Computer Communications 35 (2012) 48–61

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Data obfuscation with network coding ☆

A. Hessler [a], T. Kakumaru [a], H. Perrey [b], D. Westhoff [b,*]

[a] NEC Europe Ltd., Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
[b] Hamburg University of Applied Sciences (HAW), Department of Computer Science, Berliner Tor 7, 20099 Hamburg, Germany

ARTICLE INFO

Article history:
Received 17 June 2010
Received in revised form 21 September 2010
Accepted 10 November 2010
Available online 2 December 2010

Keywords:
Data obfuscation
Fountain code
Privacy homomorphism
Content distribution

ABSTRACT

Network coding techniques such as fountain codes are a promising way to disseminate large bulks of data in a multicast manner over an unreliable medium. In this work we investigate how to conceal such an encoded data stream on its way to numerous receivers with a *minimum investment*. Compared to conventional 'encrypt – encode/decode – decrypt' approaches, our solution is preferable for two reasons: First, it causes less CPU investment for encryption and decryption proportional to the ratio of the payload length to the signaling data length. Second, besides obfuscating the payload data from an eavesdropper, we hide the coding information from an eavesdropper. We evaluate the approach with respect to its application to various data types like MPEG-2 video streams and Java bytecode.

1. Introduction

In network coding transmission, an intermediary node can combine different packets together in order to reduce the number of transmissions, and thus increase the overall throughput of the network. To disseminate bulks of data over an unreliable medium to numerous receivers, rateless erasure codes, aka fountain codes, are an efficient way to cope with various packet losses, and reduce the need of a feedback channel. In its simplest form, a packet is composed of a payload $\mathbf{x}$ of network encoded data, and the metadata $\mathbf{c}$, the coefficient vector. $\mathbf{c}$ carries the information which of the plaintext packets $\mathbf{p}_1, \ldots, \mathbf{p}_n$ contributed to form $\mathbf{x}$. We consider the case where the encoded packets are XOR combinations of several source packets, and the metadata are bit-vectors of size n-bit. The receiver can decode the received encoded packets by solving the linear equation system formed by the received $\mathbf{c}$ s.

In a multi-hop propagation scenario, a forwarding node may not only want to forward received encoded packets $(\mathbf{x}_i, \mathbf{c}_i)$ but also generate freshly encoded packets $(\mathbf{x}_f, \mathbf{c}_f)$ by combining a subset of the already received encoded packets. Therefore, the node is required to compute $\mathbf{x}_f = \oplus_{i=1}^{l} \mathbf{x}_i$ and $\mathbf{c}_f = \oplus_{i=1}^{l} \mathbf{c}_i$. In the state of the art, the coefficient vector $\mathbf{c}$ that describes how the different plaintext packets were combined remains in clear-text, i.e. every intermediate node knows, which source plaintext packets $\mathbf{p}_i$ contributed to the received encoded packet. For example, assuming that the data stream is subdivided into packets $\mathbf{p}_1, \ldots, \mathbf{p}_8$ and in case the $\mathbf{c}_i$ equals the bit string 00000101, hence the encoded packet $\mathbf{x}_i$ has been generated by combination of the plaintext packets $\mathbf{p}_1$ and $\mathbf{p}_3$.

The introduced network coding paradigm has recently been applied to various applications, such as peer-to-peer data streaming or WSN code image update to name only a few. Obviously, there is a growing need to enrich such novel concepts with security means. Solutions regarding the integrity and authenticity of incoming encoded packets have already been proposed in [12,4,5]. Nevertheless, there is currently almost no work which explicitly deals with the confidentiality of encoded data. Exceptions are [3,1], the latter is the early version of this work. Two reasons may explain this: one can argue that the encoding in itself is already a weak mean of concealing the data and therefore no more additional protection is required. Examples following this direction are [4,6] assuming that the attacker can eavesdrop only on a subset of all transmission paths between source and destination (s). One can also argue that there is no research challenge in weaving confidentiality into the network coding paradigm by applying a cipher E: the only decision to be taken is whether to encrypt on the plaintext data or to encrypt on the encoded data. It is further possible to transmit either $(E(\mathbf{x}_i, \mathbf{c}_i))$ or $(E(\mathbf{x}_i), \mathbf{c}_i)$, or to apply the encryption on the plaintexts already $E(\mathbf{p}_i)$ before generating the $\mathbf{x}_i$. We state that all approaches, including ours, have their drawbacks either with respect to security, CPU investment or with respect to the flexibility for generating new encoded packets on its way to the final multicast destinations.

Our contribution: The contribution of this work is to conceal respectively obfuscate the data stream of network encoded packets, while allowing intermediate nodes to generate new encoded

☆ The work at hand is an extended version of the results presented at the IEEE Percom'10 Workshop SESOC [1].
* Corresponding author. Tel.: +49 40428758183.
E-mail addresses: Hessler@neclab.eu (A. Hessler), Kakumaru@neclab.eu (T. Kakumaru), Heiner.Perrey@haw-hamburg.de (H. Perrey), Westhoff@informatik.haw-hamburg.de (D. Westhoff).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.11.004

packets based on already received ones. This is done in a lightweight manner by purely hiding the information about the composition of the encoded packets. Examples where we see value for our solution are environments with restricted devices and/or in cases where energy saving of security enabled devices due to eco-IT aspects is of relevance. This may be a desirable feature particularly for wireless mesh networks. We evaluate our results with respect to two types of data streams: MPEG-2 video streams and Java bytecode.

2. Related work

Cai and Yeung introduced the problem of using network coding to achieve perfect information security against a wiretapper who can only eavesdrop on a subset of the transmission paths between source and destination(s) [4]. Feldman et. al. generalized and simplified the method by showing that the problem is equivalent to finding a linear code with certain generalized distance properties [5]. Tan and Medard have investigated a network coding scheme that has both a low network cost and a low success probability of the wiretapper [6]. Other improved solutions against a wiretapper have been proposed in [8,9]. However, in a wireless network scenario like we are assuming, an eavesdropper may collect packets which have been sent by the source. There have been other previous pieces of work relevant to the security of network coding which has been focused on pollution, in which the attacker nodes inject bogus packets in the network [11,13]. These contributions are mainly addressing the authenticity of the data sent by a legitimate sender, but do not provide any confidentiality. Jaggi et.al. introduced a solution that works in the presence of Byzantine nodes [14]. Ho et. al. has also referred to Byzantine modification detection [15]. There have been several research regarding a trade-off between network cost and network vulnerability for multipath traffic in wireless *ad hoc* networks [16,17]. When the link is to be resilient against wiretapping, multiple disjoint paths may be used. In general, it causes an increase of the network cost.

A similar approach to ours has been presented by Fan et al. [3]. They also propose to encrypt the coefficient vector. However, contrarily to our approach they use a public key based privacy homomorphism. Moreover, their focus is on convergecast traffic, where sensors report data to a sink node. The solution has the advantage to also provide source anonymity, since the source ID is masked as it is implicitly contained in the coefficient vector. However, their solution is not very practical, as the privacy homomorphism which they rely on produces a very large cipher. Hence, the encrypted coefficient vector that is sent over the network is growing to a few mega byte.

3. Network and adversarial model

3.1. Network model

The goal of the network is to transmit large bulks of data (i.e. larger than hundredfold the MTU size) which are fully available at the sender side in a multi-hop manner to a multicast group. The network medium is considered to be noisy and highly error-prone. This is why we believe our solution is most promising in wireless environments. The network model consists of numerous (wireless) nodes distributed forming a connected graph. We assume that the set of nodes is partitioned into three distinct roles: source, forwarder, and receiver. We assume that during the transmission of the data, the receivers remain connected to the source, possibly over several hops. There can be several sources, and forwarders can come and go. However, in this work we will keep a model with only one source, and the network remains static. We assume a flow of data that comes from the source to the set of receivers, and possibly relayed by the set of forwarders. A forwarder distinguishes itself from a receiver by the fact that it is not interested in the data, but cooperatively forwards it, such that eventually all the receivers completely collect the data transmitted by the source. Furthermore, a routing overlay has to be available such that the diffusion of the data can be achieved efficiently. Such an overlay is build upon an interest for the data transmitted by the source. To enhance the resiliency against eavesdroppers, a multipath message propagation is advisable.

3.2. Adversarial model and security requirements

We assume that the attacker is omnipresent and thus is in complete control of the wireless channels over the whole network. She can eavesdrop packets over the wireless broadcast medium or control the communication channel to delete, modify, and send data. However, she cannot capture nodes from the receiver or the source sets. As the forwarders do not have to store any sensitive cryptographic material, their capture is irrelevant in our model. In this sense, our adversary model is in line with the classical Dolev-Yao threat model [2]. Finally, an attacker should not be able to interfere with the decoding process infinitely. We assume to have a mechanism to recover from injection attacks: Once she stops attacking actively the network, the receivers shall be able to decode correctly the source data.

The security goal we are aiming at in this work at hand is to *obfuscate the data* while minimizing computational and transmission overhead. Of course, other means like ensuring authenticity and integrity of the encoded data are also important. An exemplary approach for the latter protection aim is described in [12].

4. CNC – concealed network coding

Our approach which we termed *Concealed Network Coding* (CNC) is simple but meaningful: we propose to encrypt the coefficient vector instead of the encoded packet such that $(\mathbf{x}, E(\mathbf{c}))$. To avoid giving up any flexibility regarding the composition of new and meaningful encoded packets on its way to the final destination, E should belong to a specific class of encryption transformations. It should belong to the encryption transformation class of privacy homomorphisms, such that it holds $E(a \oplus b) = E(a) \otimes E(b)$ for any plaintext pair a and b, a properly chosen additive operation $\otimes$ on the ciphertext, and another additive operation $\oplus$ on the plaintext. The benefits for such a construct $(\mathbf{x}, E(\mathbf{c}))$ are manifold:

- Although the encoded payload $\mathbf{x}$ is not directly encrypted, the fact that the coefficient vector is encrypted provides obfuscation for $\mathbf{x}$ too; Recall that $\mathbf{x}$ is a random combination of d plaintext packets neither providing any information about the concrete d nor revealing the chosen plaintext packets denoted as $\mathbf{p}_1, \ldots, \mathbf{p}_d$ without loss of generality.
- Although the encoded payload $\mathbf{x}$ is implicitly concealed, still meaningful and syntactically correct network coding on two (or more) encoded and encrypted packets $(\mathbf{x}_i, E(\mathbf{c}_i))$ and $(\mathbf{x}_j, E(\mathbf{c}_j))$, can be performed: $(\mathbf{x}_f, E\ (\mathbf{c}_f)) = (\mathbf{x}_i \oplus \mathbf{x}_j, E(\mathbf{c}_i) \otimes E(\mathbf{c}_j))$; the operation $\oplus$ is the concrete encoding operation, e.g. in the easiest case the bitwise XOR operation, which is in fact an addition in the vector space $\mathbf{F}_2^n$; however, it is essential to point out that other operations may also be possible.
- In a setting with $E(\mathbf{x}, \mathbf{c})$, such an in-network coding approach on encrypted data is only possible if $E()$ is homomorphic respectively to the operation $\oplus$. This property is not achieved by many ciphers.
- Since $\mathbf{c}$ is encrypted, the attacker cannot modify encoded packets at will, she lacks the knowledge of which packets she is combining together.