IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

Securing Network Coding Architectures against Pollution Attacks with Band Codes

This is the author's manuscript

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/1676752 since 2018-09-18T16:48:48Z

Published version:

DOI:10.1109/TIFS.2018.2859583

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

13 December 2025

Securing Network Coding Architectures against Pollution Attacks with Band Codes

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

***Abstract*—During a pollution attack, *malicious* nodes purposely transmit bogus data to the *honest* nodes to cripple the communication. Securing the communication requires identifying and isolating the malicious nodes. However, in Network Coding (NC) architectures, random recombinations at the nodes increase the probability that *honest* nodes relay polluted packets. So, discriminating between honest and malicious nodes to isolate the latter turns out to be challenging at best. Band Codes (BC) are a family of rateless codes whose *coding window* size can be adjusted to reduce the probability that honest nodes relay polluted packets. We leverage such property to design a distributed scheme for identifying the malicious nodes in the network. Each node counts the number of times each neighbor has been involved in cases of polluted data reception and exchanges such counts with its neighbor nodes. Then, each node computes for each neighbor a discriminative *honest score* estimating the probability that the neighbor relays clean packets. We model such probability as a function of the BC coding window size, showing its impact on the accuracy and effectiveness of our distributed blacklisting scheme. We experiment distributing a live video feed in a P2P NC system, verifying the accuracy of our model and showing that our scheme allows to secure the network against pollution attacks recovering near pre-attack video quality.**

***Index Terms*—Network coding, pollution attacks, secure video communications, peer to peer, distributed scheme**

I. Introduction

NETWORK Coding (NC) [1] has received a lot of attention lately because it increases the effective throughput of a network. Moreover, random NC enables totally-push packet scheduling schemes that simplifies the deployment of peer-to-peer (P2P) content distribution architectures [2]. In particular, content providers such as YouTube and NetFlix already generate about 30% of the overall Internet traffic [3], a figure expected to double by the end of the decade. Therefore, NC-based P2P architectures are particularly appealing to the end of distributing video contents to large user populations by leveraging the users' resources without deploying ad-hoc infrastructures [4]. In a random NC-based video streaming architecture, the video stream is subdivided into independently encoded chunks called *generations*, where each generation is further partitioned in k *blocks* of identical size. One source node holds the original video and, for each generation, relays random linear block combinations to the network nodes as coded packets. The network nodes buffer the received packets and relay random linear combinations thereof to their peer nodes. Once a node has collected enough packets, it recovers the generation solving a system of equations.

NC-based architectures are particularly vulnerable to *pollution attacks*, where one or more *malicious* nodes attempt to cripple the communication by injecting bogus data (polluted packets) in the network. The recombinations at the nodes increase the probability that an honest node relays a polluted packet: if *any* of the recombined packets is polluted, the recombined packet is polluted as well. With respect to other applications, media distribution can tolerate the loss of a few coding units, so perfect security of the communication may not be necessary. However, malicious nodes may be able to cripple the communication polluting just injecting a few packets due to the recombinations in the network. Such evidence calls for schemes suitable to distribute media contents with a sufficient level of security against pollution attacks.

Traditional countermeasures to pollution attacks generally build upon malicious nodes identification and blacklisting. First, malicious nodes are identified, for example via ad-hoc coding schemes or cryptographic approaches, e.g., [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. Second, malicious nodes are inhibited from further polluting the network via, for example, blacklisting by a central authority. With random NC, it is challenging to tell whether the source of a polluted packet is a malicious node that polluted it on purpose or an honest node that accidentally relayed a polluted packet (assuming that polluted packets could be identified in the first place). While existing solutions to pollution attacks in NC are discussed in Sec. II, many of them fall short with respect to designing a video distribution scheme secure to pollution attacks. For example, cryptographic and coding complexity represent a drawback when dealing with mobile users. Similarly, the requirements for a central authority become problematic when distributing video contents to a large users population.

In our previous work [16] we leveraged Band Codes (BC) [17] to achieve resilience to pollution attacks of limited intensity by controlling the coding parameters. BCs are a family of rateless codes originally devised to control the decoding complexity as a function of the so-called *coding window* size. With BC, the recombinations at the nodes are constrained to blocks falling within a random window of size $W \leq k$ (for $W = k$, BC resolve to random NC). We showed that the probability that a node accidentally recombines a polluted packet decreases with W. Thus, under the assumption of a limited intensity pollution attack, BC enable the nodes to decode clean generations without even identifying the malicious nodes. However, if malicious nodes inject more

Manuscript submitted Jan. 27, 2018; revised May, 19 and July 2, 2018.

A. Fiandrotti is with Télécom Paristech, Images Données Signals department, 75013 Paris, France (e-mail: attilio.fiandrotti@telecom-paristech.fr)

R. Gaeta and M. Grangetto are with Dipartimento di Informatica, Università di Torino, 10129 Torino, ITALY (e-mail: rossano.gaeta@unito.it, marco.grangetto@unito.it)

polluted packets, the honest nodes relay enough polluted packets to make decoding of clean generation almost impossible. Such shortcoming prompted us to investigate a solution for identifying and blacklisting malicious nodes suitable for the case of severe pollution intensity.

The original contribution of this work is a distributed scheme to identify and blacklist malicious nodes in video communications leveraging on multiple properties of Band Codes (BC). By comparison, the goal of [16] was to address the pollution attacks without identifying the malicious nodes. Each node computes a *honest score* for each of its neighbors as the ratio of clean packets received by the neighbor over all packets received by the neighbor. Therefore, honest scores lie in the $[0, 1]$ interval for each neighbor and neighbors that score closer to 1 are more likely to be honest. Let us associate each node in the network with a Bernoulli random variable whose parameter is s_h (s_p) for honest (malicious) nodes, respectively. The values s_h and s_p correspond to the probability that all the packets provided to the node by honest and malicious neighbors enable the node to correctly recover a pollution-less generation. With this in mind, we show that honest scores are actually estimates for the parameters s_h and s_p, that each node can compute for all its neighbors. We also analytically model s_h and s_p as a function of the BC coding window W. The model shows that $s_h > s_p$ and that both tend to 1 as the coding window size W of BC decreases, i.e. $W < k$. That is, with BC the honest score estimator converges to the expected values s_h and s_p faster than with random NC, i.e. $W = k$. Therefore, as the number of observations available at a node increases, the node estimates s_h and s_p with increased accuracy.
In detail, the key highlights of the scheme we propose in this work are:

- it is totally distributed, so no central authority is needed;
- it is lightweight, since it requires no cryptographic computations and enjoys the low decoding complexity of BC;
- its performance depends only on BC coding parameters (i.e. generation size k and coding window size W);
- it is suitable for realtime communications to mobile devices;
- its performance are described by an accurate and validated mathematical model;
- its expected performance is thoroughly validated in a fully fledged P2P protocol;
- its ability to identify malicious nodes does not decrease as the malicious nodes inject more polluted packets into the network for the values of pollution intensity we considered.

The paper is organized as follows: Sec. II discusses the relevant literature whereas Sec. III overviews Band Codes and our push-based P2P video streaming protocol *ToroStream*. Sec. IV describes the attack model and the proposed malicious nodes identification and blacklisting scheme, whereas Sec. V contains an analytical model providing the theoretical support for the proposed scheme. Sec. VI provides an experimental evaluation of our malicious nodes identification scheme and validates the relative model. Finally, Sec. VII draws the conclusions of our work.

For the sake of readability, Tab. I summarizes the key notation used throughout this paper.

BC parameters	
k, k'	Generation size, num. pkts to decode ($k' \geq k$)
W	Coding window size ($W \leq k$)
f, l	Coding window leading and trailing edges
x_i	i-th data block, $i \in [1, k]$
$P^i(y^i, g^i)$	i-th coded packet (payload, coding vector)
ϵ_c,	Code overhead, $\epsilon_c = (k' - k)/k$
ToroStream architecture and attack model	
N; N_h,N_m	Tot. num. of nodes; Num. of honest, malicious nodes
p_{poll}	Pkt pollution probability at malicious nodes
N_s	Neighborhood size
Proposed method	
$\mathcal{S}_i(\mathcal{S}_i^m)$	Set of (malicious) nodes seen by node $\mathcal{N}_i$
$nc_{i,j}, np_{i,j}$	Num. of clean and polluted $\mathcal{N}_i$ generations using $\mathcal{N}_j$ pkts
$c_{i,j}$, $p_{i,j}$	Num. of clean and polluted pkts from node $\mathcal{N}_j$ to $\mathcal{N}_i$
c_j, p_j	Total num. of clean and polluted pkts from node $\mathcal{N}_j$
s_j^i	Honest score of neighbor $\mathcal{N}_j$ computed at node $\mathcal{N}_i$
s_h, s_p	Prob. provided pkts yield clean generation (honest, malicious)
t_{black}	Time at which suspect neighbors are blacklisted
α	Blacklisting precision-recall control parameter
Mathematical model	
$c_{i,j}^{*\leftarrow *}, p_{i,j}^{*\leftarrow *}$	model counterpart of counters $c_{i,j}$ and $p_{i,j}$
$c_{i,j}^{*}, p_{i,j}^{*}$	model counterpart of counters $c_{i,j}$ and $p_{i,j}$
c_j^h, p_j^h	model counterpart of counters c_j and p_j (j honest)
c_j^p, p_j^p	model counterpart of counters c_j and p_j (j malicious)
Experimental settings and variables	
N_o	Number of observations available at the nodes
t_o	Duration of the nodes observations
ϵ_p	Pollution overhead
TPR	True Positives Rate in malicious nodes identification
CI	Continuity Index of the video stream

TABLE I
KEY NOTATION USED IN THE PAPER.

II. RELATED WORKS

Approaches to pollution attacks in NC-based P2P systems can be broadly classified in three categories.

Cryptographic or algebraic: works in [5], [6], [7], [8], [9], [10], [11] propose the design of verification techniques able to detect on-the-fly pollution of received packets. They require a secure infrastructure to pre-distribute verification keys and exhibit an high computational cost for verification thus making them unfit for live (real-time) video streaming scenarios as the one we consider in our experiments. The work in [18] represents an improvement over these approaches by exploiting a homomorphic signature function that enables intermediate nodes to verify messages and create valid signature without the need of a secure channel for key pre-distribution. Nevertheless, computational complexity remains remarkable. Another improvement over existing approaches is presented by [19] where the authors propose a lower complexity tag encoding scheme to enable pollution detection. Nevertheless, this approach still relies on key pre-distribution. In [20] two improved key distribution schemes are proposed and analyzed in terms of computation and communications costs. The authors of [21] address a weakness of verification-based techniques known as *verification attack*, whereby a large amount of content to be checked is injected at high rate in a named-data networking scenario.

Error correction: articles [12], [13], [14] exploit error correction to recover corrupted received packets. All these methods add coding redundancy information that enable the packet receivers to detect and automatically reconstruct the original data. The price to be paid is a remarkable increase in the coding overhead; furthermore, the effectiveness of these approaches heavily depends on the amount of corrupted information. In [15] the authors propose rateless codes that are resilient to Byzantine attacks when the fraction of corrupted packets is bounded above by 1/3. Unlike BC, codes in [15] do not fully support NC; indeed, the sparsity of encoding is not preserved when recombinations are performed at each intermediate node.

Probabilistic: a third approach relies on the capability of estimating the probability a node is malicious after collecting several observations on the integrity of received packets and the packets senders. In this context, [22] developed a distributed detection algorithm based on intersection operations of the set of data uploaders to progressively isolate malicious neighbors of a peer. The method is effective in a static scenario where neighborhood of peers does not change in time but falls short of when the overlay network topology is highly dynamic. Approaches in [23], [24] overcome this limitation by casting the problem of identifying malicious nodes as a statistical inference that computes the probability of each peer to be malicious or not. Nevertheless, since belief propagation is used as a mean to compute these probabilities the computational cost of this approach can be substantial. The work in [25] abstracts from actual pollution defense mechanism and models the interaction between attackers and defenders with a two-player strategic game. The final goal is to optimize the resource allocation to defenders when resources are limited in wireless networks.

The present work falls in the last class of probabilistic approaches. Probabilistic approaches are indeed less capable of guaranteeing complete security against pollution attacks (e.g., exact recovery of compromised data). However, the loss-tolerant nature of video communications makes such approaches fit to strike the performance-complexity trade-off sufficient for reasonably securing video communications against pollution attacks.

III. Background

This section first overviews those aspects of Band Codes (BC) [17] that are instrumental to understand the present work. Then, we describe *ToroStream* [26] , our random-push NC-based P2P protocol for low delay video delivery over random overlays. In the following, we assume that network nodes are connected by unicast wired links and all coding operations take place at the application-level of the ISO (layer 5) or OSI (layer 7) stack.

A. Encoding at the Source

Let us assume that one *source* node holding the original video content organizes it into independently encoded and decodable *generations*. For example, each generation accounts for one self-decodable unit of the compressed video bitstream (i.e., one Groups of Pictures - GOPs). Each generation is further divided in k blocks $(x_1, ..., x_k)$ of the same size, the block size for example approaching the size of a network packet. The parameter k defining the number of blocks per generation (identical for all generations) is known as *generation size*. In the following, for the sake of conciseness, we focus on the encoding and distribution of a single generation of video as each generation is independently encoded and decoded following the same approach. Periodically, each node is given a chance to transmit one packet to the network, which we call *transmission opportunity*. At every transmission opportunity, the source generates a random linear combination of the k blocks computed as $y = \sum_{i=1}^{k} g_i x_i$, where g_i are binary values and the summation corresponds to bit-wise XORs. Vector $g = (g_1, ..., g_k)$ is the *coding vector*: the number of non zero elements of g, which corresponds to the number of blocks encoded in the packet, is known as the packet *degree*. The source transmits packets $P(g, y)$ containing i) a payload y consisting of the encoded blocks and ii) the coding vector g used to encode y. Clearly, it is key that the coded payload is received together with the relative coding vector for the receiver to solve the associated system of equations and recover the generation.

The blocks to be combined are selected within a random *coding window*, i.e. a set of W adjacent blocks, where $W \leq k$ (if $W = k$, we have random NC). In the following, we call the first and last block spanned by a coding window the *leading edge* and *trailing edge* of the window and we indicate them as f and l, respectively. For $W < k$ integer f is drawn from the distribution

$$HD(f) = \begin{cases} \frac{W+1}{2k} & \text{if} \quad f = 0 \text{ or } f = k - W \\ \frac{1}{k} & \text{if} \quad 0 < f < k - W \end{cases} \tag{1}$$

such that blocks are equally likely to be encoded [17]. For $W = k$, we have $HD(f) = 1$ only for $f = 0$. Next, each block within the coding window is independently drawn such that $\mathcal{P}\{g_i = 1\} = \frac{1}{2}$ if $f \leq i \leq l$, $g_i = 0$ otherwise. 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 $BC(k, W)$.

B. Recombinations at the Nodes

Every node in the network receives BC coded packets that are stored into a separate input buffer for each generation. In order to retain the ability to independently decode each generation, the recombination process is constrained within each generation, i.e. only packets pertaining to the same generation are recombined. In the following we describe the process to recombine packets pertaining to the same generation at one network node, the process being identical for all generations. Let us assume that a node input buffer contains q packets that are all $BC(k, W)$. We indicate the q buffered packets as $\{P^1, \ldots, P^i, \ldots, P^q\}$, where the i-th packet is defined as $P^i = (g^i, y^i)$. As for the source node, each network node is periodically granted one transmission opportunity. When one transmission opportunity arises, the node creates a novel coded packet $P^r(g^r, y^r)$ as follows. First, the node draws the

recombination window leading edge f_r (and the corresponding trailing edge $l_r = f_r + W - 1$) according to the same probability in (1). Then, concerning the i-th packet P^i in the buffer, let s^i be the first non-null element of the coding vector g^i such that $g^i_{s^i} = 1$ and $g^i_j = 0\ \forall j < s^i$. Similarly, let t^i be the last non-null element of the coding vector g^i such that $g^i_{t^i} = 1$ and $g^i_j = 0\ \forall j > t^i$. Then, the node independently draws each scalar $c_i \in \{0, 1\}$ such 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. Finally, the recombined packet P^r coding vector g^r is computed as the linear combination of the coding vectors of the packets in the input buffer as $g^r = \sum_{i=1}^{q} c_i g^i$. Similarly, the corresponding coded payload is computed as $y^r = \sum_{i=1}^{q} c_i y^i$. Such process guarantees that P^r is still a $BC(k, W)$, i.e. it preserves the degree distribution imposed by the source and thus do not alter the packet decoding complexity.

C. Decoding at the Receivers

The network nodes decode each received packet $P(g, y)$ via a Gaussian elimination-like algorithm. The algorithm solves a system of k linear equations $GX = Y$, where G is the $k \times k$ matrix holding the coding vectors of the received packets, Y is the $k \times 1$ vector holding the coded payloads and X is the $k \times 1$ vector 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 $G_{i,j} = 0, \forall j$, we say that the i-th row of G is *empty* and write $G_i = \emptyset$. The algorithm is described in pseudo-code as Alg. 1 and is executed each time the node receives a packet $P(g, y)$, progressively filling matrix G with linearly independent equations eventually allowing to solve for the unknown X. 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. If G_s is not empty and if $g = G_s$, we say that a *collision* happens at line 8 of the algorithm (we shall for now overlook lines 9 and 10). If $g = G_s$ (or, equivalently, $y = Y_s$), then $P(g, y)$ must be a bitwise copy of (G_s, Y_s), so P is dropped and the algorithm ends at line 11. 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 rows of G, i.e. it represents redundant or duplicate information. The 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).

Once k linearly independent packets have been received, i.e. once the rank of G is equal to k, matrix G is made diagonal via standard backward-substitution, such that the unknown X can be determined. After the diagonalization the vector Y contains the recovered blocks, i.e. $Y_i = x_i$, which eventually allows to recover the generation payload. Due to the random recombinations at the nodes, not all the packets received by a node are innovative, i.e. linearly independent from the packets the node has already collected. In practical applications, it takes $k' > k$ packets to recover a generation.

Algorithm 1 Pollution detection provisioned triangularization

```
1:  receive P(g, y).
2:  while true do
3:      s ← position of leading one of g
4:      if G_s = ∅ then
5:          G_s ← g; Y_s ← y
6:          return
7:      else
8:          if g = G_s
9:              if y ≠ Y_s
10:                 t_poll = t_now
11:             return
12:         g ← g ⊕ G_s; y ← y ⊕ Y_s
13:     end if
14: end while
```

D. The ToroStream P2P Protocol

The network nodes exchange coded packets using the ToroStream [26] P2P protocol. Such protocol is representative of architectures such as [4], [27] where nodes are organized in a random, non acyclic, overlay network. Mesh-based, random overlays are especially sensitive to pollution attacks since a malicious node could easily spread the pollution to the whole network. Moreover, there is no easy way to pinpoint the source of the pollution to a single node lacking any topological hierarchy. For the above reasons, ToroStream is well suited to stress the pollution resiliency capabilities of our architecture designed around BC, which is the goal of the present work.

The overlay network is created under the coordination of a centralized tracker as follows. A node joins the streaming session issuing a join request to the tracker. The tracker maintains a directory of the nodes in the overlay network while ignoring the actual network topology. The tracker replies to join requests with a list of node addresses drawn at random. The joining node contacts the addresses in the list and, after a handshake, starts exchanging coded packets with them. Each node is allowed to exchange coded packet with no more than N_s neighbors per time. If the number of neighbors of a node exceeds N_s following a handshake, the node gracefully disconnects from some neighbors selected at random. Moreover, every node gracefully disconnects from some neighbors at random and establishes novel connections at random to guarantee node churning while keeping no more than N_s neighbors. Such mechanism guarantees that, at any time, each node has a number of neighbors approximately equal to N_s. Finally, explicit signaling such as join/leave requests and periodic keep-alive messages to the tracker make the protocol robust to network or node failures.

IV. Proposed Architecture

In this section, we first model how pollution propagates after a pollution attack due to random packets recombinations. Then, we review how a network node may exploit the BC decoding process to detect pollution attacks and adjust the BC coding window size to limit pollution propagation. Next, we introduce a scheme where a node accumulates and exchanges

periodic packet counts with its neighbors. Such counts allow each node to compute a discriminative *honest score* for each neighbor, blacklisting and isolating low-scoring nodes.

A. Pollution Attack and Propagation

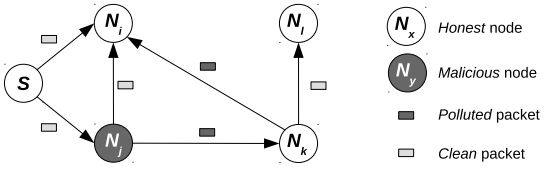

Fig. 1. Simple network composed of $N_h = 3$ honest nodes and $N_m = 1$ malicious node. The malicious node $\mathcal{N}_j$ transmits a polluted packet to $\mathcal{N}_k$ that relies a recombination thereof to $\mathcal{N}_i$.

Let a network be composed of N nodes, where N_h nodes are of the honest type and N_m nodes are of the malicious type, where $N_m + N_h = N$. For example, in the network illustrated in Fig.1, node $\mathcal{N}_j$ is malicious whereas all the other nodes are honest, therefore $N_h = 3$, $N_m = 1$, $N = 4$.

Malicious nodes transmit fake linear combinations $y^p = \sum_{i=1}^{k} g_i^p x_i^p$ to the network, where $g_i^p \neq g_i$ and/or $x_i^p \neq x_i$ in the form of polluted packet $P(g, y^p)$. To the end of crippling the communication, altering the payload or the coding vector is equivalent. However, with BC the degree of the coding vectors follows a specific distribution and altering it may deceive the malicious nodes identity. So, we consider a pollution attack model where at each transmission opportunity each malicious node alters with probability p_{poll} the transmitted packet by randomly flipping the bits of the coded payload. In Fig.1 the malicious node $\mathcal{N}_j$ relays to the honest node $\mathcal{N}_k$ a polluted packet $P(g, y^p)$ where the payload y^p disagrees with the encoding vector g. The value of p_{poll} along with the number of malicious nodes N_m determines the amount of polluted packets purposely injected in the network. We define the *pollution overhead* ϵ_p as the ratio between the polluted and correct packets flowing across the overlay.

The network nodes further propagate the pollution as they do not know if a packet they received is polluted or not. Each time a network node draws the received packets for recombination, it is sufficient that just one of the recombined packets is polluted for the relayed packet to be polluted as well. For example, in Fig. 1 the honest node $\mathcal{N}_k$ accidentally relays to $\mathcal{N}_i$ a packet which is polluted because $\mathcal{N}_k$ has received a polluted packet from malicious $\mathcal{N}_j$. Recombinations at the nodes make it difficult to tell whether a node which has relayed a polluted packet is malicious or is honest and has involuntarily propagated the pollution.

B. Detecting Pollution Attacks

The network nodes independently detect pollution attacks while decoding each received packet via Alg. 1. Let us assume that node $\mathcal{N}_i$ in Fig. 1 receives a packet $P(g, y)$ at time t_{now}. Node $\mathcal{N}_i$ executes Alg. 1 to decode $P(g, y)$, attempting to insert the coding vector g and the coded payload y in the s-th row of matrix G and vector Y respectively. If G_s is not empty ($G_s \neq \emptyset$), a collision happens at line 8 and a chance to detect an ongoing pollution attack arises. If the received packet coding vector g and G_s are identical, the received packet $P(g, y)$ and (G_s, Y_s) shall represent the same linear combination of symbols. However, if y and Y_s are not bitwise-identical (line 9), at least one of the packets received so far for the generation must be polluted. In this case, the algorithm logs the time at which the pollution attack was detected as t_{poll} (line 10) and returns a warning code.

The proposed pollution detection algorithm brings several advantages. First, no additional complexity is entailed beside a bitwise comparison between the coding vectors and the coded payloads. Second, such scheme may enable a node to detect a pollution attack even before the generation is recovered, allowing for timely countermeasures. Third, there is a chance to detect pollution attacks at each algorithm iteration, i.e. there are multiple chances to detect a pollution attack for each received packet.

Notice that such pollution detection scheme entails limitations as well. First, pollution attacks are detected only on a probabilistic basis, i.e. the reception of one or more polluted packets may go unnoticed. Second, it is not possible to understand *which* packet(s) is (are) polluted, so the node shall assume that all the packets received so far for the generation are polluted. Nevertheless, in the following we show how probabilistic pollution detection is sufficient to precisely blacklist the malicious nodes leveraging the properties of BC.

C. Coding to Limit Pollution Propagation

Upon detection of a pollution attack in Alg. 1, each node adapts its own BC parameters to limit the chances to relay polluted packets. In the case of Fig. 1, node $\mathcal{N}_j$ detects as pollution attack and broadcasts a warning to its neighbors including $\mathcal{N}_i$. We indicate with t_{poll} the time at which node $\mathcal{N}_i$ has either detected a pollution attack or has received a warning from a neighbor. Let us assume that at time $t_{now} > t_{poll}$ a transmission opportunity arises for node $\mathcal{N}_i$: the difference $t_{now} - t_{poll}$ represents the time since the last evidence of a pollution attack.

If lots of time has passed since the last pollution evidence, say $t_{now} - t_{poll} > t_{back}$, $\mathcal{N}_i$ assumes that the attack has ended or its intensity has decreased. Hence, $\mathcal{N}_i$ draws the packets to recombine as detailed in Sec. III-B from a coding window of size $W = k$.

If $t_{now} - t_{poll} < t_{back}$, node $\mathcal{N}_i$ behaves conservatively assuming that the last detected pollution attack may be still going on. Hence, $\mathcal{N}_i$ draws the packets to recombine from a random window of size $W = \lceil \frac{k}{n} \rceil$, i.e. $W < k$, i.e. $\mathcal{N}_i$ constrains the packet recombination to a subset of the buffered packets. So, $\mathcal{N}_i$ decreases the probability to relay a polluted packet with respect to the case where each packet is recombined with identical probability as in random NC.

D. Counting Polluted Packets and Exchanging Observations

With reference to Fig. 1, node $\mathcal{N}_i$ has (or has had at some point in time) node $\mathcal{N}_j$ among its neighbors: we say that $\mathcal{N}_i$ has *seen* $\mathcal{N}_j$. In the following, we indicate as S_i the set of

nodes seen by $\mathcal{N}_i$ and as S_i^m the set of malicious nodes seen by $\mathcal{N}_i$: clearly, S_i^m is a subset of S_i and $|S_i^m|$ indicates its cardinality. For example, in the case in Fig. 1, we have $S_i = \{\mathcal{N}_j, \mathcal{N}_k\}$ and $S_i^m = \{\mathcal{N}_j\}$.

We define as *observation vector* the vector indicating how many clean and polluted packets a node has received from each neighbor it has seen according to the output of Alg. 1. We recall that the algorithm only detects pollution attacks on a probabilistic basis. Namely, the algorithm flags a generation as polluted *only* if the attack is actually detected and provides no information on *which* packet(s) is (are) actually polluted. So, in the following *all* packets received for a generation flagged as polluted by Alg. 1 will be considered polluted to account for a caution criteria. With reference to node $\mathcal{N}_i$ in Fig. 1, $(c_{i,j}, p_{i,j})$ indicates the number of clean and polluted packets transmitted by $\mathcal{N}_j$ to $\mathcal{N}_i$. Note that similarly, with reference to node $\mathcal{N}_k$, $(c_{k,j}, p_{k,j})$ indicates the number of clean and polluted packets transmitted by $\mathcal{N}_j$ to $\mathcal{N}_k$. So, the whole observation vector of $\mathcal{N}_i$ is $\mathcal{V}_i = \{(c_{i,j}, p_{i,j}), (c_{i,k}, p_{i,k})\}$. Node $\mathcal{N}_l$ does not appear in $\mathcal{N}_i$ observation vector because in our example $\mathcal{N}_i$ has never seen $\mathcal{N}_l$.

We assume that the nodes are allowed exchange the respective observation vectors on a periodic basis or on handshakes at least. Depending on constraints such as bandwidth, nodes may piggyback their observations to other messages or relay differential descriptions of their observation vectors.

E. Distributed Identification and Blacklisting Scheme

The scheme that allows a network node to independently identify and blacklist its malicious neighbors is as follows.

Let us assume that in the case of Fig. 1 $\mathcal{N}_i$ and $\mathcal{N}_k$ exchange the respective observation vectors. At this point $\mathcal{N}_i$ has an estimate of the total number of good and polluted packets received from $\mathcal{N}_j$ and $\mathcal{N}_k$. Let node $\mathcal{N}_i$ compute

$$c_j = c_{i,j} + \sum_{\mathcal{N}_q \in S_i} c_{q,j}, \tag{2}$$

as the total number of clean packets transmitted by $\mathcal{N}_j$ to $\mathcal{N}_i$ and all the nodes seen by $\mathcal{N}_i$ which have also seen $\mathcal{N}_j$. In a totally analogous way, let us define

$$p_j = p_{i,j} + \sum_{\mathcal{N}_q \in S_i} p_{q,j}. \tag{3}$$

We define

$$s_j^i = \frac{c_j}{p_j + c_j}, \quad j \neq i \tag{4}$$

as the *honest score* of node $\mathcal{N}_j$ computed at node $\mathcal{N}_i$. Such score represents the fraction of clean packets transmitted by $\mathcal{N}_j$ to $\mathcal{N}_i$ and all the nodes that exchanged their observations with $\mathcal{N}_i$. Node $\mathcal{N}_i$ may sort the computed scores in increasing order, i.e. nodes that have uploaded more packets for polluted generations are listed before the others.

After collecting enough observations, $\mathcal{N}_i$ proceeds to blacklisting neighbors with lower score as follows. Let us assume that t_{black} seconds after detecting the pollution attack, node $\mathcal{N}_i$ computes the relative score s_j^i as defined above for each neighbor $\mathcal{N}_j$ it has seen. Next, $\mathcal{N}_i$ computes the mean neighbor score and the relative standard deviation as μ and σ, respectively. Let us define as *honest score threshold*

$$\theta_p = \alpha\ \sigma + \mu, \tag{5}$$

where α is a factor that can be adjusted to regulate the tradeoff between precision and recall in malicious nodes identification as described below. If node $\mathcal{N}_j$ score s_j^i is such that $s_j^i \geq \theta_p$, then $\mathcal{N}_j$ is considered honest. Otherwise, node $\mathcal{N}_j$ is to be considered as malicious and is blacklisted in two steps as follows.

As a first step, $\mathcal{N}_i$ immediately discards all the buffered packets received from $\mathcal{N}_j$ for any generation. Therefore, packets received from $\mathcal{N}_j$, more likely to be polluted, cannot be drawn for recombination and relayed any further. Next, for each generation, matrix G and vector Y are flushed and the buffered packets that were not discarded are decoded again via Alg. 1. This reduces the probability that the recovered generation payload is corrupted due to the reception of polluted packets from $\mathcal{N}_j$.

As a second step, $\mathcal{N}_j$ is permanently isolated from $\mathcal{N}_i$ as follows. First, $\mathcal{N}_i$ *gracefully* removes $\mathcal{N}_j$ from its neighborhood. Second, $\mathcal{N}_i$ will reject any novel neighborhood request that shall come from $\mathcal{N}_j$ in the future. That is, $\mathcal{N}_i$ will not receive any coded packet directly from $\mathcal{N}_j$ for the rest of the communication. Assuming that $\mathcal{N}_j$ is similarly blacklisted by all the other nodes in the network, $\mathcal{N}_j$ will be isolated from the rest of the network and will be unable to relay polluted packets. Therefore, removal of blacklisted nodes from the overlay becomes superfluous to the end of containing the pollution attack.

F. Reference Centralized Scheme

Finally, we propose a reference centralized counterpart of our distributed blacklisting scheme. On a periodic basis, the nodes envoy their observations exclusively to the tracker, that acts as a trusted central authority. After t_{black} seconds the streaming session has begun, the tracker computes the overall number of polluted and clean packets p_j and c_j contributed by each j-th node in the network to each i-th node that provided its observation vectors. Next, the tracker centrally computes a score s_j^c for each j-th node in the network according to (4) and permanently blacklists nodes for which $s_j^c < \theta_p$. Namely, each time a blacklisted node requires novel neighbor address(es) to the tracker, the tracker ignores the request. Also, each time an honest node requires novel neighbor address(es) to the tracker, the tracker never forwards the address of blacklisted nodes. Because all the nodes periodically drop parts of their neighbors at random, malicious nodes are gradually isolated from the rest of the network.

G. Why Does it Work?

Finally, we provide the theoretical ground required to understand how node $\mathcal{N}_i$ can discern malicious from honest neighbors by thresholding scores. To this end, we associate each network node with a Bernoulli random variable whose parameter is s_h for honest nodes and s_p for malicious nodes. We use s_h (s_p) to represent the probability that packets

provided by the honest (malicious) neighbor $\mathcal{N}_j$ result in a clean recovered generation at $\mathcal{N}_i$. The honest score s_j^i we defined in (4) is the fraction of packets provided by $\mathcal{N}_j$ to $\mathcal{N}_i$ that yield the decoding of a clean generation. Honest score s_j^i can thus be thought of $\mathcal{N}_i$'s estimate of the probability s_h (s_p) that $\mathcal{N}_j$ is a honest (malicious) neighbor. Thanks to the homogeneity and regularity of the overlay network organized by ToroStream, it follows that all s_j^i computed by $\mathcal{N}_i$ for its honest neighbors will converge in the long run to the same value s_h. Analogously, all s_j^i computed by $\mathcal{N}_i$ for its malicious neighbors will tend to s_p.

To understand why the proposed score-based blacklisting method is effective, we denote as $nc_{i,j}$ and $np_{i,j}$ the number of clean and polluted generations decoded by $\mathcal{N}_i$ using at least one packet transmitted by $\mathcal{N}_j$. Similarly to the definition of c_j, p_j in Sec. IV-E, we can then define $nc_j = nc_{i,j} + \sum_{\mathcal{N}_q \in S_i} nc_{q,j}$ as the number of clean generations contributed by $\mathcal{N}_j$ to all of its neighbors (similarly, $np_j = np_{i,j} + \sum_{\mathcal{N}_q \in S_i} np_{q,j}$).

If we assume $\mathcal{N}_j$ is malicious, it is well known that for a large sample size, i.e. if enough observations were collected, the sample distribution of the honest score s_j^i estimator can be approximated as $\mathcal{N}(s_p, \frac{s_p(1-s_p)}{nc_j+np_j})$, i.e., as a Normal distribution with mean value s_p and variance equal to the scaled variance of the Bernoulli random variable to estimate [28]. As the sample size nc_j+np_j gets larger, the probability $s_j^i \geq \theta_p$ decreases with the variance $s_p(1-s_p)$. Furthermore, the convergence rate depends on the variance that peaks its maximum for $s_h = 0.5$ (or $s_p = 0.5$). That is, for infinitely long experiments (i.e., infinite observations) the probability that $s_j^i \geq \theta_p$ (if $\mathcal{N}_j$ is malicious) tends to 0 and $\mathcal{N}_i$ will correctly identify all malicious neighbors. In the practical case of finite time experiments, the rate of convergence of s_j^i depends on how distant s_p is from 1.

The model we develop in Sec. V will reveal that i) both s_h and s_p tend to 1 as W decreases, and ii) $s_h > s_p$. Such findings suggest that for random NC ($W = k$), s_h and s_p are both approximately 0.5, so more observations are required to avoid $s_j^i \geq \theta_p$ if $\mathcal{N}_j$ is malicious, i.e., to avoid false negatives. On the contrary, with BC ($W < k$) s_h and s_p get increasingly close to 1 as W decreases. So, given the same experiment length, BC will enable to identify malicious nodes more precisely than random NC since higher variance increases the tail probability that $s_j^i \geq \theta_p$ for the case $\mathcal{N}_j$ is malicious.

V. Mathematical model for s_h and s_p

In this section we develop a mathematical model aimed at characterizing the values of s_h and s_p as a function of system parameters. The model development is carried out in five steps:

- with reference to Sec. III-B we derive the probability p_{sl} that one of the packets stored in a node input buffer for a generation is selected for recombination (Sec. V-A);
- we then exploit p_{sl} to obtain the probability $p_{rp}(q)$ that a node creates a polluted recombined packet for a generation when q $BC(k, W)$ packets are stored in its input buffer (Sec. V-B);
- we consider a sequence of recombinations for a given generation and from $p_{rp}(q)$ we derive the average probability a node creates and relays a recombined polluted packet p_{rl} (Sec. V-C);
- we focus on a generic node $\mathcal{N}$ and by exploiting p_{rl} we derive and expression to describe the average number of packets provided to $\mathcal{N}$ by one of its honest uploader that result in clean and polluted decoded generation. We further extend this derivation to one of the malicious neighbors of $\mathcal{N}$ (Sec. V-D);
- finally, we focus on any pair of randomly chosen nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ and we consider all cases where $\mathcal{N}_i$ and $\mathcal{N}_j$ can be honest or malicious in order to derive an expression for the s_h and s_p (Sec. V-E). Please note that the ToroStream protocol described in Sec.III-D lets us view the overlay network as a regular random graph [29] wherein the degree of all nodes is equal to N_s and where all neighborhoods are completely randomly determined, i.e., any peer can be connected to any other with the same probability. This means that the overlay network is symmetric since all nodes in the system are homogeneous and statistically indistinguishable with respect to how many and which neighbors are connected to it. Therefore, the analytical expressions for s_h and s_p we derive do not depend on a particular choice of nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ since any other pair of nodes would yield the same quantities.

In Sec. VI we numerically show that $s_h > s_p$ and that smaller BC coding windows W yield s_h and s_p values closer to 1. In the sequel we denote as $B_p^{n,x} = \binom{n}{x} p^x (1-p)^{n-x}$ the binomial probability distribution with parameters n, p computed in x. Similarly, we denote as $H_{n_1,n_2,\ldots,n_a}^{N_1,N_2,\ldots,N_a} = \frac{\binom{N_1}{n_1}\binom{N_2}{n_2}\cdots\binom{N_a}{n_a}}{\binom{N_1+N_2+\cdots+N_a}{n_1+n_2+\ldots+n_a}}$ the multivariate hypergeometric distribution with a populations whose sizes are $N_1, N_2, \ldots N_a$. Furthermore, to avoid cluttering the notation we omit the explicit dependence of all derived probabilities on k and W.

A. Probability of selecting a packet for recombination

We consider a generic node $\mathcal{N}$; as discussed in Sec. III-B, $\mathcal{N}$ first draws the recombination window leading edge f_r (and the corresponding trailing edge $l_r = f_r + W - 1$) according to (1). Packet P^i in the buffer is actually selected for recombination with probability $\frac{1}{2}$ only if $f_r \leq s^i \leq t^i \leq l_r$ (we recall from Sec. III-B that scalar $c_i \in \{0, 1\}$ is such 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). The probability that the structure of a $BC(k, W)$ packet P^i satisfies $f_r \leq s^i \leq t^i \leq l_r$ is

$$p_{st}(f_r, s^i, t^i) = \begin{cases} \frac{2^{t^i - s^i - 1}}{2^W - 1} & t^i - s^i > 0 \\ \frac{1}{2^W - 1} & t^i - s^i = 0 \end{cases}$$

This means that the probability packet P^i is *eligible* for recombination when a leading edge f_d is drawn is

$$p_{el}(f_d) = \sum_{s^i = f_d}^{l_r} \sum_{t^i = s^i}^{l_r} \sum_{f_r = 0}^{k-W} HD(f_r) \cdot p_{st}(f_r, s^i, t^i)$$

Thus, according to the recombination algorithm described in Sec. III-B, the probability packet P^i is *actually selected* by

$\mathcal{N}$ for recombination is given by

$$p_{sl} = \sum_{f_d=0}^{k-W} \frac{HD(f_d)p_{el}(f_d)}{2}. \quad (6)$$

B. Probability a node creates a polluted recombined packet

Here we derive the probability $p_{rp}(q)$ node $\mathcal{N}$ creates a recombined polluted packet for a given generation when it has q packets in its input buffer. To this end, we consider a scenario where:

- the number of neighbors of $\mathcal{N}$ is equal to N_s and all neighbors upload packets for a generation;
- the neighborhood of $\mathcal{N}$ includes m malicious neighbors;
- node $\mathcal{N}$ input buffer contains q packets for the considered generation that have been provided by its N_s neighbors. We assume there are $0 \leq q_m \leq q$ packets provided by the m malicious neighbors and the remaining $q-q_m$ packets provided by $N_s - m$ honest neighbors;
- malicious neighbors of $\mathcal{N}$ may have modified packets they transmitted with probability p_{poll};
- the distance between node $\mathcal{N}$ and the video source is equal to d hops (when $\mathcal{N}$ is a neighbor of the video source $d = 1$);
- all neighbors of $\mathcal{N}$ may have relayed a polluted $BC(k, W)$ packet they created with probability $p_{rl}(d-1)$. In Sec. V-C we derive a recursive expression for $p_{rl}(d)$.

In this scenario, the probability that all the selected packets provided by the m malicious neighbors of $\mathcal{N}$ are clean is

$$p_{cp}(q_m, d) = \sum_{b=0}^{q_m} B_{p_{sl}}^{q_m,b}[(1-p_{poll})(1-p_{rl}(d-1))]^b, \quad (7)$$

i.e., none of the selected packets received by $\mathcal{N}$ from a malicious neighbor has been voluntarily altered (factor $(1-p_{poll})^b$) and none has been created and relayed as polluted (factor $(1-p_{rl}(d-1))^b$). Similarly, the probability that all the packets provided by honest neighbors selected by $\mathcal{N}$ are clean is

$$p_{ch}(q_m, d) = \sum_{b=0}^{q-q_m} B_{p_{sl}}^{q-q_m,b}(1-p_{rl}(d-1))^b. \quad (8)$$

In this case, honest nodes may have created and relayed a recombined packet where at least one of the selected packets was polluted. It follows that the probability node $\mathcal{N}$ creates a recombined polluted packet when m out of its N_s neighbors are malicious is obtained by averaging over all possible values of q_m and over the probability that the neighborhood of node $\mathcal{N}$ includes m malicious neighbors as

$$p_{rp}(q,d) = \sum_{m=0}^{\min(N_m,N_s)} H_{N_s-m,m}^{N_h,N_m} \sum_{q_m=0}^{q} B_{\frac{m}{N_s}}^{q,q_m}[1-p_{cp}(q_m,d)p_{ch}(q_m,d)]. \quad (9)$$

This averaging is based on the assumption that the neighborhood creation of a node is completely random and can thus be modeled as a sampling without replacement of N_s nodes, operated by the tracker, among all nodes in the system.

C. Average probability to recombine and relay a polluted packet

The number of $BC(k, W)$ packets in the input buffer of node $\mathcal{N}$ for a given generation increases as long as decoding does not occur. This means that $\mathcal{N}$ creates multiple recombined packets for a given generation each time using a different (increased) input buffer occupancy q. This also means that the probability $\mathcal{N}$ creates a polluted recombined packet $p_{rp}(q, d)$ increases with time; it is thus a sensible choice to define an *average probability* for a node to create and relay a recombined polluted packet.

To achieve this goal we can represent this time-dependent process by a *recombination sequence* $\underline{q} = (q_1, q_2, \ldots q_{n_r})$ where $n_r > 0$ represents the number of recombination rounds operated by node $\mathcal{N}$ for a given generation, with the constraints that $q_1 > 0$, $q_{n_r} \geq k$, and $\forall i \in [1, n_r - 1]$, $q_i < q_{i+1}$.

The definition of a recombination sequence can be further restricted by considering that:

- the overall upload bandwidth of nodes is equal to B_u Mbit/s;
- the size of a $BC(k, W)$ packet is equal to PS kbit;
- each node $\mathcal{N}$ uses a scheduler that is cyclically activated each t_{sched} seconds to create and upload recombined packets to its neighbors. This means that a node runs $\frac{1}{t_{sched}}$ *recombination rounds* each second.

If we assume that each node equally shares its upload bandwidth among its N_s neighbors then we find that the overall *download bandwidth* of nodes is equal to B_u, as well. This implies that each node fills its input buffer at a rate equal to $\frac{B_u}{PS}$ packets/s that translates to $\delta_q = \frac{t_{sched}B_u}{PS}$ packets/recombination round. This observation allows us to further constraint the definition of a recombination sequence $\underline{q}$ by imposing that $\forall i \in [2, n_r]$, $q_i = q_{i-1} + \delta_q$.

For a recombination sequence $\underline{q}$ we define two quantities:

- the average value of probability $p_{rp}(q, d)$ defined as $\overline{p}_{rp}(\underline{q}, d) = \frac{\sum_{i=1}^{n_r} p_{rp}(q_i,d)}{n_r}$;
- the weight associated to each $\underline{q}$ that we define as $w(\underline{q}) = \xi(q_{n_r})$, where $\xi(k'_{n_r}) = 1 - \sum_{k'=1}^{k'_{n_r}} \tau(k')$ denotes the probability that decoding has not occurred when k'_{n_r} $BC(k, W)$ have been collected (here $\tau(k')$ is the probability that a generation is decoded upon the receipt of the $k'-$th packet). This also serve to model the fact that creation of recombined packets stops after decoding.

It follows we can define the average probability node $\mathcal{N}$ creates and relays a recombined polluted packet $p_{rl}(d)$ by computing the weighted average of $\overline{p}_{rp}(\underline{q}, d)$ over all possible recombination sequences $\underline{q}$, that is

$$p_{rl}(d) = \frac{\sum_{\underline{q}} w(\underline{q})\overline{p}_{rp}(\underline{q}, d)}{\sum_{\underline{q}} w(\underline{q})} \quad (10)$$

It can be noted that (10) recursively defines $p_{rl}(d)$ since (7), (8), and (9) all depend on $p_{rl}(d-1)$. We thus complete the recursive definition of $p_{rl}(d)$ with the base case $p_{rl}(0) = 0$, i.e., nodes connected to the video source can only receive polluted packets from polluter neighbors (besides the video

source) that voluntarily alter packets with probability p_{poll} and there is no honest node relaying polluted recombined packets.

D. Average number of packets to decode one generation

We consider a generic node $\mathcal{N}$ whose neighborhood includes m malicious neighbors. We focus on the generations it decodes and we assume that q $BC(k,W)$ packets stored in its input buffer allowed node $\mathcal{N}$ to decode a single reference generation. In this setting, we can write the expression for the average number of packets provided to $\mathcal{N}$ by one of its *honest* neighbors when a *single clean* generation is decoded as

$$c^h(m)=\sum_{q=k}^{\infty}\tau(q)\sum_{q_m=0}^{q}B^{q,q_m}_{\frac{m}{N_s}}(1-p_{poll})^{q_m}(1-p_{rl}(d))^q\left(\frac{q-q_m}{N_s-m}\right) \tag{11}$$

Eq. 11 weights the average number of packets provided by each honest uploader (factor $\frac{q-q_m}{N_s-m}$) by the probability that the generation does not contain polluted packets (probability $(1-p_{poll})^{q_m}(1-p_{rl}(d))^q$). It also averages over all possible values of $BC(k,W)$ packets provided to $\mathcal{N}$ by malicious neighbors and over all possible values of packets in the input buffer that are required for decoding.

Similar reasonings allow one to derive an expression for the average number of packets provided to $\mathcal{N}$ by one of its honest neighbors when a *single polluted* generation is recovered (denoted as $p^h(m)$): it suffices to replace in (11) the probability of clean decoding by its complement $1-(1-p_{poll})^{q_m}(1-p_{rl}(d))^q$. In the same vein, two analogous quantities for one of the malicious neighbors of $\mathcal{N}$ can be derived: $c^p(m)$ can be obtained from (11) and $p^p(m)$ from $p^h(m)$ by substituting factor $\frac{q-q_m}{N_s-m}$ with $\frac{q_m}{m}$ in both.

E. An expression for s_h and s_p

We now consider any two randomly chosen nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ to derive an expression for the honest score s^i_j. Since nodes are homogeneous and statistically indistinguishable with respect to how many and which neighbors are connected to it, the expression for s^i_j we derive does not depend on a particular choice of i and j and it can be used to characterize parameters s_h and s_p. To this end, we first characterize the number of clean and polluted packets transmitted by $\mathcal{N}_j$ to $\mathcal{N}_i$ for a *single* generation.

We focus on the cases where $\mathcal{N}_j$ is in the neighborhood of $\mathcal{N}_i$ and consider the composition of the remaining N_s-1 nodes to properly average the values of $c^h(m), p^h(m), c^p(m), p^p(m)$.

We first consider $\mathcal{N}_j$ as an honest node that provides packets to an honest $\mathcal{N}_i$; in this case we define the number of clean packets $\mathcal{N}_j$ provides to $\mathcal{N}_i$ as

$$c^{h\leftarrow h}_{i,j}=\frac{\sum_{m=0}^{\min(N_m,N_s-1)}H^{N_h-2,N_m,1}_{N_s-m-1,m,1}c^h(m)}{\sum_{m=0}^{\min(N_m,N_s-1)}H^{N_h-2,N_m,1}_{N_s-m-1,m,1}} \tag{12}$$

Analogously, we derive $p^{h\leftarrow h}_{i,j}$ (defined as the number of polluted packets from $\mathcal{N}_j$ to $\mathcal{N}_i$) by replacing $c^h(m)$ with $p^h(m)$ in (12). Please note, that $c^{h\leftarrow h}_{i,j}$ and $p^{h\leftarrow h}_{i,j}$ are the model counterpart of counters $c_{i,j}$ and $p_{i,j}$ defined in Sec.IV-D. If $\mathcal{N}_i$ is malicious we obtain

$$c^{p\leftarrow h}_{i,j}=\frac{\sum_{m=0}^{\min(N_m-1,N_s-1)}H^{N_h-1,N_m-1,1}_{N_s-m-1,m,1}c^h(m)}{\sum_{m=0}^{\min(N_m-1,N_s-1)}H^{N_h-1,N_m-1,1}_{N_s-m-1,m,1}} \tag{13}$$

and $p^{p\leftarrow h}_{i,j}$ is computed by replacing $c^h(m)$ with $p^h(m)$ in (13).

The case where $\mathcal{N}_j$ is malicious is similar but with an important difference: when averaging over all values of m in the neighborhood the quantities to be averaged must be computed for $m+1$ malicious neighbors and not for m since $\mathcal{N}_j$ is malicious itself. So, when $\mathcal{N}_i$ is honest we obtain

$$c^{h\leftarrow p}_{i,j}=\frac{\sum_{m=0}^{\min(N_m-1,N_s-1)}H^{N_h-1,N_m-1,1}_{N_s-m-1,m,1}c^p(m+1)}{\sum_{m=0}^{\min(N_m-1,N_s-1)}H^{N_h-1,N_m-1,1}_{N_s-m-1,m,1}}. \tag{14}$$

Again, $p^p(m+1)$ replaces $c^p(m+1)$ in the definition of $p^{h\leftarrow p}_{i,j}$ while if $\mathcal{N}_i$ is malicious we have

$$c^{p\leftarrow p}_{i,j}=\frac{\sum_{m=0}^{\min(N_m-2,N_s-1)}H^{N_h,N_m-2,1}_{N_s-m-1,m,1}c^p(m+1)}{\sum_{m=0}^{\min(N_m-2,N_s-1)}H^{N_h,N_m-2,1}_{N_s-m-1,m,1}} \tag{15}$$

with $p^{p\leftarrow p}_{i,j}$ obtained by substituting $c^p(m+1)$ by $p^p(m+1)$.

All previously defined quantities $c^{*\leftarrow *}_{i,j}$ and $p^{*\leftarrow *}_{i,j}$ are the model counterpart of counters $c_{i,j}$ and $p_{i,j}$ defined in Sec.IV-D for all possible honest/malicious combinations; they all must be weighted for the relative population sizes to obtain the overall number of clean and polluted packets as

$$\begin{aligned}
c^h_{i,j}&=c^{h\leftarrow h}_{i,j}\frac{N_h-1}{N_h+N_m-1}+c^{p\leftarrow h}_{i,j}\frac{N_m}{N_h+N_m-1}\\
c^p_{i,j}&=c^{h\leftarrow p}_{i,j}\frac{N_h}{N_h+N_m-1}+c^{p\leftarrow p}_{i,j}\frac{N_m-1}{N_h+N_m-1}\\
p^h_{i,j}&=p^{h\leftarrow h}_{i,j}\frac{N_h-1}{N_h+N_m-1}+p^{p\leftarrow h}_{i,j}\frac{N_m}{N_h+N_m-1}\\
p^p_{i,j}&=p^{h\leftarrow p}_{i,j}\frac{N_h}{N_h+N_m-1}+p^{p\leftarrow p}_{i,j}\frac{N_m-1}{N_h+N_m-1}
\end{aligned} \tag{16}$$

According to the definitions (2) and (3) given in Sec. IV-E, we define the total number of clean packets transmitted by $\mathcal{N}_j$ to $\mathcal{N}_i$ and all the nodes seen by $\mathcal{N}_i$ ($\mathcal{N}_q\in S_i$) which have also seen $\mathcal{N}_j$: in the case node $\mathcal{N}_j$ is honest we obtain

$$c^h_j=c^h_{i,j}\left[nc_{i,j}+np_{i,j}+\sum_{\mathcal{N}_q\in S_i}(nc_{q,j}+np_{q,j})\right], \tag{17}$$

where $nc_{x,j}$ denotes the overall number of generations decoded by $\mathcal{N}_x$ when $\mathcal{N}_j$ uploaded packets to it as defined in Sec. IV-G. Analogously, we define

$$p^h_j=p^h_{i,j}\left[nc_{i,j}+np_{i,j}+\sum_{\mathcal{N}_q\in S_i}(nc_{q,j}+np_{q,j})\right]. \tag{18}$$

To write (17) and (18) we exploited the homogeneity and regularity properties of nodes in ToroStream, i.e., $\forall \mathcal{N}_q \in S_i : c^h_{i,j} = c^h_{q,j} \wedge p^h_{i,j} = p^h_{q,j}$. Similar expressions can be written for c^p_j and p^p_j in the case node $\mathcal{N}_j$ is malicious by replacing $c^h_{x,j}$ and $p^h_{x,j}$ by $c^p_{x,j}$ and $p^p_{x,j}$, respectively.

According to (4), we finally define the probability that packets provided by $\mathcal{N}_j$ to $\mathcal{N}_i$ result in a clean recovered generation when $\mathcal{N}_j$ is honest as

$$s^i_j = \frac{c^h_j}{c^h_j + p^h_j} = \frac{c^h_{i,j}}{c^h_{i,j} + p^h_{i,j}} = s_h \tag{19}$$

Similarly, when $\mathcal{N}_j$ is a polluter we obtain

$$s^i_j = \frac{c^p_j}{c^p_j + p^p_j} = \frac{c^p_{i,j}}{c^p_{i,j} + p^p_{i,j}} = s_p. \tag{20}$$

VI. Experiments

In this section we first define the scenario we consider showing how pollution propagates depending on BC parameters. Afterwards, we exploit the model in Sec. V to predict the performance of the proposed identification method. Finally, we used the actual prototype to both validate the model prediction and to analyze the effects of protocol parameters on the accuracy of the identification technique.

A. Experimental scenario

We consider a network with N=1000 nodes where N_m = 20 are malicious and $N_h = N - N_m$ = 980 are honest. The neighborhood of each node is constrained to N_s = 25 nodes and each malicious node alters the payload of each transmitted packet with probability p_{poll} = 1%. We stream a live video by means of the ToroStream P2P protocol described in Sec. III-D. The protocol is implemented as a realtime multi-threaded C++ Linux application that we deploy over a 64-cores server with 128 GB of RAM and a fast array of RAID disks. Such setup allows us to capture in realtime the complex interactions between coding, packet scheduling and other aspects of a real P2P live video distribution architecture.

We encode a video feed at a constant rate of 500 kbit/s where each node can exploit an upload bandwidth up to $B_u = 1$ Mbit/s. The video stream is divided in generations of k blocks each, where each generation encompasses one or more self-decodable Groups of Pictures (GoPs). At each transmission opportunity, the server node encodes a packet as described in Sec. III-A (with $W = k$ at start-up) and uploads it to a random peer node. The server seeds coded packets for each generation of the video stream for an amount of time corresponding to the generation playout duration. Periodically (once every $t_{sched} = 100$ ms) each peer node transmits a novel BC packet out of its locally buffered packets to a neighbor drawn at random. Whenever a transmission opportunity arises, each malicious node randomly flips the bits of the coded video payload with probability p_{poll} as described in Sec. IV; otherwise, it behaves as an honest node. Each node decodes the received packets using the pollution detection capable Alg. 1 and broadcasts a message to its neighbors whenever it detects a polluted generation.

B. Pollution Propagation Analysis

As a first set of experiments, we investigate how the pollution propagates through the network as a function of the generation size k and coding window size W. Let us define the pollution overhead ϵ_p as the fraction of relayed packets that are polluted. The coding overhead $\epsilon_c = \frac{k'-k}{k}$ represents instead the fraction of packets relayed by a node that are non-innovative at the recipient. Fig. 2 (left) shows the attainable ϵ_p - ϵ_c tradeoff as a function of W and k. Only 20 nodes out of 1000 are malicious, and each malicious node pollutes on average 1% of the transmitted packets, so the pollution overhead due to the malicious nodes activity amounts to just 0.02% of the overall traffic. The *RNC* curve refers to a RNC scheme where the nodes relay random linear combinations of the received packets (i.e., $W = k$). The other three curves refer to the BC-based scheme described in this work and account for three different coding window sizes $W \in \{\frac{2k}{3}, \frac{k}{2}, \frac{k}{3}\}$. We see that with BC the pollution overhead is about 10 times lower than with RNC, and decreases with the window size W. Most important, we observe that small W and k values yield the sought conditions for correctly identifying the malicious nodes. However, small k also impair the code efficiency because the nodes exchange fewer innovative packets increasing the coding overhead (right figure). Therefore, in the rest of our experiments we consider generations of 250 kbit (k=25 blocks, as each block accounts for 10 kbit of video payload) and a packets size $PS = 10$ kbit as a reasonable tradeoff between coding and pollution overhead.

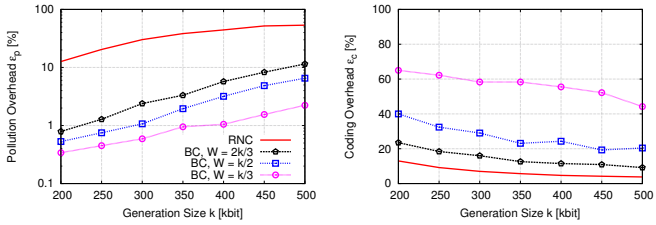

Fig. 2. Pollution overhead ϵ_p and coding overhead ϵ_c as a function of the of the generation size k [kbit] and coding window size W [blocks].

C. Honest Score Model Exploitation

The model we developed in Sec. V, through (19) and (20), allows us to predict the value of the honest score estimated by each node. As already discussed this amounts at estimating the system-wide parameters s_h and s_p. Table II (leftmost columns) shows the values obtained with (19) and (20), computed by setting the model parameters according to the scenario defined in Secs. VI-A and VI-B. These evaluations depend on (10) and (11) that require the knowledge of $\tau(k')$, i.e. the decoding probability of $BC(k, W)$. To the best of our knowledge there is not an analytical expression for such probability. Therefore, we numerically computed it by repeated decoding trials simulation for a range of coding window sizes $W \in \{k, \frac{2k}{3}, \frac{k}{2}, \frac{k}{3}\}$. It can be noted that the model predicts that $s_h > s_p$ for all values of W we considered, thus suggesting that the proposed identification mechanism is able to discern honest from malicious in the long term. Furthermore, the model

results also show that both s_h and s_p tend to 1 as W decreases; according to the remarks in Sec. IV-G, this means that the rate of convergence of the estimates increases as W decreases. In particular, the model results show that for RNC ($W = k$), s_h and s_p are both approximately 0.5, so more observations are required to avoid that $s_j^i \geq \theta_p$ if $\mathcal{N}_j$ is malicious, i.e., to reduce false negatives. On the contrary, with BC ($W < k$) s_h and s_p get increasingly close to 1 as the coding window size W is reduced.

Obviously, the proposed model must be validated against real data obtained by running our prototype. To this end we perform a 300 s long trial with our prototype; at the end ot the experiment we collect and average the honest scores for all N_h honest and N_m malicious nodes to obtain values comparable with those from (19) and (20). The measured values are show in the rightmost part of Table II. It can be noted that the model prediction and the real data are in very good agreement. To further support the model prediction, in Fig. 3 we show the honest scores computed by a randomly chosen honest node for RNC ($W = k$) and BC with $W = k/2$. The honest scores are ordered on the x axis from the lowest to the highest and honest scores corresponding to malicious nodes are represented with a red circle. The dashed line represents the decision threshold in (5) for $\alpha = 2.0$. With RNC, on the average, honest scores are lower as correctly predicted by our model (see Table II). Moreover, the honest score of several malicious nodes are higher than the decision threshold, i.e. honest scores do not properly allow to discriminate between honest and malicious nodes. This is a consequence of having both s_h and s_p very close to $\frac{1}{2}$ ($s_h = 0.511453$ and $s_p = 0.47911$). Conversely, with BC the honest scores of malicious nodes are more clearly clustered around values lower than those of honest nodes. Such behavior is the key for blacklisting malicious nodes in reasonable time, as we will experimentally prove in the following.

W	model		s_j^i (measured)	
	s_p	s_h	Malicious	Honest
$\frac{k}{3}$	0.926249	0.948308	0.923849	0.95452
$\frac{k}{2}$	0.915887	0.935425	0.895789	0.927744
$\frac{2k}{3}$	0.886283	0.904738	0.852016	0.882849
k	0.463819	0.473402	0.47911	0.511453

TABLE II
HONEST SCORE PREDICTED BY (19) AND (20) (LEFT) AND VALUES COMPUTED BY PROTOTYPE (RIGHT) FOR VARYING W.

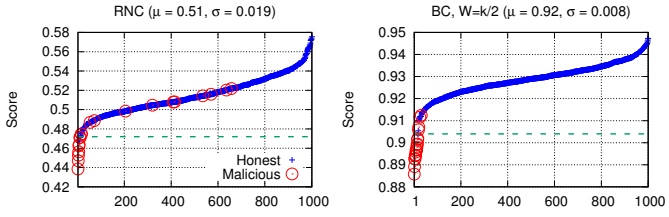

Fig. 3. Computed honest scores for different coding schemes (t_o = 300 s).

D. Malicious Nodes Identification

As a second experiment, we assess how precise is our scheme in identifying the malicious nodes as the *True Positives Ratio* (TPR) of the nodes identified as malicious. For each i-th network node we define the TPR as the fraction of true malicious nodes among the top-$|S_i^m|$ nodes in the sorted scores list, where $|S_i^m|$ is the subset of malicious nodes actually seen by the i-th node according to the ground truth (in the following, we report the TPR averaged over all the network). Namely, we study the effect of the nodes observation time t_o and number of observations N_o available at a node.

First, after an *observation time* of t_o = 300 s , each node $\mathcal{N}_i$ requests $(N_o - 1)$ other random nodes their observation vectors and computes a sorted list of suspected malicious neighbors, albeit no node blacklisting is enforced not to bias the malicious nodes identification process. Fig. 4 (left) shows the TPR as a function of N_o (when N_o=1, a node relies on its own observations only). As a general trend, malicious nodes are identified with increasing precision (i.e., the TPR increases) as N_o increases. However, the precision of the RNC scheme increases linearly only with N_o: even with N_o=200 observations available at each node, the TPR barely exceeds 50%. With RNC, honest nodes relay a lot of polluted packets, as Fig. 2 (left) shows. Thus, collected observations are unreliable and useless to discriminate between the honest and the malicious nodes. On the contrary, the BC schemes precisely identify (TPR between 90% and 100%) the malicious nodes with just about N_o = 100 observations. With BC, honest nodes relay fewer polluted packets as Fig. 2 (left) shows. Thus, observations are more reliable, enabling to precisely identify malicious nodes.

Similarly, Fig. 4 (right) shows the TPR as a function of the observation time t_o when $N_o \leq 75$. With RNC, even large t_o increases bring however modest TPR improvements only. Fig. 2 (top) shows that honest nodes relay many polluted packets with the RNC scheme, thus the observations are unreliable. That is, with RNC it is difficult to identify the malicious nodes even over long observation time because the exchanged observations are not discriminative. Conversely, with BC the malicious nodes are precisely identified over short time (e.g.: with $W = k/3$, the TPR is equal to 97% after 300 seconds). Fig. 2 (top) shows in fact that the nodes relays a limited number of polluted packets, therefore the exchanged observations are reliable. That is, with BC the malicious nodes are quickly identified because the exchanged observations are highly discriminative.

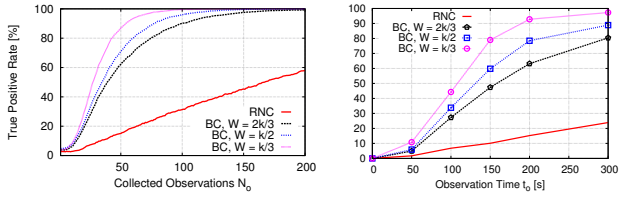

Fig. 4. TPR as a function of the number of observations collected from the other nodes N_o for different coding schemes (t_o = 300 s) and as a function of the observation time t_o for different coding schemes (N_o = 75).

E. Malicious Nodes Blacklisting

As a third and last set of experiments, we assess the effectiveness of our malicious nodes blacklisting scheme.

First, we evaluate to which extent blacklisting malicious nodes improves the video quality. Each node computes the score for each neighbor observed so far, t_{black} s after joining the streaming session. Next, the node blacklists the neighbors with lower score according to the criteria described in Sec. IV-E. In all the following experiments, we set α = 2.0 in (5), which yields the score threshold corresponding to the dashed line in Fig. 3. Such α value enables a reasonable tradeoff between blacklisting recall and precision for different coding parameters, as the figure suggests. We define as *Continuity Index* (CI) the average fraction of generations successfully recovered by all the nodes. Only generations recovered entirely, prior to the respective decoding deadlines and without polluted packets are successfully recovered. In the considered video streaming application, the CI corresponds to the inverse of the screen freezes frequency due to corrupted generations of video data. Fig. 6 (left) shows the CI as a function of the blacklisting time t_{black} and for different encoding window sizes W. The reported CI is computed, for each node, only over generations distributed by the server after t_{black}, i.e. after the blacklisting has been enforced. The case t_{black}=0 refers to the scenario where no blacklisting is performed at all and serves as a baseline. With RNC, the blacklisting mechanism does not allow to recover the full video quality for any t_{black}. As shown in Fig. 4 (right), even after collecting observations for 300 s the nodes cannot properly discriminate between honest and malicious nodes. Conversely, with BC the TPR increases together with t_{black}, so the blacklisting mechanism is increasingly more precise at excluding from the communication true malicious nodes. By t_{black} = 250 s, the CI is steadily above 90%, and by the end of the streaming session the malicious nodes are correctly identified and blacklisted ($W = \frac{k}{2}$). The experiment shows that packet counts observations collected with BC are discriminative enough to enable a simple blacklisting scheme to recover nearly-optimal video quality.

Second, we study the effect of an increase of the probability p_{poll} that a malicious node injects a bogus packet. Fig. 5 (left) shows the precision in malicious identification (TPR) as p_{poll} increases above 1%. As expected, the experiments confirm that the honest nodes relay more polluted packets to the network as p_{poll} increases. With RNC, the TPR drops as p_{poll} increases: as malicious nodes inject more polluted packets, the ambiguity between honest and malicious nodes increases and collected observations are less discriminative to identify the malicious nodes. On the contrary, with BC the TPR increases as p_{poll} increases: malicious nodes are in fact more likely to be correctly identified when they transmit more polluted packets to the network.
Fig. 5 (right) shows the corresponding post-blacklisting CI as a function of p_{poll} for t_{black} = 200s. As expected, the CI for the RNC scheme drops quickly as p_{poll} increases, i.e. the malicious nodes succeed in compromising the video communication. Conversely, with BC the CI increases as p_{poll} increases, with smaller window sizes W yielding better performance. This experiment shows that, with our proposed scheme, injecting more polluted packets in the network may be counterproductive for malicious nodes as they are more easily identified and isolated from the network.

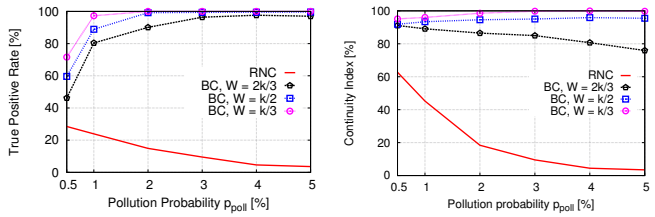

Fig. 5. Precision in malicious nodes identification (left) and video quality (right) as a function of the packet pollution probability p_{poll} (t_{black} = 200 s).

Finally, in Fig. 6 (right) we experiment with the centralized scheme in Sec. IV-F for $W = \frac{k}{2}$ and σ_p = 2. The centralized scheme identifies the malicious nodes restoring near-perfect video quality in only 200 s as the tracker collects observations from the whole network The distributed scheme achieves the same performance in about 300 s as network nodes can collects observations only from seen nodes. We also experiment with the malicious nodes attempting to throw blame on honest nodes. For each transmitted observation, each of the $N_m = 20$ malicious nodes relays bogus observations by randomly altering the observed nodes identifiers. The distributed scheme performance worsens as t_{black} increases as bogus observations cause the honest nodes scores to drop. Conversely, with our centralized scheme the number of correct observations available at the tracker is such that the weight of bogus observations is marginal only. Concluding, the centralized scheme secures the network against pollution attacks in less time and is more resilient to bogus observations in reason of the higher number of available observations at the price of deploying a centralized authority in the network.

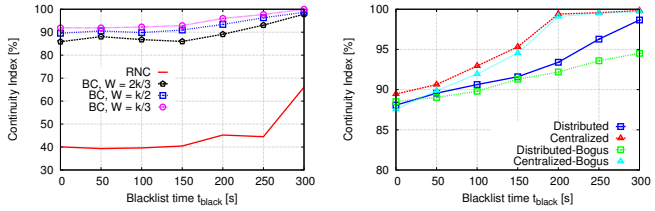

Fig. 6. Video quality as a function of blacklisting time t_{black} for different coding window sizes W_d (left) and of the observation time t_o for distributed and centralized proposed scheme, with and without bogus observations (right).

VII. Conclusions and Future Works

We proposed a distributed scheme for identifying and blacklisting malicious nodes on a probabilistic basis in pollution attacks to Network Coding (NC)-based video communications. Our approach scales with the network size because is totally distributed and is lightweight because it does not entail cryptographic primitives. We developed an analytical model to characterize the probability a node provides packets that

result in the decoding of a clean or a polluted generation as a function of the BC coding window W to show its impact on the accuracy and effectiveness of our scheme. We also showed that our method allows each node to estimate this probability for all other nodes. We experimented with P2P video streaming, demonstrating that our model is accurate and that our approach enables to precisely isolate the malicious nodes and restore the security of a video communication in reasonable time. Notably, the malicious nodes are more likely to be correctly identified and isolated when they relay more polluted packets to the network.

REFERENCES

[1] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] S. Bassel and Dongyu Qiu, "Performance analysis of network-coding-based p2p live streaming systems," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2140–2153, 2016.

[3] CISCO Inc., "White paper: Cisco VNI forecast and methodology, 2015-2020," 2016.

[4] 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.

[5] 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.

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

[7] Q. Li, D.-M. Chiu, and J.C.S. Lui, "On the practical and security issues of batch content distribution via network coding," in *ICNP*, 2006.

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

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

[10] F. Oggier and H. Fathi, "An authentication code against pollution attacks in network coding," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1587–1596, 2011.

[11] A. Esfahani, G. Mantas, and J. Rodriguez, "An efficient null space-based homomorphic mac scheme against tag pollution attacks in rlnc," *IEEE Communications Letters*, vol. PrePrint, no. 99, 2016.

[12] 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.

[13] 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.

[14] 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.

[15] A. Cohen, S. Dolev, and N. Tzachar, "Efficient and universal corruption resilient fountain codes," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4058–4066, 2013.

[16] A. Fiandrotti, R. Gaeta, and M. Grangetto, "Characterization of band codes for pollution-resilient peer-to-peer video streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 6, pp. 1138–1148, June 2016.

[17] 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 Trans. on Multimedia*, vol. 16, no. 2, pp. 521 – 532, Feb. 2014.

[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.

[19] X. Wu, Y. Xu, C. Yuen, and L. Xiang, "A tag encoding scheme against pollution attack to linear network coding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 33–42, 2014.

[20] C. Cheng, J. Lee, T. Jiang, and T. Takagi, "Security analysis and improvements on two homomorphic authentication schemes for network coding," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 993–1002, May 2016.

[21] D. Kim, J. Bi, A. V. Vasilakos, and I. Yeom, "Security of cached content in ndn," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 2933–2944, Dec 2017.

[22] 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.

[23] 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, October 2013.

[24] 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.

[25] W. Tong and S. Zhong, "A unified resource allocation framework for defending against pollution attacks in wireless network coding systems," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2255–2267, Oct 2016.

[26] A. Fiandrotti, A. M. Sheikh, and E. Magli, "Towards a p2p videoconferencing system based on low-delay network coding," in *EUSIPCO*, 2012, pp. 1529–1533.

[27] 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.

[28] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Higher Education, 4 edition, 2002.

[29] B. Bollobás, *Random Graphs*, Cambridge University Press, 2001.

Attilio Fiandrotti (M'12) received his M.Sc. and Ph.D. degrees in Computer Science from Politecnico di Torino in 2005 and 2010 respectively. Currently, he is Maître de Conférences at Télécom Paristech, Université Paris Saclay, Image Data Signals department, Multimedia group. His current research activities include adaptive HTTP video streaming, robust cooperative multimedia distribution and deep learning techniques for image and video analysis and compression.

Rossano Gaeta received his Laurea and Ph.D. degrees in Computer Science from the Università di Torino, Italy, in 1992 and 1997, respectively. He is currently Associate Professor at the Computer Science Department, Università di Torino. He has been recipient of the Best Paper award at the 14-th IEEE/ACM MASCOTS 2006 and at the 26th IFIP PERFORMANCE 2007. His current research interests include the analysis of coding techniques in distributed applications and the modeling of information diffusion in online social networks.

Marco Grangetto (S'99—M'03—SM'09) received his Electrical Engineering degree and Ph.D. degree from the Politecnico di Torino, Italy, in 1999 and 2003, respectively. He is currently Full Professor at the Computer Science Department, Università di Torino. His research interests are in the fields of multimedia signal processing and networking. In particular, his expertise includes wavelets, image and video coding, data compression, video error concealment, error resilient video coding unequal error protection, and joint source channel coding.