Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 17 (2014) 162 – 169

Procedia Technology

Conference on Electronics, Telecommunications and Computers – CETC 2013

Efficiency of rateless secure network coding

Elke Franz*, Stefan Pfennig, Tobias Reiher

Technische Universität Dresden, Faculty of Computer Science, 01062 Dresden, Germany

Abstract

Network coding allows to increase the throughput as well as the robustness of data transmissions. Regarding robustness, especially rateless network coding was shown to be beneficial. However, the vulnerability of mere network coding against attacks requires introducing security mechanisms, and as known, security implies costs. Within this paper, we evaluate the efficiency of secure network coding schemes applied in a rateless manner. Our results show that secure rateless network coding schemes can still outperform routing. The actual efficiency depends on parameters like generation size, packet size, or network topology. Since conflicting efficiency parameters cannot be fulfilled in equal measure, selection of a secure network coding scheme and its parameters should be done adapted to the actual communication requirements.

Peer-review under responsibility of ISEL – Instituto Superior de Engenharia de Lisboa, Lisbon, PORTUGAL.

Keywords: Network coding; efficiency; security; pollution attacks

1. Introduction

Network coding [1] is a promising approach for increasing throughput, energy efficiency, and robustness of data transmission, hence, its use in various contexts was investigated [2–4]. In contrast to common routing, nodes in the network do not only forward data packets, rather, they compute linear combinations of the data packets they receive. Usually, data to be transmitted is organized in generations [2]. If the size of the generation equals the network capacity, the whole generation can be transmitted at once from the sender (source) to the recipients (sinks).

In the rateless scenario, the sender sends linear combinations computed from the data packets of one generation until he gets acknowledgments from all recipients. Possibilities for rateless network coding and for managing this feedback have been studied, e.g., in [5–7]; but these studies do not focus on secure network coding schemes.

However, the vulnerability of network coding against various passive as well as active attacks requires introducing security mechanisms. Since especially the pollution of data packets is critical due to its damaging influence on the subsequent processing, we focus on schemes providing security against this type of attack. Numerous approaches for detecting polluted packets have been published. Of course, security increases costs, e.g., in terms of communication or computation overhead. To the best of our knowledge, the existing evaluations of the performance of rateless network coding do not explicitly consider secure network coding schemes.

* Corresponding author. Tel.: +49-351-463-38076 ; fax: +49-351-463-38255.
E-mail address: elke.franz@tu-dresden.de

Peer-review under responsibility of ISEL – Instituto Superior de Engenharia de Lisboa, Lisbon, PORTUGAL.
doi:10.1016/j.protcy.2014.10.224

The contribution of this paper is to provide results regarding the efficiency of selected secure network coding schemes applied in a rateless manner. Particularly, we consider the dependency of efficiency parameters on the generation size, packet size, and network topology. Our results indicate that rateless network coding still provides benefits if security mechanisms are introduced. Since the efficiency parameters may be contradictory, selecting a network coding approach and setting the necessary parameters should be done adaptively.

The paper is organized as follows. The selected network coding schemes are sketched in Section 2. In Section 3, we describe the test settings and experiments. Subsequently, the results of the experiments are discussed in Section 4. Finally, Section 5 concludes and gives an outlook.

2. Selected approaches for secure network coding

The common notation of network coding is based on a directed, acyclic graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with a set of nodes (vertices) V and a set of links (edges) E. In the multicast scenario also considered in this paper, a sender $\mathsf{S} \in \mathsf{V}$ transmits data packets over forwarders $\mathsf{F_i} \in \mathsf{V}$ to a number of recipients $\mathsf{R_i} \in \mathsf{V}$ (Fig. 1). The forwarders compute linear combinations from the data packets $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n}) \in \mathbb{F}_q^n$ they receive over their l incoming edges

$$\boldsymbol{y} = \sum_{i=1}^{l} \alpha_i \boldsymbol{x}_i \tag{1}$$

and send these combinations over their outgoing edges to successive nodes. The recipients can decode the original data by solving a system of linear equations once they received a sufficient number of linear independent data packets.

In random linear network coding (RLNC), the nodes randomly select the linear coefficients $\alpha_i \in \mathbb{F}_q$. The approaches we have selected for our evaluations are based on Practical Network Coding (PNC, introduced in [2]), a practical implementation of RLNC. In PNC, data to be sent is divided into data packets $\boldsymbol{p}_i = (p_{i,1}, p_{i,2}, ..., p_{i,m}) \in \mathbb{F}_q^m$. These data packets are amended by a unit vector $(\beta_{i,1}, \beta_{i,2}, ..., \beta_{i,h}) \in \mathbb{F}_q^h$ that represents the global encoding vector GEV. The resulting data packets of $n = h + m$ symbols are organized in matrices called generations. Only data packets from one generation can be combined. Therefore, data packets are tagged with a generation identifier *gid* that needs to be unique within the system. One generation contains h data packets, hence, h represents the generation size.

The network capacity C (max-flow min-cut, [1]) represents the amount of data that can be transmitted to the recipients at once in the multicast scenario. In rateless network coding, the sender sends data packets until he gets acknowledgments from all recipients confirming the successful delivery of the data packets. In this scenario, the generation size can be chosen independent from the network capacity. The generation size obviously influences the efficiency of network coding, e.g., while a larger generation size increases the throughput, it also increases the decoding delay [5].

Since network coding in its basic form is vulnerable against attacks, the introduction of security mechanisms have to be considered. Pollution attacks are especially critical since polluted packets that are included into the linear combinations influence the whole subsequent processing and may prevent decoding in the worst case. Hence, various approaches have been introduced in the literature that aim at preventing the success of such pollution attacks.

Within this paper, we focus on approaches based on cryptography that enable intermediate nodes in the network to detect and discard polluted data packets that do not belong to the linear span of the valid data packets. Generally, the sender computes information necessary for verifying the validity of the data packets. For our evaluations, we selected four schemes that represent different approaches (for more details, we refer to the cited articles as well as [8]):

Homomorphic hashes (HH): As an example, we selected the approach introduced by Li et al. [9]. The sender computes hash values for all packets of the generations to be sent during a session with session identifier *sid*. The authenticity of the hash values is ensured by means of digital signatures. Due to the homomorphic property, intermediate nodes can check the validity of received packets by comparing the hash for the combined data packets to the combination of the original hash values.

Homomorphic signatures (HS): The sender computes one homomorphic signature for each data packet. The signatures are included in the data packets. Since they are homomorphic, a valid signature for a combined data packet can be computed by appropriately combining the signatures of the corresponding data packets. As an example, we