies are measured in terms of *recall* and *precision*. We define the recall as the ratio of polluted generations that are correctly recognized as polluted: the higher the recall, the higher the likelihood that a pollution attack is detected. We define the precision as the ratio of generations flagged as polluted that are actually polluted: the higher the precision, the lower the number of false positives. The *Early* detection obviously enjoys ideal precision equal to 1, since no false positives can be revealed by the proposed algorithm. On the contrary with the *Late* and *Early+Late* schemes, a node may receive k' clean packets and correctly recover the generation; then one or more polluted late packets may be received and hence the generation may be incorrectly flagged as polluted, i.e. false positive case. Figure 5 shows the recall (left) and precision (right) as a function of the ratio between generation size and coding window k/W. The Early+Late strategy yields the better recall, with about 60% of the polluted generations detected as such, however its recall is only marginally better than the recall of the Early scheme. Figure 5-b shows that the precision of the *Early+Late* and *Late* strategies are significantly lower than 1, i.e. several generations are incorrectly detected as polluted (high number of false positives). As late packets are more likely to be polluted, they are in fact poor candidates for checking the received packets sanity [6]. The figures show that as W decreases, the ability to detect pollution attacks drops. Each time Alg. 1 iterates and a collision happens, pollution attacks may be detected. The higher the number of iterations, the more likely that pollution attacks are detected. The expected number of iterations depends on the number of collisions, which depends on the rank of G and the average degree of the received packet [8]. Because the average packet degree is equal to $\frac{W}{2}$, reducing W reduces the number of collisions and hence the chances to detect pollution attacks. However, as W decreases the resilience to pollution attacks improves and sets off the decreased ability to detect pollution attacks, as we show later on. The figure shows on the right the cumulative probability that a pollution attack is detected as a function of the n-th received packet (the dashed vertical line represents the average number of packets k'=51.7 after which a generation is recovered). It can be noted that, as the rank of G approaches k, it is very likely to detect pollution attacks and to stop the pollution from propagating.

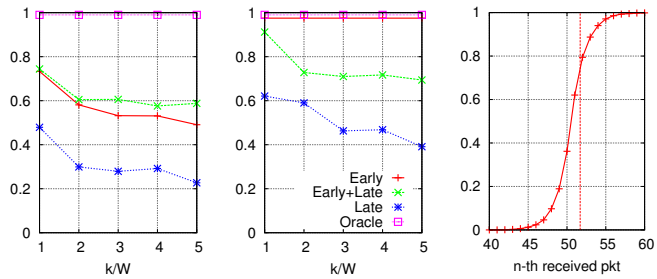

Fig. 5. Recall (left), precision (center) and cumulative distribution of detected pollution attacks (right) of three different pollution detection strategies as a function of the coding window size W.

C. Video Quality and Pollution Propagation

In this section we sweep the parameters characterizing BC, namely the generation size k and the coding windows size W, and we assess the relative impact on multiple performance metrics.

Figure 6 shows the average CI, the code overhead ϵ_c as the ratio of output bandwidth wasted transmitting non-innovative packets and the decoding complexity in terms of number of XOR between coded packets required to recover 1 Mbit of coded data as a function of $\frac{k}{W}$ for several values of k. Notice that the case where $\frac{k}{W} = 1$, i.e., $W = k$, corresponds to a classic NC scheme where the nodes recombine the packets at random. whereas larger values of $\frac{k}{W}$ represent BC with

decreasing window sizes.

The graph on the left shows that for classic NC the CI is close to 0 for k = 50, i.e. the malicious nodes succeeded in disrupting the video stream and the honest nodes recover only bogus data. Moreover, decreasing k, i.e. using shorter generations, classic NC yields a CI of about 50%; in other words, shorter generations provide a limited resilience to pollution and are not enough to nullify the effects of the attack. Conversely, using BC with a narrow coding window one is able to improve the CI well above 50%: as an example for k=20 and $\frac{k}{W} = 4$, i.e. $W = \frac{k}{4}$), the CI soars to over 90%. The lesson learned is that adjusting the BC parameters is enough to set off the effect of the attack in terms of received video quality.

The central graph shows that BC coding efficiency, and as a consequence network bandwidth usage, depend on parameters W and k. The figure shows that as W shrinks, the coding overhead increases: when $\frac{k}{W} = 4$ about one packet out of two is not innovative, wasting a significant amount of the output bandwidth available at the nodes. In a bandwidth-constrained scenario the peers may not even have enough capacity to tolerate such a large overhead to supply the video.

Finally, the right graph shows that reducing the generation size k and the coding window size W brings a further benefit in terms of lower decoding complexity. For k=25 and $\frac{k}{W} = 2$ the decoding complexity drops to about 600 XORs per Mbit, versus 1200 XORs per second for classic NC (k=50 and $W = k$). Notice that the scheme proposed in our previous work [6] exhibits comparable video quality in similar experimental conditions. However, its complexity is identical to that of classic NC as it was not based on BC and thus it did not enjoy controlled decoding complexity.

These experiments demonstrate that our BC-based architecture achieves its primary goal of pollution resilience, plus it enables controlled decoding complexity as an additional benefit.

Albeit the optimal coding setup depends on several network parameters, the above experiments suggest that in our setup the coding parameters $k = 25$ and $W = \frac{k}{2}$ offer a reasonable overall tradeoff between resilience to pollution, decoding complexity and coding efficiency. in the rest of our experiments we will consider such coding setup to deal with pollution attacks whenever a node operates within its backoff window: i.e., when a node operates within the backoff window and a transmission opportunity arises, the node draws the packets to recombine from a window of size $W = \frac{k}{2}$, where $k = 25$.

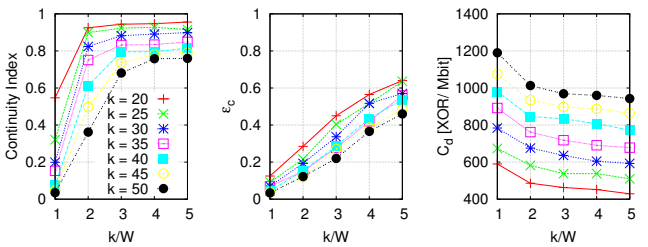

Fig. 6. Continuity index (left), coding overhead ϵ_c (center) and decoding complexity C_d (right) as a function of the generation size k and coding window size W.

In Fig. 7 we provide a better insight on the effects of BC coding parameters in terms of pollution propagation.

The left graph shows the probability that the i-th packet received by a node is polluted over a generation streaming time for different coding window sizes W. In classic NC, random recombinations increase by a 1000-times factor the probability that a honest node relays polluted packets. Conversely, with BC such probability increases at a far lower rate, constraining the pollution propagation through the network.

The central graph shows the probability P_p that an honest node relays a packet which is polluted. It can be noted that for $k = 50$ and $\frac{k}{W} = 1$ almost 50% of the recombined packets transmitted by the honest nodes are polluted. Hence, in classic NC the recombinations propagate the pollution and completely disrupt the communication (the corresponding CI in the previous figure is in fact close to 0). As k drops, P_p decreases and the corresponding CI increases; narrowing also the coding window W further avoids the propagation of the pollution and improves the video quality. For $k = 25$ and $\frac{k}{W} = 2$ P_p drops to about 0.5% (a 100-fold reduction with respect to the classic NC scheme) and the corresponding CI soars over 90%.

The graph on the right of the same figure shows the cumulative probability over time that a sample honest node relays at least one polluted packet for different values of W ($k = 25$). When $W = k$, after only 2 seconds since the pollution attack has begun, 80% of the honest nodes have relayed at least one polluted packet, i.e. almost all honest nodes have contributed to propagate the pollution. However, by reducing the coding window with $\frac{k}{W} = 2$ it takes 22 seconds before 80% of the nodes have transmitted at least one polluted packet, and this figure further improves to 56 seconds for $\frac{k}{W} = 4$.

Hence, adjusting the coding window size W reduces the likelihood that a honest node spread the pollution and improves the video quality.

D. Matrix G Refresh Policy

In this section we experiment three different policies for refreshing the rows of the decoding matrix G and we assess the policy impact on the decoding complexity and resulting video quality. We recall that whenever a node receives a packet and the coding vector is such that it collides with one row of G, such collision may be addressed with a swap at line 10 of Alg. 1: here we experiment with three distinct policies. The first policy, termed *Never*, is such that the received packet is XOR'ed with a row of G and simply processed again; in this case a packet received later has no chance to replace a packet received earlier. The second policy, termed *Degree*, is such that the degree of received packet is compared with the degree of the G row: if the degree of the received packet is lower, a swap is performed and the algorithm iterates. In this case, a packet received later on will occasionally replace a packet received earlier on. The third and last policy, termed *Always*, is such that the newly received packet always takes the place of the G row: in this case, packets received later on may replace packets received earlier in the matrix G and vector Y. Fig. 8 shows the probability that a row of G is polluted, the corresponding continuity index and the actual

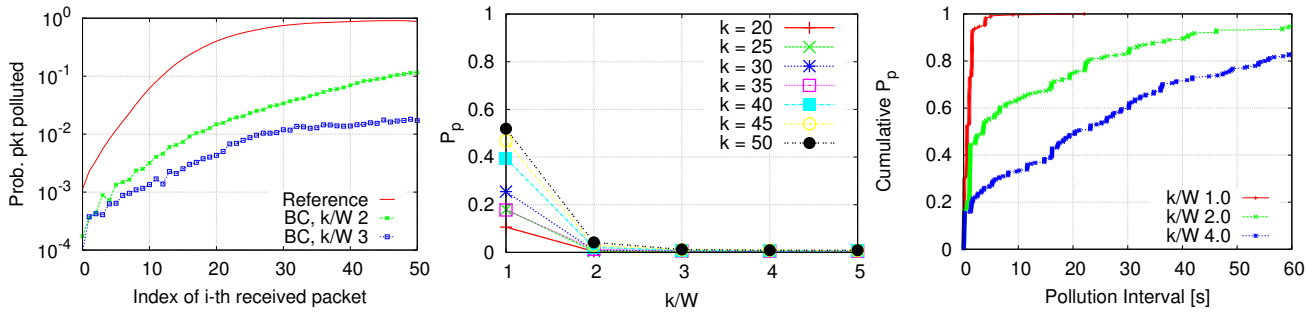

Fig. 7. Probability that a node receives a polluted packet over time (left), Probability that a node relays a polluted packet P_p (center) and cumulative probability that a node relays a polluted packet over the pollution interval (right).

decoding complexity measured as number of XOR operations required to recover a generation of size k=25 blocks. We can see that the *Always* policy yields the highest probability that a row of G is polluted (left) and thus the lowest CI (center): this is because packets received later are more likely to be polluted. The *Degree* policy yields better results in terms of probability to pollute matrix G and relative impact on the video quality: packets with lower degree are generated by nodes who recombine from a smaller coding window, thus they may be better candidates for swapping. Finally, the *Never* policy yields the best results: packets received earlier are less likely to be polluted, thus they are the best candidates to fill the matrix and shall not be replaced by packets received later on. Complexity wise, we see that the *Degree* policy shows the lowest complexity since the swap limits the degree of the rows of G and as a consequence the number of XORs required by the decoding procedure. Therefore, avoiding the swap at line 10 of Alg. 1 yields best video quality, whereas performing the swap on a lowest-degree basis yields the best tradeoff between video quality and decoding complexity.

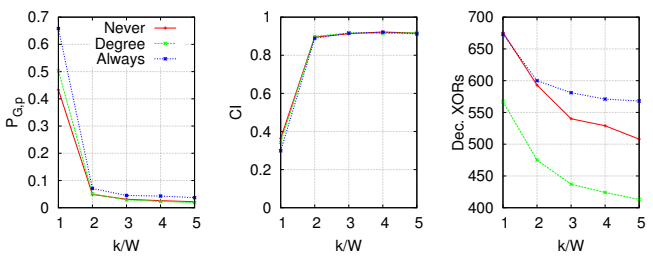

Fig. 8. Probability that at least one row of the G matrix polluted (left), continuity index (center) and decoding complexity (right) for three different swap strategies.

E. Increased Polluters Activity

We explore now the case where malicious nodes exhibit an increasingly aggressive behavior, not withstanding the possibility of revealing their identity. Namely, we evaluate the effect of a five-fold increase of the malicious nodes activity, both in terms of an increase in the number N_m of malicious nodes and in the probability that a malicious node relays a polluted packets p_{poll}. We consider four different packet recombination strategies with different levels of resilience against pollution attacks. The *Reference* strategy is our baseline scheme where the nodes are unaware of pollution attacks and relay random linear packet combinations as in classic NC. The *Proposed* strategy is our BC-based scheme, where the nodes react to the detection of a pollution attacks by recombining from a window of size $W = k/2$. The *Time* strategy where the packets are recombined with a probability that depends on their age according to the time-sensitivity function in Eq. (1) using $\beta = 2$. Finally, the *Joint* strategy explores the combined potential of BC and the age-based recombination strategy. Whenever a pollution attack is detected, the nodes start recombining from a narrower window of size $W = k/2$ plus they draw for recombination each i-th received packet according to Eq. (1) with $\beta = 4$.

In Fig. 9 we show CI, ϵ_p and ϵ_c when we progressively increase N_m from 0 to 100 nodes, while the total number of nodes N = 1000 is kept constant and p_{poll}=0.01, i.e. we increase the ratio of malicious nodes in the network from 0 to 10% while we do not chance the overlay size. As N_m increases, with the *Reference* strategy the pollution overhead ϵ_p quickly rises (up to 36% for $N_m = 100$) and so the video quality quickly drops to 0. The *Proposed* and *Time* strategies show similar CI and can cope with a larger amount of malicious nodes activity, however the video quality starts to drop when the number of malicious nodes exceeds 50, i.e. 5% of the total. Despite the *Time* strategy shows lower pollution overhead than the *Proposed* strategy, both yield pretty similar video quality. In fact, the *Time* code overhead is higher that of *Proposed*: because more packets are required to recover a generation, the higher the probability that at least one of the received packets is polluted and the lower the video quality. Hence, despite the *Proposed* strategy yields slightly lower resilience to pollution attacks than *Time*, its better coding efficiency allows to achieve a good tradeoff in terms of the resulting video quality. Finally, the *Joint* strategy yields the best results as the system can tolerate up to 10% of malicious nodes with little impairment in the video quality: such result is achieved by balancing the tradeoff between coding efficiency and pollution resilience.

Fig. 10 illustrates the complementary experiment where we increase the probability that a malicious node transmits a polluted packet p_{poll} from 0 to 20% (the number of malicious nodes is fixed at N_m=20). As in the previous experiment,

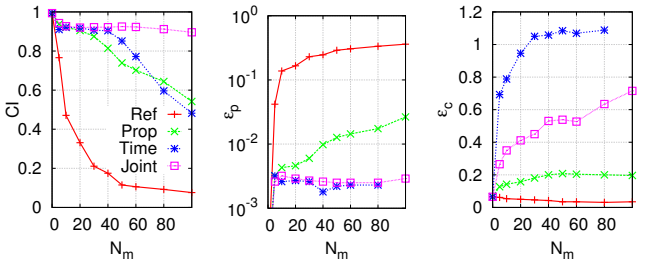

Fig. 9. Video quality (left), pollution overhead ϵ_p (center) and code overhead ϵ_c (right) as a function of the number of the number of malicious nodes N_m (over a total of 1000 nodes in the network).

the *Reference* scheme shows little resilience to increasingly aggressive polluters: pollution probabilities below 5% cripple the communication. The *Proposed* and *Time* schemes show improved resilience to pollution: the malicious nodes must pollute more than 5% of the transmitted packets before the video quality degrades. However, as a malicious node transmits more polluted packets, it exposes itself to the risk of being identified end isolated from the network, making harsh pollution attack detrimental for the attacker itself. As an example in [11] a polluter detection and identification mechanism is proposed where it can be seen that identifying polluters turns to be easier when they pollute all packets they upload. Similar behaviors are reported by [12], [13] for another identification mechanism. These remarks make clear that polluters are not interested in injecting a large amount of fake data but only the minimum possible to impair the system performance without increasing too much the risk of being identified.

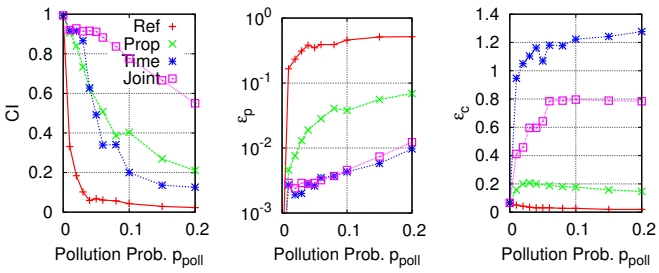

Fig. 10. Video quality (left) pollution overhead ϵ_p (center) and code overhead (right) as a function of the probability p_{poll} that a malicious node pollutes a transmitted packet.

VI. Related Works

BC have been recently proposed in [8] to obtain energy efficient network coding for P2P mobile video streaming. In [8] BC have been defined and characterized with respect to the trade-off between encoding overhead and decoding complexity. Video streaming experiments with real mobile devices showed that BC can halve the decoding complexity and extend the device operational lifetime with respect to traditional NC with just a negligible increase in coding overhead. In our previous work [7], we have worked out a preliminary analysis of BC in a scenario with malicious nodes and we have observed that BC can effectively mitigate the propagation of the pollution.

Besides BC several related works have been carried out for standard random NC. In this context, in [6] a technique able to mitigate the effects of pollution has been proposed: a recombination scheme where nodes draw packets to be recombined according to their age in the input queue is devised and its effectiveness is experimented in a practical streaming scenario.

Other approaches in the literature have proposed techniques for on-the-fly verification of the received packets so as to detect and identify the malicious peers. For instance the readers can refer to [14], [15], [16], [17], [18], [19], [20] as important contributions in this area, that proposed either cryptographic or algebraic approaches. The major drawback of these methods is the high computational costs for verification and the communication overhead due to pre-distribution of verification information. Pre-distribution of verification keys is particularly critical in case of live streaming where novel data are being forwarded at a high rate.

Other approaches, e.g. [21], [22], [23], have focused on error correction to deal with pollution attacks; these methods introduce coding redundancy to allow receivers to correct errors but their effectiveness depends on the amount of corrupted information.

An orthogonal line of research deals only with identification of malicious nodes. In this context [11] proposed a fully distributed detection algorithm based on a stochastic method that uses intersection operations to progressively isolate malicious neighbors of a peer. This method works best in a scenario with a static neighborhood. Alternatively, in [12], [13] malicious nodes identification was cast as a statistical inference problem designed to infer the probability of neighboring peers being malicious.

VII. Conclusions and Future Works

This work leverages Band Codes (BC), a family of rateless codes designed for low-complexity network coding, to introduce resilience against pollution attacks in P2P networks. First, we showed how to exploit the BC decoding process to detect ongoing pollution attacks on a probabilistic, distributed, basis before the message is recovered, enabling the nodes to adopt early countermeasures. Second and foremost, we adaptively adjust the BC coding parameters, reducing the probability that honest nodes relay polluted packets and the number of polluted packets flowing through the network. Real P2P video streaming experiments on a thousand nodes scale show that our BC-based scheme improves the video quality over classic NC, whereas controlled decoding complexity is preserved as an additional benefit beside pollution resilience.

References

[1] X. Zhang, J. Liu, B. Li, and T.S.P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *proceedings of IEEE Infocom*. Citeseer, 2005, vol. 3, pp. 13–17.

[2] G Huang, "PPLive: A practical P2P live system with huge amount of users," in *Proceedings of the ACM SIGCOMM Workshop on Peer-to-Peer Streaming and IPTV Workshop*, 2007.

[3] M. Grangetto, R. Gaeta, and M. Sereno, "Rateless codes network coding for simple and efficient P2P video streaming," in *IEEE International Conference on Multimedia and Expo, 2009 (ICME 2009).*

[4] M. Wang and B. Li, "Network coding in live peer-to-peer streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1554–1567, Dec 2007.

[5] P. Dhungel, X. Hei, K.W. Ross, and N. Saxena, "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, P2P-TV '07*, 2007, pp. 323–328.

[6] Attilio Fiandrotti, Rossano Gaeta, and Marco Grangetto, "Simple countermeasures to mitigate the effect of pollution attack in network coding-based peer-to-peer live streaming," *Multimedia, IEEE Transactions on*, vol. 17, no. 4, pp. 562–573, 2015.

[7] Attilio Fiandrotti, Rossano Gaeta, and Marco Grangetto, "Pollution-resilient peer-to-peer video streaming with band codes," in *Multimedia and Expo (ICME), 2015 IEEE International Conference on*. IEEE, 2015.

[8] A. Fiandrotti, V. Bioglio, M. Grangetto, R Gaeta, and E. Magli, "Band codes for energy-efficient network coding with application to P2P mobile streaming," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 521 – 532, February 2014.

[9] A. Fiandrotti, A. M. Sheikh, and E. Magli, "Towards a p2p videoconferencing system based on low-delay network coding," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012, pp. 1529–1533.

[10] V. Bioglio, R. Gaeta, M. Grangetto, and M. Sereno, "On the fly gaussian elimination for lt codes," *IEEE Communication Letters*, vol. 13, pp. 953–955, 2009.

[11] Y. Li and J.C.S. Lui, "Stochastic analysis of a randomized detection algorithm for pollution attack in P2P live streaming systems," *Performance Evaluation*, vol. 67, no. 11, pp. 1273 – 1288, 2010.

[12] R. Gaeta and M. Grangetto, "Identification of malicious nodes in peer-to-peer streaming: A belief propagation-based technique," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 10, pp. 1994–2003, 2013.

[13] R. Gaeta, M. Grangetto, and L. Bovio, "DIP: Distributed identification of polluters in P2P live streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 10, no. 3, pp. 24, 2014.

[14] M. N. Krohn, M. J. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," *Security and Privacy, IEEE Symposium on*, 2004.

[15] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *IEEE INFOCOM*, 2006.

[16] Q. Li, D.-M. Chiu, and J.C.S. Lui, "On the practical and security issues of batch content distribution via network coding," in *Network Protocols, 2006. ICNP '06. Proceedings of the 2006 14th IEEE International Conference on*, 2006.

[17] D.C. Kamal, D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *In Proceedings of the fortieth annual Conference on Information Sciences and Systems*, 2006.

[18] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient signature-based scheme for securing network coding against pollution attacks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008.

[19] E. Kehdi and Baochun Li, "Null keys: Limiting malicious attacks via null space properties of network coding," in *INFOCOM 2009, IEEE*.

[20] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing xor network coding against pollution attacks," in *INFOCOM 2009, IEEE*.

[21] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D.R. Karger, "Byzantine modification detection in multicast networks with random network coding," *Information Theory, IEEE Transactions on*, vol. 54, no. 6, pp. 2798 –2803, june 2008.

[22] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of byzantine adversaries," *Information Theory, IEEE Transactions on*, vol. 54, no. 6, pp. 2596 –2603, june 2008.

[23] R. Koetter and F.R. Kschischang, "Coding for errors and erasures in random network coding," *Information Theory, IEEE Transactions on*, vol. 54, no. 8, pp. 3579 –3591, august 2008.