Thwarting Pollution Attacks in Network Coding for Delay Tolerant Mobile Social Networks

Lucile Sassatelli
Université Côte d'Azur, CNRS, I3S
Sophia Antipolis, France
sassatelli@i3s.unice.fr

Muriel Médard
Massachusetts Institute of Technology
Cambridge, MA, USA
medard@mit.edu

ABSTRACT

We consider Delay Tolerant Mobile Social Networks (DTMSNs), made of wireless nodes with intermittent connections and clustered into social communities. The lack of infrastructure and its reliance on nodes' mobility make routing a challenge. Network Coding (NC) is a generalization of routing and has been shown to bring a number of advantages over routing. We consider the problem of pollution attacks in these networks, that are a very important issue both for NC and for DTMSNs. Our first contribution is to propose a protocol which allows controlling adversary's capacity by combining cryptographic hash dissemination and error-correction to ensure message recovery at the receiver. Our second contribution is the modeling of the performance of such a protection scheme. To do so, we adapt an inter-session NC model based on a fluid approximation of the dissemination process. We provide a numerical validation of the model. We are eventually able to provide a workflow to set the correct parameters and counteract the attacks. We conclude by highlighting how these contributions can help secure a real-world DTMSN application (e.g., a smart-phone app.).

CCS Concepts

•**Networks** → **Network performance modeling;** *Mobile ad hoc networks;* •**Security and privacy** → *Mobile and wireless security;*

Keywords

Network coding, Inter-session, Delay-tolerant networks, Pollution attacks, Opportunistic routing, Fluid models

1. INTRODUCTION

We consider intermittently connected networks made of human-carried wireless devices, and whose physical meeting patterns make cluster into social communities. We abbreviate them by Delay Tolerant Mobile Social Networks (DTMSNs), that are Delay Tolerant Networks (DTNs) with heterogeneous mobility. Considering that the mobility is characterized by the distribution of inter-contact times between the node pairs [13], we refer to "heterogeneous" (resp. "homogeneous") when this distribution varies (resp. does not vary) over the node pairs. The three main goals of DTMSNs in civilian applications can be deemed as: (i) to provide network access to remote communities (e.g., Bytewalla [1]), (ii) to provide cheaper content access by file exchange in ad hoc mode (e.g., PSN [11, 28], Liberouter [2]), (iii) to offload the cellular networks (e.g., [14]).

In order to decrease the transmission delay, a source of traffic has to rely on the mobility of other nodes which act as relays. In particular, if multiple copies of the same packet are allowed to spread in the network then the packet delay will be lowered. How to spread multiple copies has been investigated in several proposals to save energy and memory resources. Next we consider Spray-and-Wait (SaW) [25]. Beyond mere replication is Network Coding (NC), a networking paradigm which is a generalization of routing [3, 17] in that it allows intermediate nodes to modify the packets' payload by combining the incoming packets to forge outgoing packets. Specifically, random NC has attracted an increasing interest for DTNs [19]. The benefits are increase in throughput, as well as adaptability to network topology changes and resilience to link failures. The successful reception of information does not depend on receiving a specific packet anymore, but on receiving a sufficient number of independent packets, thereby circumventing the coupon collector problem that would emerge with single repetition of packets. There are two types of NC: intra-session NC codes (i.e., combines) together only packets belonging to the same session or connection, while Inter-Session NC (ISNC) combines packets pertaining to different sessions.

We consider pollution attacks which are a critical issue for non-protected content such as public distributed multimedia content. The packet mixing involved in NC makes a localized pollution attack likely to contaminate the whole network: such mixing may result into more corrupted packets at the destination than the number of packets that have been injected by the adversary. The risk of pollution is intensified in DTMSNs where all nodes are a priori welcome to participate in the relaying as the more relays, the more routing opportunities. Specifically, we consider the case where traffic is injected into the network by one or several nefarious nodes. Our contribution in this regard is two-fold:

- We propose a protocol which allows controlling adversary's network capacity by combining cryptographic hash dissemination and error-correction to ensure message recovery at the receiver.
- We model the performance of such a protection scheme by adapting an inter-session NC model based on a fluid approximation of the dissemination process in a DTMSN model. We provide a numerical validation of the model.

The paper is organized as follows. After Sec. 2 on the literature,

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

we first present the error-correcting scheme in Sec. 4, to identify the requirements to thwart nefarious nodes. We then we present in Sec. 5 a NC-compliant signature scheme. Eventually we detail how to distribute the hashes with authentication in Sec. 6. We then express in Sec. 7 the performance in terms of the protection parameters: the number N_s of nodes holding the public hash, the dimension K_1' of the vector space sent out by the legitimate source (stemming from the correction-control scheme). We conclude by highlighting how these contributions can help secure a real-world DTMSN application (e.g., a smart-phone app.).

2. RELATED WORKS

The authors of [29], [7] and [5] address the vulnerability of content distribution systems using NC. To thwart attacks by malicious nodes sending bad packets, they propose signature schemes for NC that are secure and allow nodes to check easily the integrity of the received packets. In [15], the authors consider error-correction NC, homomorphic signatures and error-detection hashes for NC and their relative usefulness under different scenarios. However, the combination of these techniques, as we propose here, is not explored. Ho *et al.* in [10] propose a scheme for detecting adversarial modifications in network using random linear NC. This approach assumes that the packets headers storing the linear combination coefficients are not polluted by the adversary, thereby allowing the destination to know the coding coefficients. Here, we consider the more general case, where the whole coded packet can be polluted, even in the headers. The authors of [26] introduce a broadcast-mode transformation of the network, which changes the multicast capacity of the network. However that method cannot apply to DTN because trusted gateways are required. In [16, 12], the problem of error-control in random linear NC is considered. The construction of an error-correction coding scheme for NC is described, and the conditions of successful decoding are proven. These schemes apply when the packets can be polluted entirely, i.e., the coding coefficients are modified by the adversary. Building on the NC-compliant signature scheme of [29] and on the error-control scheme of [16], we propose and optimize a protocol whose goal is twofold: (i) control the capacity of the adversary towards the destination so as to ensure the capacity of the legitimate source and (ii) achieve capacity, i.e., the maximum rate at which the destination can still recover information from the source. This is achieved by combining cryptographic hash dissemination and error-correction. We build the performance model upon the fluid model for ISNC presented in [23].

3. NETWORK MODEL

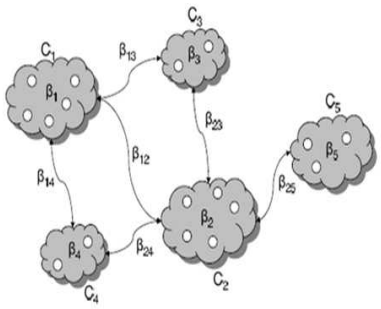

Figure 1: The multi-community model considered [6].

Symbol	*Meaning*
Network settings	
N	total number of nodes excluding the sources and the destinations
C	number of node communities
N_i	number of nodes in community i
β_{ij}	inter-meeting intensity of a node in community i with a node in community j
Bw	bandwidth: mean number of packets that can be exchanged during a contact in each direction
Communication settings	
S_1, A	source node of session 1, adversary node
D_1	destination node of session 1
K_1, K_2	number of information packets of session 1, 2
K_1', K_2'	maximum number of packets that can be released by S_1, S_2
$K_{S_1}(t)$, $K_A(t)$	number of indices realeased by S_1 and adversary A till time t
$R(t)$, $S(t)$	number of indices originally generated by S_1 (resp. A) received by D_1 by time t
M, Q	maximum number of copies of an index released by S_1, S_2
S_{11}, S_{22}	set of indices associated to pure payloads sent out by source S_1, S_2
X_{ic}, Y_{ic}	number of nodes in community c that carry i indices in S_{11} (resp. S_{22})
$\tilde{X}_{Ic}$, $\tilde{Y}_{Ic}$	number of nodes in community c that carry index I of S_{11} (resp. S_{22})
l	$l = \sum_{i=11,22} l_i$ for a $(c, \mathbf{l})$-node

Table 1: Main notation used for the model of ISNC.

We consider a network made of N nodes grouping into C communities (see Figure 1). We assume that the number of meetings per unit of time between two given nodes is invariant over time and Poisson distributed, according to the findings of [13]. The average of this distribution is named inter-meeting intensity. We consider that all nodes pertaining to the same community i have the same inter-meeting intensity β_{ij} towards any other node of community j. The concept of community imposes that $\beta_{ii} > \beta_{ij}$, for all $i \neq j$.

A source S_1 of community s_1 wants to send a file to its respective destination D_1 in community d_1. We assume that the file to be transferred needs to be split into K_1 packets: this occurs owing to the finite duration of contacts among mobile nodes or when the file is large with respect to the buffering capabilities of the nodes. The message is considered to be well received if and only if all the K_1 packets of the source are recovered at the destination. To do so, $K_1' > K_1$ Random Linear Combinations (RLCs) of the K_1 packets are spread by the source, each associated with a certain index. We do not assume any feedback. For the modeling, we assume that the number of packets that can be exchanged during a contact in each direction (thereby accounting both for the rate and the contact duration), is stochastic and follows any known distribution of mean Bw. The buffer size is assumed to be any known integer, denoted by B, equal for all the nodes in the network. Note that these assumptions are not necessary for the protection scheme to hold, but are general enough to get a meaningful model. A node is said to be a $(c, \mathbf{l})$-node, if it belongs to community c and has in its buffer $\mathbf{l} = (l_{11}, l_{22})$ indices of S_{11}, S_{22}, respectively. All notations are gathered in Table 1.

We consider a given session of interest (so-called "legitimate session") that is attacked. We consider the Byzantine form of pollution attacks that are described in [12, 16]. That means that the adversary has full knowledge of the legitimate packets, and tries to forge fake packets so as to impede the transmission as much as it can.

4. CODING FOR ERRORS IN RANDOM NETWORK CODING

The error-correction scheme, presented by Koetter and Kschischang in [16], is based on noticing that the recovery of the information packets at the receiver is possible as soon as any generating set of the space spanned by the information packets, called information space, is received. Hence, the corrupted packets introduced by the adversary prevent from decoding the information packets as soon as they are not in the information space. The error-correction coding at the source is thus meant to choose what vector space has to be sent, rather than which vectors (packets).

The flow of encoding and the corresponding notations used consistently throughout the paper is depicted in Figure 2.

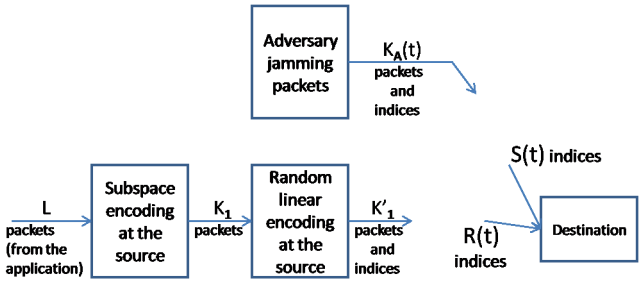

Figure 2: The flow of encoding.

Let W be a vector space over a finite field. A code with ambient space W is a nonempty collection of subspaces of W. The authors of [16] provide a Reed-Solomon-like code construction and decoding algorithm. We briefly describe the encoding technique in a simple way, and refer the reader to [16] for complete development. Let $\mathbb{F} = \mathbb{F}_{q^m}$ be a finite extension field of $\mathbb{F}_q$ (in practice, q is often taken to be 256). Let $\mathbf{u} = (u_0, \ldots, u_{L-1})$ denote a block of L information packets, each of size m over $\mathbb{F}_q$, hence each component of $\mathbf{u}$ pertains to $\mathbb{F}$. Let $A = \{\alpha_1, \ldots, \alpha_{K_1}\}$ be a set of linearly independent elements in $\mathbb{F}$, that is each $\alpha_i, i = 1, \ldots, K_1$, is a vector of m elements over $\mathbb{F}_q$. These elements span a K_1-dimensional vector space $\langle A \rangle$ over $\mathbb{F}$. We have $K_1 \leq m$. The considered ambient space, whose a subspace will correspond to a codeword, is the direct sum $W = \langle A \rangle \oplus \mathbb{F} = \{(\alpha, \beta) : \alpha \in \langle A \rangle, \beta \in \mathbb{F}\}$, a vector space of dimension $K_1 + m$ over $\mathbb{F}_q$. The codeword V is then the vector subspace of W, whose a generating set is made of v_j, $j = 1, \ldots, K_1$, with $v_j = [\alpha_j, \beta_j]$ and $\beta_j = \sum_{i=0}^{k-1} u_i \alpha_j^{q^i}$ (all operations are finite field operations, and $[.,.]$ denotes the concatenation of two row-vectors). Hence V is a subspace of W of dimension K_1.

In this adversarial environment where an adversary can pollute any bytes of a coded packet (both coding coefficients and payload), it is shown in [16] that the criterion for recovering the L packets of the legitimate session with this error-correction scheme is:

$$dim(U \cap V) \geq L + dim(U \backslash (U \cap V)) \ , \qquad (1)$$

where V is hence the coded subspace generated at the source, and U is the space received at the destination (combination of a subspace of V and a subspace of what the adversary sent). We lower-bound $dim(U \cap V)$ by $R(t)$ and we upper-bound $dim(U \backslash (U \cap V))$ by $K_A(t)$, where $K_A(t)$ is the number of degrees of freedom (RLCs) that have been spread out by the adversarial nodes up to time t. **The considered recovery criterion is hence** $R(t) \geq L + K_A(t)$. More details are provided in Sec. 7.

The problem addressed in the next section is that of secure network capacity: we see from the recovery criterion of equation 1 that, if the adversary gets access to too many relay nodes, then the successful recovery may never be possible.

5. OBTAINING A NON-ZERO CAPACITY FOR THE SECURE CHANNEL

Considering a static network and its corresponding graph, Jaggi *et al.* in [27] show that when the adversary can eavesdrop on all links and pollute z_0 links, the secure network capacity is $C - 2z_0$ [27], where C is the network capacity as defined in [3].

In DTNs, we cannot know, prior to communication, what are going to be the different opportunistic paths, from source and adversaries to the destination. However, the capacity of a DTN can be defined. Garetto at al. [9] proposed a definition of the network capacity of a DTN, by considering the corresponding graph whose nodes are those of the DTN, and an edge between two nodes is weighted by the average number of meetings per unit of time between these two nodes, i.e., their intra-meeting intensity. The authors of [9] are then able to define the network capacity as the min-cut of this graph, for a given source-destination pair. It is interesting to note that the average (over network meeting realizations) of $dim(U \backslash (U \cap V))$ relates to the min-cut capacity between the adversary and the destination. We refer the reader to [12] for a study of network capacities in the presence of Byzantine nodes, and only present shortly in the next section the option we choose for our system design to guarantee a desired non-zero secure capacity (i.e, the maximum rate at which the destination can recover information).

Our goal in this section is to ensure that the probability that D recovers the original K_1 packets of S can be greater than zero, no matter how omniscient and smart the adversary is. The probability of recovering the source packets is then expressed in Sec. 7. The recovery is possible only if the receiving rate of A's packets is no more than that of S, that is if set of relays the adversary can use is a strict subset of those the source can use. In what follows, we refer to this condition as having a non-zero secure capacity for the DTN.

In order to obtain it by another means than authentication, since we assume that the nodes do not systematically share symmetric keys, we need to make some nodes able to detect and not to relay the adversary packets. To do so, we use the signature scheme for content distribution with NC, presented by Zhao *et al.* in [29]. This is a homomorphic signature scheme that allows nodes to verify any linear combinations of pieces without contacting the original sender. As previously introduced, the vectors that are sent over the network, after error-correction encoding, are v_j in $\mathbb{F}_q^{l+m}$, $j = 1, \ldots, l$. They are the generating set of V. A received packet is a valid linear combination if and only if it belongs to V. We briefly describe the approach of [29].

Let p be a large prime such that q is a divisor of $p - 1$. Let g be a generator of the group of order q in $\mathbb{F}_p$. Let consider the random set of elements in $\mathbb{F}_p^*$: $K_{pr} = \{a_i\}_{i=1,\ldots,l+m}$, which is called the private key. As well, let $\mathbf{x}^{(1)} = \{h_i = g^{a_i}\}_{i=1,\ldots,l+m}$.

The scheme works as follows:

- The source finds a vector $\mathbf{y}$ that is orthogonal to all vectors in V.
- The source computes vector $\mathbf{x}^{(2)} = (y_1/a_1, \ldots, y_{l+m}/a_{l+m})$.
- The source publishes the public hash $H_{pu} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$.
- When a node, that has received the public hash, receives a

vector $\mathbf{w}$ and wants to verify that $\mathbf{w}$ is in V, it computes

$$d = \prod_{i=1}^{l+m} h_i^{x_i^{(2)} w_i}$$

and verifies that $d = 1$.

H_{pu} can be considered as a public key. However, it is crucial to notice that the term *public key*, in this context, does not refer to asymmetric encryption, but instead to the parameters a hash has to be computed with. Indeed, evaluating a hash function is, generally, significantly cheaper than performing encryption [22]. That is why, in the remainder, we make use of the term *public hash* instead of *public key* in order to avoid any confusion between single hash computation and asymmetric encryption.

If the public hash H_{pu} is received by all the network nodes, no error-correction such as that presented in the previous section is needed as it is impossible for an adversarial node to send nefarious packets to any relay node. However, in DTNs, as the public hash has to be disseminated to a number of nodes, we cannot ensure that all those nodes will obtain the public hash prior to transmission and within a certain limited delay.

To cope with this problem, the source chooses two parameters, N_s and P_h: the public hash H_{pu} is sent out by the source which (periodically) waits for a certain time so as to ensure that N_s nodes got H_{pu} with probability P_h. The public hash can be disseminated by any means, such as SaW with a maximum spraying counter N_s. The nodes having the hash are referred to as *secure nodes* thereafter. The number of nodes having received the hash determines the secure network capacity, as explained in the beginning of this section. The workflow to determine the desired protection and transmission parameters is shown in Figure 3.

The next section discusses the security problems that can arise in disseminating the public hash H_{pu}.

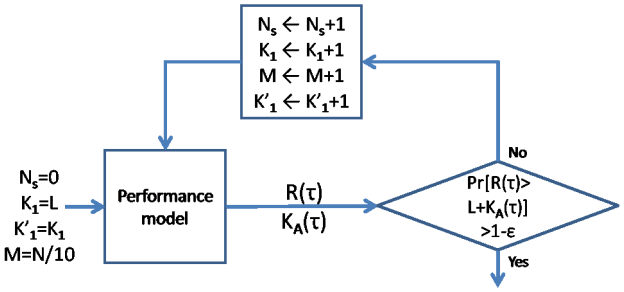

Figure 3: The general process to choose the protection and transmission parameters N_s, K_1 and M. The input are (i) initial values as exemplified on the left-hand side, (ii) the other network parameters needed by the performance model of Sec. 7, and (iii) some deadline τ and recovery probability $1 - \epsilon$ by this deadline. Once the process has succeeded, the actual transmission of first the hash then the data can get started.

6. SECURITY ASSUMPTIONS AND SYMMETRIC KEY PRE-DISTRIBUTION

The first problem one can think of regarding the above scheme is the case where adversarial nodes can disseminate fake public hashes to the nodes so as to cause Denial of Service (DoS) attack: the legitimate message will not be relayed anymore since the nodes with fake hashes will reject the legitimate packets. In infrastructure networks, the public hash is usually signed with any signature scheme, thereby requiring a Trusted Third Party such as a Certification Authority. In MANETs, such an infrastructure is not available, and the usual way to meet the authentication requirements is the use of symmetric keys. Thus, in delay tolerant MANETs, to thwart DoS attack performed by adversarial nodes sending fake public hashes, we have to assume that a fraction of the nodes in the network is preloaded with keys, so that authentication of the public hash sender can be achieved thanks to symmetric key cryptography. Thus, only the nodes preloaded with the keys drawn from a certain common pool are able to get the public hash. We assume the symmetric-key pre-distribution scheme of Eschenauer and Gligor [8]. The public hash does not have to be encrypted, it has only to come with an extra hash allowing authentication, such as provided by the HMAC function of the Message Authentication Code (MAC) [4]. The extra hashes will be changed from relays to relays. Nodes will never accept public hash without successful authentication. The key pre-distribution of [8] works in the following way. First, a large pool of P keys (e.g., $2^{17} - 2^{20}$ keys) and their identifiers is generated. Then b keys are randomly drawn out of P without replacement, for each node to be loaded. When two nodes have to communicate securely, i.e., in our context, to authenticate each other, they need to discover if they share a key. To do so, each node broadcasts, in clear text, the list of identifiers of the keys in its key ring. During the public hash broadcast phase, a relay node must not accept any fake hash upon meeting with the adversary. To avoid such a situation, the key-sharing patterns can be hidden from an adversary [8]. For instance, for every key on a key ring, each node could broadcast a list $E_{K_i}(\alpha)$, $i = 1, \ldots, b$, where α is a challenge. The decryption of $E_{K_i}(\alpha)$ with the proper key by a recipient would reveal the challenge α and establish a shared key with the broadcasting node, thereby preventing A from misleading R. It is shown in [8] that the probability that any two nodes, loaded with key-rings, share at least a key is:

$$P_s = 1 - \frac{(P-b)!^2}{(P-2b)!P!} .$$

Let us consider a time-slot between two consecutive contacts of a loaded node ν_1 with any two other loaded nodes ν_2 and ν_3, both sharing a key with ν_1. If we assume that only a fraction r of the N network nodes have been loaded with keys, then the mean time between two consecutive meetings of a given node is $\frac{1}{\beta P_s r N}$. Since the average number of mean times to disseminate N_s copies of a packet is $\lceil \log_2(N_s) \rceil$ [25], we can estimate the time needed to spread N_s copies of the public hash by: $\frac{\lceil \log_2(N_s) \rceil}{\beta P_s r N}$. Hence, the source will wait for this delay, plus a possible margin, before starting to send the message. These simple formula can be extended for the multi-community case considering [23] or [20].

Once the adversary cannot mislead the network with fake public hashes anymore, the second problem that can arise is the adversary flooding the network with wrong packets even before all the intended number of nodes have received the public hash, so as to prevent the destination from decoding. However, the injection of nefarious packets is possible only if the adversarial node has snooped over at least one packet from the source, so as to retrieve the identifiers of the unicast session of interest, and include them in the forged packets to pollute the header the the legitimate packets. In order to prevent such an attack, the identifier of the unicast session must be hidden during the public hash dissemination phase, this being done thanks to the symmetric keys described above.

Thus, the public hash is broadcast without being hidden, with an extra hash allowing public hash sender authentication (like that provided by HMAC), and comes with the identifiers of the session which is the only part to be encrypted to remain confidential as long

as the communication has not started. Once the source has started to send data packets, nothing is encrypted and adversarial nodes can send corrupted packets with the right identifiers.

Eventually, it is worth highlighting why a public hash for integrity check and symmetric keys for authentication are both needed. The public hash held by authenticated nodes allows these nodes to accept packets from unauthenticated nodes but to filter and relay them only if they are not polluted. This allows to get the most out of the relaying capability of the network, compared with the case where only authenticated nodes would be allowed to act as relays. Another possibility would be to force nodes to code correctly, as presented in [21], but this requires different cryptographic capabilities.

7. PERFORMANCE ANALYSIS IN PRESENCE OF POLLUTION ATTACKS

We now express performance in terms of the protection parameters: the number N_s of nodes holding the public hash and the dimension K_1 of the vector space sent out by the legitimate source (stemming from the correction-control scheme). To do so, we first present briefly the ISNC model of [23] (the reader is referred to [23] for details).

7.1 Inter-session NC and modeling

Figure 4 depicts the process of ISNC missing when, e.g., a node with full buffer decides to still receive packet of session 2 by mixing it with a packet of session 1 it already holds.

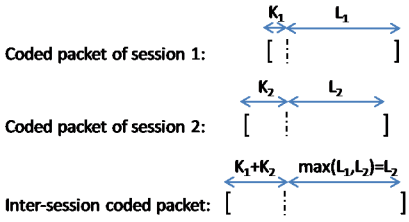

Figure 4: The basic operation performed on two packets with ISNC [23].

All nodes can identify the session number of each packet. The decoding criterion can be stated as follows. The original K_1 packets of session 1 can be recovered if and only if the matrix made of the coding coefficients (i.e., packets' headers) can undergo a Gauss-Jordan elimination resulting in (i) only elements of the K_1-size identity matrix over the K_1 columns assigned to session 1, and (ii) the other elements on the rows corresponding to these identity elements are zeros. Thereafter, the number of received Degree of Freedom (DoF) of session 1 is the number of identity elements over these K_1 columns.

The goal is to express the probability that both sessions are able to recover ("decode") their respective K_1 and K_2 original packets after a certain time τ. Unless otherwise mentioned, the notations are defined in Table 1. The quantities (in number of nodes) X_{ic} and Y_{ic}, $\tilde{X}_{Ic}$ and $\tilde{Y}_{Ic}$ are those involved in the decoding probability. Each network node is an object whose state is defined by the set of indices of each session stored in its buffer. We now present the ODEs defining the continuous-time mean field limit, called the fluid model. Its numerical accuracy is assessed in [23]. Below we consider the quantity variations when a node A may send packets to a node B upon meeting.

The ODEs for X_{ic} writes as follows. The ODEs for Y_{ic} can be deduced from those of X_{ic}, replacing 1 by 2 everywhere.

$$\frac{dX_{ic}(t)}{dt} = \beta_{s_1 c} N_c \sum_{\mathbf{l}^B} P_j(c,\mathbf{l}^B) P_{gs11}(i - l^B_{11}, \mathbf{l}^B) \ldots$$

$$+\beta_{cd_1} N_c \sum_{\mathbf{l}^B} P_j(c,\mathbf{l}^B) P_{ls11}(l^B_{11} - i, \mathbf{l}^B, \mathbf{l}^{D_1}) \ldots$$

$$+N_c \sum_{e=1}^{C} \sum_{\mathbf{l}^A,\mathbf{l}^B} \beta_{ec} N_e P_j(c,\mathbf{l}^A) P_j(c,\mathbf{l}^B) P_{grs11}(i - l^B_{11}, \mathbf{l}^A, \mathbf{l}^B) \ldots$$

$$-\beta_{cs_1} N_c \sum_{\mathbf{l}^B : l^B_{11} = i, p_{11} > 0} P_j(c,\mathbf{l}^B) P_{gs11}(p_{11}, \mathbf{l}^B) \ldots$$

$$-\beta_{cd_1} N_c \sum_{\mathbf{l}^B : l^B_{11} = i, p_{11} > 0} P_j(c,\mathbf{l}^B) P_{ls11}(p_{11}, \mathbf{l}^B, \mathbf{l}^{D_1}) \ldots$$

$$-N_c \sum_{e=1}^{C} \sum_{\substack{p_{11}\mathbf{l}^A, \\ \mathbf{l}^B : l^B_{11} = i}} \beta_{ec} N_e P_j(c,\mathbf{l}^A) P_j(c,\mathbf{l}^B) P_{grs11}(p_{11}, \mathbf{l}^A, \mathbf{l}^B) \, .$$

Let us briefly give the a high-level explanation of these equations. The first summation term in $\frac{dX_{ic}(t)}{dt}$ corresponds to the number of nodes transitioning into state (i,c) per unit of time thanks to a meeting with the source node. The second and third summation terms are similarly defined: thanks to a meeting with the destination and relay node, respectively. The last three negative summation terms correspond to symmetric quantities for the number of nodes leaving state (i,c). To ease the understanding, we use consistent subscripts where g, l and r refer to "gain" (of such an index), "loss" and "relay", respectively ($s11$ refers to the type of packet considered, X is for S_{11} indices, Y for S_{22}). The term $P_j(c,\mathbf{l})$ denotes the fraction of relay nodes that are $(c,\mathbf{l})$-nodes.

The ODEs for $\tilde{X}_{Ic}$ write as:

$$\frac{d\tilde{X}_{Ic}}{dt} = \sum_{e=1}^{C} \beta_{ce} N_e N_c A_{R11,e,c} + \beta_{s_1 c} N_c A_{S11,c} - \beta_{cd_1} \tilde{X}_{Ic} A_{D11,c} \, ,$$

where

- $A_{D11,c}$ is the fraction of nodes in community c that have I of S_{11} in their buffer and that drop it upon meeting with D_1,
- $A_{S11,c}$ is the fraction of nodes in community c that meet with S_1,
- $A_{R11,e,c}$ is the fraction of nodes in community c without index I of S_{11} that obtain I from a relay in community e.

We finally mention one last important intermediate quantity involved in the ones above: $P_{nic,e,c}\big(n_{11}, \mathbf{l}^A, \mathbf{l}^B, K_{S_1}(t), \mathbf{v}(t)\big)$ denotes the probability that for a $(e,\mathbf{l}^A)$-node and a $(c,\mathbf{l}^B)$-node, there are n_{11} indices of S_{11} at node A not in common with node B and whose corresponding spray-counters are still below M, when S_1 has already spread out $K_{S_1}(t)$ indices. As well, the expressions of the above terms are given in [23].

7.2 Modeling pollution attacks

The source nodes are now renamed from S_1 and S_2 to S (legitimate source) and A (adversarial node). For the sake of clarity, we consider only one community and only one legitimate session, which get mixed with the adversary packets. The difference

with the inter-session NC case is that the adversary forges packets with session identifiers identical to those of the legitimate session. Therefore, even if the network is operated with inter-session NC, the adversarial packets are not considered as pertaining to another session and the header part of the packets corresponding to the legitimate session gets corrupted. The source sends out K_1' RLCs of its K_1 original packets (see Figure 2). The K_1' RLCs span a vector space of dimension K_1, where K_1 is the dimension of the vector space V defined in Section 4. When sending out a packet, the adversary generates indices that can overlap those generated by the legitimate source, but with the same session identifier, rendering those indices indistinguishable by the nodes. Also, some nodes can reject the RLC if they are secured (i.e., hold the public hash).

Let us recall the successful decoding criterion introduced in Sec. 4: $R(t) \geq K_1 + K_A(t)$, with $R(t)$, K_1 and $K_A(t)$ representing, respectively, the number of different indices of S received at D by time t, the number of information packets of S to be recovered at D and the number of RLCs sent out by A by time t, as before. The main principle of our analysis, which is the core component of the flow process to determine the protection parameters (see Figure 3), is as follows. In Sec. 4, a space of dimension K_1 is created from the information space of dimension L, to counteract pollutions of intra-session NC. In layman's terms, subspace coding [16] can be loosely interpreted as preparing to accommodate for a second session that would be separable because high-dimensional intra-session NC (in dimension $K_1 + m$) would boils down to inter-session NC (in dimension $L + K_A(t)$). To put it differently, in machine learning/classification realms, it has been proven with the kernel theory that easier separability can be achieved in higher dimensions than that of the original data set. Hence, the term $dim(U \cap V)$ can be expressed as the number of rows, after a Gauss-Jordan elimination, with non-zero elements in the columns corresponding to the K_1-dimensional subspace of the $(K_1 + m)$-dimensional space. This is also called the number of DoFs, approximated by the number of indices initially generated by the source and received at D (with the same principle as the introduction of this technique in [18]). The same idea is used for $dim(U \backslash (U \cap V))$.

Below we detail how to extend the ISNC model of [23]. An important term in the node exchanges is P_{nic}, which stands for the probability that for two meeting nodes have a certain number of indices not in common. The expression of P_{nic} keeps the same but l_{11}^A on the right-hand side is replaced by $l_{11}^A + l_{22}^B$. Let s_r be the fraction of nodes that can receive at least one packet: $s_r = \sum_{\mathbf{l}: l_{11}+l_{22}<B} P_j(1,\mathbf{l})$. Let s_s be the fraction of nodes that can send at least one packet: $s_s = 1 - \frac{\sum_I \check{X}_I(t) + \sum_I \check{Y}_I(t)}{MK_S(t)+QK_A(t)}$ and let f_c be the fraction of contaminated nodes (i.e., holding packets involving DoFs of A):

$$\frac{df_c}{dt} = \beta(Nf_c s_s + 1)\left((1 - \frac{N_s}{N})s_r - f_c\right) .$$

We define P_{cgi0} and P_{ucus} as the probability that a node is contaminated given that no index of A is in its buffer, and the probability that a node is uncontaminated and unsecured given that no index of A is in its buffer, respectively.

$$P_{cgi0} = \begin{cases} \frac{(f_c - \sum_{\mathbf{l}: l_{22}>0} P_j(1,\mathbf{l}))}{\sum_{\mathbf{l}: l_{22}=1} P_j(1,\mathbf{l})} & \text{, if } \sum_{\mathbf{l}: l_{22}=1} P_j(1,\mathbf{l}) > 0, \\ 1 & \text{, otherwise .} \end{cases}$$

$$P_{ucus} = \begin{cases} \frac{1 - f_c - Ns/N}{\sum_{\mathbf{l}: l_{22}=1} P_j(1,\mathbf{l})} & \text{, if } \sum_{\mathbf{l}: l_{22}=1} P_j(1,\mathbf{l}) > 0, \\ 0 & \text{, otherwise.} \end{cases}$$

We then have the following changes, all other quantities but those mentioned above remaining the same.

- The expression of $\frac{dK_A(t)}{dt}$ has P_{sc} appended with

$$\Big((l_{22}^B > 0) + (l_{22}^B == 0)P_{cgi0} + (l_{22}^B == 0)P_{ucus}\Big) .$$

The same is appended to the expressions of P_{gs22} and A_{S22}.
- The expressions of P_{grs11}, A_{R11} and A_{R22} are appended with

$$\Big((l_{22}^A == 0)(1 - P_{cgi0}) + \big((l_{22}^A == 0)P_{cgi0} + (l_{22}^A > 0)\big) \dots .$$

$$\big((l_{22}^B > 0) + (l_{22}^B == 0)P_{cgi0} + (l_{22}^B == 0)P_{ucus}\big)\Big)$$

8. NUMERICAL RESULTS

We finally perform the numerical validation of this model adapted to the case of pollution attacks in DTNs operated with NC. In this section, we assess the accuracy of the fluid model above, that captures the effect of the joint control of routing and ISNC on various quantities. We consider a synthetic contact trace on which we run the polluted NC-based dissemination thanks to a discrete event simulator written in Matlab. The simulation results are averaged over 30 runs and the 95% confidence intervals are plotted. The trace is made of $N = 1000$ nodes, $C = 1$ for the sake of clarity of the curves and $\beta = 5.10^{-4}$. The buffer size is set to $B = 2$ packets. The bandwidth is Poisson distributed with mean $Bw = 3$ packets. The communication settings are: $K_1 = 1$, $K_1^{'} = 4$ and $M = Q = 50$.

The number of secured nodes (i.e., nodes holding the public hash), is set to $N_s = 800$, that is 80% of the nodes do not relay coded packets that have been polluted by the adversary. The evolution of the averages of $R(t)$ and $S(t)$ are plotted against time in Figure 5. The match between analytical and simulation results is verified. As well, Figure 6 shows the pdf of $R(t)$ and $S(t)$ obtained from the analysis compared against those obtained from the simulation. It shows the good prediction of the model, which can hence be used to optimize such a secure transmission scheme (see Figure 3). It is also worth noting that the model can be extended with no difficulty to the case of several communities, allowing to assess how much hash should be provided to each community.

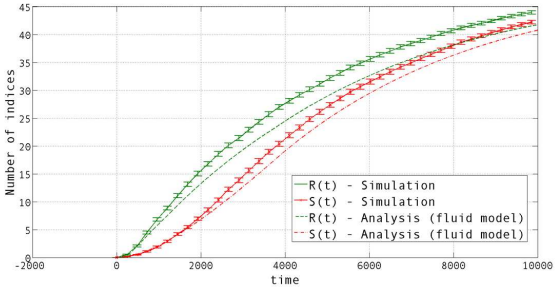

Figure 5: Average $R(t)$ and $S(t)$. PDF of $R(t)$ and $S(t)$ for $t = 2000$.

9. CONCLUSION

In this paper, we have proposed a theoretically-founded method to secure information sharing within a delay tolerant mobile social network. We have provided a performance model derived from a fluid approximation for inter-session NC. A number of papers (see [23] and references therein) show that, unlike node-to-node contacts, social interactions are much less volatile features which can

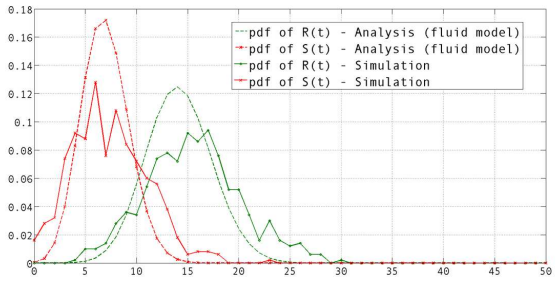

Figure 6: PDF of $R(t)$ and $S(t)$ for $t = 2000$.

be leveraged to design, e.g., efficient routing in DTMSNs. Furthermore, these features can be well estimated online by the nodes [24, Sec. V.E], as they are sufficiently persistent in time. The nodes can in turn use these estimates to decide their protection parameters upon the detection of pollution thanks to any independent intrusion detection system, or with conservative estimates of polluting nodes' density.

10. REFERENCES

[1] Bytewalla DTN on Android Phones.

[2] Liberouter: Towards autonomous neighborhood networking.

[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Inf. Theory*, 46(4):1204–1216, Jul. 2000.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Message authentication using hash functions: the HMAC construction. *CryptoBytes*, 2(1), 1996.

[5] D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *ACM Intern. Conf. on Practice and Theory in Public Key Cryptography (PKC)*, pages 68–87, Irvine, CA, USA, Mar. 2009.

[6] E. Bulut, W. Zijian, and B. Szymanski. Impact of social networks in delay tolerant routing. In *IEEE GLOBECOM*, Honolulu, HI, December 2009.

[7] D. Charles, K. Jain, and K. Lauter. Signatures for network coding. In *40th Annual Conf. on Information Sciences and Systems (CISS)*, Princeton, NJ, Mar. 2006.

[8] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *ACM Conf. on Computer and Comm. security*, pages 41–47, Washington, DC, USA, 2002.

[9] M. Garetto, P. Giaccone, and E. Leonardi. On the capacity region of MANET: Scheduling and routing strategy. *IEEE Trans. on Vehicular Technology*, 58:1930–1941, 2009.

[10] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Karger. Byzantine modification detection in multicast networks with random network coding. *IEEE Trans. on Inf. Theory*, 54(6):2798 – 2803, Jun. 2006.

[11] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *ACM SIGCOMM workshop on DTN (WDTN)*, Aug. 2005.

[12] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros. Resilient network coding in the presence of byzantine adversaries. *IEEE Trans. on Inf. Theory*, 54(6):2596 – 2603, Jun. 2008.

[13] T. Karagiannis, J.-Y. L. Boudec, and M. Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *ACM MobiCom*, Montreal, Canada, Sep. 2007.

[14] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou. Microcast: Cooperative video streaming on smartphones. In *ACM MobiSys*, Jun. 2012.

[15] M. Kim, L. Lima, F. Zhao, J. Barros, M. Médard, R. Koetter, T. Kalker, and K. Han. On counteracting byzantine attacks in network coded peer-to-peer networks. *IEEE Journal on Selected Areas in Communications: Special Issue on Mission-Critical Infrastructure*, 28(5):692–702, May 2010.

[16] R. Koetter and F. Kschischang. Coding for errors and erasures in random network coding. *IEEE Trans. on Inf. Theory*, 54(8):3579 – 3591, Aug. 2008.

[17] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. on Netw.*, 11(5):782–795, 2003.

[18] Y. Lin, B. Li, and B. Liang. Efficient network coded data transmissions in disruption tolerant networks. In *IEEE Conf. on Computer Comm. (INFOCOM)*, Phoenix, AZ, Apr. 2008.

[19] Y. Lin, B. Li, and B. Liang. Stochastic analysis of network coding in epidemic routing. *IEEE Journal on Selected Areas in Comm.*, 26(5):794–808, June 2008.

[20] A. Picu and T. Spyropoulos. DTN-Meteo: Forecasting the performance of DTN protocols under heterogeneous mobility. *IEEE/ACM Trans. on Netw.*, 23(2), Apr. 2015.

[21] R. A. Popa, A. Chiesa, T. Badirkhanli, and M. Médard. Going beyond pollution attacks: Forcing byzantine clients to code correctly. Technical report, MIT, 2011.

[22] T. Roeder and F. B. Schneider. Hashes and message digests. In *Lecture Notes*, Cornell University, NY, USA, 2005.

[23] N. Shrestha and L. Sassatelli. On control of inter-session network coding in delay-tolerant mobile social networks. In *ACM MSWiM*, Montreal, Canada, Sep. 2014.

[24] N. Shrestha and L. Sassatelli. Inter-session network coding-based policies for delay tolerant mobile social networks. *IEEE Tran. on Wireless Comm.*, PP(99), 2016.

[25] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient Routing in Intermittently Connected Mobile Networks: The Multi-copy Case. *IEEE/ACM Trans. on Netw.*, 16(1):63–76, Feb. 2008.

[26] D. Wang, D. Silva, and F. R. Kschischang. Constricting the adversary: A broadcast transformation for network coding. In *Allerton Conference on Comm., Control, and Computing*, Monticello, IL, Sep. 2007.

[27] R. W. Yeung and N. Cai. Network error correction, part I: Basic concepts and upper bounds. *Comm. in Information and Systems*, 6:2006, 2006.

[28] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *ACM MSWiM*, Chania, Greece, Oct. 2007.

[29] F. Zhao, T. Kalker, M. Médard, and K. J. Han. Signatures for content distribution with network coding. In *IEEE Int. Symp. on Inf. Theory*, pages 556–560, Nice, France, Jun. 2007.